Designing an Accurate and Efficient Classification Approach
for Network Traffic Monitoring

A thesis submitted for the degree of
Doctor of Philosophy

Adil Fahad Al Harthi,
School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

January 9, 2015
This Ph.D thesis is dedicated to all my immediate family members 
and to all my teachers.
Declaration

I certify that:

a) except where due acknowledgement has been made, the work is that of the author alone;

b) the work has not been submitted previously, in whole or in part, to qualify for any other academic award;

c) the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program;

d) any editorial work, paid or unpaid, carried out by a third party is acknowledged;

e) ethics procedures and guidelines have been followed.

Adil Fahad Al Harthi
School of Computer Science and Information Technology
RMIT University
27th August, 2014
Acknowledgments

I thank Almighty Allah for his blessings, grace and guidance. And peace of Allah be upon the noble Prophet Muhammad and upon his family.

On my uneven but worthwhile journey toward my Ph.D. degree. Over the past four years I met not only challenges in work and life, but also many supportive individuals who gave me the confidence to overcome those challenges. I take this opportunity to thank everyone who gave me their valuable assistance during my Ph.D. study at RMIT University for their remarkable guidance and help. Without these people, the completion of this thesis would have been more difficult.

First and foremost, I would like to express my greatest gratitude to my supervisor Professor Zahir Tari for his support, knowledge, patient and belief in me. I will always consider myself most fortunate to have had the opportunity to work under the supervision of Professor Zahir Tari. I would like to thank my second supervisor Dr. Ibrahim Khalil for sharing his knowledge and providing insightful comments in the research work. I owe my thanks to Dr. Abdun Mahmood, Prof. Albert Y. Zomaya, Prof. Ibrahim Habib and Prof. Hussein Alnuweiri for their observations and providing collaboration opportunities that resulted in joint publications.

I would like to thank all my fellow colleagues from RMIT University and friends who have always been my source of inspiration and helped me through the highs and lows of Ph.D. life. I am especially grateful to my friend and collaborator Abdulmohseen Almalwi for motivating me to continue my Ph.D. and for our innumerable discussions that significantly improved the work. I extend sincere thanks to Fahad Alzahrani, Fahad Alotaibi, Saeed Alzahrani, Mohammed Alkhthami, Sultan ALamri, Abdullah ALamri, Najlaa Alshatri, Zhixin Zhou, Ali Almarhabi and Ali Alghamdi for their company and friendship.

I would like to take this opportunity to thank my country, and the government of Saudi Arabia for providing me and my family with sponsorship. Also, I would like to thank the Saudi Arabian Cultural Mission in Australia for their help and support.
I would like to thank the administrative and technical staff members of the Computer Science and Information Technology (CSIT) school who have been kind enough to provide advice and assistance in their respective roles.

Last but not least, I would like to thank my father Fahad Ateeg and my mother Saleha Saeed for their endless sacrifices without which I would not have been able to pursue and achieve my dreams. I would also like to thank my brothers (Adnan, Mohammed, Ahmed, Ibrahim and Abdulaziz), my sisters (Reem, Wafa, Hajar, Ahlam, Hanof and Olaa) and my cousin Mohammad Bin Thabet who were supportive all time. I owe my deepest gratitude to my beloved wife Kurayaman, and my daughter Yara for their unconditional love, continuous sacrifices and support. Without them I would not have been able to complete my Ph.D. degree. I would like to dedicate this thesis to my beloved family.
Credits

Portions of the material in this thesis have previously appeared in the following publications:


• A. Almalawi, A. Fahad, Z. Tari and I. Khalil, ”An Efficient $k$-nearest Neighbour Approach Based on Various-Widths Clustering”, IEEE Transactions on Knowledge and Data Engineering, 2015, *To appear*.


The thesis was typeset using the LATEX document preparation system.

All trademarks are the property of their respective owners.

**Note**

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.
Contents

Abstract 1

1 Introduction 4
  1.1 Importance of Network Traffic Classification 5
    1.1.1 QoS issues 5
    1.1.2 Intrusion detection system 6
  1.2 Limitations of existing work 7
  1.3 Research problem 11
  1.4 Overview of contributions 13
  1.5 Thesis organization 15

2 Related Work 16
  2.1 Port-based classification 17
  2.2 Deep Packet Inspection (Signature based classification) 18
    2.2.1 Protocol/State Analysis 19
    2.2.2 Behavior & Heuristic Analysis 20
    2.2.3 Pattern Analysis 21
    2.2.4 Numerical Analysis 21
    2.2.5 Connection pattern-based classification 22
  2.3 Statistics-based classification 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.1</td>
<td>The Results</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Classification of the traffic based on all the features</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Evaluation of “Goodness”</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Temporal Variation of FS Goodness</td>
<td>72</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Discussion</td>
<td>75</td>
</tr>
<tr>
<td>3.6</td>
<td>The Local Optimisation Approach (LOA)</td>
<td>76</td>
</tr>
<tr>
<td>3.6.1</td>
<td>The Proposed Algorithm</td>
<td>79</td>
</tr>
<tr>
<td>3.6.2</td>
<td>An Illustrative Example</td>
<td>79</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Result and Analysis</td>
<td>82</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Choice of Parameters</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Runtime performance</td>
<td>84</td>
</tr>
<tr>
<td>3.6.5</td>
<td>Impact of FS Techniques on Runtime</td>
<td>86</td>
</tr>
<tr>
<td>3.6.6</td>
<td>Comparing FS Techniques Computational Performance</td>
<td>88</td>
</tr>
<tr>
<td>3.6.7</td>
<td>Summary of Results with different Datasets and Limitations of LOA Approach</td>
<td>89</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusion</td>
<td>91</td>
</tr>
<tr>
<td>4</td>
<td>Optimal and Stable Feature Set for Traffic Classification</td>
<td>92</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>93</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Contributions</td>
<td>94</td>
</tr>
<tr>
<td>4.2</td>
<td>Optimality vs Stability</td>
<td>96</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Selecting Feature Set from Global Perspective</td>
<td>98</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Initial Investigation</td>
<td>98</td>
</tr>
<tr>
<td>4.3</td>
<td>GOA – Global Optimization Approach</td>
<td>102</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Integration of Feature Selection</td>
<td>103</td>
</tr>
<tr>
<td>4.3.2</td>
<td>The Adaptive Threshold</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>Conceptual View of the Adaptive Threshold</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>Extracting Stable Features</td>
<td>108</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Intensive Search Approach</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>109</td>
</tr>
</tbody>
</table>
5.4.2 Baseline Methods .................................................. 152
5.4.3 Quality Evaluation using Benchmarking Machine Learning Techniques 153
5.4.4 Experiment Setup .................................................. 153
5.4.5 Experiment Results and Comparison .................................. 155
5.4.6 Experiment Results .................................................. 155
  Overall Accuracy ...................................................... 155
  Precision and Recall ............................................... 157
  F-measure .......................................................... 160
5.4.7 Computational Efficiency .......................................... 162
  Efficiency of Transformation Methods ............................... 162
  Efficiency of Transformed Data on Classifiers ..................... 163
5.4.8 Scalability Test .................................................... 165
5.4.9 Quantifying Privacy ................................................ 167
5.4.10 Discussion and Summary ......................................... 170
5.5 Conclusion .......................................................... 170

6  SemiTra: A Semi-supervised Approach for Network Traffic labelling  172

  6.1 Introduction ........................................................ 173
  6.2 The Proposed Semi-supervised Traffic Flow labelling .......... 175
    6.2.1 Multi-view Layer ............................................... 177
    6.2.2 Initial Clustering Analysis .................................... 182
    6.2.3 Ensemble Clustering ........................................... 184
      Consensus Function of Clustering ................................ 185
      Cluster Mapping Process ........................................ 186
    6.2.4 Local Self-training ........................................... 190
      Choice of Supervised Learning Algorithms ..................... 190
    6.2.5 Global Self-training on Meta-level Features ............... 193
    6.2.6 Function Agreement and Labelling .......................... 194
  6.3 Experimental Evaluation .......................................... 196
    6.3.1 Datasets Used in Experiments ............................... 196
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.2</td>
<td>The Baseline Methods</td>
<td>197</td>
</tr>
<tr>
<td>6.3.3</td>
<td>The Experimental Setup</td>
<td>199</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Performance Metrics</td>
<td>200</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Analysis of the Experimental Results</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>Results on the Two-Classes Problem</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>Results on Multi-Classes Problem</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td>Running times and Scalability</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>Stability</td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>Discussion and Summary</td>
<td>211</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusion</td>
<td>212</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>214</td>
</tr>
<tr>
<td>7.1</td>
<td>Contribution</td>
<td>216</td>
</tr>
<tr>
<td>7.2</td>
<td>Future Work</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>222</td>
</tr>
</tbody>
</table>
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Evolution of network traffic classification approaches (between 1992-2014)</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>The process of network traffic classification consists of four parts: (1) Traffic Data Repository (from/to which traffic data are retrieved and stored), (2) Data Pre-processing (for traffic flow feature selection), (3) Classification Engine (which comprises of various types of classification methods), and (4) Dispersion Graph (for traffic visualisation) [Lee et al., 2011]</td>
<td>55</td>
</tr>
<tr>
<td>3.2</td>
<td>Feature selection process [Liu and Yu, 2005]</td>
<td>57</td>
</tr>
<tr>
<td>3.3</td>
<td>Final subset validation process</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>Classification of the traffic based on all features</td>
<td>70</td>
</tr>
<tr>
<td>3.5</td>
<td>Classification of the traffic based on features of the candidate FS techniques</td>
<td>71</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparison of effectiveness of existing FS techniques on three randomly chosen datasets (D1,D5,D9)</td>
<td>73</td>
</tr>
<tr>
<td>3.7</td>
<td>Comparing feature selection stability on traffic data</td>
<td>74</td>
</tr>
<tr>
<td>3.8</td>
<td>The LOA approach</td>
<td>77</td>
</tr>
<tr>
<td>3.9</td>
<td>Procedure of Local Optimization Approach (LOA)</td>
<td>81</td>
</tr>
<tr>
<td>3.10</td>
<td>Evaluation LOA against the selected FS techniques</td>
<td>83</td>
</tr>
<tr>
<td>3.11</td>
<td>Influence of parameterising LOA</td>
<td>85</td>
</tr>
<tr>
<td>3.12</td>
<td>Evaluation LOA against the selected FS techniques(a value 1 represents the lowest build and classification time)</td>
<td>87</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

3.13 Comparison of runtime performance ........................................ 88

4.1 Stability and optimality of FS techniques on real-traffic data ............... 102
4.2 The proposed Global Optimization Approach ................................. 103
4.3 Stability and optimality of GOA approach and the baseline FS techniques on real-traffic data ......................................................... 116
4.4 Comparing the accuracy and the performance of classification using the output set of GOA, FCBF-NB and BNN .................................... 120
4.5 Classification of the traffic using the candidate features (Temporal Stability and Spatial Accuracy) ..................................................... 123
4.6 The average effect of discretisation on the three traffic datasets ............. 126
4.7 Impact of the output of GOA and the discretisation technique on different classification algorithm ...................................................... 129

5.1 Data collection and data publishing ............................................. 133
5.2 A schematic representation of the privacy-preserving architecture for traffic data 137
5.3 Simulation of a water distribution system ..................................... 147
5.4 Comparison of the precision values of the PrivTra framework against the baseline transformation methods ................................. 158
5.5 Comparison of the recall values of the PrivTra framework against the baseline transformation methods ............................................ 159
5.6 Comparison of the scalability of the four transformation methods ........... 166
5.7 Comparison of privacy level for the four preserving privacy methods based on Friedman test ................................................................. 169
5.8 Privacy level comparison of all preserving privacy methods against each other based on Nemenyi test ...................................................... 169

6.1 Overview of the proposed SemTra approach .................................. 176
6.2 Different views of the traffic data using the multi-view layer ................ 181
6.3 The basic process of cluster ensemble ........................................ 185
6.4 Portion of clusters types (e.g. atomic, non-atomic and novel clusters) based
on the number of clusters .............................................................. 189
6.5 Overall accuracy comparison of all semi-supervised methods with each other
on the binary class datasets ......................................................... 204
6.6 F-measure comparison of all semi-supervised methods with each other on the
binary class traffic datasets ......................................................... 204
6.7 Overall accuracy comparison of all semi-supervised methods with each other
on the multi-class traffic datasets ................................................ 207
6.8 F-measure comparison of all semi-supervised methods on the multi-class traffic
datasets ....................................................................................... 207
6.9 Scalability of semi-supervised methods on DARPA dataset .................. 209
6.10 Runtime comparison of all semi-supervised methods with the Nemenyi test . 209
6.11 Stability comparison of all semi-supervised methods with the Nemenyi test . 211
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>An example of network applications</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>An example of features used as input for traffic classification [Zuev and Moore, 2005]</td>
<td>67</td>
</tr>
<tr>
<td>3.3</td>
<td>Flow statistics (percentages of flows) according to applications</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>The Goodness Rate (GR) of FS techniques on the ten datasets</td>
<td>72</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparing feature selection similarity on traffic data</td>
<td>74</td>
</tr>
<tr>
<td>3.6</td>
<td>Evaluation of FS techniques on the categorisation framework</td>
<td>76</td>
</tr>
<tr>
<td>3.7</td>
<td>The GR of LOA approach on 10 different datasets</td>
<td>82</td>
</tr>
<tr>
<td>3.8</td>
<td>Influence of different setting of support threshold</td>
<td>84</td>
</tr>
<tr>
<td>3.9</td>
<td>Summary of the datasets used for performance evaluation</td>
<td>89</td>
</tr>
<tr>
<td>4.1</td>
<td>The Process of selecting features globally</td>
<td>99</td>
</tr>
<tr>
<td>4.2</td>
<td>Procedure of Sequential Forward Selection (SFS)</td>
<td>110</td>
</tr>
<tr>
<td>4.3</td>
<td>Data statistics number of the flows</td>
<td>114</td>
</tr>
<tr>
<td>4.4</td>
<td>Evaluation of FS techniques on the categorisation framework</td>
<td>116</td>
</tr>
<tr>
<td>4.5</td>
<td>Standard confusion metrics for evaluation accuracy</td>
<td>117</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparison of GOA against FCBF-NB and BNN in terms of classification rate, subset size and runtime</td>
<td>118</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Comparative ranking of the most valuable features. FCBF-NB rank refers to [Moore and Zuev, 2005]. BNN rank refers to [Auld et al., 2007]. GOA rank refers to the proposed approach described in Section 4.3.</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Datasets used in the experiments</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of the overall accuracy of different classifiers using different transformation methods</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Comparing F-measure values of different classifiers using different transformation methods</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison of runtime performances taken for transformation (ms)</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of the performance of different classifiers based on transformed data</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Quantifying privacy of geometric data transformation</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Compliance of the proposed PrivTra framework and the related methods to desirable requirements.</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Summary of datasets used in the experiments</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Comparing overall accuracy and F-measure values of semi-supervised methods on eight binary-class traffic datasets</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>Comparing overall accuracy and F-measure values of semi-supervised methods on eight multi-class traffic datasets</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Comparison of runtime performances taken for labelling</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Comparison of the stability the SemTra approach and the baseline methods</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>Compliance summary of the semi-supervised performance based on empirical evaluation metrics</td>
<td></td>
</tr>
</tbody>
</table>
Abstract

Traffic classification is the process of identifying various applications and protocols existing in a network, which is crucial to both network management and security. In particular, a well-architected network should ensure the presence of a traffic classification module to prioritize various applications over the limited bandwidth for an effective Quality of Service (QoS). It is also important for the network operator to properly understand applications and protocols regarding network traffic in order to appropriately develop and implement an effective security policy.

Over the past decade, as traffic capacity continues to increase rapidly, traffic classification has been regarded with much concern both industrially and academically. In particular, three types of traffic classification methods are used to identify network flows: including port-based, payload-based, or flow statistics-based methods. The port-based method depends on scrutinizing standard ports utilized by popular applications. However, such a method cannot be relied upon all the time as not all present applications utilize standard ports. A few applications even overshadow themselves by using definite ports of distinct applications.

The payload-based method basically searches for the application’s signature in the payload of the IP packets. As a result, this method overcomes the problem of dynamic ports and hence is used widely in many industrial products. In spite of its popularity, this payload-based method does not work with encrypted traffic and requires a significant amount of processing and memory. In the recent academic research, the flow statistics-based method classifies traffic by creating additional new features from Transport Layer Statistics (TLS)
(e.g. packet length and packet arrival time) without necessitating Deep Packet Inspection (DPI), and then applying either supervised or unsupervised machine learning algorithms on the TLS data to categorize network traffic into pre-defined categories depending on identified applications.

This thesis is concerned with improving the accuracy and the efficiency of network traffic classification. Four research issues are being addressed to achieve the main aim of this thesis. The first research task is to optimize various feature selection techniques for improving the quality of the Transport Layer Statistics (TLS) data. The second research is intended to identify the optimal and stable feature set in the temporal-domain and the spatial-domain networks. The third research task is related to the development of preserving the privacy framework to help network collaborators in the spatial-domain network to publish their traffic data and making them publicly available. The final research task is related to automatically provide sufficient labelled traffic flows for constructing a traffic classification model with a good generalization ability, and to evaluate the generated traffic classification.

Firstly, a Local Optimisation Approach (LOA) is proposed to improve the quality of transport-layer statistics data and find representative features for accuracy and the efficiency network classifier. In particular, a Local Optimisation Approach (LOA) optimizes various feature selection techniques and uses the concept of support to filter out irrelevant and redundant features which provide no information about different classes of interest.

Secondly, the instability issue of the Local Optimisation Approach (LOA) and other existing feature selection techniques raises serious doubts about the reliability of the selected features. Thus, with the aim of enhancing the confidence of network operators, a Global Optimisation Approach (GOA) is proposed to select not only an optimal, but also a stable feature set to validate the accuracy and efficiency of traffic classification in the temporal-domain and the spatial-domain networks. In particular, the Global Optimisation Approach (GOA) selects optimal features set from a global prospective to avoid a situation where the dependence between a pair of features is weak, but the total inter-correlation of one features to the others is strong. Then, multi-criterion fusion-based feature selection technique, information-theoretic method and then a Random Forest framework with a new goodness
measure are proposed to estimate the final optimum and stable feature subset.

Thirdly, the sharing of traffic data among organizations is important, to create a collaborative and an accurate and a global predictive traffic classification model across the spatial-domain networks. However, the chance that such traffic data may be misused can threaten the privacy and security of data providers. Thus, a novel privacy-preserving framework is proposed for publishing traffic data and make them publicly available for the common good. In particular, the proposed privacy-preserving framework is designed to satisfy the privacy requirements of traffic data in an efficient manner by dealing with various types of features, including numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure.

Fourthly, in order to identify both the optimal and stable features, and also to build a traffic classification model with a good generalization ability using the supervised or unsupervised techniques, the traffic flows must be labelled in advance. Thus, a novel semi-supervised is proposed to reduce the effort of labelling traffic flows by exploiting a small subset of labelled data along with a larger amount of unlabelled once. Also, in the proposed semi-supervised approach, both supervised and unsupervised learning concepts are incorporated from local and global perspectives to improve the accuracy of the labelling process, and adaptively handle the presence of the new traffic applications.
Chapter 1

Introduction

In recent years, knowing what information is passing through the networks is rapidly becoming more and more complex due to the ever-growing list of applications shaping today’s Internet traffic. Consequently, traffic monitoring and analysis have become crucial for tasks ranging from intrusion detection, traffic engineering to capacity planning. Network Traffic Classification is the process of analysing the nature of the traffic flows on the networks, and classifies these flows mainly on the basis of protocols (e.g. TCP, UDP, IMAP etc.) or by different classes of applications (e.g. HTTP, P2P, Games etc.). Network Traffic Classification has the capability to address fundamentals to numerous network management activities for Internet Service Providers (ISPs) and their equipment vendors for better Quality of Service (QoS) treatment. In particular, network operators need an accurate and efficient classification of traffic for effective network planning and design, applications prioritization, traffic shaping/policing and security control. It is essential that network operators understand the trends in their networks so that they can react quickly to support their business goals. Traffic classification can also be a part of Intrusion Detection Systems (IDS) where the main goal of such systems is to detect a wide range of unusual or anomalous events, and to block unwanted traffic.
CHAPTER 1. INTRODUCTION

1.1 Importance of Network Traffic Classification

Accurate traffic classification is essential for addressing QoS issues (including provisioning, Internet pricing and Lawful Interception [LI]) and for security monitoring tasks.

1.1.1 QoS issues

One of the major challenges in the development of appropriate and effective QoS is the lack of a proper pricing strategy. An effective pricing strategy is central to the classification of the QoS that customers receive. A pricing strategy is also important because it facilitates generation of resources for the ISPs. Traffic classification has the capacity to sustain a realistic pricing mechanism. In the last few years, several pricing mechanisms have been proposed to create a suitable pricing plan. Generally, a good pricing model should charge consumers for the resources they utilise. This ensures transparency by eliminating opportunities for overcharging customers.

ISPs can develop effective and profitable business models through traffic classification. Most of the recommended Internet pricing techniques are effective because they ensure that consumers are charged fairly for the QoS. However, no QoS solution has been implemented extensively to satisfy customers’ needs. Consequently, appropriate QoS solutions should be implemented by taking into account technical efficiency, financial efficiency, and social effects. Technical effectiveness refers to the costs associated with using the technology of a given pricing scheme. Economic effectiveness refers to the effects of a pricing model on utilisation of a network. Hence, a good pricing model should be implemented consistently and transparently.

The cost of implementing QoS is important and should not exceed the revenue that is likely to be generated from it. Network stability and consistency should also be taken into consideration when implementing the new QoS. In addition, a programmed traffic classification should be incorporated in the QoS-based pricing model. Currently, ISP networks in most countries are required to provide lawful intercept abilities (LI). Traffic categorisation is a major solution to this legal requirement. Governments execute LI at different levels of abstraction. In the communications industry, a law enforcement group can appoint an
CHAPTER 1. INTRODUCTION

individual to gather intercepted information.

The traffic patterns in an ISP system can be detected through traffic classification. In addition, traffic classification can be used to identify the categories of applications that are being used by a client at a particular time. This information can be retrieved from the network without contravening privacy laws that regulate the use of the Internet.

Hence, IP traffic classification is important in the following ways. First, it facilitates the use of a class-based pricing model, which is fair to the customer and ensures sustainability. In this pricing model, ISPs are able to recover the cost of delivering the QoS by charging customers with different needs based on the services that they receive (as suggested in [Nguyen, 2009] and [Burgstahler et al., 2003]). Second, real-time IP traffic classification facilitates the development of automated QoS architectures. This leads to an effective transfer of information concerning QoS needs between Internet-based applications and the network. The resulting improvement in QoS signalling enhances the use of IntServ and DiffServ. Finally, the classification enables ISP providers to comply with the requirement that their networks must provide L1 capabilities.

1.1.2 Intrusion detection system

Apart from resolving QoS issues for ISPs, the other primary task of network traffic classification is to help network operators to recognize and identify anomalous behaviour. In particular, network operators have always been interested in keeping track of the anomalies occurring on their network in order to protect customers from external or internal threats. Over the past ten years, the number of vulnerabilities and attacks over the Internet, not only potentially targeting individuals’ security, but also national security, has increased enormously. In particular, the increased connectivity to the Internet and corporate networks by SCADA (Supervisory Control and Data Acquisition) systems for controlling the national critical infrastructures (e.g. electricity, gas, water, waste, railway etc) has expanded the ability of outsiders to breach security.

Examples of threats to SCADA include an attack on a SCADA-run sewage plant in Maroochy Shire, Queensland, causing 800,000 litres of raw sewage to be released into local
parks and rivers, causing the death of local marine life as well as discoloring the water and generating a noxious stench that permeated the atmosphere [Miller and Slay, 2010]; and the Davis-Besse nuclear power plant in Oak Harbor, Ohio, was attacked by the Slammer SQL server worm, which disabled a safety monitoring system of the nuclear power plant for nearly five hours [Poulsen, 2003]. More recently, Stuxnet [Falliere et al., 2011], a threat specifically written to target industrial control systems, was discovered. The threat was designed to damage nuclear power plants in Iran [Thomas, 2003]. Hence, the threat posed to critical infrastructures is far greater in terms of impact and scale of attack than common computer vulnerabilities, and have the potential to cause financial disasters and/or loss of life.

To cope with an increasing number of attacks and threats, a network traffic classification has been formulated as Intrusion Detection Systems (IDSs), and has become an important security tool for managing risk, and an indispensable part of the overall security architecture. In particular, an IDS is used as a second line of defence to identify suspicious and malicious activities in network traffic. It gathers and analyzes information from various sources within computers and networks, and once an attack has been detected, it informs the network administrator of the incident so that an appropriate response can be made. Therefore, an accurate network classification approach plays an important role in assisting network operators to protect their networks against possible threats and attacks.

1.2 Limitations of existing work

A number of network traffic classification schemes have been investigated, proposed and developed by the research community and the networking industry over the past ten years. To show the evolution of traffic classification approaches between 1992 and 2014, we used the search of Microsoft Academic to calculate the number of papers matching the phrase of “traffic classification”, “traffic flows” or “traffic identification” in the area of computer science (see Fig. 1.1). Firstly, well-known port numbers have been used to identify Internet traffic [Estan et al., 2003; Karagiannis et al., 2005]. Such an approach was successful because traditional applications used fixed port numbers; however, extant studies show that the current generation of P2P applications try to hide their traffic by using dynamic port numbers.
numbers. In addition, applications whose port numbers are unknown cannot be identified in advance. Another technique relies on the inspection of packet contents [Moore and Papagiannaki, 2005; Karagiannis et al., 2004; Haffner et al., 2005], and it analyses packets’ payload contents to see if they contain signatures of well-known or anomalous applications. Features are extracted from the traffic data and later compared to well-known signatures of applications provided by human experts. These approaches work very well for Internet traffic; however, studies [Auld et al., 2007; Erman et al., 2007b] show that these approaches have a number of drawback and limitations. First, they cannot identify new or unknown attacks and applications for which signatures are not available, so these techniques need to maintain an up-to-date list of signatures. This is a problem because new applications and attacks emerge every day, hence, it is not practical and sometimes impossible to keep up with the latest signatures. Secondly, deep packet inspection is a difficult task; since it requires significant processing time and memory. Finally, if the application uses encryption, this approach no longer works. Promising approaches [Auld et al., 2007; Erman et al., 2007b; Kim et al., 2008] that have recently attracted some attention currently are based on Transport Layer Statistics (TLS) data and efficient machine learning. This assumes that applications typically send data in some sort of pattern, which can be used as a means of classification.

![Figure 1.1: Evolution of network traffic classification approaches (between 1992-2014).](image-url)
CHAPTER 1. INTRODUCTION

of connections by different traffic classes. To extract such patterns, only TCP/IP headers are needed to observe flow statistics such as mean packet size, flow length, and total number of packets. This allows the classification techniques [Auld et al., 2007; Erman et al., 2007b; Kim et al., 2008] to have sufficient information to work with.

As can also be seen from Fig. 1.1, research in machine-learning-based network classification has been considered as a substantial domain of knowledge for traffic classification tasks. However, there are still a number of fundamental issues which need to be taken into consideration and resolved in order to improve the accuracy and efficiency of network security and network traffic engineering.

In this section, we briefly highlight the limitations of existing work.

• Improve the quality of transport-layer statistics data for accurate and effective network traffic classification

To classify Internet traffic data using Transport Layer Statistics (TLS) as a set of features, a dataset is prepared for analysis. In general, the size of Internet traffic data is very large, including thousands of traffic records with a number of various features (such as flow duration, TCP port and packet inter-arrival time). Ideally, the use of a large number of features should increase the ability to distinguish network traffic applications [Chou et al., 2008]. However, this is not always true in practice, as not all the features of traffic data are relevant to the classification task. Among a large number of features present in TLS, some may not be relevant, and therefore could mislead the classifier, while some others may be redundant due to high inter-correlation with each other [Guyon and Elisseeff, 2003]. If irrelevant and redundant features are involved in the analysis, both the efficiency and the accuracy of the classification can be affected. Nevertheless, a number of research studies have applied machine learning (ML) algorithms to the TLS data to address the problem of network traffic analysis. However, the quality of Transport Layer Statistics (TLS) data can degrade the performance of these ML techniques [Auld et al., 2007; Moore and Zuev, 2005; Lee et al., 2011].

• Identify the optimal and stable feature in the temporal-domain and the spatial-domain for accurate and effective network traffic classification
CHAPTER 1. INTRODUCTION

The issue of improving the accuracy of network classification in both the temporal-domain (across different periods of time), and the spatial-domain (across different network-locations) has been the subject of current studies [Li et al., 2009; Fahad et al., 2013]. However, many of these classical studies in this area neglect the insensitivity of feature selection techniques when selecting the representative set in the temporal-domain and the spatial-domain traffic data. For example, a given feature selection technique may select largely different subsets of features under small variations of the traffic training data. However, most of these selected features are as good as each other in terms of achieving high classification accuracy and better efficiency. Such an instability issue will make the network operators less confident about relying on any of the various subsets of selected features.

- **Preserve the privacy for traffic data publishing for accurate network traffic classification**

A number of efficient and accurate network traffic classification and intrusion detection systems using machine learning algorithms have been developed and attracted attention over the past ten years [Soysal and Schmidt, 2010; Govindarajan, 2014; Mahmood et al., 2010]. This is due to the ability of machine learning algorithms to (i) learn without being explicitly programmed, and (ii) cope with a vast amount of historical data, making it difficult for human beings to infer underlying traffic patterns from such an enormous amount of data. However, a key problem in the research and development of such efficient and accurate network traffic classification and intrusion detection systems (based on machine learning) is the lack of sufficient traffic data, especially for industrial network (Supervisory Control and Data Acquisition SCADA) systems [Chan et al., 2011; Mahmood et al., 2010]. Unfortunately, such data are not so easy to obtain, because organizations do not want to reveal their private traffic data for various privacy, security and legal reasons [Mahmood et al., 2010; Liu et al., 2010; Khelil et al., 2012]. Therefore, network traffic data should be further protected before being published, to prevent privacy leakage while still providing a maximal utility to data analysts using privacy-preserving methods.

10

(January 9, 2015)
• **Automatically labelling of raw traffic data for accurate and effective network traffic classification**

To overcome the problems of both supervised-classification and unsupervised-classification models, a limited number of semi-supervised-classification models have been proposed [Erman et al., 2007a; Rotsos et al., 2010]. These models work by utilizing a small set of labelled data along with a larger amount of unlabelled data to improve the performance of the traffic classification. However, most of these models suffer from accuracy and efficiency problems. This is due to (i) the assumption that unlabelled flows must be classified or belong to fixed traffic classes (known as force assignments), and (ii) ignore to discover the emergence of new patterns and applications. As such, an automatically labelling process for efficient and accurate creation of ground truth to train and test the different ML algorithms is needed instead of the tedious and costly manual labelling procedure.

### 1.3 Research problem

The main goal of this thesis is to answer the following research questions:

**A) How to optimize various feature-selection methods and improve the quality of transport-layer statistics data for accurate and effective network traffic classification?**

This research question focuses mostly on improving the quality of the transport-layer statistics data. In particular, the accuracy of the classification process will be affected by the large number of irrelevant features which provide no information about different classes of interest and worsen the accuracy. The efficiency of the classification process will also be poor due to highly correlated features (referred to as redundant), which increases the number of features that need to be learnt, and consequently increases the runtime of building and validating the classifier. Therefore, improving the quality of the transport-layer statistics data is required in order to find representative features by optimizing various feature selection techniques which are used as a knowledge discovery tool for identifying robust and truly relevant underlying characteristic features.
CHAPTER 1. INTRODUCTION

B) How to identify the optimal and stable feature set in the temporal-domain and the spatial-domain for accurate and effective network traffic classification?

Many feature selection (FS) techniques have been developed in the literature (e.g. [Moore and Zuev, 2005; Auld et al., 2007; Yuan et al., 2010; Williams et al., 2006]) with a focus on improving accuracy and performance by discarding the relevant and/or redundant features. However, these studies neglected the insensitivity of the output of FS techniques to variations in the training dataset across different period of time (known as temporal-domain), and across different network-locations (known as spatial-domain). The instability issue of the feature selection raises serious doubts about the reliability of the selected features to validate the accuracy and efficiency of traffic classification in the temporal-domain and the spatial-domain network. As such, extensive analysis is desirable to provide insight into the main factors that affect the stability of the feature-selection process, and the relationship between stability and predictive performance (known as optimality) of feature selection.

Nevertheless, it would be ideal to ensure the globally optimal feature subset and address the principal causes of stability we are concerned with. This is important to build traffic classification models that will remain accurate regardless of such time and location heterogeneity.

C) How to preserve the privacy for traffic data publishing for accurate intrusion detection systems and network traffic classification?

Preserving the privacy of network traffic data has specific and unique requirements that differ from other applications. In particular, network traffic data have various types of attributes: numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure. Thus, the vast majority of current privacy-preserving approaches are not readily applicable to private data in traffic networks. This is because their design assumes that the data being protected have to be numeric. To help organizations to publish their traffic data and make them publicly-available for the common good, a privacy-preserving approach must be
CHAPTER 1. INTRODUCTION

devised to improve the anonymization schemes and preserve data utility for accurate
data analysis by specifically dealing with the unique characteristics of network traffic
data.

D) How to “automatically” label raw traffic data for evaluating and building an
accurate network traffic classification?
The assessments of either the supervised or unsupervised traffic classification models
require labelled data. Nevertheless, in order to construct a traffic classification model
with a good generalization ability, the availability of a large amount of labelled data is
required. Unfortunately, labelled traffic data is scarce, time-consuming, expensive and
requires intensive human involvement. As such, it would be ideal to reduce the need
and effort to label traffic flows by exploiting a small subset of labelled data along with
a larger amount of unlabelled once. However, the subset of labelled data often can be
limited to a fixed number, which can diminish the accuracy of the labelling process,
especially with the emergence of new classes at any time in the network traffic flows.
Thus, the goal of this research question is to address such an issue, and improve the
accuracy of the labelling process by making it more adaptive to the presence of new
classes.

1.4 Overview of contributions

In response to the research questions discussed in Section 1.3, the following contributions are
made in this thesis:

1. Improve the quality of transport-layer statistics data for accurate and ef-
fective network traffic classification
A key issue with many feature selection techniques [Almuallim and Dietterich, 1994;
Duda and Hart, 1996; Hall, 2000; Liu and Motoda, 1998] used to select a small subset
from the original features of the Transport Layer Statistics (TLS) is that they are de-
signed with different evaluation criteria (e.g. information-based measure, dependence-
based measure, etc.). To address this issue, new metrics are presented to extensively
CHAPTER 1. INTRODUCTION

evaluate and compare such techniques based on different criteria and from different perspectives. In addition, a Local Optimization Approach (LOA) [Fahad et al., 2013] is proposed to address the limitations of existing feature selection techniques and generate a highly discriminant set of features.

2. Identify the optimal and stable feature in the temporal-domain and the spatial-domain for accurate network traffic classification

A Global Optimisation Approach (GOA) [Fahad et al., 2014b] is proposed with respect to both stability and optimality criteria, relying on multi-criterion fusion-based feature selection techniques and an information-theoretic method. Moreover, a new strategy based on a discretisation method is presented to significantly improve the accuracy of different ML algorithms which suffer from the presence of continuous-valued features in the temporal-domain and the spatial-domain traffic data.

3. Preserve the privacy for traffic data publishing for accurate network traffic classification

A privacy-preserving framework [Fahad et al., 2014a] is proposed for publishing network traffic data in an efficient manner while preserving privacy of data providers. Unlike traditional privacy-preserving approaches that are still frequently used in many real-world applications, the proposed privacy framework is designed specifically to deal with various types of attributes present in the traffic data, including numerical, categorical, and hierarchical attributes.

4. Automatically label raw traffic data for accurate network traffic classification

A new Semi-Supervised Approach is proposed for automatically Traffic Flows labelling (SemTra). SemTra alleviates the shortage of labelled data by incorporating the predictions of multiple unsupervised and supervised models. In particular, the prediction information for unlabelled instances is derived from diversified and heterogenous models, the strength of one usually complements the weakness of the other, thereby maximizing the agreement between them can boost the performance of the labelling process.
CHAPTER 1. INTRODUCTION

1.5 Thesis organization

The objectives of our research are addressed in seven chapters, with the current chapter presenting an introduction to the thesis. The remaining chapters of the thesis are structured as follows:

- **Chapter 2** provides a comprehensive literature review on the network traffic classification topic. In particular, the objective of this study is to critically analyse the state-of-the-art network classification techniques and categorizes them into different groups.

- **Chapter 3** presents new metrics (namely goodness, stability and similarity) to compare the effectiveness of existing feature selection techniques. In this chapter, and we proposed a Local Optimization Approach (LOA) to identify the “best” and to improve the quality of a network classifier.

- **Chapter 4** proposes a Global Optimization Approach (GOA) to obtain not only optimal, but also stable features sets from the temporal-domain and the spatial-domain, relying on fusion multi-criterion feature selection techniques and an information-theoretic method.

- **Chapter 5** presents a new privacy-preserving framework to satisfy the privacy requirements of traffic data by dealing with various types of attributes, including numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure.

- **Chapter 6** introduces a new semi-supervised approach for network traffic labelling (SemTra) to obtain sufficient and reliable labelled data for effective training. In this chapter, supervised and unsupervised learning are incorporated from local and global perspectives to discover the emergence of a new class and significantly boost the performance of the labelling process.

- **Chapter 7** summarises the main contributions of this thesis and discusses the possibility of further research to increase the performance of network classification task.
Chapter 2

Related Work

The main purpose of the network scheduler is to classify differently processed packets. Today, myriads of different techniques are used to attain the network classification. The simplest of these would be to correlate parts of data patterns with the popular protocols. A rather advanced technique statistically analyzes the packet inter-arrival times, byte frequencies, as well as packet sizes in order. After the traffic flow classification has been done through a certain protocol, a pre-set policy is used for the traffic flow, including the other flows. This process is conducted in order to achieve a particular quality, i.e. QoS. This application should be conducted at the exact point when traffic accesses the network. It should also be carried out in a manner that allows the traffic management to take place, isolating the individual flows and queue from the traffic. These individual flows and queue will be shaped differently as well. The next network traffic classification approaches [Estan et al., 2003] [Moore and Papagiannaki, 2005] [Moore and Zuev, 2005] are considered as the most reliable, as they involve a full analysis of the protocol. However, these approaches have certain disadvantages, the first being the encrypted and proprietary protocols. As they do not have a public description, they cannot be classified. Although the implementation of every single protocol possible in the network is a thorough approach, in reality this is extremely difficult. A single-state tracking protocol might demand quite a lot of resources. Consequently, the method loses its meaning and becomes impractical and unattainable.

This work focuses on analyzing each method, with its advantages and disadvantages.
CHAPTER 2. RELATED WORK

The following are the four methods of network traffic classification available:

1) Port-based classification
2) Deep-Packet Inspection
3) Connection pattern-based classification
4) Statistics-based classification

2.1 Port-based classification

One of the most popular methods used to classify the traffic on the Internet involves analyzing the packet’s content found at a certain point in the network. These packets typically contain source and destination ports, i.e. their addresses. Although ports represent the endpoints of the logical connections, their purpose does not end there. They also represent the means by which the program of the client determines the computer’s server program in the network. This method relies on the concept that port numbers [Estan et al., 2003] [Karagiannis et al., 2005] [Chen et al., 2008a] in TCP or UDP packets are constantly used by the applications. TCP SYN packets are analyzed by the middle network classifier. The port number of TCP SYN packet is then referenced with the Internet Assigned Numbers Authority (IANA)’s list [Cotton et al., 2011], which has all the registered ports. TCP SYN packets need to know the server side, which belongs to the TCP connection of the new client-server, in order for the classification to take place. UDP packets follow the similar process as the TCP SYN packets. Ranging from 0-65536, port numbers can be classified into three types. The first type belongs to the ports that are set for the privileged services (0-1024), i.e. the popular ports. The second type belongs to the ports known as registered (1024-49151). The third type are the private ports (above 49151), including the dynamic ones. The port-based classification of the traffic is determined by associating one popular port number with a provided traffic type, i.e. of correlating the transport layer’s port number with its application(s). For example, the port number 80 of the TCP correlates with the traffic of the http, whereas 6346 represents traffic of Gnutella etc. This is why the port-based method is seen as the easiest. It just requires insight into the packets’ header. And this is where its strength lies, in its simplicity and low cost. However, there are several disadvantages of using this method as well, the
first being that it cannot be applied to the allocations of the dynamic ports [Fraleigh et al., 2003]. For instance, web-classified traffic might be a different traffic that is using http. Hence, there is no method for matching a certain application to its port number, which is dynamically allocated [Moore and Papagiannaki, 2005]. Furthermore, a certain number of applications use port numbers which are assigned to different applications by IANA. In this way, they avoid detections and blockings from the access control operating systems. Many peer-to-peer (P2P) applications will often use other applications’ port numbers [Estan et al., 2003] [Karagiannis et al., 2005], simply because they have not registered their port numbers with the Internet Assigned Numbers Authority [Moore et al., 2001] [Keralapura et al., 2010]. And finally, there is a certain number of IP layer encryptions which hide the header (TCP or UDP), subsequently preventing the port numbers from being seen. All these disadvantages make the port-based classification method insufficient for all the applications. Subsequently, the idea of using more complex network classification methods has been suggested in the recent literature.

2.2 Deep Packet Inspection (Signature based classification)

As the recent literature has noted, the port-based technique often leads to traffic estimates that are not correct. This causes problems with the quality of the network management as well as with the wrongly-identified intrusions, i.e. viruses. Many have turned to toward the Intrusion Detection Systems (IDS). The need for the IDS appeared when the Internet suffered a number of virus outbreaks back in 2004. As the packet header inspection was not enough for the detection of the virus, the IDS vendors began conducting a deep analysis of the packet. Hence, the term “Deep Packet Inspection” as well as efficient and accurate methods [Moore and Papagiannaki, 2005] [Karagiannis et al., 2004] [Haffner et al., 2005] has been formed. Many applications can be classified using the information L3 and L4. However, this does not apply to all of them. Some applications have to use a certain message type, such as IM streams voice, or an additional sub-classification, such as URL, in order to be classified. The deep packet inspection will provide all of the above, doing both classification and sub-classification. Predefined byte patterns are examined within the packets in a stateful
or stateless manner to enable the protocol recognition. For example, the P2P traffic from
the eDonkey has the string “e3 38”, whereas the traffic from the web has the “GET” string.
This is possible only if both the packet header and payload are accessible. Deep packet
inspection techniques apply Signature Analysis to identify unique features, i.e. signatures
of each application. These signatures are then combined into a reference database, which is
used for comparing the particular traffic. This is conducted so that the classification engine
will identify that particular application. Subsequently, reference updates must be conducted
often so that recent developments, together with the applications, are combined with the
existing protocols.

There are different signature analysis methods [Moore and Papagiannaki, 2005] [Karagiannis et al., 2004] [Haffner et al., 2005] [Chen et al., 2008b]. The most popular methods
include:

1. Protocol/State analysis
2. Behavioral & Heuristic analysis
3. Pattern analysis
4. Numerical analysis

2.2.1 Protocol/State Analysis

A certain sequence of steps should be followed with certain applications. For example, when
the client requests the normal FTP GET, the server should provide a proper response to
it. When the communication protocols have already been defined and identified, then the
application that incorporates a certain communication mode will be identified. P2P ap-
lications can be identified by using the application level signatures, according to Sen et
al. [Sen et al., 2004]. To support the thesis, there has been an examination of BitTorrent,
DirectConnet, Kazaa, eDonkey and Gnutella, all of them being P2P protocols. The exami-
nation included different protocol stages: from the signaling and download, to the keep-alive
messages and synchronization. On the other hand, the analysis conducted by Dreger et al.
[Dreger et al., 2006] included the application-layer protocols as a means of detecting different
network intrusions, such as SMTP, FTP, HTTP and IRC. Whereas, the analysis conducted
by Ma et al. [Ma et al., 2006] concentrated entirely on the flow content by using the structural and statistical features so that the traffic can be identified. The traffic utilizes the same application-layer protocol. The analysis of the Fast Track, WNP and OpentNap P2P protocols was conducted by Spognardi et al. [Spognardi et al., 2005] so that the payload signatures could be identified. These signatures acted as a reference for Snort NIDS in order to monitor the network traffic. Dewes et al. [Dewes et al., 2003] conducted the analysis on a number of chat protocols in order to accurately identify different payload signatures. Their results showed the rate of 91.7% for the recall regarding every chat connection. The precision of their technique was at 93.13%. The protocol-based method fails with some applications simply because they might use protocols that are private and are not defined by traffic classification engine. Furthermore, there are applications which have communication orders that are almost identical, which impede this method. This paper indicates that using one analysis method is not enough for complete network traffic classification. In order for the network traffic to be classified completely, different approaches should be used.

### 2.2.2 Behavior & Heuristic Analysis

Communication behavior of an application differs when in the running mode, subsequently affecting network traffic differently. For instance, for each application there are two modes: interactive and sleep. They both differ according to the volume of the network traffic. When in the interactive mode, the data exchanged between the server and client is extensive, thereby sharply increasing the network traffic. When in the sleep mode, there will be a period communication with light packet that the server sends to determine whether the client is alive. This is done periodically, whereas the interactive mode involves constant communication. Subsequently, the analysis of the traffic behavior should be done, as it will provide insights into the applications which are running. This analysis will provide the basis for the classification of the applications. Furthermore, the underlying protocol might be classified using a statistical (heuristic) analysis of the packets that have already been inspected. These two analyses, behavior and heuristic, usually, complement each other perfectly. This is why Karagiannis et al. [Karagiannis et al., 2005], and Iliofotou et al. [Iliofotou et al., 2007] suggested techniques
where host behavior patterns are analyzed and identified on the transport layer. In order to observe the traffic flow, the application, functional and social levels of the patterns should be analyzed. Furthermore, this method is used by different anti-viral programs in order to detect viruses and worms.

### 2.2.3 Pattern Analysis

The classification engine can use a certain pattern (string/bytes/characters) [Risso et al., 2008] [Roughan et al., 2004], which is incorporated into the packet’s payload, in order to identify the protocols. Depending on the application, the pattern can be observed at different packet’s positions, not just at off-set. However, this does not create an issue for the classification engine to identify the packets. What does create an issue is that certain protocols do not contain these patterns, string and characters according to which the classification can be conducted. Therefore, this approach cannot be applied to all the protocols.

### 2.2.4 Numerical Analysis

Numerical characteristics, including the offsets, payload size and response packets, are a part of the numerical analysis [Bonfiglio et al., 2007] [Crotti et al., 2007]. An excellent subject for this analysis is the Older Skype version (pre-2.0), where the client’s request is 18 bytes, whereas the message that the client sends is 11 bytes. As there are many packets that need to be analyzed, the classification based on this analysis will take longer than the other ones. As there are a number of communications that are encrypted nowadays, one classification method is not sufficient for classifying all the applications. For instance, if the communication is encrypted, Deep Packet Inspection cannot inspect the information found in the upper layers. Hence, many classification methods have began employing the behavior and heuristic analysis, together with intelligent and clustering algorithms, which can help identify certain encrypted traffic. However, the issue of not being able to identify all the traffic still remains. This issue cannot be resolved by a single communication method, but rather a combination of different methods and techniques.

The advantage of Deep Packet Inspection methods [Moore and Papagiannaki, 2005] [Kara-
giannis et al., 2004] [Haffner et al., 2005] is that such methods can work well in the case of well-documented open protocols. Thus, with well-defined signatures, a correct and accurate decision can be guaranteed. However, the Deep Packet Inspection method required the availability of the real traces to give a good and sufficient feedback for choosing the perfect and best performing byte signatures. Some applications can be missed, or the method can produce false positives if the signatures are not kept up to date. Moreover, this Deep Packet Inspection method is based on a strong assumption that any packet payload could be inspected. However, the encryption of packet contents prevents the classification engine from extracting signatures or ports information.

### 2.2.5 Connection pattern-based classification

The communication pattern of a certain host is compared with the behavior pattern of different activities, i.e. applications in the connection pattern-based classification. [Karagiannis et al., 2004] utilize this idea, using the classification algorithm on P2P traffic. [Karagiannis et al., 2005] (BLINC) expanded the idea, thereby providing a general method applicable to a number of different applications. This general method used the source of destination ports, sets cardinality of unique destination ports, IPs and the sets of the magnitude in order to describe characteristics of the network flow, which match different applications. Thus, the entire network traffic was observed prior to constructing the nodes’ graphs, i.e. communicating hosts. Using filters, such as an edge on the packet, on SYN packet, etc, the edges are constructed. After the graph has been constructed, it is analysed, using the properties of quantitative graph description, including node degree distribution, joint degree distribution, connectivity metrics etc.

This method does not employ the packet payload in order to do the traffic classification [Karagiannis et al., 2005], which enables the encrypted content to be identified. However, some behavior patterns of the application cannot always be found easily, especially in cases where several different applications are being deployed simultaneously and using one host. There are some other disadvantages of the method, including the longer start-up time, lack of local decision, the need for many flows so that the communication pattern can be iden-
CHAPTER 2. RELATED WORK

tified. Finding the hosts takes time, and it cannot be conducted before the communication flows have been collected. Additionally, this connection pattern-based technique requires a large amount of memory since all hosts are collected. Certain problems might arise while conducting some graph metrics calculation, as the CPU load might be high as well.

2.3 Statistics-based classification

Machine learning has been extensively used in many elds, such as load prediction, medical diagnosis and search engines. In last decades, many algorithm based on statistical machine learning have been proposed [Auld et al., 2007] [Erman et al., 2007b] [Kim et al., 2008] [Nguyen and Armitage, 2008] in flow classification or bandwidth management. These approaches were able to achieve over 80% flow accuracy on average on their data sets. However, many open challenging still exists, such as imbalance characteristics of training data sets, and concept drifting of data distribution. In this section, we focus on presenting a detailed review of previous works on this topic.

2.3.1 Feature Selection

A feature is a calculated statistic from one or several packets, such as a standard deviation of inter-arrival times or mean packet length. A flow is described using a set of statistical features as well as related feature values. The set of statistical features is the same for every traffic flow, whereas the feature values depend on the network traffic class and thus differ from flow to flow. In [Zuev and Moore, 2005] [Moore et al., 2005], different datasets are used to define as many as 249 features, such as features of the flow duration, flow activity, and packets’ inter-arrival time. Even through there are many available features, the curse of dimensionality still remains a problematic issue for learning the data distribution in high dimensional datasets. As redundant features negatively influence the performance of algorithms, there are better options than training a classifier by utilizing the maximum number of features obtainable. One of the options requires the features to be divided into further sub-features based on their usefulness. However, how this is done is still one of the central problems of machine
learning. Recently, there have been several attempts to address this problem by using the reduction feature, which utilizes different requirements in order to define a feature as the most useful, based on the working constraints in the practical network traffic classification. The representative quality of a feature set considerably influences the level of effectiveness of machine learning algorithms.

By using feature selection algorithms, the process of carefully selecting the number and types of features used to train the machine learning algorithm can be automated. Feature selection algorithms [Tan, 2007] are broadly categorized as the filter, wrapper [Chawla et al., 2005] and hybrid models. The filter method scores and ranks the features relying on certain statistical metrics and chooses the features with the highest ranking values. Typically used statistical criteria include t-test, chi-square test, mutual information and principal component analysis. Even though filter approaches have low computation expense, they lack robustness against feature interaction. The wrapper method evaluates the performance of different features using specific machine learning algorithms, thereby producing feature subsets “tailored” to the algorithm used [Kohavi and John, 1997]. It searches the whole feature space to find the features to improve classification or clustering performance, but it also tends to be more computationally expensive than the filter model [Liu and Yu, 2005]. It is well-known that searching for optimal features from a high dimensional feature space is an NP-complete problem. The hybrid model attempts to take advantage of the filter and wrapper models by exploiting their different evaluation criteria in different search stages [Liu and Yu, 2005]. For example, the hybrid methods of t-test and genetic algorithm, principal component analysis and ant colony optimization, and the mutual information and genetic algorithm, have been proposed.

Van Der Putten et al. [Van Der Putten and Van Someren, 2004] found that the choice of feature selection is more important for obtaining high performance than the choice of traffic classification methods. Dunnigan and Ostrouchov use principal component analysis (PCA) to choose the most important features which contribute to the covariance matrix of observation data. In [Zander et al., 2005], Zander et al. use the feature selection to find an optimal feature set and determine the influence of different features. In [Roughan et al., 2004], Roughan et
CHAPTER 2. RELATED WORK

al. used up to four features to train the classifiers and achieved high traffic classification accuracy. Lei et al. in [Lei et al., 2008] for the first time proposed a hybrid feature selection method combined with Chi-Squared and C4.5 decision tree algorithm. This method also gives superior performance compared with the original C4.5 decision tree algorithm without selecting useful features. Valenti and Rossi in [Valenti and Rossi, 2011] considered both the nature of the input data and of the target traffic. The behavior features for P2P traffic flow are selected using two statistical metrics.

Because most feature selection algorithms are not effective for online traffic classification, Zhao et al. in [Zhao et al., 2008] proposed a real-time feature selection method for traffic classification. The underlying idea is that the selected feature subset is calculated based on the first several packet in the flow. To evaluate the performance, the feature selection method is combined with a decision tree classification method. Experimental results show that the proposed method can achieve good performance for online traffic flow classification. In [Jamil et al., 2014], Jamil et al. studied the online feature selection methods for P2P traffic. They discovered that the methods of Chi-squared, Fuzzy-rough and Consistency-based feature selection algorithms were the three best for P2P feature selection out of more than ten feature selection algorithms. They extended their previous works in [Jamil et al., 2013] to determine the optimal online feature selection algorithms for P2P traffic classification using J48 algorithm. In particular, J48 is a machine learning algorithm which makes a decision tree from a set of training data examples, with the help of information entropy idea. They also showed that it can obtain high accuracy 99.23% with low running time with the proposed feature selection method.

While most of the current feature selection methods have been proposed for balanced traffic data, in the case of imbalanced data, the feature selection is skewed and many irrelevant features are used. In [Zhen and Qiong, 2012a], a new filter feature selection method called 'balanced feature selection' (BFS) is proposed. The certainty coefficient is built in a local way guided by entropy theory, and the symmetric uncertainty is used in a global way. A search method is developed to select an optimal feature for each class.

Even though many feature selection algorithms in machine learning have been proposed
for the problem of imbalanced data distribution and concept drifting, more recently, Zhang et al. in [Zhang et al., 2012] proposed a method of weighted symmetrical uncertainty with the metric of area under ROC curve, called as WSU\_AUC, to optimize flow classification when both the issues of imbalanced learning and concept drifting exist.

### 2.3.2 Classification Methods

Based on the usage, machine learning techniques [Auld et al., 2007] [Chen et al., 2012] [Erman et al., 2007b] [Kim et al., 2008] [Nguyen and Armitage, 2008] can be classified into four different categories: numerical prediction, association mining, clustering and classification. Numerical prediction is a part of the supervised machine learning. It utilizes the models formed by the instances that were selected earlier in order to classify the unlabelled instances. Clustering is a part of the unsupervised machine learning as it combines similar instances into a particular cluster. Association mining searches for interaction between a subset of features. Classification and numerical prediction are almost identical, except for the difference in the output, which belongs to the continuous values category, not to the discrete one. Of these four, clustering and classification are considered as the most important techniques. Based on the machine learning, flow classification techniques can be classified into supervised, unsupervised and semi-supervised categories. Thus, the machine learning algorithms, utilized for the network traffic classification can belong to one of the three categories below:

1. Supervised Machine Learning algorithms
2. Unsupervised Machine Learning algorithms
3. Semi-Supervised Machine Learning algorithms

#### 2.3.3 Supervised Machine Learning Algorithms

Classification models [Bernaille et al., 2006] [Erman et al., 2006a] [McGregor et al., 2004] [Zander et al., 2005] [Moore and Zuev, 2005] [Chen et al., 2012], are constructed by utilizing a training set of instances, which corresponds to a particular class. The class needs to be known prior to learning, as supervised algorithms do not utilize the unknown classes. Subsequently, the classification model predicts the class memberships for the new instances.
CHAPTER 2. RELATED WORK

This prediction is conducted based on the examination of the unknown flows' feature values. The supervised learning establishes the knowledge which helps classify new instances into pre-defined classes. The learning machine is then provided with example instances which have already been pre-classified into classes. Previous instances are analyzed and generalized in order to construct the classification model, i.e. learning process's output. Hence, the main emphasis of the supervised learning focuses on modeling the input/output relationships. Hence, the aim of the supervised learning is to identify the mapping of the input features into an output class. The knowledge learnt is presented in the form of classification rules, a flowchart, decision tree etc. Also, it is utilized for the classification of the new unseen instances in the later stages. Supervised learning involves two stages: testing and training. In the learning stage, the training is used to analyze the given data, i.e. the training dataset, and construct a classification model. The testing stage, or classifying stage, utilizes the model from the training stage so that the new, unseen instances are classified.

In supervised learning, the learning algorithms could be grouped into two categories: parametric and non-parametric classifiers. For the parametric classifiers, the data distribution for each class is assumed to be known except for its distribution parameters. The class-conditional distribution can be obtained by estimating the parameters with training data. Some typical parametric classifiers include naive Bayesian, Gaussian mixture, and so on. For the non-parametric classifiers, the posterior probability is estimated directly from the data without any assumption about data distribution form. The common non-parametric classifiers are nearest neighbors, neural network, support vector machine, Gaussian process, etc. Researchers found that the application of these supervised machine learning algorithms to the traffic classification problem is able to achieve great performance. Moore and Zuev in [Moore and Zuev, 2005] introduced the correlation-based feature method to eliminate irrelevant and redundant features from traffic data, and then built an efficient Naive Bayes classifier in combination with a kernel estimation method to classify the traffic into different types of services and applications. The classification performance is very promising, up to 96%, for their collected datasets with the choice of 10 flow-behaviour features. It is worth noting that the concept of the Naive Bayes classifier assumes that the relation between fea-
tasures of a particular object is independent. However, in their recent work in [Auld et al., 2007], they pointed out that the traffic data, which is extracted from the header of packets and manually labelled using packet content, exhibit redundancy and interdependence among features describing each flow. Auld et al. in [Auld et al., 2007] proposed a Bayesian neural network which can incorporate the dependence among features, and thus more robust results are obtained.

In [Early et al., 2003], a decision tree classifier is trained with the probabilistic features of average packet size, Round Trip Time of a flow and FIN and PUSH in packets for accurately classifying http, ftp, smtp and telnet applications. They also reported that an accuracy of 93% in the flow classification was obtained. In [Roughan et al., 2004], the nearest neighbour method in combination with linear discriminant analysis is applied and built using only the features of average packet size and duration time of a flow to classify Internet traffic. For real-time classification of network traffic flows, Roughan et al. also used the support vector machine (SVM) method to generate a network classifier to aid network operators to classify the real-time traffic flows into seven classes of pre-defined applications [Roughan et al., 2004]. In particular, they considered the coarse classes of flows, including interactive, bulk data transfer, streaming and transactional etc. Also, to identify the best combination of the features for better classification performance, they developed a feature selection method.

Williams et al. in [Williams et al., 2006] investigated the elimination of non-informative features on five popular machine learning algorithms for Internet traffic classification, including Nave Bayes, C4.5, Bayesian Network and Nave Bayes Tree algorithms. In particular, only the features of packet lengths, total packets, total bytes, flow duration, inter-arrival times and protocol are used for training these classifiers. Meanwhile, empirical studies were conducted with little training data to show the computational performance of these algorithms. They show that similar classification accuracy can be obtained by these algorithms, and the computational performance of C4.5 has the fastest classification speed in comparison with the remaining algorithms.

Using supervised algorithms, the performance of the CorelReef approach which is port-based, the BLINC approach which is host behavior-based and also seven common statistical
feature-based methods have been evaluated extensively on seven different traces by Lim et al. in [Lim et al., 2010]. Because the traffic flows would dynamically change and the concept of flow would not be constant, the resulting trained classifiers have limited ability to adapt seamlessly. Instead of trying to differentiate one traffic application from another, in [Xie et al., 2012], Xie et al. propose a bootstrapping approach where the classifier learns to classify the types of traffic applications in isolation. They demonstrated that the proposed method is robust against any change to traffic flows, such as the emergence of new applications.

More recently, a network traffic classification benchmark called as NetraMark was presented in [Lee et al., 2011]. They considered six design guidelines comprising comparability, reproducibility, efficiency, extensibility, synergy, and flexibility, and integrated seven different state-of-the-art traffic classifiers. These seven machine learning algorithms include C4.5 decision tree, Naive Bayes, Naive Bayes Kernel Estimation, Bayesian Network, k-Nearest Neighbors, Neural Networks and Support Vector Machine. The final decision hypothesis is derived with a weighted voting process to obtain single best classification results. Chen et al., in [Chen et al., 2012] have proposed an efficient Malware Evaluator tool to categorize malware malwares into species and detects zero day attacks. In particular, the Malware Evaluator tool defines its taxonomic features based on the behavior of species throughout their life-cycle to build efficient learning models using both support vector machines and decision trees. In [Zhang et al., 2013c], Zhang et al. proposed a new learning framework to address the issue of very few training samples for traffic classification using correlation information. The correlation of traffic flow is incorporated into a nonparametric approach to improve the classification accuracy. The performance of the proposed approach was evaluated in terms of average overall accuracy against different size of training dataset varying from 10 to 50 per class. The experimental results show that the accuracy of the proposed approach outperformed the existing classifications approaches when only small size of traffic samples is available.

The main task of the network traffic classification involves identifying the traffic of the known applications inside the network packets of unseen streams. However, the challenge of correlating classes of network traffic, separated by the machine learning, to the applications
that cause the network traffic is one that should be dealt with. Hence, supervised machine learning requires the training stage, in order to provide the necessary link between the classes and applications. A priori flow classification is needed within the training datasets during the training stage, which makes supervised machine learning attractive for identifying a particular pattern/patterns as well as application/applications, which are of interest. Training of the supervised machine learning classifier, by using the examples of every possible class to be seen in practice, is the most efficient technique. In spite of that, the performance of the supervised machine learning classifier might deteriorate if the classifier is not trained using a mix of traffic or if the network links which are monitored begin noticing traffic of the applications that were previously not known. When the evaluation of the supervised machine learning scheme within an operational context is conducted, it is important to take into account: 1. how the new applications will be detected by the user, 2. how supervised training examples will be provided to the classifier, and 3. when the re-training will occur.

2.3.4 Unsupervised Machine Learning Algorithms

Unsupervised machine learning algorithms [Erman et al., 2006b] are also known as ‘clustering algorithms’. In unsupervised machine learning scenarios, no labelled data are available, and the clustering algorithms attempt to group traffic flows into different clusters according to similarities in the feature values. Because the clusters are not predefined, the algorithm itself determines class distributions in a statistical manner. This is useful for cases where several network traffics are unknown.

Clustering methods do not utilize the training instances that are predefined, as the supervised machine learning algorithms do. Instead, they utilize internalized heuristics in order to identify natural clusters [Fisher Jr et al., 1991]. These natural clusters are formed when the same-property instances are grouped together. Thus, by utilizing clustering methods that are different, different categories of clusters can be formed. For instance, when there is an instance that can be a part of several clusters, then it belongs to the overlapping group. If an instance belongs to one group, then it is a part of the exclusive cluster group. If, on the other hand, it is a part of the group that has a particular probability, then it belongs to the
probabilistic cluster category. If instances are divided into groups at the top level, and are then divided further all the way to the individual instances, then these types of instances belong to the hierarchical category [Fisher Jr et al., 1991]. Three clustering methods can be identified: incremental clustering (where instances are grouped according to the hierarchy), the probability-based clustering method (where instances are grouped into classes probabilistically [Witten et al., 1999]) and the K-means algorithm (where instances are grouped in separate clusters in domains that are numeric).

McGregor et al. in [McGregor et al., 2004] was one of the earliest works to use the concept of the unsupervised machine learning technique to cluster and group network traffic flows using transport layer features with Expectation Maximization (EM) algorithm. This approach specifically partitions traffic flows into different groups of applications based on similar observable properties of flows. However, the authors do not evaluate the accuracy of the clustering as well as identify the optimal features of traffic flows that help the EM clustering to produce the best outputs. In [Zander et al., 2005], Zander et al. proposed AutoClass approach which uses Bayesian clustering techniques and an extension of Expectation Maximization (EM) algorithm to address the limitations McGregor et al. in [McGregor et al., 2004] work by guaranteeing the converge of a global maximum, and also define the best set of features for better clustering performance. To find the global maximum rather than the local maximum, AutoClass repeats EM searches starting from pseudo-random points in parameter space, and thus it performs much better than the original EM algorithm. Both the early works in [Erman et al., 2006a] and [Zander et al., 2005] have shown that building a network classifiers using the clustering algorithms and the transport layer characteristics has the ability to improve the identification of Internet traffic applications. Erman et al. in [Erman et al., 2007a] proposed to use the K-means clustering algorithm which is a partition-based algorithm and the DBSCAN algorithm which is a density-based algorithm to evaluate the predicating performance instead of the AutoClass algorithm which is a probabilistic model-based algorithm. Similar to other clustering approaches, the K-means and DBSCAN clustering algorithms used Euclidean distance to measure the similarity between two flow instances. While the K-means algorithm produces clusters that are spherical in
shape, the DBSCAN algorithm is able to produce arbitrary shaped clusters, which enable a network classifier to find an accurate set of clusters with minimum amount of analysis. They also demonstrated that both K-means and DBSCAN perform better and work more quickly than the clustering method of AutoClass used in [Zander et al., 2005]. The K-means and DBSCAN clustering algorithms are tested on a real dataset and can achieve a recall of 80%. In [Erman et al., 2007b], the same clustering algorithms were applied to the transport layer characteristics of Internet traffic to build accurate network classifiers that can differentiate between Web and P2P applications, the resultant classifier models have obtained an accuracy of between 80% and 95%, precision of between 71.42% and 97.84%, and recall of between 62.10% and 97.32%.

To extract sets of traffic flows that have common of communication patterns, Bernaille et al. in [Bernaille et al., 2006] generated natural clusters of traffic applications by using only the first few packets of a TCP flow. In contrast to the previously published works, this method is the first to use the size of the first few packets of a TCP flow as the features rather than extracting the features based on the whole packets of a flow to accurately classify the traffic application at early stage. The underlying intuition is that the first few packets give sufficient information as they carry the negotiation phase of an application, which is a predefined sequence of messages proceeding through distinct phases. The traffic classification mechanism includes two phases. Firstly, learning phase is performed offline to cluster TCP flows into distinct groups using a set of training data. In their experiments, one-hour packet trace of TCP flows from a mix of applications was fed to the K-means clustering algorithm as a training set. In particular, the K-means clustering algorithm used the Euclidean distance to measure the similarity between flows resulting in forms of natural clusters that are used to define a set of rules. Secondly, classification phase is performed online to identify the accurate application type by using the previously generated rules to assign a new flow to an appropriate and corresponding cluster.

In [Yuan et al., 2008], Yuan et al. developed a new unsupervised machine learning approach for network traffic flow classification based on the concept of information entropy. Firstly, in order to partition the traffic flows into different levels, a clustering algorithm
CHAPTER 2. RELATED WORK

Based on the concept of the information entropy technique is sequentially applied on traffic data collected from different active hosts. Secondly, during the clustering process the parameters of the clusters and the dynamic properties of the clusters are adapted to categorize traffic flows into broad-based application types. The experimental results show that the classification accuracy of the entropy-based clustering are significantly better than existing clustering approaches. However, to further improve the accuracy of such approach a combination of the unsupervised learning method with a supervising learning method based SVM algorithm is proposed. More recently, a graph-based framework for clustering P2P traffic classification has been proposed by Iliofotou et al. in [Iliofotou et al., 2009]. In particular, the authors used the Traffic Dispersion Graphs (TDGs) to capture network-wide interaction. For a graph, each node is an IP address, and the edge between two nodes indicates some type of interaction between those two nodes. This graph is able to detect network-wide behaviour which is common among P2P applications and different from other traffic. In their work, graph-based classification first clusters traffic flows into natural groups using flow-level features in an unsupervised way without using prior application-specific knowledge. Then the generated TDGs are used to classify new coming flows into their corresponding clusters. Two bidirectional flows are considered in this method. The advantage of this method is not only its high predicative capacity, but also its visualization ability. Zhang et al. in [Zhang et al., 2013b] proposed a novel approach to tackle the issue of unknown applications in the extreme difficult circumstance of small supervised training samples. The superior capability of the proposed approach to detect unknown flows originating from unknown applications is relying on the sufficiently utilization of correlation information among real-world network traffic flows. To do so, two techniques have been introduced to first enhance the capability of the nearest cluster-based classifiers, and then combine the flow predictions to further boost the classification performance. Wang et al. in [Wang et al., 2013] proposed a novel approach for clustering the traffic flow, which is based on Random Forest (RF) proximities instead of Euclidean distances. The approach firstly measures the proximity of each pair of data points by performing a RF classification on the original data and a set of synthetic data. After that, a K-Medoids clustering is employed to partition the data points into K groups based on the
proximity matrix. Compared with the classic clustering algorithms, the results show that this method performs much better for traffic clustering in terms of both overall accuracy and per-class performance.

The major advantage of the unsupervised machine learning lies in the automatic discovery of classes that recognizes natural patterns, i.e. clusters in the datasets. However, these clusters need to be labelled so that the new instances are mapped to applications in an appropriate manner. The issue of mapping clusters one-on-one to the applications still remains in the unsupervised machine learning schemes. Theoretically, the number of clusters would be equal to the number of application classes, with every application dominating just one cluster group. However, in practice, there is discrepancy between the number of clusters and the number of application classes. In reality, there are always a larger number of clusters than there are application classes. Furthermore, there is the possibility that an application can spread over several clusters and even dominate them. Hence, the issue will arise when mapping from one cluster to the application that is the source. When conducting the assessment of the unsupervised machine learning within the operational context, it is important to take into account the way the clusters are labelled, i.e. the way they are mapped to certain applications. It is also important to consider how the labelling can be updated with every new application detected as well as the optimal amount of clusters (computational complexity, labelling and label lookup costs as well as accuracy balancing) [Nguyen and Armitage, 2008].

2.3.5 Semi-supervised Machine Learning Algorithms

A semi-supervised machine learning algorithm falls between supervised machine learning and unsupervised machine learning. Semi-supervised learning is able to make use of a small amount of labelled training data and a large amount of unlabelled data, and is widely used in the Big Data era because labelling the data is always an expensive task and unlabelled data is always easy to obtain.

In contrast to supervised machine learning, an accurate representation of data distribution is difficult to obtain in semi-supervised machine learning with a small amount of training data,
so that supervised learning is not possible in this scenario. Also, instead of simply using the knowledge of clusters grouped with the limited training data for the external validation, semi-supervised machine learning tries to use the limited knowledge to guide the further learning process. There are two major methods in semi-supervised machine learning: by adapting the similarity measure or modifying the search for better clusters. In similarity measure-based methods, a similarity measure has already been applied to the limited training data to obtain the initial clusters, but the similarity measure is adopted to satisfy the available constraints. Several semi-supervised methods fall into this category, including Euclidean distance with a shortest path algorithm, Mahalanobis distance with a convex optimization, hierarchical single or complete link, and K-means. In search-based methods, the algorithm of searching clusters is adapted to assist the clusters to fit the new constraints or labels. In terms of statistical learning, the semi-supervised learning methods include the methods of generative models, graph-based methods and low-density separation. The semi-supervised learning is similar to the process of concept learning for humans, where a small amount of instruction is provided before the self-learning and the experience or knowledge is accumulated during his/her future learning with a large amount of unlabelled input data. Because of this self-learning characteristic, the methods of semi-supervised machine learning are also introduced for the Internet traffic classification and achieved promising learning performance.

In [Erman et al., 2007a], Erman et al. introduced a robust semi-supervised learning method relying on a well-known partition-based clustering algorithm, namely K-means, for an accurate off-line and on-line traffic classification. First, K-means clustering is employed to partition a small amount of training data. Second, a mapping from the clusters to the various known classes is obtained according to the available labelled flows. The introduced mapping is adapted with the unlabelled data, and thus the clusters are learnt by mapping to the different flow types. The self-learning performance is promising as reported in [Lin et al., 2010]: high flow and byte classification accuracy (greater than 90%) is obtained over a six-month period with a small number of labelled and a large of unlabelled flows.

A graphical model is a common framework to incrementally learn domain-specific knowledge. In [Rotsos et al., 2010], Rotsos et al. proposed a probabilistic graphical models for
CHAPTER 2. RELATED WORK

semi-supervised traffic classification. They assumed the data samples satisfy the Gaussian
distribution, and extend the naïve Bayesian classifier to learn the unlabelled data. Unlike
methods such as SVM, the model described in this paper is able to obtain a set of well-defined
parameters that easily adapt the model to the requirements of the classification process and
achieve very good results with a significantly reduced training dataset. However, their works
depend on the accuracy of IP address detection; the performance would be poorer when
training and testing environments are different.

In [Qian et al., 2008], a Gaussian Mixture Model-based (GMM) was proposed as a new
semi-supervised classification method to accurately categorize different Internet flows. To
achieve an optimum configuration, a wrapper-based Feature Subset Selection method and
CEM clusters (FSSCEM) algorithm are combined. More recently, Zhang et al. [Zhang et al.,
2013b] introduced a semi-supervised clustering method based on the extended K-means clus-
tering algorithm. In particular, as the quality of K-means clustering outputs are affected by
the random selection of the clusters’ centers in the initialization phase, Zhang et al. used
the variance of the traffic flows to initialize clusters centers instead to boost the clustering
performance. Meanwhile, they selected the few labelled instances to perform a mapping from
the clusters to the predefined traffic class sets.

Instead of focusing only on the data instances, Wang et al. [Wang et al., 2011] consid-
ered the other available background information in the network domain to detect unknown
applications. They described this available information in the form of pair-wise must-link
constraints and incorporated them in the process of clustering. In particular, the three avail-
able constraints in the Internet traffic were used along with variants of the K-means algorithm
to perform hard or soft constraint satisfaction and metric learning. A collection of real-world
traffic traces from various locations of the Internet has been used to show the benefit of the
widely available flow constraints in improving the accuracy and purity of the cluster.

In [Li et al., 2013], Li et al. proposed a semi-supervised network traffic classification
method based on incremental learning to improve accuracy, time consumption and limited
application range in traditional network traffic classification. The proposed method takes full
advantage of a large number of unlabelled samples and a small amount of labelled samples
to modify the SVM classifiers. The utilization of incremental learning technology can avoid unnecessary repetition training and improve the situation of low accuracy and inefficiency in original classifiers when new samples are added. Wong et al. in [Wong et al., 2012] examined the P2P sharing protocols of BitTorrent. They proposed a new detection method that is based on an intelligent combination of Deep Packet Inspection (DPI) and Deep Flow Inspection (DFI) with semi-supervised learning.

A semi-supervised approach, proposed by Shrivastav and Tiwari in [Shrivastav and Tiwari, 2010], incorporates clustering and classification as two main stages. The training dataset is partitioned into several separate groups, i.e. clusters during the clustering stage. After the clusters have been formed, the classification, i.e. assigning class labels to the clusters, takes place by utilizing labelled data. In order to test the approach, a dataset, KDD Cup 1999, including both the attack and normal data, has been taken. The results from the testing are compared to the SVM-based classifier. And they are comparable. A self-training architecture was proposed in [Gargiulo et al., 2012]. This architecture incorporates a few base classifiers which allow for the traffic database to be automatically built up without prior knowledge about data. Furthermore, this database is based on raw tcpdump traces. The results of the real and emulated traffic traces show that intrusion detection systems trained on the given dataset have the same performance level as the systems trained on the hand-labelled data.

A combination of the unsupervised and semi-supervised machine learning techniques, called MINETRAC has been proposed by Casas et al. in [Casas et al., 2011]. This combination allows for different IP flow classes with the similar characteristics to be identified and classified. MINETRAC employs clustering techniques, utilizing Sub-Space Clustering, Evidence Accumulation as well as Hierarchical Clustering algorithms in order to inspect the inter-flows structure. It also allows for the traffic flows to be categorized according to natural groupings by connecting the data structure evidence. The evidence is provided by the same set of traffic flows. Semi-supervised learning is utilized by the automatic classification. A small part of the ground truth flows are used for the mapping of the known clusters into their applications, or network services. These applications/ network services are probably
CHAPTER 2. RELATED WORK

the source of the known clusters.

Zhang et al. in [Zhang et al., 2014] proposed a semi-supervised learning method to target the problem of new network traffic generated by previously unknown applications in a traffic classification system, called a zero-day traffic problem. By incorporating a generic unknown class into conventional multi-class classification framework, this problem becomes how to obtain the training samples of zero-day traffic. After that, they extracted the zero-day traffic information from a set of unlabelled traffic which are randomly collected from the target network.

There are two main reasons why semi-supervised machine learning is very useful in traffic flow classification. First, fast and accurate classifiers can be trained with a small number of labelled flows with a large number of unlabelled flows which are easily obtained. Second, the semi-supervised machine learning is robust and can handle previously unseen applications and the variation of existing concepts. However, the semi-supervised machine learning approaches would be misleading in their learning process, specifically when there are few labelled training data. Hence, the assumed model has to be accurate at the beginning of the learning process.

2.3.6 Ensemble Learning

To date, much research and work has been done in order to obtain a good classification or clustering performance using machine learning. Ensemble methods have many advantages in comparison with single learner methods that we previously discussed for traffic classification. In ensemble learning methods, weak learners work together and build final decision considering the result of each learner. Many different methods such as bagging and boosting have been proposed in classical machine learning for making the final decision.

In [Yan et al., 2012], Yan et al. proposed a weighted combination technique for traffic classification. This approach first takes advantage of the confidence values inferred by each individual classifier, then assigns a weight to each classifier according to its prediction accuracy on a validation traffic dataset. In [Reddy and Hota, 2013], Reddy and Hota proposed to use stacking and voting ensemble learning techniques to improve prediction accuracy. Several base classifiers were used in their method, including Naive Bayes classifier, Bayesian Network,
and Decision trees. Before training the classifiers, feature selection techniques were applied to reduce the number of features, thereby reducing the training time. They showed that a high classification accuracy up to 99.9% is obtained in their experiments. Meanwhile, their experimental results also showed that stacking performs better over voting in identifying P2P traffic.

He et al. in [He et al., 2008] and in [He et al., 2009] combined an ensemble learning paradigm with semi-supervised co-training techniques for network traffic classification. Co-training semi-supervised learning utilizes both labelled and unlabelled samples. Because unlabelled samples are used to refine the classifiers, a high accuracy can be obtained by training with a small number of labelled samples mixed with a large number of unlabelled samples.

In [Aliakbarian and Fanian, 2013] Aliakbarian and Fanian proposed a new ensemble method for network traffic classification. Firstly, they choose the best subset of features using a multi-objective evolutionary algorithm so a new dataset is produced with them. As comparative algorithms, two ensemble methods, bagging and boosting are used in this paper. In the bagging method, the feature vector is chosen randomly; then each feature subset inputs some weak learners. The final decision is obtained by using majority voting of these learners based on each learner’s accuracy. In their boosting algorithm, the difference is that if a weak learner classifies a sample incorrectly, the probability of choosing this sample will increase in the next weak learner. Results show the proposed ensemble method has better classification performance in comparison with other methods, specifically for P2P traffic.

Govindarajan in [Govindarajan, 2014] proposed a hybrid ensemble approach for network intrusion detection. The ensemble classifier was designed using a Radial Basis Function (RBF) and Support Vector Machine (SVM) as base classifiers. It was constructed by voting with the modified training sets which are obtained by resampling the original training set. They showed that this proposed RBF-SVM hybrid system is superior to the individual approach in terms of classification accuracy.

Both empirical results and theoretical analysis show that the ensembles tend to yield better results compared to the single base classifier. The advantage of ensemble methods
CHAPTER 2. RELATED WORK

comes from the diversity among the models. Moreover, the lower computation cost of weak classifiers makes them easier to apply network traffic analysis.

2.4 Issues Related to the Network Traffic Classifications

In this chapter, we also present a comprehensive study of other relevant areas of network classification, including summarization, sampling, ground truth and privacy-preserving for traffic data. This is an important task to discuss the open issues and challenges in the field that would help in improving the classification accuracy and efficiency.

2.4.1 Summarization

Network traffic monitoring can be considered as a knowledge discovery process in which the data of network traffic is analyzed. In a data mining scenario, data summarization is a useful tool to discover underlying knowledge in the data. Recently, many researchers have introduced the data summarization method for network traffic monitoring and intrusion detection [Xu et al., 2005a] [Cohen et al., 2007] [Mahmood et al., 2008] [Mahmood et al., 2011] [Hoplaros et al., 2014]. Currently, there are many existing data summarization methods for different applications, such as frequent itemset mining [Chandola and Kumar, 2007] and clustering [Aggarwal et al., 2005] [Cohen et al., 2007] [Mahmood et al., 2008] [Mahmood et al., 2011].

Xu et al. in [Xu et al., 2005a] considered that compact summaries of cluster information to provide interpretive report help network operators achieve security and management by narrowing down the scope of a deeper investigation into specific clusters and explain the observed behavior. To do so, an information-theoretic and data mining approach are used to extract clusters of significance on the basis of the underlying feature value distribution (or entropy) in the fixed dimension. Given the extracted clusters along each dimension of the feature space, the compact summarization method is used to discover “structures” among the clusters, and build common behavior models for traffic profiling. Hence, the essential information about the cluster such as substance feature values and interaction among the free dimensions is revealed in the structural behavior model of a cluster with a compact summary.
Cohen et al. [Cohen et al., 2007] developed algorithms that collect more informative summaries through an efficient use of available resources. Unbiased estimators that use these more informative counts were derived. The superior of these estimators are those with smaller variance on all packet streams and sub-populations. The proposed summarization algorithm generates a sketch of the packet streams, which allows us to process approximate sub-population-size queries and other aggregates.

Mahmood et al. [Mahmood et al., 2008] proposed a summarization framework based on the concept of clustering to provide network operators with a concise report. They investigated the use of BIRCH which is hierarchical-based clustering algorithm to efficiently discover the interesting traffic patterns. Their framework was designed to deal with mixed types of attributes including numerical, categorical and hierarchical attributes. In order to find multidimensional clusters and to deal with multivariate attributes in the network traffic records, they introduced three new distance functions within the BIRCH clustering algorithm. Then, these three distance functions are used to calculate the various types of traffic attributes and accurately describe the relationships among different records and clusters. The index nodes for the BIRCH can be considered as holding summaries for each cluster and then used to form the final summary report.

In [Mahmood et al., 2011], Mahmood et al. proposed a three-level technique to summarize network traffic. They generated a reasonably compact and accurate summary report from a given network traffic trace. They firstly applied hierarchical cluster formation to a traffic trace to identify a detailed set of aggregate traffic flows, and then extract a compact summary report by applying a summarization algorithm to the clusters. Similar to their previous work, each node corresponds to a cluster $C$, and the CF-entries in the node correspond to the sub-clusters $C_1, \cdots, C_l$ of $C$. Two summarization methods were proposed using the size and homogeneity of clusters.
2.4.2 Privacy-Preserving

Network customers have become increasingly concerned about their personal data, especially with the great development in communication and IT systems. Thus, to protect the right to privacy, numerous statutes, directives, and regulations have been developed in recent decades [Guarda and Zannone, 2009]. The “right to privacy” is defined as “the right to be let alone” in Warren and Brandeis’ report [Warren and Brandeis, 1890]. The current regulations in many countries enforce strict policies for storing and processing the personal data, which aim to guarantee people control of the flow of their personal data. The services suppliers in their IT systems are trained to implement these privacy regulations when handling personal data.

Guarda and Zannone in [Guarda and Zannone, 2009] helped researchers by providing them with referenced work for clear definition of privacy and data protection policies when developing privacy-aware systems, including the languages and methodologies. They also analysed the current proposals in the corresponding research area for explicitly addressing privacy concerns. They reviewed the state-of-the-art in Privacy Requirements Engineering, Privacy Policy Specification, and Privacy-Aware Access Control and the relationships among these research areas in different countries to guarantee the consistency of enterprise goals, data protection and privacy policies.

Park Yong in [Park Yong, 2011] undertook an empirical study to examine the relationship between online market structure and the provision of privacy protection in a composite sample of 398 heavily trafficked and randomly selected U.S. commercial sites to answer the following question: How do online market conditions and website business characteristics affect the level of privacy protection?. Their analysis shows that most corporate awareness does not readily translate into specific provisions of privacy protection, even though other scholars have found that managerial attention to privacy concerns in fact has increased recently, indicating a possible role of the markets in regulating privacy in different ways.

However, it is remarkably hard to keep Internet communication private. General privacy-preservation methods are committed to data protection at a lower privacy level, and the research into privacy protection methods is focused on data distortion, data encryption, and
so on. One method of protecting the privacy of a network connection is to use an encrypted link to a proxy or server. Bissias et al. in [Bissias et al., 2006] presented a straightforward traffic analysis attack against this kind of encrypted HTTP stream. Surprisingly, the source of the traffic can be effectively identified in their proposed attack. A designed attacker firstly creates a profile of the statistical characteristics of web requests from interesting sites, including distributions of packet sizes and inter-arrival times. After that, they compare the candidate encrypted streams with these profiles. They show that the attacker achieves 40% when 25 candidate sites are considered, and achieves 100% accuracy for three candidate sites. However, the accuracy would be decreased when there are longer delays after training.

The works of Bissias demonstrated that the supposedly secure channels on the Internet are prone to privacy infringement due to packet traffic features leaking information about the user activity and traffic content such as packet lengths, directions, and times. Iacovazzi and Baiocchi [Iacovazzi and Baiocchi, 2014] called this technique “traffic masking”. In [Iacovazzi and Baiocchi, 2014], they defined a security model that indicates what the best target of masking is, and then proposed the optimized traffic masking algorithm that removes any leaking (full masking). After that, the trade-off between traffic privacy protection and masking cost, namely the required amount of overhead and realization complexity feasibility, can be determined.

### 2.4.3 Discretization

We have to note that the network traffic flow data is continuous. Most of the current machine learning methods such as decision tree-based learning and Naive Bayes methods, cannot be applied directly to the continuous features. To make those machine learning methods work, discretization is one of the methods used to cut the data into ranges and apply a variable number of cuts to the continuous attributes. Mazumder et al. [Mazumder et al., 2012] considered a discretization solution which partitions numeric variables into a number of sub-ranges and treats each such sub-range as a category. We measured the contribution of a given interval corresponding to a particular decision (normal or anomaly). An ideal discretization method which minimizes the number of intervals without significant loss of class-attribute
CHAPTER 2. RELATED WORK

mutual dependence was proposed by maximizing the interdependence between class labels and attribute values. There are three sub-modules to divide the task of discretization prior to the learning process. They firstly determine the number of discrete intervals, find the width or the boundaries of the intervals depending on the range of values of each continuous attribute, and map the attribute values from the continuous domain to the discrete domain.

As the Naive Bayes classifier suffers from continuous attributes, Liu et al. [Liu et al., 2008] applied the discretization method on traffic data for accurate Internet identification. The underlying idea of this method is that the discretization provides an alternative to probability density estimation when Naive Bayes learning involves continuous and quantitative attributes. The efficiency of the Naive Bayes and the discretization method has been demonstrated with AUCKLAND VI and Entry traffic datasets.

In [Lim et al., 2010], Lim et al. investigated the performance of the C4.5 Decision Tree algorithm when used with ports and the sizes of the first five consecutive packets. The performance results showed that the C4.5 Decision Tree algorithm was able to achieve the highest accuracy on every trace and application, with 96.7% accuracy on average, due to its own entropy-based discretization capability. Based on this observation, they proposed an entropy-based discretization technique to discretise the input flow features and to improve the classification tasks of other machine learning algorithms as well (e.g. Naive Bayes and k-NN). The experimental study showed that the entropy-based Minimum Description Length algorithm can significantly improve the performance of the candidate ML algorithms by as much as 59.8%, making all of them achieve more than 93% accuracy on average without considering the tuning processes of any algorithm. The authors have compared the performance of the entropy-based discretization against one of the simplest discretization technique, namely the Equal-Interval-Width, and they found the proposed technique can significantly improve the classification accuracy by about 13%.

2.4.4 Sampling

A typical network traffic flow monitoring involves a collection of flow records at various intermediate network nodes/points, such as routers. While the monitoring of a fundamental
task of network traffic flow seems to be easy, collecting and observing the traffic flows at high speeds is an extremely challenging task especially under excessively resource-constrained environments. It is impractical to record all the traffic flow data and learn the patterns of these traffic flows because the resource requirements (e.g. memory and CPU) in routers are mostly used for number of vital functions such as route computation, forwarding, scheduling, protocol processing and so on. Therefore, in network traffic flow monitoring, routers randomly select a subset of packets using sampling techniques to meet this challenge.

The high demands of flow measurement as a fundamental ingredient in most network management tasks has attracted the attention of router vendors, including Cisco and Juniper, and motivated them to solve a basic flow measurement problem. The NetFlow [Benoit, 2004] in routers was designed under this situation. NetFlow records and maintains the statistics features of traffic flow, including packet and byte counters and information about TCP flags (e.g. Urgent, Ack, Syn, Fin etc), timestamps of the first and last packets among other information [Benoit, 2004]. It observes each packet which enters or exits a router interface and checks to see if there is already a flow record for that packet. In the case where the flow record has been previously observed, the information from incoming packet is fused into the existing flow record. Otherwise, it creates a new flow record according to this new packet. The NetFlow also helps to sample the coming packets according to a configurable sampling ratio. Several variation methods have been proposed on the basis of this idea. While Flow sampling uses hash-based flow selection for flow sampling, rather than the traditional random packet sampling, FlowSlices [Kompella and Estan, 2005] combines both hash-based flow selection and the random sampling method together. In particular, it uses different types of sampling for the resources of the routers, including memory, CPU. For instance, it uses packet sampling for regulating the CPU usage, and flow sampling to regulate the memory usage.

Honh and Veicht [Hohn and Veitch, 2003] used the results of sampling from a theoretical perspective for recovering traffic statistics. They applied their proposed approach, namely Inverted Sampling, to both packet and flow filtering. Three statistical information recovery layers were defined in their approach, including the packet layer which observes the spectral
density for packet arrival, the flow layer which deals with the distribution for flow packet, and finally an internal flows layer which investigates the average rate of the packet arrival per flow packet. Extensive experimental analysis shows that the proposed Inverted Sampling approach could even retrieve exact characteristics and attributes, such as its spectrum or the distribution of flow size, from the raw traffic data.

Kumar and Xu [Kumar and Xu, 2006], proposed a sketch-guided sampling method relying on a probability function to estimate the flow size. Their method is intended to decrease the sampling rate for large flows, while increasing the sampling rate for smaller flows. Consequently, using this method for sampling, the accuracy of small flows can improve significantly; while the accuracy of large flows can marginally deteriorate.

For the application of traffic flow size estimation, Ramachandran et al. proposed the FlexSample [Ramachandran et al., 2008] to explicitly improve the flow coverage by advocating the use of an online sketch to obtain flow size estimates. Based on the idea of the FlexSample both the volume of the flow and the flow coverage can be accurately estimated.

Saxena and Kompella [Saxena and Kompella, 2010] improves the flexibility of flow monitoring by introducing a novel class-based sampling framework, namely CLAMP. This CLAMP framework increases the fidelity of flow measurements for a certain class of flows based on the interest of the network operators. In particular, the core idea of CLAMP is to encapsulate various class definitions using the composite Bloom filter (CBF) to work together, and also maximizing the objectives of flow coverage and the accuracy of certain class of flows during the implementation of the CLAMP framework.

He and Hou [He and Hou, 2006] studied three self-similarity sampling techniques for Internet traffic: including static systematic sampling, stratified random sampling and simple random sampling. Three of the most important parameters for a self-similarity process have been taken also into account and investigated in their studies. These parameters include the mean (first order statistics), the Hurst parameter (second order statistics), and the average variance of the sampling results. Their work has made several major observations: firstly, they showed that all three sampling techniques fail to identify the mean (first order statistics) precisely, this is due to the nature characteristics of the Internet traffic, secondly, they
also demonstrate that the Hurst parameter (second order statistics) of Internet traffic can be captured accurately on the three sampling techniques, thirdly, they showed that static systematic sampling can cope with the smallest variation of sampling results across different sampling techniques; fourthly, an important observation of a self-similarity process showed that the sampled mean is usually far less than the real mean because a sufficiently high sampling rate requires large values of samples to be available in advance which is less likely. To address this limitation, they proposed a biased systematic sampling (BSS) method to provide much more accurate mean estimations, and keep the overhead of sampling low.

Another important aspect of the sampling method is the re-sampling of the multi-class data for traffic flow classification. The re-sample of the multi-class is important to address the issue of imbalance of the Internet flows which occurs when some applications, known as 'majority classes', generate much more traffic flows than other applications, known as 'minority classes'. For this imbalance scenario, the majority classes always have much better classification performance than the minority classes. However, the minority classes are also important to network traffic management. Similar to the imbalance learning problem in classic machine learning [He and Garcia, 2009], many solutions have been proposed for network traffic flow classification. Zhen and Qiong [Zhen and Qiong, 2012b] investigated the use of flow-rate-based cost matrix (FCM) which is cost-sensitive learning to improve the minority classes with a few bytes in Internet traffic classification, and then they proposed a new cost matrix known as weighted cost matrix (WCM) to calculate the optimal weight for each cost of FCM. they also consider the data imbalance degree of each class to further boost the performance of network classifier on the issue of minority classes.

Yang and Michailidis [Yang and Michailidis, 2007] examined the problem of non-parametric estimation of network flow characteristics, namely packet lengths and byte sizes, based on sampled flow data. Two approaches were proposed: the first one is based on a single-stage Bernoulli sampling of packets and their corresponding byte sizes. Subsequently, the flow length distribution is estimated by an adaptive expectation-maximization (EM) algorithm that in addition provides an estimate for the number of active flows. The flow sizes (in bytes) are estimated by using a random effects regression model that utilizes the flow length
information previously obtained. The second one combines a two-stage sampling procedure in which in the first stage samples flows amongst the active ones, while the second stage samples packets from the sampled flows. Subsequently, the flow length distribution is estimated using another EM algorithm and the flow byte sizes based on a regression model. In [Fernandes et al., 2008], Fernandes et al. explored the use of a stratified sampling method to select adequate samples. After the evaluation with two partitioning clustering methods, namely clustering large applications (CLARA) and K-means, the superiority of the stratified sampling method on both size and flow duration estimate is validated.

More recently, Zander et al. [Zander et al., 2012] proposed the method of subflow packet sampling (SPS) to reduce ML sub-flow classifier’s resource requirements with minimal compromise of accuracy. Two different classifiers, C4.5 decision trees and Naive Bayes, were used to evaluate the classification performance.

2.4.5 Ground Truth

The evaluation of the learning approaches for network traffic flow classification requires the availability of accurate ground truth. This is necessary to compare the results of such learning approaches with the right answer. However, it is impossible to obtain publicly-available ground truth traffic packets included with known payload data; this is due to the privacy matter. Meanwhile, it is also usually inefficient and hard to generate ground truth data by manually triggering applications on different machines and labelling the corresponding generated flows.

In [Gringoli et al., 2009], Gringoli et al. presented a distributed system, named GT, for capturing Internet traffic in a computer network. The authors developed a special software agent, named ‘client daemon’ which is deployed on each monitored machine in the network. The agent retrieves from the kernel the name of the application that generated each flow and sends this information to a remote back-end. The agent periodically scans a list of opened network sockets and searches the names of applications that own them. For each socket, the information is stored, including current time-stamp, local and remote IP address and port number, transport protocol, and application name. At the same time, a packet capture
engine runs on the gateway router so that all the traffic coming from and into the local network is captured, while a database server collects the labels assigned by client daemons. Finally, a post-processing tool, called ipclass, is run to process the gathered information and packet. The tool connects the agent gt which collects socket information with the packet capture engine on routers. The ground truth is produced by labelling each flow with an application and pinpointing the flow’s characteristics. The authors validated their method on a 218GB dataset, and showed more than 99% of bytes and 95% of flows.

Dusi et al. in [Dusi et al., 2011] qualified the error that the classical approaches, such as port-based and DPI-based, make when establishing ground truth related to application-layer protocols. They also compared their developed “gt” tool [Gringoli et al., 2009] to these traditional approaches. The data they analyzed demonstrated that port numbers can still be a good source of ground truth information for web and email traffic, specifically in non-firewalled networks. Their analysis from experiments also showed that there is poor accuracy of ground truth with transport ports for P2P, Streaming or Skype traffic, as well as with DPI in which no more than 14% of bytes from P2P traffic, but almost 100% of Skype on TCP and Streaming can be achieved.

Gargiulo et al. in [Gargiulo et al., 2012] developed a self-training system to build a dataset of labelled network traffic based on raw tcpdump traces without prior knowledge of data. Each packet is labelled either as normal or as belonging to an attack pattern, based on DempsterShafer theory. Results for both emulated and real traffic traces have shown that intrusion detection systems trained on such a dataset perform as well as the same systems trained on correctly hand-labelled data.

A novel architecture, Flowsing, has been proposed by Lu et al. in [Lu et al., 2011]. This architecture concentrates on generating the correct ground truth, such as payload data, automatically. As a multi-agent based offline ground truth generator, Flowsing’s main aim is to generate correct full-scale ground. Ground truth database that the Flowsing generated includes the traffic traces and the flow statistical information that corresponds with them. A network traffic collection, traffic split, and traffic aggregation are three models that are a part of the Flowsing. The network traffic packets together with their process information are
collected in the traffic collection part. These packets are then separated and categorized into pure flows offline, which are then assembled in the traffic aggregation part. Here, statistical information is calculated as well. After all three steps have been completed, full-scale ground truth is produced.

2.5 Conclusion

The accurate network traffic classification has been the basis of many network management activities. Many of these activities involve flow prioritization, diagnostic monitoring as well as traffic policing and shaping. The main goal of network traffic classification is to find the network traffic mixture. Even though a number of classification techniques have been proposed recently, none of them have been validated entirely, as most validations have been poor and ad hoc. There are many reasons for such poor validations, including that unavailability of dependable validation techniques. Furthermore, there are no reference packet traces that use well-defined and clear content. All the techniques used for the network classification have shown one consistency and that is that they cannot be used for a broad range of application traffic on the Internet.
Optimizing Feature Selection for Improving Transport Layer Statistics Quality

There is significant interest in the network management and industrial security community about the need to improve the quality of Transport Layer Statistics (TLS) and to identify the “best” and most relevant features. The ability to eliminate redundant and irrelevant features is important in order to improve the classification accuracy and to reduce the computational complexity related to the construction of the classifier. In practice, several feature selection (FS) techniques can be used as a preprocessing step to eliminate redundant and irrelevant features and as a knowledge discovery tool to reveal the “best” features in many soft computing applications. In this chapter, we investigate the advantages and disadvantages of such FS techniques with new proposed metrics (namely goodness, stability and similarity). We continue our efforts toward developing an integrated FS technique that is built on the key strengths of existing FS techniques. A novel way is proposed to identify efficiently and accurately the “best” features by first combining the results of some well-known FS techniques to find consistent features, and then use the proposed concept of support to select the smallest set of features and cover data optimality. The empirical study over ten high-dimensional network traffic datasets demonstrates significant gain in accuracy and improved run-time performance of a classifier compared to individual results of well-known FS techniques.
3.1 Introduction

Network traffic classification has attracted a lot of interest in various areas, including Supervisory Control and Data Acquisition (SCADA) (industrial network) security monitoring, Internet user accounting, Quality of Service, and user behaviour. Classification-based techniques [Auld et al., 2007; Moore and Zuev, 2005] rely on a set of “good” features (that can provide a better class separability) in order to develop accurate and realistic traffic models. The identification of good features for classification is a challenging task because: (i) this requires expert knowledge of the domain to understand which features are important; (ii) datasets may contain redundant and irrelevant features which greatly reduces the accuracy of the classification process; and (iii) the efficiency of the classifiers (e.g., based on machine learning techniques) is reduced when analysing a large number of features. Indeed, a number of studies (e.g. [Blum and Langley, 1997; Kohavi and John, 1997]) have shown that irrelevant/redundant features can degrade the predictive accuracy and intelligibility of the classification model, maximise training and testing processing time of the classification model, and increase storage requirements. This chapter addresses these issues and proposes a new technique that identifies a small set of “good” features that can increase the accuracy and efficiency of network traffic classification.

Previous classification approaches that used the basic information from IP headers and payload (such as the packet content) for classification did not work well. IP headers contained a few features (such as IP addresses, port numbers, protocols) cannot accurately distinguish between applications. Payload-based techniques relied on deep inspection of packet content which resulted in significant processing and memory constraints on the bandwidth management tool. Recent approaches [Auld et al., 2007; Erman et al., 2007b; Kim et al., 2008; Lee et al., 2011] have addressed the above limitations by i) avoiding deep packet inspection by creating additional new features from Transport Layer Statistics (TLS), (e.g., statistical information in features of the traffic such as packet length and packet arrival time (see Section 3.4.1), and ii) applying machine learning techniques to learn from the data. Even though these approaches provide a promising alternative, they suffer from the presence of a large number of irrelevant/redundant TLS-based features. To improve such approaches, we
need to properly eliminate redundant features and identify the most relevant features (which we refer to as best features).

Existing machine learning-based approaches [Moore and Zuev, 2005; Auld et al., 2007; Erman et al., 2007b; Lee et al., 2011; Zhang et al., 2013a] focus on achievable classification accuracy through the use of various ML techniques such as classification and clustering; however, they suffer from irrelevant and redundant features. On the other hand, feature selection techniques [Blum and Langley, 1997; Kohavi and John, 1997] can be used for identifying the best features by eliminating irrelevant features. Feature selection techniques can be divided into two main categories: the wrapper method and the filter method [Blum and Langley, 1997; Kohavi and John, 1997]. The former method [Kohavi and John, 1997] employs an existing ML technique (e.g Support Vector Machine (SVM) [Yuan et al., 2010] and Bayesian Neural Network (BNN) [Auld et al., 2007] etc) as a classifier and uses the classifier’s accuracy as the evaluation measure to select the best possible features. Such a method tends to be not only computationally expensive but also inherits bias toward the predetermined learning algorithm. The latter method [Blum and Langley, 1997] relies on the natural characteristics of the data (e.g. correlation) and does not require a predetermined mining algorithm to select feature subsets. As a result, this method does not inherit the bias [Liu and Yu, 2005] of any mining algorithm and it is also computationally effective. However, different filter-based methods use different evaluation criteria (e.g. information-based measure, dependence-based measure, consistency-based measure and distance-based measure). Therefore, one of the key challenges (in selecting a filter method) is to define appropriate metrics that can be used to properly compare existing FS techniques to classify traffic. This chapter proposes new metrics, called goodness (to measure the quality of the generated feature set by each FS technique), stability (to measure the sensitivity of a FS technique under variations to the training traffic data) and similarity (to measure the diversity and disagreement between FS techniques). These three metrics enable us to compare and understand the inner instruments of each FS technique and the common differences between them. As shown in the proposed experiments (in Section 3.4), each FS technique has its own advantages and no single technique performs equally well on all three metrics.
The other key challenge (for traffic classification) is to preserve the maximum number of relevant features for traffic classification. It is found that classification accuracy is related to the number of relevant features used in the classification process. However, different FS techniques choose different sets of relevant features. Even worse, they do not always choose the same number of relevant features. This is problematic for the following reasons: (i) different feature selection techniques may yield feature subsets that can be considered local optima in the space of feature subsets; (ii) the representative power of particular feature selection techniques may constrain its search space such that the optimal subset cannot be reached; and (iii) a “combined” approach can give a better approximation to the optimal subset or ranking of features (which is often not applicable with a single FS technique). In addition to the new metrics, the second contribution of this chapter is an algorithm that combines the benefits of several well-known feature selection techniques, which is inspired by similar work in sensor fusion [Verma and Rahman, 2012], classifier combination [Kittler et al., 1998; Kodovsky et al., 2012], and clustering ensemble algorithms [Yang and Chen, 2011; Zhuang et al., 2012].

To the best of our knowledge, this is the first ensemble-based technique to be used for feature selection. In this approach, a feature selection technique is considered as a domain expert. Features that were supported by many domain experts are considered important in contributing to high classification accuracy and more efficient computation. Our proposed approach presents an efficient way of selecting the “best” features for network traffic classification by introducing the concept of support as an optimality criterion to keep the size of the feature set small. Figure 3.1 provides an overview of the implementation of the FS techniques and ML algorithms for traffic classification in practice.

The proposed metrics and approach are evaluated using four publicly-available benchmark traffic datasets. Our extensive experiments show that the proposed approach indeed provides a robust and meaningful way of identifying “best” features for traffic classification by exploiting the advantages of each FS technique (see Section 3.6.3 for details).

[1] The limitations of data repository in terms of space required and recycling of storage are out of scope for this project.
This chapter is organised as follows. Section 3.2 describes the steps of the general feature selection process. Section 3.3 introduces the concept of the three new metrics. Section 3.4 describes the experimental methodology, including benchmark traffic datasets. Section 3.5 presents our initial investigation based on the proposed metrics, followed by a discussion of the experimental results. Section 3.6 presents our proposed Local Optimization Approach. Section 4.6 presents the conclusion and discussions on future research directions.
3.2 The Feature Selection (FS) Techniques Used for Benchmarking

There is a wide variety of FS techniques in the literature. However, with high-dimensional network traffic data, neither the wrapper method nor complex search algorithms are applicable. In this chapter, we resorted to the use of filter methods [Cover et al., 1991; Han and Kamber, 2006; Jolliffe, 1986; Liu and Setiono, 1995; Yu and Liu, 2003; Dash and Liu, 2003], since they do not rely on the use of any data mining algorithm. They are simpler to implement and have fewer parameters to be tuned. However, filter methods are designed with different evaluation criteria and a given method is often tailored to a specific domain, which therefore may not work well on other domains. To identify the best features for network traffic, we have analysed some well-known FS techniques, each being the best one for a specific criterion. In the end, we selected six (6) FS techniques, which cover the following evaluation criteria: information, dependence, consistency, distance, and transformation. These are: Information Gain (IG) [Cover et al., 1991] and Gain Ratio (GR) [Han and Kamber, 2006] (for information-based criteria), Principal Component Analysis (PCA) [Jolliffe, 1986] (for transformation-based criteria), Correlation-based Feature Selection (CBF) [Yu and Liu, 2003] (for dependence-based criteria), Chi-square, [Su and Hsu, 2005] (for statistical criteria), and Consistency-based Search (CBC) [Dash and Liu, 2003] (for consistency-based criteria).

Before describing the specifics of each of the six FS techniques, let us first look at their common characteristics. All these techniques (as shown in Figure 3.2) share a similar process for selecting a best subset [Dash and Liu, 1997], in which the selection process of the best subset (by each FS technique) has four steps which include: subset generation, subset evaluation, stopping criterion, and final subset validation. Consequently, a feature is selected if additional information is obtained when it is added to the previously selected feature set, and discarded in the opposite case since the information obtained is already contained (redundant) in the previous set.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Figure 3.2: Feature selection process [Liu and Yu, 2005].

Here are the specifics of the six selected FS techniques.

- **Information Gain** [Han and Kamber, 2006]: This is one of the approaches used for decision tree construction of the ID3\(^2\) (Iterative Dichotomiser 3 algorithm) [Quinlan, 1993]. It measures the number of bits of information provided in class prediction by knowing the value of features [Wang et al., 2005]. The feature with the highest value of information gain is considered as the *splitting point*, while a feature with the minimum value reflects the *impurity* in data partitions. Information gain is defined as the difference between the original information (which is based on the proportion of the class) and the new information (which is obtained after partitioning). A variety of FS techniques based on information criterion have been proposed including Mutual Information and Term Strength. However, Yang [Yang and Pedersen, 1997] reported that Information Gain performed much better on their multi-class benchmarks due to its ability to aggressively reduce non-informative features. Therefore, Information Gain has been chosen in our approach as a generalised form for the information-based criterion.

- **Gain Ratio** [Han and Kamber, 2006]: This approach incorporates “split information” of the feature into the information gain measure. Gain ratio attempts to overcome the bias of information gain toward the feature with a large number of distinct values by applying normalisation to information gain using a split information measure (which

\(^2\)It uses the information theory to determine the most informative attributes and to have tree with minimal branching.
represents the potential information generated by splitting the training dataset into partitions). The features are ranked based on the value of gain ratio. Therefore, the ratio becomes unstable if the value of the splitting point reaches zero. In general, gain ratio is an information theoretic measure that selects features with an average-or-better gain and its advantage over Information Gain is that it does not consider features with a large number of distinct values.

- **Principal Component Analysis (PCA) [Jolliffe, 1986]**: This approach searches for $K$ n-based vectors used to represent the original traffic data. Such data is projected onto a much smaller space. PCA combines the essence of attributes by creating a small set of variables. The input data are a linear combination of principal components, and explain the entire changes with several components. The purpose is to provide an effective explanation through dimension reduction using linear equations. Although the $p$ components are required to reproduce the total system variability, often much of this variability can be accounted for by a small number, say $k$, of the principal components. If so, there is almost as much information in the $k$ components as there is in the original $p$ variables. The $k$ principal components can then replace the initial $p$ variables, and the original dataset, consisting of $n$ measurements on $p$ variables, is reduced to one consisting of $n$ measurements on $k$ principal components. PCA and Linear Discriminant Analysis (LDA) approaches transform the data in the high-dimensional space to a space of fewer dimensions, and they are considered to be the only two feature selection techniques available for this criterion [Belhumeur et al., 1997]. However, Yan et al [Yan et al., 2007] report that LDA suffers from two intrinsic problems: (i) singularity of within-class scatter matrices and (ii) limited available projection directions. Therefore, we have chosen PCA (in our approach to represent the transformation-based criterion) since it outperforms LDA [Zhang et al., 2010].

- **Correlation-based Feature Selection (CBF) [Yu and Liu, 2003]**: CBF is a widely-used filtering algorithm. For a given traffic dataset, the algorithm tries to find an optimal subset which is best related to the predicted class and does not contain
any redundant features. Two aspects are noteworthy: feature class correlation and feature-feature correlation. The former indicates how much a feature is correlated to a specific class while the latter represents the correlation between two features. Fayyad and Irani [Fayyad and Irani, 1993] used an information theory method to discretise numeric features and then used *symmetrical uncertainty* to measure feature-feature correlation where there is no notion of one feature being a class [Yu and Liu, 2003]. The advantage of such a method is that it is fast, and can identify relevant features as well as redundancy among relevant features without pairwise correlation analysis [Yu and Liu, 2003].

- **Chi-square [Su and Hsu, 2005]**: This approach uses a discretisation technique based on a statistical measure and evaluates flows individually with respect to the classes. It measures the association between the class and input feature $F$. The range of continuous valued features needs to be discretised into intervals. A numeric feature is initially stored by placing each observed value into its own interval. The next step, Chi-square $X^2$ measurement determines whether the relative frequencies of the classes in adjacent intervals are similar enough to justify merging. The merging process is controlled by a predetermined threshold, which is determined by attempting to maintain the validity of the original data. [Yang and Pedersen, 1997] reported that Chi-square performs well due to its ability to potentially perform as a feature selection and discretise numeric and ordinal features at the same time. In other words, it works as a combined discretisation and feature selection technique [Su and Hsu, 2005].

- **Consistency-based Search (CBC) [Dash and Liu, 2003]**: This technique uses a consistency measure that does not attempt to maximise the class separability but tries to retain the discriminating power of data defined by original features. In other words, using this measure, feature selection is formalised as finding the smallest set of features that can identify flows of a class as consistently as the complete feature set. Therefore, the consistency measure is capable of handling irrelevant features in the original space reflected as a percentage of inconsistencies. For instance, if two instances of the pattern
represent different classes, then that pattern is considered to be inconsistent. The consistency measure can help remove both redundant and irrelevant features. This type of evaluation measure is characteristically different from other measures because of its heavy reliance on the training dataset and use of Min-Features bias in selecting a subset of features [Dash and Liu, 2003].

The aforementioned FS techniques involve relevant feature adding and redundant feature removal to identify best subset. Here we first provide some basic definitions of relevant and redundant, and then we introduce these mechanisms.

**Definition 1 (Irrelevant Feature)** A feature is said to be irrelevant if it carries no information about the different classes of interest. Such features have no discriminative power.

**Definition 2 (Redundant Feature)** A feature is said to be redundant if it has a high correlation with another feature. This feature can either decrease the accuracy or increase over-fitting.

Let us introduce the mechanism of relevant feature adding and redundant feature removal:

**Definition 3 (Adding)** For a given feature set \( S_i \), let \( f^+ \) be the feature such that

\[
  f^+ = \arg \max M_i(S_i)
\]

where \( M_i \) denotes the criterion used by the FS techniques to generate the best feature subset. \( ADD(S_i) \) is the operation that adds a feature \( f^+ \) to the current set \( S_i \) to obtain the set \( S_{i+1} \) if

\[
  ADD(S_i) = S_i \cup f^+ = \{S_{i+1}, S_i, S_{i+1}\}
\]

**Definition 4 (Removing)** For a given feature set \( S_i \), let \( f^- \) be the feature such that

\[
  f^- = \arg \max M_i(S_i)^4,
\]

\[^3\text{arg max stands for the argument of the maximum.}\]
\[^4\text{The notation of backslash stands for removing a feature.}\]
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

where $M_i$ denotes the criterion used by the FS techniques to generate the best feature subset. Thus, $REM(S_i)$ is the operation of removing a feature $f^-$ to the current set $S_i$ to obtain set $S_{i-1}$ if

$$REM(S_i) \equiv S_i \setminus \{f^+\} = S_{i-1}, \quad S_i, S_{i-1} \subset X$$

(3.4)

3.3 Proposed New Metrics

Section 3.2 described six well-known FS approaches covering various criteria. However, one of the major problem is the lack of metrics to properly compare such techniques in order to reveal the best features in network traffic. Here, we propose three new metrics to address such a problem:

- **Goodness** refers to how well a generated subset can accurately classify the traffic flows.
- **Stability** refers to the property of selecting the same set of features irrespective of variations in the traffic data that have been collected over a period of time.
- **Similarity** compares the behaviour of multiple FS techniques on the same data, and also evaluates how different criteria differ in generating an optimal set for a given dataset.

3.3.1 Evaluating Goodness

The aim here is to evaluate the accuracy of the final output of the selected FS techniques (described in Section 3.2). In practice, a straightforward way is to measure the results directly using prior knowledge about the data. In network traffic data, however, we often do not have such prior knowledge. Therefore, we need to rely on indirect methods [Dash and Liu, 1997] (e.g error rate), such as the one that monitors the change in classification performance caused by the change of features. On extremely imbalanced dataset, the error rates cannot provide the information on minority class (e.g. Attack), thus the Goodness Rate (GR) is used as a performance metric. For a selected feature subset, we simply conduct the before-and-after experiment to compare the Goodness Rate (GR) of the classifier learned on the full set of features and that learned on the final selected subsets.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

The goal is to explain how to evaluate the goodness of the final set, since varying an independent measure (denoted as \( M_i \)) will produce different sets of features. The following steps are used to validate (the goodness of) the output set generated by an independent measure:

- Apply a Naive Bayes classifier to the data with only the optimal subset produced by independent measure \( M_i \). Naive Bayes is chosen because it does not require feature weighting [Fung et al., 2011]; therefore, its performance depends solely on the number of features selected. Moreover, Naive Bayes has been shown to work better than more complex methods [Witten and Frank, 2005]. Hence, we emphasise the advantages of using the simplest of the computational methods to ensure the process is tractable in time.

- Validate the goodness of the results using the fitness function, which is defined as follows:

\[
Goodness(S_i) = \frac{1}{Y} \sum_{i=1}^{Y} \frac{N_{tp}^i}{N_i}
\]  

(3.5)

where \( Y \) is the number of classes in the dataset, \( N_{tp}^i \) denotes the number of true positive of each class, and \( N_i \) is the total number of instances for class \( i \).

3.3.2 Evaluating Stability

The aim here is to measure the stability of the selected FS techniques, motivated by the need to provide network experts with quantified evidence that the selected features are relatively robust to variations in the traffic data. In practice, network operators tend to have less confidence in FS techniques that produce a different set of features on datasets taken over a period of time. Therefore, a candidate set of features that not only yields high prediction but also has a relatively stability is preferable [Somol and Novovicova, 2010]. Let \( S = \{S_1, S_2, \ldots, S_{|D|}\} \) be a collection of feature subsets obtained by running a single FS technique \( t \in T \), each time with the same configuration, on different traffic datasets (say \( D \), where \( |D| \) is the total number of datasets used). Let \( X \) be a subset representing all the features that occur anywhere in \( S \):

62
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER
STATISTICS QUALITY

\[ X = \{ f_i | f_i \in S, F_{f_i} > 0 \} = \bigcup_{i=1}^{\lvert D \rvert} S_i, X \neq 0 \]  \hspace{1cm} (3.6)

where \( F_{f_i} \) is the frequency of the feature \( f_i \). In a situation when the confidence of a feature needs to be measured, then the following formula is used:

\[ stab(f_i) = \frac{F_{f_i} - 1}{|D| - 1} \]  \hspace{1cm} (3.7)

where \( F_{f_i} \) is the frequency of feature \( f_i \in X \) in the collection \( S \), and \( |D| \) denotes the total number of generated subsets. Thus, all confidence values are normalised between \([0,1]\). The measure of stability of the feature \( f_i \in X \) in collection \( S \) takes the following properties:

- \( stab(f_i)=0 \): \( f_i \) does not appear anywhere in the observed subsets.
- \( stab(f_i)=1 \): \( f_i \) appears in each subsets of the system.

To evaluate the average confidence of all features in the collection \( S \), we need to extend Equation 3.7. Let \( N \) be the total number of frequencies of any feature \( f_i \) that appears in collection \( S \). \( N \) will then be

\[ N = \sum_{i \in X} F_i = \sum_{i=1}^{\lvert D \rvert} |S_i|, \{ N \in IN, N \geq n \} \]  \hspace{1cm} (3.8)

Therefore, the stability over all features \( f_i \in X \) in collection \( S \) is defined as

\[ stab(S) = \sum_{f_i \in X} \frac{F_{f_i}}{N} \times \frac{F_{f_i} - 1}{|D| - 1} \]  \hspace{1cm} (3.9)

\( \frac{F_{f_i}}{N} \) represents the relative frequency of the features \( f_i \in X \) in a subset. If \( stab(S) \) value is close to 1, this indicates that all subsets are identical, in particular, only if \( N = |D| \times |X| \). In contrast, suppose \( stab(S) \) value is close to 0 (if \( N = |X| \)), then this implies a low level of stability in overall subsets.
3.3.3 Evaluating Similarity

The stability measure can only evaluate the stability of an FS technique on different traffic datasets [Somol and Novovicova, 2010]. However, it is important to compare the behaviour of different FS techniques on the same traffic dataset, or evaluate how, for a given dataset, the candidate FS techniques differ in their preference for particular features. Therefore, we propose a similarity measure to allow a comparison of multiple FS techniques results. This will, in turn, enable the evaluation of how an optimal subset generated (using one criterion) can differ according to another criterion.

Given $F_{f_i}$ is the number of frequency of features $f_i$ in a collection $S$, the computation of the desirable properties of the proposed similarity measure is done as follows. If the value of $\text{Sim}(|T|)$ is close to 1, this will indicate high similarity; and any value close to 0 will indicate low similarity. Similarity is defined as follows:

$$\text{Sim}(T) = \sum_{f_i \in X} \frac{F_{f_i}}{N} \times \frac{F_{f_i} - 1}{|T| - 1}$$  \hspace{1cm} (3.10)

where $|T|$ denotes the total number of feature selection techniques that have been applied on a single dataset.

Let $|D|$ be the number of used traffic datasets. The similarity between two candidate FS techniques, say $t_1$ and $t_2$, (across different datasets), is defined as

$$\text{Sim}(t_1, t_2) = 1 - \frac{1}{2} \sum \left| \frac{F_{f_i}^{t_1}}{N_{t_1}} - \frac{F_{f_i}^{t_2}}{N_{t_2}} \right|$$  \hspace{1cm} (3.11)

where $F_{f_i}^{t_1}$ denotes the number of occurrences (frequencies) of feature $f_i$ in $t_1$ and $F_{f_i}^{t_2}$ is the frequency of the same feature in $t_2$.

Both $\text{Sim}(T)$ and $\text{Sim}(t_1, t_2)$ take value from [0,1], with 0 indicating that there is no similarity between the candidates FS techniques’ outputs, and 1 indicating that such techniques are generating identical subsets.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

3.4 Experimental Methodology

The main focus of this section is to demonstrate the benefits of the proposed metrics as an evaluation framework (to find an appropriate FS technique that improves the performance of the classification process). In what follows, we describe the network traffic trace data collected over different periods of time. We also show the performance results of the different FS techniques for the network traffic classification.

3.4.1 Datasets

We compare the candidate FS techniques on labelled Internet traffic data [Moore et al., 2005]. The TCP traffic flows in such data have been manually classified and collected by monitoring a high-performance network. We limit ourselves to the available traffic data. This data consists of ten datasets of flows taken from two days of network activity. Each dataset consists of flows (objects), and each flow is described by a set of features and its membership class. Each set covers randomly the same length of time throughout the 24-hour period.

Data Collection

Publicly-available labelled traffic datasets are very rare due to security and privacy concerns [Mahoney and Chan, 2003]. The traffic datasets collected by the high-performance network monitor (described in [Moore et al., 2003]) are one of the largest publicly-available network traffic traces that have been used in our experiment. These datasets are based on traces captured using its loss-limited, full-payload capture to disk where timestamps with resolution of better than 35 nanoseconds are provided. The data was taken for several different periods of time from one site on the Internet. This site is a research facility which hosts up to 1,000 users connected to the Internet via a full-duplex Gigabit Ethernet link. Full-duplex traffic on this connection was monitored for each traffic set. The site hosts several biology-related facilities, collectively referred to as the Genome Campus (Cambridge Lab).
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Traffic Categories

Classes of traffic are common groups of applications. Some approaches have simpler definitions of classes (e.g., Normal versus Attack), but others have more complex definitions (e.g., the classification of specific applications) [Silveira et al., 2010]. We have used the class descriptions provided in [Moore et al., 2005], which can be used as the basis for evaluating the candidate feature selection techniques to identify important features for traffic classification. Table 3.1 shows the classes for the corresponding applications. The complete description can be found in [Moore et al., 2005].

**Table 3.1: An example of network applications**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>BULK</td>
<td>FTP</td>
</tr>
<tr>
<td>DATABASE</td>
<td>Postgres, Sqlnet Oracle, Ingres</td>
</tr>
<tr>
<td>INTERACTIVE</td>
<td>SSH, klogin, rlogin, telnet</td>
</tr>
<tr>
<td>MAIL</td>
<td>imap, pop2, SMTP</td>
</tr>
<tr>
<td>SERVICES</td>
<td>X11, DNS, ident, ldap, NTP</td>
</tr>
<tr>
<td>WWW</td>
<td>http</td>
</tr>
<tr>
<td>P2P</td>
<td>KazaA, Bittorrent, Gnutella</td>
</tr>
<tr>
<td>ATTACK</td>
<td>Internet worm and virus attacks</td>
</tr>
<tr>
<td>GAMES</td>
<td>Microsoft Direct Play</td>
</tr>
<tr>
<td>MULTIMEDIA</td>
<td>Windows Media player, Real</td>
</tr>
</tbody>
</table>

Flow Features

Each flow is characterized by a set of unique features that correspond to a specific class. Such features allow for discrimination between various traffic classes. Table 3.2 provides a few examples drawn from the 249 per-flow features that are available from the dataset. A full description of these features can be found in [Moore et al., 2005]. Our aim is to identify the best features that are independent of a particular network configuration.
Table 3.2: An example of features used as input for traffic classification [Zuev and Moore, 2005]

<table>
<thead>
<tr>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow metrics (duration, total packets)</td>
</tr>
<tr>
<td>Packet inter arrival time (mean, variance)</td>
</tr>
<tr>
<td>Size of TCP/IP control fields</td>
</tr>
<tr>
<td>Total packets (in each direction of flow)</td>
</tr>
<tr>
<td>Payload size (mean, variance, ...)</td>
</tr>
<tr>
<td>Effective bandwidth based</td>
</tr>
<tr>
<td>Fourier-transform of packet</td>
</tr>
<tr>
<td>TCP specific values derived from tcptrace (e.g., total of pushed packets)</td>
</tr>
</tbody>
</table>

Classification Flows

The application of a classification scheme requires the features of the objects to be classified. By using these features, the classifier allocates an object (flow) to a specific class. In this dataset, the object classified is a TCP/IP traffic-flow, which is represented as a flow of single or multiple packets between a given pair of hosts. The flow is defined by an n-tuple consisting of the IP addresses of the pair of hosts and the TCP port numbers used by the server and client. In this work, we are limited to the training and testing sets available (10 datasets), which consist only of TCP and semantically complete TCP connections. Semantically complete flows are flow events for which a complete connection setup and tear-down was observed.

3.4.2 Experimental Setup

To evaluate the effectiveness of the selected FS techniques (to compute the “best” features for network traffic), we have used various sizes of datasets. Table 3.3 shows information about the structure of datasets for evaluation. It is clear from the numbers that there are a different number of flows in each dataset. This is due to a higher density of traffic during each block of 28 minutes. In these sets, each record is comprises of a set of features and a class label of...
the flow. The features are either continuous or discrete. The former is quantitative and the latter is on a qualitative scale. The goodness of each FS technique is evaluated by applying

**Table 3.3:** Flow statistics (percentages of flows) according to applications

<table>
<thead>
<tr>
<th>Total Flows</th>
<th>WWW</th>
<th>MAIL</th>
<th>BULK</th>
<th>SERV</th>
<th>DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>378101</td>
<td>86.77%</td>
<td>7.56%</td>
<td>3.05%</td>
<td>0.56%</td>
<td>0.70%</td>
</tr>
</tbody>
</table>

K-fold-cross validation on each traffic dataset. In this process, the dataset is divided into $K$ subsets. Each time, one of the $K$ subsets is used for testing while the remaining $K - 1$ subsets form the training set. Performance statistics are calculated across all $K$ trails. In these experiments, the value of $K$ is set to 10, since this was suggested by Kohavi [Kohavi, 1995] as the best empirical number for accuracy estimation and model selection. Therefore, we also expect that this $K$ value will provide a good indication of how well the classifier performs and classifies unseen data based on all features (of the original datasets).

**Figure 3.3:** Final subset validation process

Figure 3.3 shows the different steps involved in evaluating the goodness of the candidate FS techniques. The first step provides the FS techniques with the required information (i.e. IP traffic generated by different applications). In practice, such techniques identify the smallest
set of features that can be used to differentiate between applications. In the second step, the filtered traffic data is used to train a supervised machine learning algorithm (e.g. Naive Bayes) and to create the classifier model. This process of using a cross-validation to generate goodness results is repeated for each dataset. The statistics of goodness are accumulated for all ten datasets.

In this chapter, these FS techniques were implemented using version 3.7.7 of the WEKA software suite (readers are referred to [Hall et al., 2009] for more details).

3.5 Preliminary Experiments

The six selected FS techniques are compared using the new metrics (see Section 3.3). For a given selected subset, an experiment is conducted using a Naive Bayes classifier to compare the goodness of the optimal subsets generated by different FS techniques.

3.5.1 The Results

Here we discuss the various experimental results.

Classification of the traffic based on all the features

The Naive Bayes classifier is used to characterize the network traffic flows using the original full dataset without applying any FS technique. From the results shown in Figure 3.4, the classifier achieved goodness on average of 67.94%, which means that on average only, 67.94% of flows have been correctly classified according to their target classes using all the features. As expected, this result is not satisfactory because of the presence of irrelevant and redundant features in the datasets. However, it can also be seen in Figure 3.4 that the model trained on some datasets (e.g. datasets #3, #4, #5 and #6) outperforms the remaining sets. This suggests that there is a low similarity between the corresponding flows of the datasets. The results also suggest good class separability, and this is why there is a significant increase in the number of correctly classified flows. Therefore, in the remainder of this chapter, we will use different FS techniques to discard the irrelevant and redundant features.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Evaluation of “Goodness”

Both Table 3.4 and Figures 3.5, 3.6 compare the classification Goodness Rate (GR) of the six FS techniques on the ten datasets. For the classification task, the applications of network traffic are expected to be correctly classified. Therefore, an FS technique with high GR is desired. Note that the sequence CBF, Chi-square, InfoGain, PCA, CBC and GainRatio roughly order the FS techniques according to increasing GR. This ordering is notable in Figures 3.5, 3.6 and Table 3.4 on the achieved criterion values. From the results, the FS techniques achieve a higher classification of goodness in comparison with the outcomes of using a full features set. Overall, the goodness rate (GR) of the classification model has been substantially improved (mainly by removing these irrelevant and redundant features from network traffic data), except for GainRatio. The average GR using GainRatio is 61.73%, which is much lower than for Naive Bayes with all features. This indicates that the optimal subset selected by GainRatio may include some features that provide poor class separability. As a result, such features would reduce the accuracy of the (Naive Bayes) classifier.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Figure 3.5: Classification of the traffic based on features of the candidate FS techniques
Table 3.4: The Goodness Rate (GR) of FS techniques on the ten datasets

<table>
<thead>
<tr>
<th>FS Techniques</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$D_7$</th>
<th>$D_8$</th>
<th>$D_9$</th>
<th>$D_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBF(%)</td>
<td>96.98</td>
<td>94.60</td>
<td>93.65</td>
<td>95.47</td>
<td>94.91</td>
<td>87.29</td>
<td>94.60</td>
<td>92.69</td>
<td>44.19</td>
<td>93.8</td>
</tr>
<tr>
<td>InfoGain (%)</td>
<td>87.78</td>
<td>88.96</td>
<td>95.95</td>
<td>83.06</td>
<td>50.36</td>
<td>86.75</td>
<td>94.99</td>
<td>89.70</td>
<td>87.71</td>
<td>48.40</td>
</tr>
<tr>
<td>Chi-square(%)</td>
<td>67.36</td>
<td>92.68</td>
<td>95.94</td>
<td>85.55</td>
<td>48.12</td>
<td>84.92</td>
<td>95.51</td>
<td>76.50</td>
<td>90.99</td>
<td>89.32</td>
</tr>
<tr>
<td>PCA(%)</td>
<td>78.89</td>
<td>65.23</td>
<td>79.57</td>
<td>82.41</td>
<td>90.76</td>
<td>90.57</td>
<td>71.38</td>
<td>81.99</td>
<td>84.35</td>
<td>86.93</td>
</tr>
<tr>
<td>CBC(%)</td>
<td>21.58</td>
<td>93.58</td>
<td>76.92</td>
<td>27.79</td>
<td>96.40</td>
<td>92.80</td>
<td>67.06</td>
<td>93.38</td>
<td>42.57</td>
<td>69.02</td>
</tr>
<tr>
<td>Original (%)</td>
<td>57.89</td>
<td>61.70</td>
<td>84.45</td>
<td>74.51</td>
<td>79.29</td>
<td>90.07</td>
<td>51.86</td>
<td>58.35</td>
<td>67.12</td>
<td>54.20</td>
</tr>
<tr>
<td>GainRatio(%)</td>
<td>9.48</td>
<td>12.28</td>
<td>96.27</td>
<td>89.72</td>
<td>88.11</td>
<td>88.64</td>
<td>94.52</td>
<td>93.91</td>
<td>29.77</td>
<td>14.63</td>
</tr>
</tbody>
</table>

Temporal Variation of FS Goodness

Figure 3.6 shows that most FS techniques (except for GainRatio) enable the classification scheme to perform better than the base case (i.e., complete feature set). It is also shown in Table 3.4 and Figure 3.6 that no FS performs well on all datasets. Figure 3.5 shows a comparison of the performance of six widely-used FS techniques on 10 different datasets. Overall, CBF has the best performance on all of the datasets, except for the dataset #9. Chi-square achieves the best performance on the datasets #2, #3 and #7, but has the worst performance on datasets #1, #5 and #8. Information gain peaked on datasets #3, #7 and #8, but has the worst performance on datasets #4, #5 and #10. PCA achieves the best performance on dataset #5 and #6, but has the worst performance on the datasets #1, #2, #3, #4 and #7. CBC achieves the best performance on the datasets #2, #5 and #6, but has the worst performance on the other datasets (i.e. #1, #4, #7, #9 and #10). Gain Ratio peaked on the datasets #3, #7 and #8, but performed significantly worse than the other techniques on datasets #1, #2, #9 and #10. We therefore conclude that we cannot rely on a single technique, and this is our main reason for developing a hybrid approach to identify a reliable (best) set of features that help classifiers to perform well on all datasets.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER
STATISTICS QUALITY

Figure 3.6: Comparison of effectiveness of existing FS techniques on three randomly chosen datasets (D1,D5,D9)

Evaluating Stability of Candidate Feature Selections

Figure 3.7 shows the stability results obtained from each FS technique. Firstly, it can be seen that the clear winner is InfoGain, as this achieves the highest stability value of 0.87 for the ten traffic datasets under consideration. Secondly, Chi-square appears to have a better stability result than GainRatio, CBF and CBC respectively with a value of 0.70. Interestingly, GainRatio has a better stability score with 0.60. Notably better (higher) values in terms of method stability can be observed amongst InfoGain, Chi-square and GainRatio, with InfoGain being the most different from the other two methods. However, these three methods yield more stable results than CBF and CBC, which can be explained by the fact that they provide feature preference in a global respective. Finally, the stability of CBF and CBC is quite similar in terms of stability evaluation, but they achieved the worst stability scores with 0.42 and 0.39 respectively. The main reason for this is that the features provided by CBF and CBC focus on the top ranked or selected subsets. Consequently, they account poorly for feature inter-dependencies.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Figure 3.7: Comparing feature selection stability on traffic data

Evaluating Similarity of Candidate Feature Selections

Table 3.5 shows the similarity results obtained by performing \( n \) runs of feature selection techniques. This information given by the proposed similarity measure reveals the behaviour of \( n \) FS techniques on the same dataset. It can be seen from the figure that there is low similarity between feature subsets (produced by InfoGain, Chi-square, CBF, CBC, GainRatio) on each traffic dataset, with similarity values between 0.24 and 0.30. As suggested in Section 3.2, each FS technique produces an optimal set considerably different from those produced by other techniques. This outcome leads us to conclude that an optimal subset selected using one criterion may not be the optimal subset when using another criterion.

Table 3.5: Comparing feature selection similarity on traffic data

<table>
<thead>
<tr>
<th>Datasets</th>
<th>( D_1 )</th>
<th>( D_2 )</th>
<th>( D_3 )</th>
<th>( D_4 )</th>
<th>( D_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity</td>
<td>0.28</td>
<td>0.27</td>
<td>0.29</td>
<td>0.26</td>
<td>0.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>( D_6 )</th>
<th>( D_7 )</th>
<th>( D_8 )</th>
<th>( D_9 )</th>
<th>( D_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity</td>
<td>0.27</td>
<td>0.28</td>
<td>0.25</td>
<td>0.24</td>
<td>0.27</td>
</tr>
</tbody>
</table>
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

3.5.2 Discussion

As can be seen from the previous section, the results are not conclusive for any single FS technique. As such, the more FS techniques available, the more challenging it is to find a suitable one which will identify the best features for network traffic data.

This section provides a simple tabular approach to categorise the different FS techniques based on the proposed metrics. This way of comparing can serve two purposes: (i) grouping FS techniques with similar characteristics as well providing a way to compare such techniques on the same framework; and (ii) providing an intermediate step toward building an integrated FS technique to choose the best set of features for network traffic data. We categorise the normalised values of the proposed evaluation metric (EM) into three categories: low, medium and high using the following criteria:

\[
\begin{align*}
0 \text{ Low} & \quad \text{if } 0 \leq EM \leq \sigma \\
1 \text{ Med} & \quad \text{if } \sigma < EM < \frac{H-(M-\sigma)}{2} \\
2 \text{ High} & \quad \text{if } \frac{H-(M-\sigma)}{2} \leq EM \leq 1
\end{align*}
\]

(3.12)

where M and H denote medium and high respectively. The value of \(\sigma\) is set according to the various experimental results (presented in Section 3.5). In particular, the value of \(\sigma\) is set to 0.60 for evaluating goodness as this is the lowest goodness rate among the candidate FS feature selection techniques. The value of \(\sigma\) is set to 0.4 to evaluate stability as this is lowest score for stability among the candidate feature selection techniques. The same applies to similarity as this value indicates most selected features were supported by less than two techniques.

Table 3.6 summarises the values for each technique with regard to goodness, stability and similarity. We use these comparisons to help illustrate the appropriateness of each FS technique using equation 3.12.

1. For the goodness metric, prevailing ordering can be recognised among FS techniques: all FS techniques have an average value, except for CBF, whose value depends on the good quality of its output compared to the other techniques.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

This suggests that CBF is recommended for cases when FS techniques fail to produce good quality output.

2. In terms of stability, all FS techniques are almost unstable, with the exception of Information Gain and Chi-square.

3. From Table 3.6, one can notice constant low similarity between values yielded by the candidate FS techniques on the same dataset. This suggests that a large number of features are consistently excluded while the rest appear in the selected subsets with low similarity. Also, it suggests that an optimal subset selected using one criterion is almost not optimal according to another criterion [Liu and Yu, 2005].

Based on the developed categorisation approach, it can be seen that in most cases, there is no visible “winner” among the FS techniques. As a result, there is no FS technique that satisfies all evaluation criteria. Hence, we cannot rely on a single feature selection technique to select the best set of features.

Table 3.6: Evaluation of FS techniques on the categorisation framework

<table>
<thead>
<tr>
<th>FS Tech</th>
<th>Goodness</th>
<th>Stability</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBF</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>InfoGain</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Chi-square</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>PCA</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>CBC</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>GainRatio</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

3.6 The Local Optimisation Approach (LOA)

The previous section showed that any single FS technique cannot perform well on all datasets, and that different FS techniques generally produce different results. Given a traffic dataset, and without any a priori knowledge, the problem still remains regarding the selection of an FS technique that will perform the best. Therefore, instead of choosing a particular
FS technique, we have looked at “combining” five FS techniques so to “aggregate” their benefits in selecting the best features. This new approach is called the *Local Optimisation Approach* (LOA). The reader may later notice that we have excluded *Principle Component Analysis* (PCA), as this technique transforms the original features to new ones to produce the best approximation of the original features. Therefore, this does not strictly fall into the category of feature selection. Figure 3.8 depicts the overall idea of LOA: given a dataset, local optimisation aims to select the most reliable subset of features based on feature subsets selected by FS techniques. As different FS techniques produce different subsets of features, we introduce the concept of *support* to indicate the importance of a specific feature. The idea behind this concept support is that the judgement of a group is superior to that of individuals. The underlying assumption is that an important feature for traffic analysis is very likely to be supported by most FS techniques.

**Figure 3.8: The LOA approach**

**Definition 5 (Support)** Let $F=\{f_i|1 \leq i \leq m\}$ be a set of features in a given dataset, and $T=\{t_j|1 \leq j \leq n\}$ be a set of existing FS techniques. We then use a matrix $A$ to record the occurrence of features for different techniques, where $\alpha_{i,j}$ are binary values indicating whether the feature $f_i$ is selected by a technique $t_j$ (1 for selected, 0 for not selected). Therefore, the support of feature $f_i \in F$ is defined as follows:

$$support(f_i) = \frac{\sum_{j=1}^{n} \alpha_{i,j}}{|T|}$$  \hspace{1cm} (3.13)

(January 9, 2015)
where $|T|$ is the number of techniques that have been applied.

The following steps are taken by LOA to identify the most reliable subset of features to be used for a particular training set.

- Apply the five FS techniques on a training dataset and keep all the selected features in an initial pool. As different FS techniques use different ways to generate feature subsets, finding salient features is often hard. Therefore, to make the best of the different techniques, LOA applies the five FS techniques on a training set to generate an initial pool of five sets of features. Features that have not been selected by any of the five FS techniques are discarded.

- Calculate the frequency of the observed features. Let $S_{\text{best}}$ be the set of selected features, where $S_{\text{best}} = \{f_1, ..., f_n\}$. Then, the frequency of $f_i$ is defined by $F(f_i) := m_i$, where $m_i$ is the number of times $F$ has the value $f_i$.

- Order the features value of $F$ based on their occurrences (frequency). Let have $\hat{f}_1, \hat{f}_2, ..., \hat{f}_n$ such that $F(\hat{f}_1) \geq F(\hat{f}_2) \geq ... F(\hat{f}_n)$.

- Using Equation 3.13, calculate the support of the features in the $S_{\text{best}}$ by counting the number of times they are selected by FS techniques divided by the cardinality of $T$, where $0 < F(f_i) \leq |T|$. This is based on our assumption that the importance of a feature is indicated by the number of occurrences in the “optimal” feature subsets generated by the different FS techniques we applied. We hypothesise that a larger count of occurrences implies more distinct and reliable features.

- Examine the degree of support in the observed features. To do so, we apply an arbitrary threshold to retain only the top $N$ features whose supports are above the threshold. The features in the $S_{\text{best}}$ have been selected by at least one of the five FS techniques; but to retrieve an optimal set of features, a threshold must be set to keep only those features that are sufficiently distinct and reliable. For instance, if a feature selected by at least three out of five FS techniques is considered reliable enough, then we apply a threshold of $\text{supp} \geq 0.60$. 

78 (January 9, 2015)
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

3.6.1 The Proposed Algorithm

Algorithm 1 summarises the various steps of LOA to compute the most informative features in a single dataset, which we denote as DATA. This algorithm is divided into two parts. In the first part (Lines 1-17), the algorithm extracts an initial pool of five sets of features by applying the five FS techniques, and returns the corresponding generated feature subsets. In particular, the search starts with an initial set of features, say $S_0$, then this set $S_0$ is evaluated by an independent measurement, say $t_i$. Evaluate each newly generated subset $S_i$ using $t_i$. Compare the current subset, $S_i$, with the previous optimal set $S_{optimal}$. As a result, if the current set is better than the previous set, then it is considered as the current optimal subset. Iterate the search until a sufficiently optimal set, $S_{optimal}$, is found based on the independent measurement $t_i$. Output $S_{optimal}$ as the optimal set. Then add the optimal set $S_{optimal}$ of technique $t_i$ to the initial pool set $S_{Sel}$. The second part of the algorithm (Lines 18-30) measures the support of each feature value in $S_{freq}$ and includes those whose support exceeds the threshold into the set of reliable features, $S_{best}$. Finally, the selected features are significant features that contain indispensable information about the original features. The algorithm would need an $O(k)$, (where $k$ is the number of FS techniques) for identifying a pool of features, followed by $O(g)$ for the operation of calculating the support of features and selecting the most supportive set. Thus, the algorithm has a total complexity of $O(k + g)$ for choosing the final set of features (see Section 3.6.6).

3.6.2 An Illustrative Example

A simple example is given below to illustrate the use of the proposed LOA approach to select the best possible features. Figure 3.9(a) represents a unity of the sets of rows which correspond to the various FS techniques $T$ (where $t_i \in T$) and the columns represent the features themselves (where $f_i \in F$). The binary values of Figure 3.9(a) indicate whether or not a feature is selected by the corresponding FS technique $t_i$, where 1 stands for selected, and 0 for not selected. For instance, FS technique $t_1$ selects features $\{f_1,f_2,f_4,f_7,f_{10}\}$. In the last row of the Figure 3.9(a), the frequency of a feature that has been selected is calculated by counting the number of times we observe $f_i$ taking the binary value (1).
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Algorithm 1: Local Optimisation Algorithm

input:
1. DATA $\leftarrow \{f_1, f_2, ..., f_{n-1}\}$;
2. Feature Selectors(T) $\leftarrow \{CBF, Chi, ..., t_n\}$;

output:
3. $S_{best}$; // a best subset of features
4. Apply FS techniques to obtain initial pool of features for $Selector_t \in [1, T]$ do
5. $S_0$ $\leftarrow$ initialize(DATA);
6. $\gamma_{optimal}$ $\leftarrow$ evaluate($S_0$, DATA, $t_i$);
7. repeat
8. // Evaluate $S_0$ by using independent FS technique
9. $S$ $\leftarrow$ generate(DATA);
10. // Generate a subset $S$ for evaluation
11. $\gamma$ $\leftarrow$ evaluate($S$, DATA, $t_i$);
12. // Evaluate the current subset $S$ by $t_i$
13. if $\gamma$ is better than $\gamma_{optimal}$ then
14. $\gamma_{optimal}$ $\leftarrow$ $\gamma$;
15. $S_{optimal}$ $\leftarrow$ $S$;
16. end
17. until (reach $\delta$); // Add the final $S_{optimal}$ of $t_i$ to the initial pool
18. $S_{Sel}$ $\leftarrow$ $S_{sel} \cup S_{optimal}$;
19. Return $S_{Sel}$; // Return the initial pool of features
20. end

18. Preserve the maximal of Relevant Feature
19. $\beta$ $\leftarrow$ 0.60;
20. $S_{freq}$ $\leftarrow$ $\emptyset$;
21. $S_{best}$ $\leftarrow$ $\emptyset$;
22. FindBestSet($\beta$, $S_{sel}$, $S_{best}$);
23. $S_{freq}$ $\leftarrow$ ComputeFreq($S_{Sel}$);
24. $S_{sort}$ $\leftarrow$ Sort($S_{freq}$);
25. for $f_i \in S_{sort}$ do
26. if $Support(f_i) \geq \beta$ then
27. $S_{best}$ $\leftarrow$ $S_{best} \cup \{f_i\}$;
28. end
29. end
30. Return $S_{best}$;
Figure 3.9(b) shows a list of the features sorted by frequency. Then the support of a feature is calculated using Equation 3.13. A predetermined threshold is applied to retrieve the best features. For example, if the predefined threshold of $\text{support}(f_i) \geq 0.60$ is applied, then the features \{f_7, f_1, f_4, f_{10}\} are selected.

<table>
<thead>
<tr>
<th>Feature#</th>
<th>Frequency</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_7</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>f_1</td>
<td>4</td>
<td>0.80</td>
</tr>
<tr>
<td>f_4</td>
<td>3</td>
<td>0.60</td>
</tr>
<tr>
<td>f_{10}</td>
<td>3</td>
<td>0.60</td>
</tr>
<tr>
<td>f_2</td>
<td>2</td>
<td>0.40</td>
</tr>
<tr>
<td>f_3</td>
<td>2</td>
<td>0.40</td>
</tr>
<tr>
<td>f_8</td>
<td>2</td>
<td>0.40</td>
</tr>
<tr>
<td>f_5</td>
<td>1</td>
<td>0.20</td>
</tr>
<tr>
<td>f_6</td>
<td>1</td>
<td>0.20</td>
</tr>
<tr>
<td>f_9</td>
<td>1</td>
<td>0.20</td>
</tr>
</tbody>
</table>

(a) Applying a set of T on a training set

(b) Frequency

**Figure 3.9:** Procedure of Local Optimization Approach (LOA)
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

3.6.3 Result and Analysis

The aim here is to evaluate the performance of the proposed LOA algorithm. We first compare the performance of LOA against the five FS techniques (see Section 3.2). Then, we evaluate the effect of various parameter settings on the performance of LOA. For each FS technique, Naive Bayes is used to evaluate the classification goodness rate on selected features, as we have no prior knowledge about the most reliable features for Internet traffic data.

Table 3.7 summarises the Goodness Rate (GR) of LOA on 10 datasets using the Naive Bayes algorithm. As can be seen from Table 3.7, LOA performs well and was stable on all datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodness Rate</td>
<td>97.51</td>
<td>95.94</td>
<td>97.89</td>
<td>96.03</td>
<td>97.48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$D_6$</th>
<th>$D_7$</th>
<th>$D_8$</th>
<th>$D_9$</th>
<th>$D_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodness Rate</td>
<td>97.09</td>
<td>96.05</td>
<td>97.32</td>
<td>90.51</td>
<td>93.84</td>
</tr>
</tbody>
</table>

From the results shown in Figure 3.10, we observe that the LOA achieves an average goodness of 95.97%. Given the average goodness shown in Figure 3.10a, it can be seen that we were able to achieve higher GR in comparison with the remaining FS techniques. The experimental results shown in Figure 3.10b clearly demonstrate the performance of LOA across all the datasets. Notice, the GR on the datasets number #9 and #10 are not as good for either LOA or any other FS technique. The reason is that the HTTP class in these two datasets includes 2600 records, which are related to a different class HTTPS. However, LOA has the best GR among all the techniques on #9 and #10 datasets.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

3.6.4 Choice of Parameters

As discussed in the previous section, the main parameter in LOA is the *support* threshold, which we refer to as $\beta$, used for selecting the best feature set. The performance of LOA critically depends on the value of $\beta$. Thus, the choice of the parameter $\beta$ not only affects the goodness rate of the selected feature sets.
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

Table 3.8: Influence of different setting of support threshold

<table>
<thead>
<tr>
<th>Supp.Threshold</th>
<th>Goodness</th>
<th>Set Size</th>
<th>Run Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GD</td>
<td>Stdev</td>
<td>RT</td>
</tr>
<tr>
<td>0.2</td>
<td>70.39</td>
<td>21.13</td>
<td>25</td>
</tr>
<tr>
<td>0.4</td>
<td>87.04</td>
<td>7.60</td>
<td>14</td>
</tr>
<tr>
<td>0.6</td>
<td>95.97</td>
<td>4.11</td>
<td>6</td>
</tr>
<tr>
<td>0.8</td>
<td>78.95</td>
<td>39.00</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>77.86</td>
<td>32.00</td>
<td>1</td>
</tr>
</tbody>
</table>

the training and the testing time of classification, but also influences the goodness of the classification model. As discussed previously in Section 3.6, the choice of \( \beta \) is a trade-off between lower GR and higher processing requirements due to the increased number of features in the selected feature set. In this section, we investigate the effects of this parameter setting on the performance of LOA.

Runtime performance

We apply the proposed LOA approach to samples of traffic selected randomly from the 10 datasets, and the final result set is fed to the machine learning classifier (Naive Bayes) to generate and test the classifier model. We then measure the execution time required by Naive Bayes for different threshold settings.

Figure 3.11 and Table 3.8 show the run-time performance of the classifier with the threshold varying between 0.2 to 1.0. For various threshold settings, the test was repeated ten times to give the average execution time and the GR. As predicted earlier, the complexity of the classifier is linear with respect to the number of input features. Furthermore, LOA shows a significant reduction in computation time for the classifier when compared to using the full set of features. Figure 3.11c shows how classification goodness evolves with the value of the threshold, along with run-time performance. It can be shown that the maximum value of GR for the parameter \( \beta \) is achieved when the support is set to 0.60. This suggests that the goodness would not increase if \( \beta \) were to be achieved. On the other hand, it can be seen that
CHAPTER 3. OPTIMIZING FEATURE SELECTION FOR IMPROVING TRANSPORT LAYER STATISTICS QUALITY

(a) Influence of LOA’s parameters on building the classifier model

(b) Influence of LOA’s parameters on testing the classifier model

(c) Influence of PLOA’s parameters on the goodness rate

Figure 3.11: Influence of parameterising LOA
the high runtime performance is achieved when $\beta$ set to 1.0 and that the higher parameter of support decreases the accuracy. Therefore, we have found that $0.8 > \beta > 0.4$ provides the appropriate and stable region for the trade-off between increasing the goodness rate and lowering the processing time.

3.6.5 Impact of FS Techniques on Runtime

One of the key motivations for using FS techniques is to reduce the amount of time required by any ML algorithm (e.g. Naive Bayes) to build the model and evaluate new incoming flows. This is particularly important because the model-building phase is computationally time-consuming. Therefore, we use the output of the LOA and the other candidate FS techniques to measure the execution time required by Naive Bayes to build the classification model on a Core dual 2.2-GHz Intel processor machine with 2 Gbytes of RAM.

Figure 3.12a shows the normalized build time for Naive Bayes when using the output of the LOA approach in comparison to the candidate FS techniques. The dataset comprises network traffic [Moore et al., 2003] from all days of week, and the number of instances in the dataset was varied between 1000 and 10000 (the sized of the dataset is ultimately limited by the amount of memory since Naive Bayes needs to load the entire training data into the memory before building the model). For each of the feature sets, the test was repeated ten times to give the average execution time and to achieve greater confidence in the results. It can be seen that LOA shows a significant reduction of times in comparison to InfoGain, GainRatio and Chi-square. Note also that there is substantially smaller variance in computational performance for Naive Bayes when using LOA in comparison to CBF and CBC. Figure 3.12b illustrates the classification speed to evaluate new flows based on selected features by LOA and the candidates FS techniques. This is particularity important when considering real-time classification of potentially thousands of simultaneous network flows. The results show that we can successfully reduce the computation time if our selected feature subset is used in comparison to InfoGain, GainRatio and Chi-square. However, it is obvious that there is a smaller variance in computational performance for Naive Bayes when using LOA in comparison to CBF and CBC.
Figure 3.12: Evaluation LOA against the selected FS techniques (a value 1 represents the lowest build and classification time)
3.6.6 Comparing FS Techniques Computational Performance

In this subsection, we compare the execution time of LOA against the candidate FS techniques to generate optimal features. For the analysis, the performance of each technique was tested with traffic samples varying from approximately 1000 to 10000 traffic records, and all operations were performed on a Toshiba Satellite with Intel Pentium Core dual 2.2 GHz processor and 2 Gbytes of RAM. Figure 3.13 shows the time needed by InfoGain, GainRatio, Chi-square and CBF techniques is quite low. This is because these techniques use a sequential search which is fast in producing results as the order of the search space is usually $O(m((n^2 - n)/2))$ or less (where $m$ is the number of instances and $n$ is the initial number of features). It is also notable that the cost of CBC is very high compared to the other FS techniques, as it requires $O(mnp^p)$ (where $p$ is the number of relevant features). On the other hand, the LOA execution time was significantly higher than other FS techniques; this is because LOA relies on all FS techniques to generate the initial feature set. A promising future research direction would be to reduce the execution time of LOA by using parallel computing such as multi-cores CPU or Graphics Processing Units (GPU).

![Figure 3.13: Comparison of runtime performance](image)
3.6.7 Summary of Results with different Datasets and Limitations of LOA Approach

In addition to the ten datasets collected by the high-performance network monitor [Moore et al., 2003] (discussed in Section 3.4.1), the capabilities of the proposed LOA have been further assessed against the baseline FS techniques with two of the recent and most widely-used datasets.

The first one is wide2009 [Doe, 2009] dataset, where its flows are categorized into 6 classes: P2P, DNS, FTP, WWW, CHAT and MAIL. The second dataset is KDD99 [MIT, 1999], which is the most widely-used dataset for the evaluation of anomaly detection methods. The KDD99 dataset consists of 60000 single connection vectors and labelled as either normal or an attack. Table 3.9 gives an overview of the datasets used along with their associated information.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Instances</th>
<th># Features</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-performance network [Moore et al., 2003]</td>
<td>377526</td>
<td>249</td>
<td>13</td>
</tr>
<tr>
<td>wide2009 [Doe, 2009]</td>
<td>20000</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>KDD99 [MIT, 1999]</td>
<td>60000</td>
<td>41</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 3.14a compares the performance of the proposed LOA approach and the baseline FS techniques by considering the three metrics goodness, stability and similarity. The values of these three metrics are computed as explained in Section 3.5.2. It can be seen from Figure 3.14 that the LOA approach has an advantage over the other related FS techniques. First, the features obtained by LOA help the Naive Bayes classifier to achieve a higher goodness rate in comparison with the remaining FS techniques on all the four datasets. Second, the LOA approach preserves the maximum number of relevant features for traffic classification by considering only highly-supported features.
### Chapter 3. Optimizing Feature Selection for Improving Transport Layer Statistics Quality

<table>
<thead>
<tr>
<th>FS Tech</th>
<th>Goodness</th>
<th>Stability</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>CBF</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>InfoGain</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Chi-square</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>CBC</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>GainRatio</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

(a) On high-performance network dataset

<table>
<thead>
<tr>
<th>FS Tech</th>
<th>Goodness</th>
<th>Stability</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>CBF</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>InfoGain</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Chi-square</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>CBC</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>GainRatio</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>

(b) On wide2009 dataset

<table>
<thead>
<tr>
<th>FS Tech</th>
<th>Goodness</th>
<th>Stability</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>CBF</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>InfoGain</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Chi-square</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>CBC</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>GainRatio</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>

(c) On DARPA (KDD99) dataset

**Figure 3.14**: Comparing the performance of FS techniques on two more traffic datasets, namely: wide2009 [Doe, 2009] and KDD99 [MIT, 1999]

In general, the experimental results shown in Figure 3.14 indicate that the three metrics are mostly satisfied by the proposed LOA approach (in comparison to the related approaches). However, it can be seen from Table 3.14c that the LOA approach still suffers from the...
stability issue on traffic data of the high-performance network monitor. This is due to the high variations in these datasets, since these datasets are collected for different periods of times and from different locations. In Chapter 4, we will work on developing a new approach to address the sensitivity of the baselines methods and the LOA approach to variations in the traffic datasets.

3.7 Conclusion

Identifying the best and most robust (in terms of similarity) features from large datasets of Internet traffic is of critical importance in light of the emergence of new and distributed applications. This chapter made three contributions with regard to the problem of computing the best features (in network traffic). We introduced three new metrics, namely goodness, similarity and stability. The primary purpose of these metrics is to gain a deeper understanding of the properties of the FS techniques as well as to compare the quality of their outputs (selected subsets). The experimental results have shown that no existing FS technique performs well on all three metrics. Motivated by this, we proposed a method that exploits the advantages of individual FS techniques to obtain an optimal feature set that is better than any individual set. We also showed how to select the best subset based on the concept of support to extract the optimal set. The proposed LOA (Local Optimisation Approach) technique was analysed in light of the optimality criteria. Results obtained on real network traffic data illustrates the ability of LOA to identify the best features for traffic classification. As expected, the joint contributions of the five well-known feature selection techniques had a compensatory effect. Experimental results also showed that LOA performs significantly better than an individual technique.

Integrated FS approaches are computationally more expensive than the single run; however, as demonstrated empirically, once computed, they provide increased robustness and performance, being able to identify the best features for traffic classification, not adequately handled by these techniques. We have identified the need for developing an adaptive threshold instead of the fixed threshold beta used in LOA.
Optimal and Stable Feature Set for Traffic Classification

Feature selection techniques can be used as a preprocessing step to eliminate meaningless features, and also as a tool to reveal the set of optimal features. Unfortunately, as demonstrated in Chapter 3, such techniques are often sensitive to a small variation in the traffic data collected over different periods of time. Thus, obtaining a stable feature set is crucial in enhancing the confidence of network operators. This chapter proposes robust approach, called the Global Optimisation Approach (GOA), to identify both optimal and stable features, relying on a multi-criterion fusion-based feature selection technique and an information-theoretic method. The proposed GOA first combines multiple well-known FS techniques to yield possible optimal feature subsets across different traffic datasets; then uses the proposed adaptive threshold, which is based on entropy to extract the stable features. A new goodness measure is proposed within a Random Forest framework to estimate the final optimum feature subset. The effectiveness of GOA is demonstrated through several experiments on network traffic data in spatial and temporal domains. Experimental results show that GOA provides up to 98.5% accuracy, exhibits up to 50% reduction in the feature set size and finally speeds up the run-time of a classifier by 50% compared with individual results produced by other well-known feature selection techniques.
4.1 Introduction

Many factors can contribute to the usefulness of machine learning (ML) algorithms for Internet traffic classification. The quality of network traffic data (e.g., TLS [Chou et al., 2008]) is one of these factors [Auld et al., 2007][Moore and Zuev, 2005]. If the data contains irrelevant or redundant features, then the knowledge discovery process during the training becomes noisy and unreliable. In practice, FS techniques play a fundamental role in the success of many classification tasks where data analysis is a challenge due to high dimensionality, e.g. text classification [Fung et al., 2011], handwritten signature classification [Kim et al., 2011b], bioinformatics [Xing et al., 2001], Intrusion Detection System (IDS) [Chou et al., 2008][Qu et al., 2005] and so on. Indeed, feature subset selection enables a classifier to selectively focus its attention on relevant features whilst ignoring the (possibly misleading) redundant features. The main advantage of an FS method is that by concentrating on predictive features only and not considering the irrelevant ones, the accuracy of the classifier may be higher and the association between features and the target class may be easier to learn. However, as demonstrated in Chapter 3, most of the FS techniques concentrate on feature relevance and neglect the stability issue. Such an issue is important in traffic analysis when high-dimensional data is used, and FS is used as a knowledge discovery tool for identifying characteristic discriminators. For example, in traffic data analysis, a given FS method may select largely different subsets of features, called discriminators, due to variations in the traffic training data. Such instability dampens the confidence of network operators in investigating any of the various subsets of selected features for network traffic identification (e.g. arbitrarily picking the same set of features under training data variation). It is important to note that the stability of feature selection results should be investigated together with classification accuracy since network operators tend to have less confidence in feature sets that change radically on datasets taken over a period of time. Moreover, unstable features in traffic application are problematic, as there is no prior knowledge about the data and therefore, in most cases, these features are subsequently analysed further, requiring much time and effort. Therefore, when using feature selection to identify the “best” discriminators for network classification, it is preferable to generate a candidate set of features that not only
yields high prediction, but also has a relative stability. However, for the purpose of network traffic classification, there has been very little attempt to identify such features.

Apart from identifying stable and optimal features for traffic classification, transport layer statistics (TLS) involves several continuous-valued features. Examples of such features include the number of packets, number of bytes, and duration for each connection. As a consequence, these features can have a negative impact on some machine learning algorithms, in terms of both accuracy and/or training time [Ferreira and Figueiredo, 2012]. Therefore, the main focus of this chapter is to address the issues of stability and the presence of continuous-valued features.

4.1.1 Contributions

This chapter deals with the issues described above, and it proposes a new FS technique as well as a discretisation algorithm to enhance the capabilities of the network classification task.

The significant contributions of this chapter are:

• A general framework that not only provides the optimal features, but also automatically discovers the stable features for network traffic. For this purpose, the proposed GOA technique proceeds in three phases. The first phase combines multiple FS techniques to yield the optimal feature subsets across different traffic datasets. In the second phase, instead of relying on a fixed threshold, GOA adapts the concept of maximum entropy\(^1\) [Csiszár, 1996] that culls stable features based on feature distribution. Intuitively, features with a distinct distribution are considered to be stable and are therefore extracted. This process automatically adapts to feature distribution (i) to yield feature subsets with a distinct distribution (with highest distribution) and (ii) to help narrow the scope for a deeper investigation into specific features set (Section 4.3.2 for details). In the third phase, the extracted features (obtained from the first and second phases) are passed to a more computationally intense procedure, called Random Forest filtering, to determine the most representative features that are strongly related to target

---

\(^1\)It is a general technique for estimating probability distributions from the data
classes (e.g. WWW, FTP, Attack). The feature subset with the highest goodness is chosen as the final optimal set for network classification (Section 4.3.3).

- Optimal discretisations produced by *entropy minimization heuristics* method [Fayyad and Irani, 1993]. The necessity of using such a method on traffic data can have many reasons. Many machine learning (ML) algorithms primarily handle nominal features [Wu et al., 2008][Liu et al., 2002][Dougherty et al., 1995], or may even deal only with discrete features. Even though ML algorithms can deal with continuous features, learning is less efficient and effective [Wu et al., 2008]. Another advantage derived from discretisation is the reduction and simplification of data which makes the learning faster and produces a more accurate, compact and smaller output. Also, the noise present in the traffic data is reduced. In particular, features discretisation involves partitioning the range of continuous-valued features into a set of mutually exclusive intervals with interval boundaries so that the loss of class/attribute interdependence is minimized (Section 4.5.2).

- The proposed approach is evaluated using publicly-available benchmark traffic datasets [Moore et al., 2005]. In particular, we compare the effectiveness and efficiency of the candidate features set against two well-known techniques, namely FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]. Also we studied the robustness of the candidate features to classify a range of applications in both the temporal domain: comparing across different period of time, and the spatial-domain: comparing across different network-locations.

The rest of this chapter is organized as follows. Section 4.2 briefly reviews some well-known FS techniques and also analyses their performance according to the new optimality vs stability metrics. Section 4.3 describes the GOA approach, and Section 4.4 shows the various performance results with the various benchmark datasets. We conclude with some remarks in Section 4.6.
4.2 Optimality vs Stability

The performance of ML algorithms degrades when there are many irrelevant and redundant features. To achieve the best possible performance with a particular ML algorithm, FS techniques should remove such irrelevant and redundant features from the data. However, we are faced with two problems: (i) each FS technique conducts a search for an optimal subset using its own independent criteria (e.g. distance, dependence); therefore, an optimal subset selected by one criterion may not be optimal according to other criteria; also (ii) to evaluate the output of a particular FS technique, we need to use prior knowledge about the data. For dynamic network traffic data, we often do not have such prior knowledge. Therefore, we need to rely on some indirect methods [Dash and Liu, 1997][Liu and Yu, 2005], such as the one that monitors the change in classification performance caused by the change of features.

To simplify further discussion, in this section we define the optimal subset selection with respect to the proposed goodness measure in Chapter 3.

**Definition 6 (Optimality)** Given a dataset, say $D$, with subset of features, say $S_D$, which is selected by a given FS technique $t_i \in T$, with a particular classification algorithm, then $S_{D}^{t_i}$ is said to be an optimal feature subset if the goodness of the generated classifier is maximal.

$$\text{Goodness}(S_{D}^{t_i}) = \arg \max \left[ \frac{1}{Y} \sum_{i=1}^{Y} \frac{N_{tp}^i}{N_i} \right] \times 100 \quad (4.1)$$

where $Y$ is the number of classes in the dataset, $N_{tp}^i$ denotes the number of true positives of each class, and $N_i$ is the total number of instances for class $i$. Note the goodness measure takes into account the bias of the majority of classes, which is important since the distribution of traffic applications is different.

The other important metric used to evaluate FS techniques is stability. This is motivated by the need to provide network experts with quantified evidences that guarantee that the selected features are relatively robust against variations in the traffic data. In a practical classification scenario, network operators tend to have less confidence in the use of FS tech-
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

niques (to produce different sets of features from samples of traffic data over a period of time). Thus, it is preferable to have a candidate set of features that not only yields high prediction accuracy but also has higher relative stability over different samples. Given an FS technique, say $t_i$, the desirable properties of stability measures of $t_i$, denoted as $Stab(t_i)$, are

- $0 \leq Stab(t_i) \leq 1$.
- $Stab(t_i)$ close to 1 indicates high stability.
- $Stab(t_i)$ close to 0 indicates low stability.

Next, we define the stability of FS techniques and outline a framework to measure the stability index based on entropy.

**Definition 7 (Stability)** Let $t_i$, where $t_i \in T$, be an FS technique applied on two samples $D_1$ and $D_2$ of traffic dataset $D$, which generates two subsets of features $S_1$ and $S_2$. Then $t_i$ is said to be stable if its stability index takes the value of one, meaning that $t_i$ selects the same set of features for both data samples irrespective of minor variations in the traffic data. Therefore, the stability index of FS technique $t_i \in T$ is defined as follows:

$$Stab(t_i) = [1 - RU(X)] \times 100$$ (4.2)

where

$$RU(X) = \frac{H(X)}{\log(|N|)}$$

where

$$H(X) = \frac{1}{N} \sum_{i=1}^{N} \frac{N_k^i}{S} \log\left(\frac{N_k^i}{S}\right)$$

where $N$ is the total number of features, $S$ is the number of runs, $N_k^i$ is the frequency of specific feature $f_i$ observed across different datasets $|D|$.

In general, the proposed stability index of the FS technique $t_i \in T$ has the following properties:

- $stab(t_i) = 0$, if $t_i$ does not select the same features in each run.
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

- \( \text{stab}(t_i) = 1 \), if \( t_i \) selects identical subset of features in each run.

4.2.1 Selecting Feature Set from Global Perspective

Previously discussed feature selection techniques (in Chapter 3) eliminate both irrelevant and redundant attributes from a local perspective, and thus they can be tricked in a situation where the dependence between a pair of features is weak, but the total inter-correlation of one features to the others is strong. Thus, in this chapter, we introduce a new procedure to select informative features from a global perspective. The process of discarding irrelevant and redundant features from a global perspective and only keeping the optimal features is presented in Table 4.1. In particular, the procedure of removing the irrelevant and redundant features is divided into two parts. In the first part, an evaluation criterion (e.g. information Gain, consistency-based etc.) is used to evaluate the relevant degree and the reliability of each individual features for predicting the accurate class label. Therefore, features whose relevant value is zero are undesirable and thus removed from the feature space, which means that features do not have the power to distinguish between different types of traffic flows and applications. The remaining features are then ranked in descending order according to their relevant degrees, and the mean of the relevant degrees for features whose relevant degree greater than zero is calculated, \( \mu_{rv} \).

In the second part, inter-correlation between previously selected features is computed, and the total values of the redundancy degree of the related to that features are added. The weight factor \( w \) is computed (as in line (5.a)) to be used for selecting informative features features from a global perspective. Finally, features greater than zero are selected, which means that they not only can accurately predict the class, but also have a low correlation to other features.

4.2.2 Initial Investigation

There is a wide variety of FS techniques in the literature, which have been categorized into groups broadly based on the following [Almuallim and Dietterich, 1994][Duda and Hart, 1996][Hall, 2000][Liu and Motoda, 1998]: information-based criterion, dependency-based cri-
Table 4.1: The Process of selecting features globally.

**Input:**
Given the input dataset $D$
Specify the number of optimal features $K$.

**Remove irrelevant features**
1. Compute the relevant score for each feature, $x_i$.
   1.a $RS(x_i, Y) =$ e.g. Information Gain $= 2.0 \times \left[ \frac{\text{gain}}{H(Y) + H(x_i)} \right]$.
2. Rank the features in descending order based on the value of $RS(x_i, Y)$.
3. Select $x_i$ whose relevant score is greater than 0.
   3.a If $RS(x_i, Y) > 0$ then $X_{rr} = X_{rr} \cup \{x_i\}$.
4. Compute the mean of relevant scores.
   4.a $\mu_{rv} = \frac{\sum_{i=0}^{X_{rr}} RS(x_i, Y)}{|X_{rr}|}$.

**Remove redundant features**
5. For each $x_j \in X_{rr}$.
   5.a Compute the inter-correlation score between features, as
      $IS(x_i, x_j) =$ e.g. Information Gain $= 2.0 \times \left[ \frac{\text{gain}}{H(Y) + H(x_j)} \right]$.
6. Compute the mean of the inter-correlation score as
   6.a $\mu_{rd} = \frac{\sum_{i=0}^{X_{rr}} IS(x_i, x_j)}{|X_{rr}|}$.
7. Compute the weight value based on both the relevant and redundant scores.
   7.a $w = \frac{\mu_{rd}}{\mu_{rv}}$.
8. For each $x_j \in X_{rr}$.
   8.a Use the weight value to calculate the importance of features
      $S(x_i) = w \cdot x_{rv}^i - x_{rd}^i$.
   8.b Select the optimal features $S_{optimal}$.
      If $S(x_i) > 0$ then $S_{optimal} = S_{optimal} \cup x_i$.
9. Return the final set of optimal features, $S_{optimal}$.
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

terion, consistency-based criterion, statistical-based criterion and distance-based criterion. In this section, the selection of a feature set from a global perspective (as explained in Section 4.2.1), using the valuation criteria of these FS techniques, is investigated with respect to the optimality and stability metrics. The aim is to identify an FS technique that not only improves the model performance but also yields unambiguous outputs. Therefore, to make the best use of the traffic data and obtain stable results, a cross-validation strategy is used.

For each dataset, the order of the flows is randomised because many of ML techniques exhibit order effects [Fisher et al., 1992]. In the proposed experiments, we obtain $N$ feature subsets and the corresponding goodness rate for each FS technique on each dataset. We also obtain the stability for each FS technique across different datasets. Algorithm 2 below shows the various steps for measuring the goodness and the stability for each FS technique on traffic data.
Algorithm 2: Experimental Procedure

1 Input:

2 Parameter $N := 10$; $M := 10$;

3 Feature selector $T := \{t_1, t_2, \cdots, t_m\}$;

4 $DATA = \{D_1, D_2, \cdots, D_n\}$;

5 Output:

6 Goodness & Stability;

7 foreach $Selector^i_t \in [1, T]$ do

8    foreach $D_i \in DATA$ do

9        Subset = $Selector^i_t(D_i)$;

10       Superset$^i_t = Superset^i_t \cup Subset$;

11       foreach $times \in [1, M]$ do

12          randomise instance-order for $D_i$;

13          generate $N$ bins from the randomised $D_i$;

14          foreach $fold \in [1, N]$ do

15              $TestData = bin[fold]$;

16              $TrainData = data - TestData$;

17              $Train'_Data = select Subset from TrainData$;

18              $Test'_Data = select Subset from TestData$;

19              Classifier = learner($Train'_Data$);

20              Result = apply Classifier to ($Test'_Data$);

21              Goodness = ComputeGoodness(Result);

22       Stability = ComputeStability (Superset$^i_t$);

Figure 4.1 shows the comparison of the candidate FS techniques in terms of the proposed stability values and optimality scores. It can be seen that, in most cases, there is no clear winner amongst the FS techniques. As a result, there is no FS technique that satisfies both of the evaluation criteria. For example, CBF performs very well on the optimality metric but performs poorly on the stability metric. GR performs equally poorly on both metrics.
Therefore, the final conclusion is that each of these FS techniques has its own advantages, and also does identify features that are both stable and accurate (i.e. with a high goodness value). This is our motivation for developing a multi-criterion fusion-based approach to identify an optimal and stable set of features that helps classifiers to perform well and gain the confidence of network experts.

**Figure 4.1:** Stability and optimality of FS techniques on real-traffic data

### 4.3 GOA – Global Optimization Approach

As explained earlier, existing traffic classification approaches (e.g. [Zhang et al., 2013c][Yuan et al., 2010][Auld et al., 2007][Chou et al., 2008][Moore and Zuev, 2005]) rely on a single FS technique. However, a single FS technique does not perform well for both evaluation criteria. Thus, the Global Optimization Approach (GOA) is proposed here with the aim of discovering most-valuable features for the description of traffic flows with respect to both stability and optimality criteria. GOA is based on a hybrid FS technique that can reflect the trade-off between optimal and stable features. The overall process and methodology of GOA is depicted in Figure 4.2, where the first phase combines several well-known FS techniques (i) to provide an initial pool of feature subsets with good generality across different traffic datasets (Section 4.3.1), and (ii) to reduce the possibility of including the irrelevant features.
in the subsequent analysis. In the second phase, instead of relying on a fixed threshold, an entropy-based technique is proposed to adaptively select only robust (i.e., both stable and accurate) features from the larger initial pool of features. Relying on the information-theoretic method, the algorithm effectively finds the optimal cut-off of the robust features based on the underlying distribution of the selected feature set, substantially reducing the number of features that are input to the third phase (Section 4.3.2). Finally, the third phase uses a more computationally intensive procedure of Random Forest filtering to choose the best candidate feature subset that is then passed to the classification algorithm (described in Section 4.3.3) for network traffic classification.

**Figure 4.2:** The proposed Global Optimization Approach

### 4.3.1 Integration of Feature Selection

There are two essential steps in creating an FS integration. The first step involves running a set of different FS techniques on traffic datasets. The second step aggregates the output
features of the different FS techniques. To make the best of the different FS techniques, the first step of the integration approach combines five well-known FS techniques that cover various independent criteria including Information Gain [Han and Kamber, 2006][Fayyad and Irani, 1993], Gain Ratio [Han and Kamber, 2006][Quinlan, 1993], Chi-Square [Liu and Setiono, 1995], CBF [Yu and Liu, 2003][Hall and Holmes, 2003] and CBC [Dash and Liu, 2003]. The reader may notice that we excluded Relief [Kira and Rendell, 1992], as this approach has several shortcomings. First, this method searches only for one nearest hit and one nearest miss. Noisy data could make this approximation inaccurate. Second, if there are missing values for features, the algorithm will crash because it cannot calculate the distance between those instances [Robnik-ˇSikonja and Kononenko, 1997].

In general, the procedure of selecting informative features first is used for each FS technique (explained in Table 4.1), and thus the output would be dependent on its evaluation criterion (e.g. Information Gain [Han and Kamber, 2006][Fayyad and Irani, 1993], Chi-Square [Liu and Setiono, 1995] etc.)

The second step aggregates the outputs of different FS techniques as follows:

- Let $S^* = \{S^*_1, \cdots, S^*_n\}$, where $n > 1$, be an initial pool of feature subsets $S^*_i = \{f_k | k = 1 \cdots d, f_k \in S\}$, obtained from applying $n$ FS techniques on a given dataset $D$, where $D > 1$.

- Calculate the frequency of features $f \in Y$ in the initial pool. Let $X$ be a subset of $Y$ representing all features that appear anywhere in the initial pool $S^*$:

$$X = \{f | f \in Y, F^*_f > 0\} = \bigcup_{i=1}^{n} S^*_i, \ X \neq 0 \quad (4.3)$$

- Order the features of $X$ based on their frequency. Let $\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n$ be such as $F(\hat{f}_1) \geq F(\hat{f}_2) \geq \cdots F(\hat{f}_n)$. Then $S_{\text{sorted}} = \{\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n\}$.

- Pass the sorted set of features $S_{\text{sorted}}$ to the second phase of GOA.
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

4.3.2 The Adaptive Threshold

This section introduces the concept of *maximum entropy* [Csiszár, 1996] to compute an optimal cut-off and to automatically cull robust features from largely (unstable) selected features. Maximum entropy has been shown to be a viable and competitive concept in many domains including language modelling [Rosenfeld, 2005] and part-of-speech tagging [Ratnaparkhi et al., 1996]. It is a general technique for estimating probability distribution from data. One of its advantages is that when nothing is known, the distribution should be as uniform as possible; that is, it has maximal entropy. Another advantage is that maximum entropy can satisfy any given constraints to find the optimal solution.

The maximum entropy is used to find the optimal cut-off and automatically cull robust features by estimating the conditional distribution of given features obtained from the first stage. Therefore, the selected features should satisfy the given constraints before they are passed to the third stage. In addition to selecting stable features, the other motivation of this adaptive threshold is to reduce computational time required by the intensive search approach by selecting only a small set, since only $2^m - 1$ are needed to be checked compared to $2^n - 1$ (where $m \ll n$, $m$ refers to number of features in $S$, where $n$ the total number of features in the datasets).

In the remaining parts of this section, we introduce the concept of the entropy-based adaptive threshold.

**Conceptual View of the Adaptive Threshold**

Let $X$ be a categorical random variable whose value is one of $N$ possible categories or values $c_1, \cdots, c_N$, where $N \geq 2$. We observe a sequence of $m$ realizations of $X$, i.e. $m$ independent draws $x_1, \cdots, x_m$ from $X$, where $m > 2$. Let $m_i$ be the number of times the $i^{th}$ category appears in our sample. So $0 \leq m_i \leq m$ for all $i = \{1; 2; \cdots; N\}$. Then the relative frequency of each category in our sample gives us the empirical distribution of $X$, which induces an empirical probability distribution on $X$ as follows

$$p_i = p(X = c_i) = m_i/m,$$  \hspace{1cm} (4.4)
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

So if some category $c_j$ does not appear in our sample, then $p_j = 0$. If all $m$ draws come from only one category $c_k$, then $p_k = 1$ and $p_i = 0$ for $i \neq k$. Similarly, we define the empirical entropy of $X$ with

$$H(X) = - \sum_{i=1}^{N} p(x_i) \log p(x_i)$$ (4.5)

where $0 \log 0 = 0$. Moreover $H(X)$ is bounded above as follows. We assume $m < N$, then

$$H(X) = - \sum_{i=1}^{N} p_i \log(p_i)$$

$$= - \sum_{i=1}^{N} p_i \log(m_i/m)$$

$$= \sum_{i=1}^{N} p_i(\log(m) - \log(m_i))$$

$$\leq \sum_{i=1}^{N} p_i \log(m) = \log(m)$$ (4.6)

because $\sum_{i=1}^{N} p_i = 1$. If $m > N$, a similar logic as above would give us $H(X) \leq \log(N)$. So, in general we have $0 \leq H(X) \leq H_{\text{max}}(X) = \log(\min\{m; N\})$. The upper bound $H_{\text{max}}(X)$ allows us to “standardize” $H(X)$ by putting it on a scale of 0 to 1, independent of $m$ or $N$. In this chapter, we consider the standardized entropy as the relative uncertainty:

$$RU(X) = \frac{H(X)}{H_{\text{max}}(X)}$$ (4.7)

Let $A \subset \{c_1, \cdots, c_N\}$ be the set of observed values of $X$, i.e., the distinct draws that make up our random sample. Then

1. $RU(X) = 0$ implies that $H(X) = 0$, which only happens if $\log(p_i) = 0$ or $p_i = 0$. If $\log(p_i) = 0$ and $p_i \neq 0$ then $p_i = 1$ for some $i$ and $p_j = 0$ for all $j \neq i$. In other words, $H(X) = 0$ if and only if our random sample consists of $m$ draws, all of which come from
the same $i^{th}$ category. In this situation, we can say that there is no uncertainty, i.e. zero entropy.

2. If $m < N$, then some categories must be absent in our sample, i.e., $A$ is a strict subset of $X$. In this situation, $RU(X) = 1$ implies that $H(X) = \log(m)$, which can happen only if $p_i = 1/m$ for all $i$ such that $c_i \in A$ (and $p_j = 0$ for all $c_j$ not in $A$). This is shown below:

$$
H(X) = -\sum_{j=1}^{N} p_j \log(p_j) \\
= -\sum_{i=1}^{m} \frac{1}{m} \log(1/m) \\
= -\log(1/m) \\
= \log(m)
$$

So when $m < N$ then $RU(X)$ is at its maximum of 1 if every category $c_i \in A$ occurs the same number of times as every other $c_j \in A$; which means that the empirical distribution is a uniform distribution over $A$ and we have maximum unpredictability.

3. If $m > N$ then for $RU(X)$ to be equal to 1 we must have $H(X) = \log(N)$ which, using the same logic as above, can only happen if $p_i = 1/N$ for all $i$. In other words, the empirical distribution is a uniform distribution over $X$. This can only happen if all the categories in $X$ are present in our sample an equal number of times. In this case, $A$ and $X$ represent the same set.

4. If $m = N$, then $p_i = 1/m$ and $p_i = 1/N$ give us the same result. In this situation, every category in $X$ is represented in our random sample exactly once.

We can generalize the above formulation by letting $H_{\text{max}}(X|A) = \log(|A|)$. When $m < N$ then $|A| = m$ and $H_{\text{max}}(X|A) = \log(m)$. When $m < N$ then $|A| = N$ and $H_{\text{max}}(X|A) = \log(N)$. So either way we have $RU(X) = 1$ if and only if $p_i = 1/|A|$. In this chapter, we refer to the Relative Uncertainty as the Confidence measure.
Extracting Stable Features

We identify stable features using the confidence measure defined in (4.7). Let \( N \) be the total number of frequency of any feature \( f \in Y \) in set \( S_{\text{sorted}} \). Then the (induced) probability distribution \( P_A \) on \( A \) is given by

\[
P_A(\hat{f}_1) = \frac{f_i}{N}
\]

(4.9)

where \( f_i \) is the frequency of a feature. Then the (conditional) relative uncertainty (referred to as a confidence measure). \( C(P_A) = C(X|A) \), measures the degree of uniformity in the observed features in \( A \). If \( C(P_A) \) is close to 1, say \( \beta = 0.9 \), then the observed features are uniformly distributed, and thus the features are considered to be important. We say a subset \( S_{\text{best}} \) of \( A \) contains the best features if \( S_{\text{best}} \) is the smallest subset of \( A \) such that (i) the probability of any value in \( S \) is larger than the remaining values; (ii) \( R = A - S \), is close to being uniformly distributed (e.g. \( C(P_A) = C(X|R) > \beta \)). Consequently, \( S \) contains the best features in \( A \), while the remaining features are less frequently selected.

4.3.3 Intensive Search Approach

As stated previously in Section 4.2, high stability does not necessarily imply a high accuracy rate and vice versa. In this section, the goal is to select feature subsets from the candidate set (obtained from previous stages) that lead to good generalization. By using filters in the first stage, we intend to find a small and robust set of candidate features that pass the second stage (confidence measure), which are inputs into a more computationally intensive subset selection procedure referred to as Random Forest filtering. The strength of the intensive search approach is that it focuses directly on optimizing the performance of the prediction Random Forest filtering by maximizing the goodness measure presented in Section 4.2. Consequently, a feature is eliminated if it gives little or no additional information beyond that subsumed by the remaining features. In particular, this will be the case for both irrelevant and redundant features. Therefore, while this method has encountered some success in selecting optimal features, it is often prohibitively expensive to run and can break down when a large of number of features are present. Thus, to mitigate such a problem, we need to select only a small
number of the original features. This is achieved in our approach via the adaptive threshold discussed in the previous section. However, the choice of a machine learning algorithm for the intensive search approach and a search strategy needed to be considered. This will be discussed in the following subsections.

**Random Forest**

Ensemble algorithms have achieved success in machine learning by combining multiple weak learners to form a strong learner [Fumera et al., 2008]. The Random Forest method [Breiman, 2001] centres around this idea by adding $n$ additional layers of randomness to *bagging* [Fumera et al., 2008]. Such a method builds each tree using a different bootstrap sample of the data, and it can change how the classification or regression trees are built. While in traditional trees, each node is split using the best split among all variables, in Random Forest, each node is split using the best among a subset of predictors randomly chosen at that node. This method appears to give better performance than other ML algorithms (such as Neural Networks and Support Vector Machines), and also it can be robust against over-fitting [Breiman, 2001].

In what follows, we briefly discuss the steps of the Random Forest approach that is used for identifying the final optimal features for traffic classification:

- Split the data into training sets and testing sets.

- Learning sets used to train classifiers based on RF and determine the importance of features. This can be done by growing the regression tree with the following modification: rather than choosing the best split among all predictors, randomly sample $m$ of the predictors and choose the best split from those features.

- At each bootstrap iteration, predict the data not in the training sets (referred to as “out-of-bag”, OOB data) using the tree grown in the previous steps.

- Calculate the goodness rate by aggregating the OOB predictions.

- Estimate the importance of features by examining the extent to which the goodness
rate increases when (OOB) data for that feature is permuted while all others are left unchanged. In this way, a set of features with a high goodness rate is selected.

**Search Strategy**

As mentioned earlier, Random Forest filtering requires a larger number of training sets to search for the best performing features subset [Schuschel and Hsu, 2002]. For instance, let us consider a traffic dataset with $N$ features, there exist $2^N$ candidate subsets. This search space is exponentially prohibitive for an exhaustive search. Therefore, an important issue in identifying the “best” candidate features (for traffic classification) is the choice of a wrapper as an efficient search strategy. These strategies broadly fall into three categories: exponential, randomized, and sequential. In this chapter, we consider one of the well-known sequential search strategy so-called SFS (Sequential Forward Selection) [Jain and Zongker, 1997], as the complexity of the search scheme for Random Forest is in the order of $N^2$.

In general, SFS determines the “best” set of features for extraction by starting from an empty set and sequentially adding a single feature in the superset to the subset if it increases the value of the goodness (described in Section 4.2). Table 4.2 shows how the forward selection search has been modified to produce a ranked list of features.

**Table 4.2: Procedure of Sequential Forward Selection (SFS)**

<table>
<thead>
<tr>
<th>Iteration #</th>
<th>Feature Set</th>
<th>Score</th>
<th>Best addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 0</td>
<td>$\ldots, f_1, \ldots, \ldots, \ldots$</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$f_1, \ldots, f_2, \ldots, \ldots$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Iteration 1</td>
<td>$\ldots, f_2, \ldots, \ldots, \ldots$</td>
<td>40</td>
<td>$f_2$</td>
</tr>
<tr>
<td></td>
<td>$\ldots, \ldots, f_3, \ldots, \ldots$</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\ldots, \ldots, \ldots, f_4$</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Iteration 2</td>
<td>$f_1, f_2, \ldots, \ldots, \ldots$</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\ldots, f_2, f_3, \ldots, \ldots$</td>
<td>65</td>
<td>$f_3$</td>
</tr>
<tr>
<td></td>
<td>$\ldots, \ldots, f_2, \ldots, f_4$</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>Iteration 3</td>
<td>$f_1, f_2, f_3, \ldots, \ldots$</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\ldots, f_2, f_3, f_4$</td>
<td>57</td>
<td>$f_1$</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>$f_1, f_2, f_3, f_4$</td>
<td>62</td>
<td>$f_4$</td>
</tr>
</tbody>
</table>
For example, if we provide a set of features (e.g. \( f_1, \ldots, f_4 \)) to the search process, it starts by adding a single feature (e.g. \( f_1 \)) to the empty set and evaluates its score. In iteration 1, the best single feature is \( f_2 \) with a score of 40; therefore, this will be added to the subset. In iteration 2, all 2-feature subsets that include \( f_2 \) are evaluated. In this case, the addition of \( f_3 \) results in the best score, which is equal to 65. In iteration 3, \( f_1 \) will be added to the subset but the best score is only 60 (by adding \( f_1 \)), which is worse than the previous score. The search terminates since no single feature addition can improve the best subset from the previous iteration. In this case, the search has been forced to stop after iteration 3. Therefore, the selected best features are \( \{ f_2, f_3 \} \).

The Algorithm

Algorithm 3 has three parts, and selects the best and stable features from the original space. In the first part (Lines 1-12), the algorithm extracts an initial pool of features by applying the global selection procedures (presented in Section 4.2.1) on different traffic data taken over different periods of time, and returns a consensus rank feature subset. In the second part (Lines 13-28), an efficient approximation is performed to identify the most stable features in \( S \) from \( A \). The algorithm starts with an appropriate initial value (e.g. \( \beta \)), and searches for the optimal cut-off threshold from above via “linear approximation” (increasing the threshold \( \beta \) by linear growth factor at the \( i \) steps). The algorithm iterates to find the most stable subset as long as the confidence measure of the (conditional) probability distributed \( P_R \) on the remaining features sets \( R \) is less than \( \beta \). The algorithm examines each feature in \( R \) and includes it in \( S \) if its probability exceeds the threshold. The algorithm stops either if the probability of a feature exceeds the maximum probability value or if the probability distribution of the remaining feature value is close to being uniformly distributed.

However, the earlier step has a high probability of producing a subset of representatives and stable features. It may include some features that are most likely to be more strongly correlated, and this can degrade the performance traffic classification task. Thus, in the third part (Lines 29-41), the algorithm uses an intensive search technique based on the Random Forest learning approach to guarantee the quality of the final subset features.
Algorithm 3: Global Optimization Approach

1. **Input:** Feature Selectors \( T := \{ t_1, t_2, \ldots, t_m \} \);
2. **DATA** = \{ \( D_1 \), \( D_2 \), \ldots, \( D_n \) \};
3. Parameters \( S_{Optimal} := \emptyset \), \( S_{initPool} := \emptyset \), \( S_{frequency} := \emptyset \), \( S_{Sorted} := \emptyset \);
4. **Output:** \( S_{final} \); // Return the best features in terms of Optimality and Stability
5. Obtaining Optimal Features \( \text{ApplyFS}(\text{DATA}, T, S_{initPool}, S_{Optimal}, S_{frequency}, S_{Sorted}) \);
6. for \( D_i \in \text{DATA} \) do
7.   for \( t_j \in T \) do
8.     \( S_{Optimal} := \text{SelectGlobalFeatures}(D_i, t_j, \alpha) \); // select features as explained in Section 4.2.1
9.     \( S_{initPool} := S_{initPool} \cup S_{best} \);
10. \( S_{frequency} := \text{CountFrequency}(S_{initPool}) \);
11. \( S_{Sorted} := \text{SortFeatures}(S_{frequency}) \);
12. return \( S_{Sorted} \);
13. Entropy-based Stable Features Extraction
14. Parameters \( \beta := 0.98, i = 1 \); Initialization \( S_{Stable} := \emptyset \), \( R := S_{Sorted} \);
15. \( \text{ExtractStableFeatures}(S_{Stable}, R, \beta, i) \);
16. \( P_R := \text{ComputeProb}(R) \);
17. \( \delta := \text{ComputeRU}(P_R) \);
18. \( \mu := \text{FindMax}(P_R) \);
19. while \( (\beta \leq \delta) \) do
20.   \( \mu := \mu \times i \);
21.   \( i := i + 1 \);
22.   for \( f_i \in R \) do
23.     \( P_{f_i} := \text{ComputeProb}(f_i) \);
24.     if \( P_{f_i} \geq \mu \) then
25.       \( S_{Stable} := S_{Stable} \cup \{ f_i \} \);
26.       \( R := R \setminus \{ f_i \} \);
27.   \( P_R := \text{ComputeProb}(R) \), \( \delta := \text{ComputeRU}(P_R) \);
28. return \( S_{Stable} \);
29. Intensive Search Approach
30. Parameters \( \gamma := 0, \theta_{goodness} := 0 \); Initialization \( S := \emptyset \), \( S_0 := S_{Stable} \), \( S_{final} := S_0 \);
31. \( D = \{ f_1, f_2, \ldots, f_n-1 \} \);
32. \( \text{EvaluateFinalSet}(S_{final}, \theta_{goodness}, \gamma) \);
33. \( \theta_{goodness} := \text{EvaluateFeatures}(S_0, D, \text{RF}) \); // Evaluate \( S_0 \) by Random Forest \( \text{RF} \)
34. repeat
35.   \( S := \text{Generate}(D) \); // Generate a subset for evaluation
36.   \( \gamma := \text{EvaluateFeatures}(S, D, \text{RF}) \); // Evaluate the current a subset \( S \) by \( \text{RF} \)
37.   if \( \gamma \geq \theta_{goodness} \) then
38.     \( \theta_{goodness} := \gamma \);
39.     \( S_{final} := S \);
40. until \( \text{Maximized} \{ \theta_{goodness} \} \);
41. return \( S_{final} \);
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

It starts the search from initial subset $S_0$ and iterates to find the best subsets using sequential forward selection. Each generated set $S$ is evaluated using the goodness measure (defined in Section 4.2) and compared with the previous best one; if $S$ is better, it becomes the current best subset. The search iterates until the best subset of features is found and the goodness measure provides a natural stopping criterion. Thus, the respective algorithm tends to produce better feature subsets, since the mining algorithm uses the goodness measure as a dependent measure. Finally, the remaining features are all significant features that contain indispensable information about the original features set.

4.4 Evaluation

This evaluation shows how the output features of GOA improve the performance and the accuracy of the classification. We have compared the performance of GOA with that of two well-known network traffic FS techniques: FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]. These two approaches are inherently different. FCBF-NB selects the most valuable features derived using Fast Correlation Based Feature selection (FCBF) and threshold of Naive Bayes(NB) [Moore and Zuev, 2005], whereas Bayesian Neural Network (BNN) uses feature inter-dependent ranking described in [Auld et al., 2007]. In particular, the proposed GOA approach, and the baseline techniques (FCBF-NB and BNN) would follow the same network environment of LOA approach presented in Figure 3.1.

To provide a quantitative comparison, we require test data containing network traffic with a range of application types. In the absence of such publicly-available data with annotated labels (to indicate application type), we decided to use widely available and acceptable traffic datasets from Cambridge Lab [Moore et al., 2005]. This is one of the largest publicly-available network traffic traces that were collected by a high-performance network monitor.

The collated dataries are based on traces captured using loss-limited, full-payload capture to disk where timestamps with resolution of better than 35 nanoseconds are provided. The data was acquired over several different periods of time from two different sites. These two sites are both research centres but conduct research in different departments and are located in different countries. These sites are referred to as Site A and Site B. Each site hosts up to

113 (January 9, 2015)
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

1,000 users connected to the Internet via a full-duplex Gigabyte Ethernet link. Full-duplex traffic on this connection was monitored by a high-performance network for each traffic set. From site A, we use 3-day-long datasets taken over weekdays in 2003, 2004 and 2006 (for simplicity we refer to these as Day #1, Day #2 and Day #3). From site B, we use data-sets collected on weekdays in late 2007 (referred to as Site B). Table 4.3 lists the number of flows alongside the type of applications for each site and time period.

Table 4.3: Data statistics number of the flows

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>279477</td>
<td>140875</td>
<td>218620</td>
<td>208214</td>
<td>6000</td>
</tr>
<tr>
<td>MAIL</td>
<td>28124</td>
<td>16487</td>
<td>3978</td>
<td>10598</td>
<td>543</td>
</tr>
<tr>
<td>BULK</td>
<td>12151</td>
<td>10793</td>
<td>5351</td>
<td>545</td>
<td>264</td>
</tr>
<tr>
<td>ATTACK</td>
<td>1751</td>
<td>987</td>
<td>35</td>
<td>3932</td>
<td>600</td>
</tr>
<tr>
<td>CHAT</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>500</td>
<td>51</td>
</tr>
<tr>
<td>P2P</td>
<td>2085</td>
<td>2762</td>
<td>22287</td>
<td>17323</td>
<td>408</td>
</tr>
<tr>
<td>DATABASE</td>
<td>2794</td>
<td>2606</td>
<td>91181</td>
<td>0</td>
<td>786</td>
</tr>
<tr>
<td>MULTIMEDIA</td>
<td>496</td>
<td>4</td>
<td>19</td>
<td>11</td>
<td>48</td>
</tr>
<tr>
<td>VOIP</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>1025</td>
<td>102</td>
</tr>
<tr>
<td>SERVICES</td>
<td>1808</td>
<td>1111</td>
<td>70</td>
<td>455</td>
<td>316</td>
</tr>
<tr>
<td>INTERACTIVE</td>
<td>86</td>
<td>36</td>
<td>323</td>
<td>310</td>
<td>693</td>
</tr>
<tr>
<td>GAMES</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>147</td>
<td>139</td>
</tr>
<tr>
<td>GRID</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>93</td>
<td>50</td>
</tr>
</tbody>
</table>

4.4.1 Evaluating FS based on the Proposed Metrics

In this section, we compare the effectiveness of the proposed GOA approach and the baseline feature selection techniques for network traffic (namely, FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]) according to both proposed metrics, optimality and stability. In particular, we follow the same experimental procedures (presented in Section 4.2) to assess
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

all feature selection techniques.

Figure 4.3 shows a comparison of the stability and optimality of the GOA approach and the baseline feature selection techniques (FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]) over the traffic datasets. From the estimated optimality rate in Figure 4.3, we can observe that the GOA approach and FCBF-NB have achieved slightly improvement over BNN technique. Also, when it comes to the stability comparison between all FS techniques, we observe that GOA approach achieves substantial improvement in comparison to both FCBF-NB and BNN techniques (refer to Figure 4.3). The good results of the GOA approach with respect to the proposed metrics prove the strength of multiple-criteria fusion to produce not only an optimal but also a stable features set for traffic classification task.

Also, we used the simple tabular approach (introduced in Chapter 3, Section 4.4.1) to categorise the different FS techniques based on both proposed metrics, optimality and stability. This way of comparing can serve two purposes, including grouping FS techniques with similar characteristics, and also providing a better and fair way to compare such techniques on the same framework. In particular, we compare the proposed GOA approach with the candidate FS techniques (discussed in Section 4.2.1) as well as two well-known network traffic FS techniques: FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007].

Table 4.4 summarises and categorises the stability and optimality values of GOA approach and the baseline FS techniques into three categories: low, medium and high using equation 3.12, which was introduced in Chapter 3. Based on the developed categorisation approach, it can be seen from Table 4.4 that the performance of the proposed GOA approach has satisfied both optimality and stability metrics compared to the well-known FS techniques, including the previously proposed (in Chapter 3) Local Optimization Approach (LOA), FCBF-NB and BNN methods.

4.4.2 Comparison between GOA, FCBF-NB and BNN

In order to avoid being biased toward our proposed metrics, the capabilities of the GOA approach and the baseline methods (FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]) have been further assessed with three commonly used types of tests including
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

Figure 4.3: Stability and optimality of GOA approach and the baseline FS techniques on real-traffic data.

Table 4.4: Evaluation of FS techniques on the categorisation framework

<table>
<thead>
<tr>
<th>FS Tech</th>
<th>CBF</th>
<th>Chi2</th>
<th>IG</th>
<th>CBC</th>
<th>CBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimality Stability</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Stability</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>FS Tech</td>
<td>GR</td>
<td>LOA</td>
<td>FCBF-NB</td>
<td>BNN</td>
<td>GOA</td>
</tr>
<tr>
<td>Optimality Stability</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Stability</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
classification accuracy, subset size and performance. To quantify the accuracy of classification models, we use standard measurements such as overall Classifier Accuracy (CR), Precision (PR) and Recall (RC), which are defined as follows in terms of metrics defined in Table 4.5.

- **Overall Accuracy**: the percentage of correctly classified instances over the total instances.
  \[
  CR = \frac{TP + TN}{TP + TN + FP + FN}
  \]  
  (4.10)

- **Precision**: the number of class members classified correctly over the total number of instances classified as class members for a given class.
  \[
  PR = \frac{TP}{TP + FP}
  \]  
  (4.11)

- **Recall**: the number of class members classified correctly over the total number of class members for given class.
  \[
  RC = \frac{TP}{TP + FN}
  \]  
  (4.12)

**Table 4.5: Standard confusion metrics for evaluation accuracy**

<table>
<thead>
<tr>
<th>Actual connection label</th>
<th>Predicted connection label</th>
<th>E1</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

**Classification Accuracy**

The aim of this experiment is to check whether the output features are able to help classifiers to distinguish different types of network applications (e.g. WWW, P2P, Attack etc). We use three metrics to compare the three techniques. Half of the traffic data has been used as training data and the remaining half has been used for testing.

Figure 4.4a and Table 4.6 show the classification accuracy for each method in terms of Classifier Accuracy (CR), Precision (PR) and Recall (RC). The results have been computed
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

Table 4.6: Comparison of GOA against FCBF-NB and BNN in terms of classification rate, subset size and runtime

<table>
<thead>
<tr>
<th>Methods</th>
<th>Classif. rate</th>
<th>Subset Size</th>
<th>Run Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR</td>
<td>PR</td>
<td>RC</td>
</tr>
<tr>
<td>GOA</td>
<td>97.7</td>
<td>97.01</td>
<td>97.7</td>
</tr>
<tr>
<td>FCBF-NB [Moore and Zuev, 2005]</td>
<td>96.5</td>
<td>96.33</td>
<td>96.86</td>
</tr>
<tr>
<td>BNN [Auld et al., 2007]</td>
<td>92.55</td>
<td>92.87</td>
<td>92.55</td>
</tr>
</tbody>
</table>

based on the classification of the flows. For a traffic classification task, traffic applications are expected to be correctly classified. Therefore, achieving higher percentages of such metrics is desirable. Note, the proposed approach (GOA) consistently achieves higher average classification accuracies and produces a feature set that yields results comparable to, and in most cases better than, both FCBF-NB and BNN. These results support the stability of our approach in terms of classification accuracy.

Runtime Performance

A key motivation for using GOA is to reduce the size of the features set required to classify traffic data. First, we built a classification model using training data and measured the execution time needed to build the classifier. This is an important consideration because the model-building phase is computationally time-consuming. Second, we measured the time required by the classification task. This is particularly important when considering real-time\(^2\) classification of potentially thousands of simultaneous traffic flows. In order to test the computational performance of the output of the GOA in comparison with the other two approaches, we use data size of 10000 flows. The size of the training set is ultimately limited by the amount of available memory because the classification algorithm must load the entire training set into memory before building the model. For the analysis, all operations are performed on an Intel Pentium Dual Core 3.4 GHz processor machine with 2 GB of RAM.

\(^2\)The proposed GOA will not be applied for real-time, but its candidate features set which is identified in this chapter
Figure 4.4b shows the runtime performance of the classification algorithm using the output of all methods. This test was repeated 5 times to give the average execution time required to build the classifier and to classify the traffic. Note that GOA shows a significant reduction in computation time in comparison with both BNN and FCBF-NB. However, from Figure 4.4b, it can be observed that the pre-processing time of the proposed GOA is more computationally expensive than the other two approaches, BNN and FCBF-NB. This is because the GOA approach incorporates multiple feature selection evaluation criteria to produce the candidate set of features for traffic classification. Thus, future work will be devoted to improving the speed-up factor of the GOA approach by using (i) the GPU environment and/or (ii) parallel computing.

**Subset Size**

The aim of FS techniques is to select a small subset of features that have the highest discriminating power. Therefore, the output feature subset size is used to compare the GOA approach and the other two FS techniques. This is important since network classifiers need to analyze a large volume of traffic data; therefore, the smaller subset results in greater classifier efficiency and quicker classification task. The second column of Table IV shows the size of the subset selected by each FS technique. Results show that different approaches produce a different number of features. It is interesting to note that GOA produces a significantly smaller subset than both the approaches (50% smaller than FCBF-NB and 75% smaller than BNN). This suggests that the proposed adaptive threshold and the learning algorithm-based stopping criterion are effective in finding the optimal number of the candidate features.

**4.4.3 Relevance of Selected Features**

Previous sections showed a quantified evaluation, where GOA approach outperformed the benchmarks techniques in both classification accuracy and performance. This section investigates whether features identified by GOA are indeed meaningful in networking terms when observed by someone without access to knowledge of the class values (i.e., type of application) associated with the flows.
(a) Comparing the Accuracy, Recall and Precision of GOA, FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]

(b) Normalized build time speed and classification speed for each feature set obtained by (GOA, FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007])

Figure 4.4: Comparing the accuracy and the performance of classification using the output set of GOA, FCBF-NB and BNN
### Table 4.7: Comparative ranking of the most valuable features

FCBF-NB rank refers to [Moore and Zuev, 2005]. BNN rank refers to [Auld et al., 2007]. GOA rank refers to the proposed approach described in Section 4.3

<table>
<thead>
<tr>
<th>GOA Rank</th>
<th>FCBF-NB Rank</th>
<th>BNN Rank</th>
<th>Feature #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>Port Number Server → Client</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>Port Number Client → Server</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>The count of all packets seen with the PUSH bit in the TCP Header. Server → Client</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
<td>The total number of bytes sent in the initial window i.e, bytes seen in the data before receiving the first ack packet from endpoint. Client → Server</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>The total number of bytes sent in the initial window. Server → Client</td>
</tr>
<tr>
<td>-</td>
<td>6</td>
<td>-</td>
<td>6</td>
<td>The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes filed divided by the actual data packet reported</td>
</tr>
<tr>
<td>-</td>
<td>7</td>
<td>-</td>
<td>7</td>
<td>Median of the total bytes in IP packets Client → Server.</td>
</tr>
<tr>
<td>-</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>The count of all the packets with at least one byte of TCP data payload. Client → Server</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>-</td>
<td>9</td>
<td>Variance of bytes in (Ethernet) packet. Server → Client</td>
</tr>
<tr>
<td>-</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>Minimum segment size Client → Server</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>7</td>
<td>11</td>
<td>The total number of Round-Trip Time (RTT) Samples found</td>
</tr>
<tr>
<td>-</td>
<td>12</td>
<td>-</td>
<td>12</td>
<td>The count of all the packets seen with the PUSH bit set in the TCP header. Client → Server</td>
</tr>
<tr>
<td>-</td>
<td>13</td>
<td>-</td>
<td>13</td>
<td>Variance of bytes in (Ethernet) packet. Client → Server</td>
</tr>
<tr>
<td>-</td>
<td>14</td>
<td>-</td>
<td>14</td>
<td>The maximum window advertisement seen. Client → Server</td>
</tr>
<tr>
<td>-</td>
<td>15</td>
<td>-</td>
<td>15</td>
<td>The average throughput calculated as the unique bytes sent divided by the elapsed(e.g. the time deference between the capture of the first and the last packets). Client → Server</td>
</tr>
<tr>
<td>-</td>
<td>16</td>
<td>-</td>
<td>16</td>
<td>The maximum window advertisement seen. Client → Server</td>
</tr>
<tr>
<td>-</td>
<td>17</td>
<td>-</td>
<td>17</td>
<td>The number of transitions between transaction mode and bulk transfer mode.</td>
</tr>
<tr>
<td>-</td>
<td>18</td>
<td>-</td>
<td>18</td>
<td>Median of bytes in (Ethernet) packet in both directions.</td>
</tr>
</tbody>
</table>
Table 4.7 provides insight into the ranking of the optimal features that have been identified by our GOA approach with the other two FS methods based on feature-independent selection (Naive Bayesian) and correlated-features (Bayesian Neural Network). As mentioned earlier, the majority of these features are derived directly by observing one or more TCP/IP headers using a tool such as tcptrace or by performing simple time analysis of packet headers. Upon comparing each reduction, we note that the features selected by GOA are identified by previous studies and some of the prominent features are supported by FCBF-NB [Moore and Zuev, 2005] and BNN [Auld et al., 2007]. On the other hand, we note a limited overlap in a number of significant differences between the other two feature-reduction methods. In addition to this, the values of features (selected by Bayesian neural network BNN [Auld et al., 2007]) are dependent upon the RTT which will be subject to change depending on the monitored site, which makes the features less stable than those features selected by GOA.

4.4.4 Temporal Decay and Spatial Robustness

We evaluate the temporal robustness of the candidate features that are selected by GOA. This experiment illustrates the temporal stability of selected features when classifying new traffic. As a preliminary investigation, we perform the FS techniques on the four available traffic datasets. Three datasets are taken over different periods of time; for simplicity, we refer to them as Day #1 (2003), Day#2 (2004) and Day #3 (2006). First, we apply a machine learning algorithm GOA to construct the training model using filtered features from the Day #1 datasets (2003). We test the generated model for classification of datasets from Day #2 (2004) and Day #3 (2006). It is assumed that there will be some, if not considerable, change in the variety of applications and the composition of traffic in the period 2003-2006.

The objective of this experiment is to test whether candidate features of the proposed GOA are able to help a classifier to distinguish between traffic applications across different periods of time. It is evident from Figure 4.5a that the classification accuracy for GOA features remains stable at an average CR of 98.07%. This suggests that the candidate features result in high and stable classification results for traffic from the same site and across different periods.
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

Figure 4.5: Classification of the traffic using the candidate features (Temporal Stability and Spatial Accuracy)

(a) Overall accuracy of classification traffic over different period of time using the candidate features set (temporal stability)

(b) Accuracy of a model trained from one site and applied to each other site (spatial accuracy)
To evaluate the effectiveness of the spatial independence of the candidate features, the model is trained using network data from one site and tested against data from a different site. Therefore, we first construct the model using a subset of each of the Day #3 and site B datasets. We then validate each model against the remaining dataset from that site. Note that there is no overlap between the training sets and testing sets, and we do not validate the accuracy of the model using the same datasets. Three experiments were performed to evaluate the ability of the candidate features to accurately classify the traffic from different sites. The first experiment uses Day #3 (to represent Site A) with filtered features to construct the model and uses the generated model to classify the traffic from Site B. The second experiment used 50% of data from Site B and the remaining 50% from Day #3 to build the model.

The generated model is then used to measure the accuracy across different networks. Figure 4.5b shows the accuracy results for the model trained on Day #3 and tested on Site B, and vice versa. It is clear from this figure that the accuracy of the candidate features used to classify traffic on different sites is 95.02%. However, we notice that there are drops (in accuracy) in comparison to classifying the traffic on the same sites. The main reason for this happening is that there are some variations in the distribution of applications and the composition of the traffic has changed, since the collected traffic at Site B is from a totally different network and almost two years later. To overcome this difficulty, combining decisions of several classifiers could lead to better classification results. A full investigation of such properties would be a valuable contribution by future work. In the second experiment, we evaluate the model built with multi-site training datasets. Figure 4.5b shows that there is a substantial increase in overall accuracy at 99.1% in comparison with the model which is specifically built from a given site. This indicates that collaboration to share traffic data between multi-site (spatial-networks) and ISPs organizations can generate a more efficient and representative classification model than just building the model from a single site. However, this collaboration is very beneficial, ISPs and Web sites are extremely reluctant to share their traffic data among them. The reason being they are competitors in business, and they are worried if the customers’ privacy would be affected if they share data. To address
such an issue, a new privacy-preserving framework for traffic data publishing is introduced in Chapter 5.

4.5 Impact of the Candidate Features on different ML Algorithms

In this section, we examine the performance of the candidate feature set optimized by the GOA approach on different machine learning techniques. This is an important task as the third stage of the proposed approach uses a predetermined learning algorithm to guide the search to determine the final set of best features. Such a method uses its predictive accuracy as the primary measure and, therefore, it may inherit bias towards the predetermined learning algorithm. To avoid such case, we evaluate the output of the GOA approach on five standards classification algorithms, namely K-Nearest Neighbours algorithm, Naive Bayes [John and Langley, 1995; Duda and Hart, 1996], ID3 [Quinlan, 1986], Support Vector Machine [Boser et al., 1992; Cortes and Vapnik, 1995; Vapnik, 2000] and Logistic Regression (LR) [Han and Kamber, 2006; Fayyad and Irani, 1993]. These ML algorithms can achieve superior performance and they are the top five out of ten evaluated machine learning algorithms. In addition, these five ML algorithms work differently and represent an ideal cross-section of learning algorithms to use for a test on learning bias.

4.5.1 The Sensitivity of the Candidate Features on different ML Algorithms

As mentioned in the previous section, one very important advantage of using GOA is the trust that a system administrator can have in the output of GOA to classify traffic data regardless of the ML algorithm that is used. To evaluate the sensitivity of GOA’s output to different ML algorithms, the GOA approach was performed as described in Section 4.3. The performance of each ML algorithm is evaluated by applying $K$-fold-cross validation to the three datasets. In this process, each dataset is divided into $K$ subsets. Each time, one of the $K$ subsets is used for testing while the remaining $K - 1$ subsets form the training set. Performance statistics are calculated across all $K$ trials. Throughout this experiment, the value of $K$ is set to 10, a widely-accepted empirical value [Kohavi, 1995] for accuracy estimation and model selection. Figure 4.6 demonstrates the accuracies achieved by NB,
CHAPTER 4. OPTIMAL AND STABLE FEATURE SET FOR TRAFFIC CLASSIFICATION

Figure 4.6: The average effect of discretisation on the three traffic datasets

ID3, 7K-NN, LR and SVM using the candidate feature set (obtained by GOA). The results confirm our hypothesis that there is little difference between the accuracies induced by the five learning algorithms. It can be seen that the accuracies of all algorithms using such features are high and equivalent. The only case where there is a significant difference is the accuracy of NB and SVM, suggesting that these algorithms suffer from the presence of continuous-valued features in the traffic data. Throughout the remainder of this chapter, we examine various ways to improve the performances of such ML algorithms by discretizing the input features. In the following subsections, we first briefly describe the discretisation technique, then we present the experimental results of the ML algorithms after applying the discretisation method.

4.5.2 Discretisation to Improve Classification Accuracy

The candidate features for the classification of traffic datasets involve continuous-valued features (“continuous” refers to features taking numerical values). As discussed in the previous section, the accuracy of some ML algorithms suffer from the presence of such features. To address this problem, we use the state-of-the-art supervised discretisation technique devel-
opied by Fayyad and Irani [Fayyad and Irani, 1993], due to its ability to improve performance compared to other methods (such as Equal Width Interval Binning and Holte’s 1R Discrétizer) [Dougherty et al., 1995]. In general, this discretisation technique partitions the range of the features into sub-ranges (at least two). Essentially, the technique combines an entropy-based splitting criterion such as information gain, with a minimum description length stopping criterion. The best cut point is the one that makes the sub-intervals as pure as possible, i.e., where the information value is smallest. The technique is then applied recursively to the two subintervals. For a set of instances \( S \) a feature \( f_i \) and a cut point \( T \), the class information entropy of the partition obtained by \( T \) is given by:

\[
E(f_i; T; S) = \frac{S_1}{S} \text{Ent}(S_1) + \frac{S_2}{S} \text{Ent}(S_2) \tag{4.13}
\]

where \( S_1 \) and \( S_2 \) are two intervals of \( S \) bounded by cut point \( T \), and \( \text{Ent}(S) \) is the class entropy of a subset \( S \) given by

\[
\text{Ent}(S) = \sum_{i=1}^{C} p(C_i; S) \log(p(C_i; S)) \tag{4.14}
\]

To determine the optimal stopping criteria, Minimal Description Length Principle is applied. This strategy is used to partition \( T \) if and only if the cost of encoding the partition and the classes of the instances in the intervals induced by \( T \) is less than the cost of encoding the classes of the instances before splitting. Therefore, the partition is accepted if and only if the cut point \( T \) is

\[
\text{Gain}(f_i; T; S) > \frac{\log(N - 1)}{N} + \frac{\Delta(f_i; T; S)}{N} \tag{4.15}
\]

where

\[
\text{Gain}(f_i; T; S) = \text{Ent}(S) - E(f_i; T; S)
\]

and
In equation 4.15, \( N \) represents the number of instances, \( c, c_1 \) and \( c_2 \) are the number of distinct classes present in \( S, S_1 \) and \( S_2 \) respectively. The first component is the information needed to specify the splitting point; the second is a correction due to the need to transmit which classes correspond to the upper and lower sub-intervals.

### 4.5.3 Impact of Discretising the Candidate Features

Here we compare the effectiveness of using the candidate feature subset and discretisation technique as a pre-processing step to improve the model performance of various ML techniques. The discretisation technique was performed in the way described in Section 4.5.2. Throughout this experiment, the number of bins, \( K \), is set to 10, since this was suggested by [Dougherty et al., 1995] as the best heuristics setting, based on S-Plus’s histogram binning algorithm [Dougherty et al., 1995].

Figure 4.7a shows that the discretisation technique prior to the aforementioned machine learning techniques can substantially improve accuracy. Specifically, we found the performance of SVM and NB was significantly increased. This is because the entropy-based discretisation (i) approximates the class distribution and thus helps to overcome the normality assumption used for continuous features, and (ii) provides regularisation to such ML techniques.

The performance of the remaining ML techniques on traffic data using the entropy discretisation did not degrade, but remained the same. The only slight decrease was in C4.5, one possible reason being that such ML technique did not take full advantage of the local entropy discretisation that could be performed on the traffic data. In general, from Figure 4.7, it can be seen that the aim of the GOA approach and discretisation method to improve the
performance of the network traffic task has been fulfilled.

![Graph](image1)

(a) Influence of discretisation technique on the three datasets

![Graph](image2)

(b) Comparing the average accuracy for each ML technique with and without discretisation technique

**Figure 4.7:** Impact of the output of GOA and the discretisation technique on different classification algorithm
4.6 Conclusion

In light of the emergence of new services and distributed applications, it has become critically important to identify not only the best but also most robust (in terms of stability and optimality) features from large network traffic datasets. In this chapter, we first introduced a novel Global Optimisation Approach (GOA) to exploit the optimality and stability metrics to address the limitation of existing FS techniques and to produce representative features that satisfied both metrics from a global prospective. The GOA approach involves (i) combining well-known FS techniques to filter out a mass of irrelevant features in the first step, (ii) adopting a threshold based on information theory to extract only stable features, and (iii) obtaining a more compact subset and avoiding over-fitting Random Forest filtering. Finally, we presented a strategy based on a discretisation method to enhance the performance of GOA approaches and to significantly improve the accuracy of different ML algorithms. An extensive study using a publicly-available traffic data benchmark has proved the strength of the proposed GOA approach in comparison to the baseline FS techniques.
Chapter 5

PrivTra: Privacy-Preserving Framework for Traffic Data Publishing

As demonstrated in Chapter 4, sharing network traffic data has become a vital requirement in machine learning algorithms when building an efficient and accurate network traffic classification and intrusion detection system. However, inappropriate sharing and usage of network traffic data could threaten the privacy of companies and prevent sharing of such data. In this chapter, we present a privacy-preserving strategy-based permutation technique called PrivTra framework, in which data privacy, statistical properties and data mining utilities can be controlled at the same time. In particular, the proposed approach involves: (i) vertically partitioning the original dataset to improve the performance of perturbation, (ii) developing a framework to deal with various types of network traffic data including numerical, categorical and hierarchical attributes: (iii) grouping the portioned sets into a number of clusters based on the proposed framework; and (iii) accomplishing the perturbation process by the altering the original attribute value by a new value (clusters centroid). The effectiveness of the PrivTra framework is demonstrated through several experiments real network traffic, intrusion detection and simulated network datasets. Through experimental analysis, we show that compared with previous approaches, our PrivTra framework effectively deals with multivariate traffic attributes, produces compatible results as the original data, improves the
CHAPTER 5. PRIVTRA: PRIVACY-PRESERVING FRAMEWORK FOR TRAFFIC DATA PUBLISHING

performance of the five supervised approaches and provides a high level of privacy protection.

5.1 Introduction

As discussed in Chapter 4, sharing traffic data between spatial-domain networks is highly desirable to create an accurate and global predictive traffic classification model. However, such collaborative spatial-domain classifiers are deficient in privacy protections. In particular, ISPs and Web sites are extremely reluctant to share their operational data among them. The reason being they are competitors in business, and they are worried that their customers’ privacy could be affected if they share data. Moreover, many customers are reluctant to install software from Web analytics services such as Alexa [Kahle and Gilliat., 2009]. They are worried that this kind of software will keep track of all websites the customers visit and report all the sites they visited. Tragically, even good intentions do not certainly turn to good security and privacy protections. This can be confirmed by the notion that large-scale data breaches have become more frequent [Clearinghouse, 2009]. Eventually, we trust the fact that many useful distributed data-analysis applications will be gaining good grip only if privacy is guaranteed.

Nevertheless, in order to counter the emergence of new applications and patterns, a number of network classifiers and intrusion detection systems (IDs) based on machine learning techniques [Zhang et al., 2011; Linda et al., 2009; Fahad et al., 2013; Tsang and Kwong, 2005; Almalawi et al., 2013a; Kim et al., 2011a; Tze-Haw et al., 2011] have been proposed to assist network experts to analyse the security risks and detect attacks against their systems. However, a key problem in the research and development of such efficient and accurate network traffic classification and intrusion detection systems (based on machine learning) is the lack of sufficient traffic data, especially for industrial network (e.g. Supervisory Control and Data Acquisition SCADA) systems [Chan et al., 2011; Mahmood et al., 2010]. Unfortunately, such data is not so easy to obtain because organizations do not want to reveal their private traffic data for various privacy, security and legal reasons [Mahmood et al., 2010; Liu et al., 2010; Khelil et al., 2012]. For instance, organizations do not want to admit that they have been attacked and therefore are unwilling to divulge any information about this. Thus, it has
been widely-recognized today that traffic data confidentiality and privacy are increasingly becoming an important aspect of data sharing and integration [Khelil et al., 2012; Alcaraz et al., 2012; Lisovich et al., 2010].

5.1.1 Contribution

This chapter proposes a new privacy-preserving data framework to facilitate the publishing of network traffic data while ensuring that private data will not be disclosed. Figure 5.1 described a typical scenario for the data collection phase and publishing phase. In the former phase, a data publisher collects the data from the record owner (network traffic companies/organizations). In the latter phase, the data publisher releases the transformed data to a data miner or to the public, called a data recipient, who will then conduct data mining on the published data.

Figure 5.1: Data collection and data publishing

The contributions of this chapter can be summarized as follows:

- A privacy-preserving framework (PrivTra) based on a permutation technique is proposed to deal with network traffic data. Although the vast majority of existing approaches (e.g [Oliveira and Zaiane, 2010; Vidya Banu and Nagaveni, 2013; Ghinita et al., 2011]) to privacy-preserving computation have been active in other domains including marketing data and biomedical data, such studied schemes are not readily
applicable to private data in traditional and industrial networks. This is because their design assumes that the data being protected have to be numeric. A key challenge with network traffic data is the need to deal with various types of attributes: numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure. For example, byte counts are numerical, protocols are categorical, and IP addresses have a hierarchical structure [Mahmood et al., 2008]. Consequently, the *PrivTra* framework is presented to satisfy the privacy requirements while maintaining sufficient data utility. First, the traffic mixed dataset is subdivided into the attributes of flow record, creating fragments including the pure categorical dataset, the pure numerical dataset and the pure hierarchal dataset. Next, well-established similarity measures to deal with various types of attributes are proposed to help produce more meaningful clusters. Last, the clustering results on the numerical, categorical and hierarchal datasets are combined as a categorical dataset, on which the machine learning classifiers are employed to obtain the final output. In particular, the objective of such a framework is to enforce privacy-preserving paradigms, such as k-anonymity and l-diversity, while minimizing the information loss incurred during the anatomizing process.

- The proposed *PrivTra* framework is evaluated on both synthetic and real-life datasets. In particular, we compare the effectiveness of the *PrivTra* against a new class of privacy-preserving data mining approaches, namely: *PCA-DR* [Vidya Banu and Nagaveni, 2013], *SDP* [Oliveira and Zaiane, 2010] and *RDP* [Oliveira and Zaiane, 2010]. A general observation indicates that the proposed framework outperforms the existing approaches with respect to a comprehensive set of criteria including: *Dealing with multivariate data, Efficiency, Scalability, Data quality and Privacy level* (see Section 5.4 for details).

The rest of the chapter is organized as follows: Section 5.2 introduces *PrivTra* as a new technique for privacy-preserving data publishing. In Section 5.3, we describe our SCADA platform, as a case study for generating industrial traffic data, its main components, and data processing. In Section 5.4, we evaluate the performance of *PrivTra* in anatomizing the traditional and industrial network traffic datasets. In Section 5.5, we conclude the chapter.
and discuss future research.

5.2 Preserving the Privacy Framework for Network Traffic Data

A number of studies in many domains, including marketing data and biomedical data have been proposed to prevent privacy leakage while still presenting a maximal utility to data analysts. In particular, as encrypted data (using the traditional RSA and AES methods) cannot be used for data analysis [Zhong et al., 2012], various privacy-preserving methods [Oliveira and Zaiane, 2010; Vidya Banu and Nagaveni, 2013; Ghinita et al., 2011; Mahmood et al., 2013; Domingo-Ferrer et al., 2006] for different data publishing scenarios have been proposed over the past few years. These methods have been divided roughly into three categories, namely data generalization methods (e.g. [Ghinita et al., 2011]), data transformation methods (e.g. [Oliveira and Zaiane, 2010; Vidya Banu and Nagaveni, 2013]) and data micro-aggregation methods (e.g. [Mahmood et al., 2013; Domingo-Ferrer et al., 2006]). For data generalization methods, the process is done by mapping sensitive attributes to more generalized values. For the transformation methods, the privacy of the data is preserved by transforming the original data into new values based on random multiplication or by projecting the original data into lower dimensional random space. For data micro-aggregation methods, the original data is partitioned into a small-sized group, and then replace the private values in each group are replaced with the group average.

Although privacy-preserving in data publishing has been studied extensively [Oliveira and Zaiane, 2010; Vidya Banu and Nagaveni, 2013; Ghinita et al., 2011], most of the existing studies focus on data with numerical and continuous attributes. However, network traffic flows contain data with various types of attributes (e.g. IP address, port numbers etc). Consequently, we need to address the challenge of how to preserve the privacy of such data while maintaining the quality of the data since they degrade. To address this problem, the following subsection describes the requirements of a well-developed privacy-preserving framework.
5.2.1 Desired Requirements

As a countermeasure and to mitigate the potential threats in the publishing of network traffic data, a well-developed privacy-preserving framework should include the following properties:

- **Dealing with multivariate data**: the privacy framework should have the ability to deal with various types of attributes: numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure.

- **Efficiency**: the assessment of the resources used by a privacy-preserving data mining algorithm depends on its efficiency, which represents the ability of the algorithm to execute with good performance in terms of the transformation process and improve the performance of machine learning techniques. Therefore, the privacy framework should consider these resources.

- **Data quality**: traffic data are mainly utilized by machine learning techniques to drive certain patterns, such as the type of flow (attacks or normal). Therefore, data quality should be at an acceptable level according to the intended data usage. If data quality is too degraded, the released dataset is useless for the purpose of knowledge extraction. Therefore, the privacy framework needs to be designed in an effective way to preserve the quality of the original data.

- **Privacy level**: a privacy-preserving mining method should incorporate a privacy protection mechanism in a careful manner, in order to prevent the discovery of sensitive information that is contained in published data.

Based on this comprehensive set of criteria, we have assessed the proposed framework and other baseline methods (in Section 5.4) to determine which approach meets specific requirements.

5.2.2 Overview of PrivTra Framework

This section describes the proposed privacy-preserving framework for traffic data publishing. In particular, the proposed *PrivTra* framework is based on the above requirements and
it attempts to preserve the privacy of network traffic data by modifying values of sensitive attributes, based on clustering transformation. The corresponding architecture of our Privacy-Preserving Traffic Analysis (PrivTra) framework is shown in Figure 5.2, where the first level involves the partitioning of the original data.

![Figure 5.2: A schematic representation of the privacy-preserving architecture for traffic data](image)

The primary reason for such a step, as mentioned before, is that network traffic data have various types of attributes: numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure. The partitioned data are then clustered based on similarity distance. The cluster assignment values produced by the different base clusters and the original class values are merged to form the input for various machine learning techniques (e.g., classifiers). The different steps of the proposed model are detailed in the following subsections.
5.2.3 Partitioning

Network flows are usually represented as vectors in a multi-dimensional space. Each dimension represents a distinct attribute describing the flow. Thus, flows are represented as an $m \times n$ matrix $D$, where there are $m$ rows, one for each flow, and $n$ columns, one for each attribute. This matrix contains categorical, numerical and hierarchical attributes. For example, byte counts are numerical, protocols are categorical, and IP addresses have a hierarchical stricture. A key issue for this scheme is how to deal with such various types of attributes, since the framework relies on distance-based clustering to ensure the privacy of the original data. To resolve this challenging problem, we exploit domain knowledge to intelligently split the original data for the individual records into $N$ partition sets according to certain flow attributes (e.g. IP address, port numbers etc). Consequently, the original mixed datasets are vertically partitioned into three sub-datasets: a pure categorical dataset, a pure numerical dataset and a pure hierarchal dataset.

5.2.4 Preserving the Privacy Based on Clustering Concept

The proposed PrivTra framework takes the clustering concept to find similar flows and replaces it with cluster assignments. However, a key problem with existing clustering algorithms [Anderberg, 1973; Gower and Ross, 1969; Xu et al., 2005b] for multivariate traffic data is the calculation of distance for different types of traffic attributes in order to accurately group similar patterns together. Thus, we need to address such challenges by presenting a framework that takes into consideration three types of attributes: numerical, categorical and hierarchal.

In this section, we address this issue and present Algorithm 4 which highlights the general steps that are performed on the traffic data to preserve the privacy of multivariate attributes.

Given the traffic dataset $[x_{ij}] = [d_{ij}] \cdot [class_i]$ where $1 \leq i \leq N_{instances}$ and $1 \leq j \leq N_{attributes}$, the process starts by partitioning the traffic data into multiple traffic data types (as discussed in the previous subsection). Then the partitioned data is passed to the clustering algorithm. The purpose of the clustering algorithm is to group the similar dataset into a number of $N_{cluster}$ clusters. Clustering is performed separately on the corresponding data.
types (e.g. \([N_{ij}, C_{ij}\) and \([H_{ij}]\)). At the completion of each clustering, each row of \([d_{ij}]\) is replaced with clusterings id. The assignment ids produced by base clusters (including \([N_{M_{ij}}], [C_{T_{ij}}]\) and \([H_{R_{ij}}]\)) that are combined to form the final modified data \([y_{ij}] = [M_{D_{ij}}] \cdot [class_{i}]\). In general, the output of the clustering algorithm depends on the attributes types, the clustering types and the similarity measurements. In the following subsections, we will focus on the similarity measurement that is developed for each type of attribute and also discuss the corresponding clusterings.

Algorithm 4: Privacy-Preserving Framework for Network Traffic Data

input:
1. Data ← \(\{f_1, f_2, ..., f_{n-1}\}\);
2. Sim ← \(\{Sim_{NM}, Sim_{HR}, Sim_{CT}\}\);

output:
3. \([y_{ij}] = [M_{D_{ij}}] \cdot [class_{i}]\); // modified dataset
4. 
5. \([N_{ij}, C_{ij}, H_{ij}] ← PartitionData(Data)\);
6. if Data is Partitioned then
   7. switch Attribute Type do
      8. case Numerical
         9. \([N_{M_{ij}}] ← NumericalCls(Sim_{NM}, [N_{ij}]\);
      10. case Categorical
          11. \([C_{T_{ij}}] ← CategoricalCls(Sim_{CT}, [C_{ij}]\);
      12. case Hierarchical
          13. \([H_{R_{ij}}] ← HierarchicalCls(Sim_{HR}, [H_{ij}]\);
      14. \([M_{D_{ij}}] ← CombModData([N_{M_{ij}}], [C_{T_{ij}}], [H_{R_{ij}}]\);

5.2.5 Numerical Attributes

Choosing an appropriate similarity measure is crucial for transforming the network traffic data into a new value, based on the clustering concept. Therefore, before transforming numerical attributes, we now formally define a distance/similarity for the numerical types. In general, there is a wide variety of distance/similarity measures for numerical data available in the literature [Sneath et al., 1973]; however, many of them are only good for linear dependence, not for nonlinear ones. Therefore, we have chosen the Euclidean distance to measure...
the distance between numerical attributes due to its ability to deal with non-linear dependence. Nevertheless, it has been shown that no metric can outperform Euclidean distance for ratio-based measurements [Stegmayer et al., 2012]. The process of calculating the distance between two numerical attributes is formulated as follows:

\[
d^i_n = (x_1, x_2) = \|x_1 - x_2\| = \|(x_1 - x_2)\|^2
\]  

(5.1)

where the centroid of numerical attributes \(i\) in cluster \(C\) having \(N\) points is given by

\[
c[i] = \frac{1}{N} \sum_{j=1}^{N} x_j[i].
\]

(5.2)

Our approach to transform the numerical attributes of the original data into new values

**Algorithm 5: Transformation of Numerical Attributes**

1. **Input:**
2. \([N_{ij}]\);
3. Parameter\# of clusters \(K\);
4. **Output:**
5. \([NM_{ij}]\);
6. // the corresponding centroid
7. foreach columns in \([N_{ij}]\) do
8.     foreach \(C_k \in [1, K]\) do
9.         // Initialize cluster \(C_k\) center \(I_k\)
10.           initialize \((I_k)\);
11.         for \(d_i \in \text{columns}_j\) do
12.             \(d^i_c \leftarrow \text{DetermineMember}(d_i, C_k)\);
13.             // Assign \(d_i\) to the cluster \(C_k\) whose center is close to \(d_i\)
14.             \(I_k \leftarrow \text{UpdateCenter}(C_k)\);
15.             // Update cluster centers \(I_k\)
16.             if \(I_k\) not changed then
17.                 \(d_i \leftarrow K\);
18.                 // Replace \(d_i\) with cluster centroid \(I_k\)
19.             else
20.                 go to step (9);
21.     end foreach
22. end foreach
23. \(\text{col}_j \leftarrow \text{col}_j \cup d_i\);
24. \(NM \leftarrow NM \cup \text{col}_j\);
builds on the standard the K-means clustering algorithm [Anderberg, 1973]. We have chosen K-means clustering algorithm, as a suitable means of grouping numerical data for the following reasons: (i) it is a data-driven method with relatively few assumptions on the distributions of the underlying data, and (ii) the greedy search strategy of K-means guarantees at least a local minimum of the criterion function, thereby accelerating the convergence. In this work, the K-means clustering algorithm is performed on numerical data instances using each feature column separately. Each K-means [Anderberg, 1973] cluster represents a region of similar instances, “similar” in terms of Euclidean distances between the instances and their cluster centroids. Algorithm 5 summarises the transformation mechanism used for the numerical attributes.

For each column in the numerical matrix $([N_{ij}])$, the process starts by initializing the center of cluster $j$. Then membership of each data item $d_i$ is assigned (based on equation (5.1)) to the cluster whose center is closest to $d_i$. After that, cluster centers are updated as in equation (5.2), in which the center of cluster $C_j$ is set to be equal to the mean of all data items in cluster $j$, $\{\forall j : 1 \leq j \leq k\}$. The process is repeated until the centroid does not change. Then if the cluster centers do not change then the original attribute value are replaced with the cluster assignment.

The time complexity of the transformation mechanism of the numerical attributes is $O(nkl)$ where $n = |D|$, $k$ is the number of clusters, and $l$ is the number of iterations. However, in practice, the $l$ is considered as a constant. Thus, the time complexity of the numerical transformation mechanism is $O(nk)$. The space complexity of K-means is $O(n)$.

### 5.2.6 Categorical Attributes

Measuring the notion of similarity or distance for categorical attributes is not as straightforward as for numerical attributes, this is due to the fact that there is no explicit notion of ordering between categorical values. Categorical attributes (also known as nominal or qualitative multi-state attributes) have been studied for a long time in various contexts [Boriah et al., 2010], leading to several categorical measures [Cramér et al., 1955; Maung, 1941; Pearson, 1916]. More recently, however, the overlap [Stanfill and Waltz, 1986] measure has
become the most commonly-used similarity measure for categorical data. The popularity of this measure is due to its simplicity and ease of use. Essentially, in determining the similarity between categorical attributes, a $N \times N$ similarity matrix is constructed for each attribute member, denoted as $S_m, m = \{1 \cdots M\}$. Matrix entries represent the similarity between two categorical attributes, $x_i$ and $x_j$ (as Equation 5.3), and the matrices are effectively merged to form the co-association ($CO(x_i, x_j)$) matrix (Equation 5.3):

$$S_m(x_i, x_j) = \begin{cases} 
1 & \text{if } x_i = x_j \\
0 & \text{otherwise}
\end{cases}$$

$$CO(x_i, x_j) = \sum_{k=1}^{d} w_k S_k(x_i, x_j) \quad (5.3)$$

where $S_k(x_i, x_j)$ is the per-attribute similarity between two values for the categorical attribute $A_k$, and $w_k = \frac{1}{d}$ quantify the weight assigned to the attribute $A_K$. Having obtained the $CO(x_i, x_j)$ matrix, the clusters centroids are then produced by applying the single-link (SL) method [Gower and Ross, 1969] on the resultant matrix, where the original value of attributes are then replaced with the corresponding centroid. Note, the single linkage method has been chosen due to its simplicity of implementation for massive data.

Algorithm 6 summarizes the transformation mechanism done for the categorical attributes. The cost complexity of categorical attributes transformation is tied to the pairwise similarity between the categorical attributes $O(N^2)$, and also to the time complexity of single-link clustering which is $O(N^2)$.

### 5.2.7 Hierarchical Attributes

The traffic data of both traditional and industrial networks includes hierarchical attributes (e.g. SrcIP and DstIP). Thus, one of the key challenges is how to calculate the distance between IP addresses. In general, a global IP address is unique and allocated by a central body (IANA) which tries to assign groups of contiguous IP addresses to organizations or geographic regions. This helps in keeping routing tables small and also in managing multi-cast routing information. IP addresses can be grouped into subnetworks based on the hierarchical
Algorithm 6: Transformation of Categorical Attributes

1. **Input:**
2. \( [C_{ij}] \);
3. **Parameter** \# of clusters \( K \);
4. **Output:**
5. \( CT[i,j] \);
   // the corresponding centroids
6. \( Sim[i,j] \leftarrow \) null, \( Co[i,j] \leftarrow \) null;
7. **for** \( x_i, x_j \in C_{ij} \) **do**
8. \( Sim[x_i,x_j] \leftarrow \) ComputeSimilarity\( (x_i,x_j) \);
   // Compute the similarity between each pair as eq [5.3]
9. \( CO[x_i,x_j] \leftarrow \) UpdateMatrix\( (Sim[x_i,x_j]) \);
   // For each pair update the co-association matrix as eq [5.3].
10. **IDs**\( \leftarrow \) ComputeDendrogram\( (CO[i,j]) \);
    // Compute the Single-linkage of the co-association matrix.
11. \( CT[i,j] \leftarrow \) ReplaceOriginal\( (IDs) \);

structure of an IP address. If a set of IP addresses belongs to the same subnetwork, then they are more likely to exhibit similar behavior than two random IP addresses. Exploring hierarchical structure of IP address has been highlighted in several studies of network traffic analysis and intrusion detection [Wang et al., 2006] [Mahmood et al., 2008]. In particular, Abdun Mahmood et al. [Mahmood et al., 2008] have successfully introduced a new hierarchal similarity measure to cluster sources that have similar network traffic behavior. Following the general approach of ECHIDNA [Mahmood et al., 2008], the IP address spaces a 32-level binary prefix tree corresponding to the 32 bits in an IP address, covering all \( 2^{32} \) possible IP addresses. Using this \( L \)-level generalization hierarchy, the hierarchal similarity measure can be defined as

\[
S_{HSM} = \frac{|path(root,n_1) - path(root,n_2)|}{L} \tag{5.4}
\]

where the numerator determines the length of the common segment between \( n_1 \) and \( n_2 \), and \( L \) is the maximum depth of generalization hierarchy. For example, the distance between 128.0.0.252/32 and 128.0.0.254/31 is \((32-30)/24=0.083\).

Algorithm 7 summarises the transformation mechanism for the hierarchial attributes.
The steps for the hierarchial transformation are as follows:

- Calculate the relationship between any pair of IP addresses using equation 5.4, leading to a new similarity matrix $n \times m$ between IP addresses. In particular, the IP addresses are mapped to a co-association matrix, where entries can be interpreted as votes ratios on the pairwise co-occurrences of IP addresses and are computed as the number of times each pair of IP addresses has a common and most-significant bit-group in their IP address.

- Convert the values of the corresponding $n \times m$ similarity matrix to distance values and change the format from square to vector.

- Apply the Single-linked method to the corresponding $n \times m$ similarity matrix. The underlying assumption is that IP addresses that have similar traffic behaviour are very likely to be co-located in the same cluster.

- Replace the original attributes of IP addresses with the corresponding clustering results.

The hierarchial attributes transformation has complexity of $O(2^N)$ for computing similarity of $N$ IP addresses and $O(N^2)$ for computational complexity of Single-linked method. The overall of the computational complexity of the hierarchial attributes transformation is $O(2^N + N^2)$.

5.3 Case Study: SCADA Platform and Processing

Industrial control system security (SCADA) has been a topic of scrutiny and research for several years, and many security issues are well-known. However, a key challenge in the research and development of security solutions for SCADA systems is the lack of proper modeling tools due to the fact that it is impractical to conduct security experiments on a real system because of the scale and cost of implementing stand-alone systems. The second contribution of this chapter is the development of a SCADA platform to provide a modular SCADA modeling tool that allows real-time communication with external devices using SCADA protocols. Such a platform is important not only to evaluate (i) the proposed privacy-preserving framework
Algorithm 7: Transformation of Hierarchial Attributes

1. **Input:**
2. \([H_{ij}]\);
3. *Parameter* \# of clusters \(K\);
4. **Output:**
5. \([HR_{ij}]\);
   // the corresponding centroid
6. \(\text{Sim}[i,j] \leftarrow \text{null}, HSM[i,j] \leftarrow \text{null};\)
7. **for** \(i_p \in [H_{ij}]\) **do**
   8. \(\text{Sim}[i_p, i_p] \leftarrow \text{ComputeSimilarity}(i_p, i_p, HSM);\)
      // Compute the similarity between each pair as eq [5.4]
   9. \(HSM[i,j] \leftarrow \text{UpdateMatrix}(HSM[i,j], \text{Sim}[i_p, i_p]);\)
      // For each pair update the \(HSM\) as eq [5.4].
10. \(\text{IDs} \leftarrow \text{ComputeDendrogram}(HSM[i,j]);\)
    // Compute the Single-linkage of the hierarchial similarity matrix, \(HSM\)
11. \(\text{HR}[i,j] \leftarrow \text{ReplaceOriginal}(\text{IDs})\)

\(\text{(PrivTra)}, \) but also (ii) enabling an additional benefit of testing real attacks and trying different security solutions for such systems. In this section, we first present the water platform and then describe data processing.

5.3.1 The Water Platform

The success of penetrations and attacks on industrial networks (e.g. SCADA systems) are hardly and rarely reported, and this is due to the sensitive nature of such systems [East et al., 2009]. As a consequence, network traffic and logged data are not publicly-available for security experts to mine normal/abnormal patterns. Therefore, a robust privacy-preserving data mining algorithms for SCADA systems are an optimal way to address this issue. However, up-to-date, data collection of SCADA systems are not publicly-available to enable us to evaluate the \(\text{PrivTra})\) framework. Thus, we opted to build a virtual SCADA lab and simulate a water distribution system (WDS) as the supervised infrastructure. In particular, we used visualization features to represent the key parts of the SCADA system. For example, the field devices (such as PLC and RTU) and control devices (such as MTU and HMI) are represented by a number of virtual machines after installing the library [Software, 2011]
CHAPTER 5. PRIVTRA: PRIVACY-PRESERVING FRAMEWORK FOR TRAFFIC DATA PUBLISHING

of the widely-used Modbus protocol [IDA, 2004]. The virtualized network is used as the communication infrastructure at all SCADA network levels (e.g. field and control levels).

To simulate the supervised infrastructure, we used the library of the well-known and free hydraulic and water quality model, called EPANET [Lewis, 1999], to develop a WDS server to act as a surrogate for a real WDS. The EPANET model involves three modules, namely hydraulic, water quality and water consumption modules. We fed the consumption module with a specific model (i.e. the 2010 Melbourne water consumption [Melbourne Water, 2012]) so as to simulate the realistic behaviour of a water distribution system. One virtual machine is assigned to the WDS server. This server feeds the simulated data to virtualized field devices, and receives the Modbus/TCP control messages via a proxy. This proxy is used as an interface between virtualized field devices and the WDS server. For realistic simulation, the WDS server reads and controls process parameters such as water flow, pressure and valve status in response to message commands from field devices. The manipulated process parameters in the WDS server include:

- Water flow, pressure, demand and level
- Valve status and setting
- Pump status and speed

5.3.2 A Water Distribution System (WDS) Scenario

Figure 5.3 depicts an example of a simple WDS for a small town. This type of town could be divided into three areas (A, B and C). Each area has an elevated tank to supply it with water at a satisfactory pressure level. The supplied water is pumped out by three pumps from the treatment system into $\text{Tank}_1$. The water is also delivered to $\text{Tank}_2$ by two pumps. $\text{Tank}_3$ is supplied through gravity because of the elevation of $\text{Tank}_2$ which is higher than $\text{Tank}_3$. $\text{Tank}_1$ is twice as big as $\text{Tank}_2$ and $\text{Tank}_3$ because it is considered to be the main water source for areas $B$ and $C$. 

146 (January 9, 2015)
The water network is monitored and controlled by the SCADA system. In this scenario, some of the PLCs, namely PLC1...PLC4, depend on each others’ readings to control their end devices. Therefore, the MUT server plays a key role in coordinating and exchanging the data readings among these devices, in addition to storing the acquired data in the Historian. The PLCs are logically programmed as follows:

- **PLC4** controls the operation of pumps P1, P2 and P3 according to the water level reading of Tank1, which is exchanged between PLC4 and PLC3 by the MUT server.

- **PLC3** controls pumps P4 and P5, which pump out the water from Tank1 into Tank2, according to the water level reading of Tank2. It reads the water level of Tank1, and regulates the valve V2 to maintain a satisfactory pressure level at the area A. As the highest water level is in Tank1 the greatest water pressure is in area A; therefore PLC3 adjusts the valve V2 according to the water level reading of Tank1.
• *PLC*$_{1}$ opens and closes the valve *V*$_{1}$ according to the water level reading of *Tank*$_{3}$ and reads the water level reading of *Tank*$_{2}$.

• *PLC*$_{2}$ reads the water level reading of *Tank*$_{3}$ in order to be sent to *PLC*$_{1}$ by the MUT server.

### 5.3.3 A Scenario of Attacks

The purpose of this scenario is to affect the normal behaviour of the Water Distribution System (WDS). The public network (e.g. Internet) is used to interconnect all WDS’s components through Ethernet modules. The Modbus/TCP application protocol is setup as a communication protocol. However, all TCP vulnerabilities are inherited and therefore the system is susceptible to external attacks such as DoS (Denial of Service) attacks and spoofing.

We have opted to simulate high-level control attacks, as they are difficult to detect because they do not fully stop the service (as is the case with DoS attacks), but they drastically reduce the performance of the SCADA system. These types of attacks require prior knowledge of the target system, and this can be obtained by the specifications, or by correlation analysis for the network traffic of that system. As mentioned from the specifications of the simulation system, *PLC*$_{4}$ controls the pumps *P*$_{1}$, *P*$_{2}$ and *P*$_{3}$ in accordance with the water level reading of *Tank*$_{1}$. This can be read by the MUT server from *PLC*$_{3}$, and sent from the MUT server to *PLC*$_{4}$. We launch a man-in-the-middle attack to intercept the message sent from the MUT server to *PLC*$_{4}$. This is done by acting as a proxy between these devices.

We modified the intercepted messages to send false readings of the water levels. Two false reading of the water level of *Tank*$_{1}$ are sent to *PLC*$_{4}$: (i) when the water level of *Tank*$_{1}$ reaches the level at which *PLC*$_{4}$ should turn on the three pumps, a false reading is sent to inform *PLC*$_{4}$ that the water level is above 98%. This type of attack will be performed repetitively until the water goes down to the lowest level. (ii) When *PLC*$_{4}$ should turn on the two pumps, a false reading is sent to let *PLC*$_{4}$ know that the water level is between 70%-80%. This type of false reading will be sent till the water level becomes lower than 70%. These types of attacks can be launched in a number of ways, and they are hard to detect because: (1) the false message is still legitimate in terms of the Modbus/TCP protocol.
specifications; and (2) the attack cannot be detected by any individual process parameter (such as water level readings) unless the statuses of the three pumps are taken into account.

5.4 Evaluation

The objective of this section is to study the performance of the PrivTra framework on data quality in terms of accuracy and discernibility cost. Specifically, we present the followings: (i) a brief description of the datasets, and (ii) an introduction to the experimental setup; then we seek answers to the following questions numerically:

1. How well are the well-known classifiers able to distinguish between distinguish Normal and Attack flows based on the transformed data?

2. How does the proposed scheme affect the runtime of classification techniques?

3. What is the runtime performance of the proposed scheme in compared with existing methods?

4. How closely can the original value of an attribute be estimated from the transformed datasets?

5.4.1 Datasets

To verify the advantages of the proposed privacy-preserving framework, eight simulated datasets are used in the experiments. We experimented also with two other publicly-available datasets, namely DARPA and Internet traffic data. These two datasets have become a benchmark for many studies since the work of Andrew et al. [Moore and Zuev, 2005]. They have different types of attributes: continuous, categorical and hierarchal. Table 5.1 summarizes the proportion of normal and anomaly flows, the number of attributes, and the number of classes for each dataset.
Table 5.1: Datasets used in the experiments

<table>
<thead>
<tr>
<th>data</th>
<th># instances</th>
<th># attributes</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>699</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>MHORD</td>
<td>2500</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>SPFDS</td>
<td>1000</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>DOSDS</td>
<td>4000</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>SPDOS</td>
<td>2500</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>SHIRD</td>
<td>1800</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>SHORD</td>
<td>400</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>ITD</td>
<td>21000</td>
<td>149</td>
<td>12</td>
</tr>
<tr>
<td>WTP</td>
<td>512</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>DARPA</td>
<td>10000</td>
<td>42</td>
<td>5</td>
</tr>
</tbody>
</table>

The characteristics of the ten datasets used are described below:

- **Internet Traffic Data (ITD):** the traffic datasets collected by the high-performance network monitor [Moore et al., 2003] are some of the largest publicly-available network traffic traces that are used in our experiment. These datasets are based on traces captured using its loss-limited, full-payload capture to disk where timestamps with a resolution of better than 35 nanoseconds are provided. The data was taken for several different periods in time from one site on the Internet. This site is a research facility which hosts up to 1,000 users connected to the Internet via a full-duplex Gigabyte Ethernet link. Full-duplex traffic on this connection was monitored for each traffic set. The site hosts several biology-related facilities, collectively referred to as Genome Campus (Cambridge Lab). There are three institutions on-site that employ about 1,000 researchers, administrators and technical staff. This campus is connected to the Internet via a full-duplex Gigabyte Ethernet link. It was on this connection to the Internet that the monitor was placed. Each traffic set consists of a full 24-hour, weekday period in both link directions. Refer to Chapters 3 and 4, for more details about this dataset.

- **DARPA Data:** since 1999, the DARPA’99 dataset has been the most widely-used
dataset for the IDS evaluations that use machine learning techniques. This dataset was prepared by Stolfo et al. [Stolfo et al., 2000] and is built based on the data captured in the DARPA'99 IDS evaluation program [Lippmann et al., 2000]. This dataset contains raw traffic flow records with an associated label to indicate whether the record was labelled as either normal or an attack. In particular, the simulated attacks fall in one of the most common types of attacks including: a Denial of Service Attack (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) and a Probing Attack. The original DARPA datasets is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. To validate our work, a sample of the original DARPA dataset is selected (the sized of the dataset is ultimately limited by the amount of memory since transformation methods need to load the entire training data into the memory). This sample dataset consists of approximately 10,000 single connection vectors each of which contains 41 attributes.

- **Water Treatment Plant (WTP):** this dataset was collected from the daily measures of sensors in an urban waste water treatment plant. The objective is to classify the operational state of the plant in order to identify abnormality through the state variables of the plant at each of the stages of the treatment process. As shown in Table 5.1, this dataset consists of 527 instances and each instance contains 38 attributes and is labelled either normal or abnormal.

- **Labelled Wireless Sensor Network Data:** The labelled wireless sensor network provides four datasets; include Multi-hop Outdoor Real Data (MHORD), Multi-hop Indoor Real Data (MHIRD), Single-hop Outdoor Real Data (SHORD) and Single-hop Indoor Real Data (SHIRD). These datasets were collected from a simple single-hop and a multi-hop wireless sensor network deployment using TelosB motes. Each dataset consists of correlated process parameters (e.g. humidity and temperature) collected during a 6-hour period at intervals of 5 seconds. The single-hop data was collected on 9th May 2010, and the multi-hop data was collected on 10th July 2010.
• **Simulated SCADA System Data**: There are so many types of attacks targeting SCADA systems with some researchers listing more than 50 different types of attacks targeting DNP3 [East et al., 2009] and Modbus [Huitsing et al., 2008]. However, for the sake of simplicity we used our SCADA simulator (described in Section 5.3) to simulate two attacks commands. The first attack is *Spoofing*, where an intruder connects to the local filed network to send fake messages to the PLCs as they were coming from a real MTU. The second attack is denial of service *DOS*, where the attackers launch flood attacks against the water distribution system. In order to construct the sets of SCADA flows, the simulated traces is splinted into three datasets. The first dataset (detonated as SPFDS) contains normal/Spoofing traffic. The second dataset (donated as DOSDS) contains normal/DOS traffic. The third datasets (detonated as SPDOS) is a combination of the Spoofing, DOS and normal traffic, which are useful to show how the different high-level control attacks can mislead the proposed privacy framework.

### 5.4.2 Baseline Methods

We compare the *PrieTra* framework with the most current and relevant privacy-preserving methods, including

- **The Principle Component Analysis Based Transformation (PCA-DR) [Vidya Banu and Nagaveni, 2013]**: this method preserves the privacy of a confidential attribute by replacing the original attributes of a dataset with a smaller number of uncorrelated variables called the principle components. The transformed matrix then shifted by multiplying it with an arbitrarily selected shifting factor to enhance security.

- **The Scaling Data Perturbation Method (SDP) [Oliveira and Zaiane, 2010]**: this method perturbed the confidential attributes by using a multiplicative noise perturbation. In particular, the used noise could be either positive or negative, and the set of operations takes only the value multiplied corresponding to a multiplicative noise applied to each confidential attribute.
• The Rotation Data Perturbation Method (RDP) [Oliveira and Zaiane, 2010]: this method is similar to SDP. However, the operation set used of this method takes only the value rotation (instead of multiply value) to identify a common rotation angle between the confidential attributes. Unlike the previous method, RDP can be applied more than once to some confidential attributes.

5.4.3 Quality Evaluation usingBenchmarking Machine Learning Techniques

To evaluate the quality of the datasets that are optimized by the PrivTra framework and to ensure that the output of the PrivTra does not convolve with specific ML techniques (e.g. classifiers). In particular, we evaluated and trained both original data and transformed data on various machine learning classifiers (deeply discussed in Chapter 4). These machine learning techniques include: K-Nearest Neighbours (K-NN) [Duda and Hart, 1996], Naive Bayes (NB) [John and Langley, 1995], Decision Tree (J48) [Quinlan, 1986], Support Vector Machine (SVM) [Vapnik, 2000] and Multi-layer Perceptron [Pal and Mitra, 1992] (MLP). The reasons for choosing these machine learning techniques are (i) as reported in [Wu et al., 2008], these ML algorithms can achieve superior performance and are the top five out of ten evaluated machine learning algorithms, (ii) previous studies show the capability of such techniques to handle high dimensional data, and (iii) these five ML algorithms work differently and represent an ideal cross-section of learning algorithms to use for testing learning bias.

5.4.4 Experiment Setup

Algorithm 8 shows the experimental procedures to evaluate the impact of the framework and the baseline methods on the classification quality. In particular, a cross-validation strategy is used to make the best use of the traffic data and to obtain stable result. For each dataset, all instances are randomised and divided to two subsets as training and testing sets. Consequently, we evaluate the effect of the proposed PrivTra framework and the baseline methods by building a classifier on a training set and measuring the classification accuracy on the testing set. Since the previously presented five ML techniques can exhibit an order effect, the result of each classifier is averaged over 10 runs on each transformation method.
Algorithm 8: Experimental Procedure

1 Input:

2 Parameter $N = 10$; $M = 10$;

3 Transformations($T$) = {PrivTra, SDP, RDP, PCA – RD};

4 DATA = {DARPA, WTP, · · · , $D_n$};

5 Classifiers = {$K$ – NN, NB, J48, SVM, MLP};

6 Output:

7 PerfoMetrics = {Accuracy, F-measure, Recall, Precision};

8 foreach Transformations$_i$ $\in [1, T]$ do

9     foreach $D_i \in$ DATA do

10        TransData = apply Transformations$_i$ to (Test$'_i$Data);

11        TransData = Transformations$_i$($D_i$);

12        foreach times $\in [1, M]$ do

13            randomise instance-order for TransData;

14            generate $N$ bins from the randomised TransData;

15            foreach fold $\in [1, N]$ do

16                Test$Data = bin[fold]$;

17                Train$Data = TransData – TestData$;

18                Train$'_Data = select Subset from TrainData$;

19                Test$'_Data = select Subset from TestData$;

20                foreach Classifier$_i$ $\in$ Classifiers do

21                    Classifier$_i = learner(Train'_Data)$;

22                    Results$_i = apply Classifier$_i to (Test'_Data)$;

23                    OverallResult = OverallResult $\cup$ Results$_i$;

24                    PerfoMetrics = average (OverallResult);

Note, we removed the categorical and hierarchal attributes from the datasets when we apply the baseline methods. This is because the baseline methods deal only with numerical data. However, our PrivTra framework applies to all traffic datasets regardless of the types
of attributes in these datasets.

5.4.5 Experiment Results and Comparison

This section tests the effectiveness of the output of PrivTra on different supervised ML techniques and compare it with the output of the baseline methods. The purpose of this investigation is also to see if PrivTra can significantly improve the performance of the chosen classifiers. Four types of external evaluation metrics are used to verify the performance of the PrivTra framework. In particular, we used recall, precision and overall accuracy metrics which were defined in the Chapter 4, as well as F-measure is the equally-weighted (harmonic) mean of precision and recall, which is defined as follows:

$$F\text{-measure} = \frac{Recall \times Precision}{Recall + Precision}$$

5.4.6 Experiment Results

We reinforce our motivation which was stated in Section 5.2 by measuring the data quality on the classifiers before and after transformation. We expect that the transformed data to maintain a reasonable degree of utility. The classification results including: overall accuracy, precision, recall and F-measure, over the original data and transformed datasets are given in the following sections.

Overall Accuracy

Table 5.2 shows the overall accuracy derived without transformation methods and the accuracy that was of the closest magnitude after the application of each of the transformation methods. In general, it can be observed from Table 5.2 that all the transformation methods are always effective in improving the performance of the five classifiers in comparison to the original data, except on the transformed data of the PCA-DR method (the accuracy values in bold indicate better performance). This is likely due to the fact that this method could weaken the dependence among different attributes. Nevertheless, it was evident from Table 5.2 that the proposed PrivTra framework achieves improvements as good as the base-
line methods on all ten datasets. This similarity in accuracy values shows that the positive benefits of transformation methods can be achieved without significant negative effect on classification accuracy. However, it is notable from Table 5.2 that the J48 classifier performs poorly on the three datasets (namely: MHIRD, DOSDS and WTP) using the PrivTra framework. As the output of the PrivTra framework contains binary data, the J48 classifier suffers from such data type.

**Table 5.2: Comparison of the overall accuracy of different classifiers using different transformation methods**

<table>
<thead>
<tr>
<th>ML Tech</th>
<th>Methods</th>
<th>MHIRD</th>
<th>MHORD</th>
<th>SPFDS</th>
<th>DOSDS</th>
<th>SPDOS</th>
<th>SHIRD</th>
<th>SHORD</th>
<th>ITD</th>
<th>WTP</th>
<th>DARPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>Original</td>
<td>96.93</td>
<td>98.43</td>
<td><strong>98.55</strong></td>
<td>97.61</td>
<td>98.31</td>
<td>97.25</td>
<td>98.30</td>
<td>91.19</td>
<td><strong>97.98</strong></td>
<td>98.55</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td><strong>98.47</strong></td>
<td>97.96</td>
<td>98.15</td>
<td><strong>98.21</strong></td>
<td>97.70</td>
<td>98.22</td>
<td>98.63</td>
<td>90.36</td>
<td>96.96</td>
<td>77.94</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>97.18</td>
<td>97.85</td>
<td>97.83</td>
<td>97.20</td>
<td><strong>98.78</strong></td>
<td>97.70</td>
<td>98.57</td>
<td>91.11</td>
<td>97.41</td>
<td>98.63</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>97.71</td>
<td><strong>98.76</strong></td>
<td>98.50</td>
<td>97.03</td>
<td>97.19</td>
<td><strong>98.05</strong></td>
<td>98.15</td>
<td>97.50</td>
<td>97.94</td>
<td><strong>98.78</strong></td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>98.14</td>
<td>97.63</td>
<td>98.42</td>
<td>97.39</td>
<td>97.24</td>
<td>98.04</td>
<td><strong>98.74</strong></td>
<td>91.22</td>
<td>96.74</td>
<td>97.22</td>
</tr>
<tr>
<td></td>
<td>J48</td>
<td><strong>98.16</strong></td>
<td>97.71</td>
<td>98.82</td>
<td>96.81</td>
<td>97.72</td>
<td>97.49</td>
<td>96.62</td>
<td>96.41</td>
<td><strong>97.80</strong></td>
<td>98.36</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>97.15</td>
<td><strong>98.60</strong></td>
<td>98.03</td>
<td><strong>96.95</strong></td>
<td>96.84</td>
<td>97.44</td>
<td>97.26</td>
<td>96.60</td>
<td>96.01</td>
<td>77.06</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>97.54</td>
<td>97.14</td>
<td>98.69</td>
<td>96.12</td>
<td><strong>98.52</strong></td>
<td>98.69</td>
<td>98.29</td>
<td>95.91</td>
<td>96.42</td>
<td>97.57</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>94.93</td>
<td>96.51</td>
<td>98.44</td>
<td>96.83</td>
<td>97.65</td>
<td>98.49</td>
<td>97.44</td>
<td><strong>97.11</strong></td>
<td>94.85</td>
<td><strong>98.78</strong></td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>97.91</td>
<td>97.51</td>
<td>98.12</td>
<td>95.98</td>
<td>98.42</td>
<td>98.38</td>
<td>97.62</td>
<td>96.69</td>
<td>96.69</td>
<td>97.33</td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>97.38</td>
<td>97.07</td>
<td>97.21</td>
<td>97.70</td>
<td>97.62</td>
<td>97.70</td>
<td><strong>98.64</strong></td>
<td>87.70</td>
<td>96.67</td>
<td>96.95</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>98.00</td>
<td><strong>98.16</strong></td>
<td>98.44</td>
<td>97.40</td>
<td><strong>97.64</strong></td>
<td>97.05</td>
<td>97.66</td>
<td>74.07</td>
<td>96.11</td>
<td>76.73</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td><strong>98.88</strong></td>
<td>97.33</td>
<td>96.85</td>
<td><strong>98.89</strong></td>
<td>97.42</td>
<td><strong>98.55</strong></td>
<td>97.98</td>
<td>88.06</td>
<td>96.11</td>
<td>97.73</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>98.65</td>
<td>97.79</td>
<td>98.42</td>
<td>97.49</td>
<td>97.57</td>
<td>97.58</td>
<td>98.08</td>
<td><strong>98.07</strong></td>
<td><strong>97.84</strong></td>
<td><strong>98.56</strong></td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>97.50</td>
<td>97.63</td>
<td>97.15</td>
<td>96.76</td>
<td>97.35</td>
<td>98.50</td>
<td>97.55</td>
<td>85.57</td>
<td>96.49</td>
<td>97.50</td>
</tr>
<tr>
<td></td>
<td>7K-NN</td>
<td>97.07</td>
<td>95.50</td>
<td>97.91</td>
<td>98.48</td>
<td><strong>98.16</strong></td>
<td>97.03</td>
<td>97.83</td>
<td>92.33</td>
<td>96.17</td>
<td>96.98</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>98.18</td>
<td>97.85</td>
<td>97.48</td>
<td>98.52</td>
<td>97.29</td>
<td>98.50</td>
<td>97.82</td>
<td>91.44</td>
<td>95.86</td>
<td>77.90</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>98.18</td>
<td>97.51</td>
<td>97.96</td>
<td><strong>98.56</strong></td>
<td>97.63</td>
<td><strong>98.69</strong></td>
<td>97.15</td>
<td>92.75</td>
<td><strong>97.25</strong></td>
<td>97.99</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>98.76</td>
<td><strong>97.91</strong></td>
<td>96.97</td>
<td>97.06</td>
<td>97.30</td>
<td>97.61</td>
<td>98.03</td>
<td><strong>98.18</strong></td>
<td>96.30</td>
<td>97.37</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td><strong>96.93</strong></td>
<td>97.49</td>
<td><strong>98.56</strong></td>
<td>97.79</td>
<td>97.02</td>
<td>98.78</td>
<td><strong>98.09</strong></td>
<td>93.49</td>
<td>98.85</td>
<td><strong>98.49</strong></td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>97.72</td>
<td>97.02</td>
<td>98.15</td>
<td>96.69</td>
<td>97.38</td>
<td>96.82</td>
<td><strong>98.52</strong></td>
<td>72.59</td>
<td>96.40</td>
<td><strong>97.97</strong></td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>97.14</td>
<td>96.98</td>
<td>98.38</td>
<td>97.09</td>
<td>97.73</td>
<td>96.16</td>
<td>97.13</td>
<td>53.88</td>
<td>95.52</td>
<td>77.16</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td><strong>97.88</strong></td>
<td>97.50</td>
<td><strong>98.61</strong></td>
<td>96.33</td>
<td><strong>98.46</strong></td>
<td>96.57</td>
<td>96.87</td>
<td>73.19</td>
<td>97.41</td>
<td>97.79</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>97.53</td>
<td><strong>98.96</strong></td>
<td>97.36</td>
<td>97.90</td>
<td>97.42</td>
<td><strong>97.33</strong></td>
<td>97.97</td>
<td><strong>97.81</strong></td>
<td><strong>97.93</strong></td>
<td>97.31</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>97.81</td>
<td>97.73</td>
<td>98.14</td>
<td>96.25</td>
<td>96.99</td>
<td>95.91</td>
<td>97.08</td>
<td>72.07</td>
<td>97.87</td>
<td>97.46</td>
</tr>
</tbody>
</table>
Precision and Recall

Figures 5.4 and 5.5 compare the precision and recall values acquired by the classifiers based on the transformed data of the framework and other baseline methods. This is an important task as it gives a more informative picture of classifiers’ performance as these classifiers tend to be affected by skewed classes (i.e. Attack and Normal) distributions. A general observation is that most of the datasets transformed with PrivTra framework have the best precision compare to other baseline methods. This is mainly because PrivTra provides a true representation of original data. However, the transformed data of PrivTra framework often leads to dramatic degradation on 7K-NN performance. This can possibly explained by the fact that 7K-NN applies Euclidean distance, which is not adapted to binary data. According to the calculated F-measures, all of the transformation methods improved the performance of most classification methods, though in many cases performance did not vary considerably between untransformed (i.e. original) and transformed data or between the varied transformations and classification methods that were applied to the various datasets. There are certain notable exceptions, in particular datasets for which specific classification systems seemed wholly unsuited regardless of any transformation method applied, and also a few cases in which a transformation method greatly improved the performance of a classification method. The DOSDS dataset and the DARPA dataset both resulted in lower overall performance across transformation and classification methods, yet there is a substantial spread in the performance of transformation and classification methods. For the DOSDS dataset, the MLP classification system was the strongest performer overall, and performance was significantly improved for several classification systems through the use of the PrivTra transformation method. In the case of the SVM classification method, performance was almost doubled following this transformation. The PrivTra transformation method generally had the greatest positive impact on performance across datasets and transformation methods, although it actually greatly reduced performance when the 7K-NN classification method was applied to the DOSDS dataset. Performance on the DAPRA set was somewhat higher on average than the DOSDS dataset, although the MLP classification method showed especially poor performance with this dataset.
Figure 5.4: Comparison of the precision values of the PrivTra framework against the baseline transformation methods.
Figure 5.5: Comparison of the recall values of the PrivTra framework against the baseline transformation methods.
Overall, when performance was already relatively high for a dataset (and this tended to be relatively consistent across classification methods), the transformation methods did not show tremendous variance, though in certain cases of difficult datasets particular transformation methods did have significant impacts on the performance of specific classification methods. Performance was increased by the PrivTra transformation method to the greatest degree, and the MLP classification method was similarly the highest performer across a greater number of datasets than other classification methods, although not to as significant a degree as PrivTra outperformed other transformation methods, on average. Given these results, it is necessary to question the benefits of conducting additional transformation processing on datasets that show high performance without any transformation being performed; depending on the resource and time intensity of the additional transformation, the benefit achieved by such a transformation might not be worthwhile and, in many cases, might not be significant enough to affect the outcome of other measures and analyses. On the other hand, poor performance in pre-transformation analysis can be greatly improved through transformation in some cases, and multiple transformations might be necessary to achieve the best results.

F-measure

On an extremely imbalanced dataset, the accuracy rates cannot provide information on minority class (e.g. Attack). Therefore, the purpose of this section is to compare the PrivTra framework and the baseline methods according to the so popular F-measure which equally weights precision and recall. Table 5.3 shows the performance of the five classifiers on the transformed datasets. Again, it can be found that the transformed methods help the classifiers to achieve better performance than the original datasets, which is consistent with the previous observation. To focus on key trends, it can be seen from Table 5.3 that the degree of performance improvement of other transformed methods depends on the type of the classifiers. For instance, as shown in Table 5.3, the F-measure values for all datasets of the proposed PrivTra performs the best for SVM, NB and MLP classifiers. However, the only case result in a significant reduction in performance is with 7K-NN and J48 classifier due to the presence of irrelevant attributes and the binary data. Nevertheless, it is notable that the
baseline methods achieve superior performance with the MLP, 7K-NN and J48. The only case where there is a significant difference is the F-measure values of NB and SVM, suggesting that these algorithms suffer from the presence of continuous-valued attributes that are generated by the baseline methods.

Table 5.3: Comparing F-measure values of different classifiers using different transformation methods

<table>
<thead>
<tr>
<th>ML Tech</th>
<th>Method</th>
<th>MHIRD</th>
<th>MHORD</th>
<th>SPFDS</th>
<th>DOSDS</th>
<th>SPDOS</th>
<th>SHIRD</th>
<th>SHORD</th>
<th>ITD</th>
<th>WTP</th>
<th>DARPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>Original</td>
<td>95.74</td>
<td>94.79</td>
<td>98.55</td>
<td>94.20</td>
<td>98.31</td>
<td>96.36</td>
<td>95.13</td>
<td>86.01</td>
<td>88.92</td>
<td>58.24</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>96.07</td>
<td>94.31</td>
<td>96.99</td>
<td>92.43</td>
<td>95.26</td>
<td>98.20</td>
<td>96.54</td>
<td>83.27</td>
<td>77.10</td>
<td>33.71</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>96.00</td>
<td>94.21</td>
<td>97.83</td>
<td>93.78</td>
<td>98.78</td>
<td>96.81</td>
<td>95.40</td>
<td>87.47</td>
<td>88.35</td>
<td>58.02</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>97.71</td>
<td>98.76</td>
<td>98.50</td>
<td>97.03</td>
<td>97.19</td>
<td>98.95</td>
<td>98.15</td>
<td>93.06</td>
<td>97.94</td>
<td>44.95</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>96.95</td>
<td>93.99</td>
<td>98.42</td>
<td>93.97</td>
<td>97.24</td>
<td>97.15</td>
<td>95.57</td>
<td>87.58</td>
<td>87.68</td>
<td>58.02</td>
</tr>
<tr>
<td>J48</td>
<td>Original</td>
<td>97.87</td>
<td>89.91</td>
<td>98.82</td>
<td>47.66</td>
<td>97.72</td>
<td>96.21</td>
<td>96.61</td>
<td>93.57</td>
<td>79.73</td>
<td>70.65</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>94.74</td>
<td>94.97</td>
<td>91.18</td>
<td>47.80</td>
<td>92.25</td>
<td>96.11</td>
<td>95.17</td>
<td>87.78</td>
<td>74.37</td>
<td>48.63</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>97.25</td>
<td>83.33</td>
<td>98.69</td>
<td>46.96</td>
<td>98.52</td>
<td>97.41</td>
<td>97.25</td>
<td>92.91</td>
<td>79.74</td>
<td>76.63</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>46.52</td>
<td>74.88</td>
<td>98.44</td>
<td>47.68</td>
<td>97.65</td>
<td>98.49</td>
<td>97.44</td>
<td>92.69</td>
<td>66.06</td>
<td>66.74</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>97.62</td>
<td>83.71</td>
<td>98.12</td>
<td>46.83</td>
<td>98.42</td>
<td>97.10</td>
<td>96.50</td>
<td>93.69</td>
<td>80.01</td>
<td>76.63</td>
</tr>
<tr>
<td>NB</td>
<td>Original</td>
<td>96.80</td>
<td>79.93</td>
<td>92.32</td>
<td>72.69</td>
<td>92.11</td>
<td>97.48</td>
<td>93.24</td>
<td>85.92</td>
<td>83.22</td>
<td>71.99</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>97.42</td>
<td>86.94</td>
<td>97.27</td>
<td>68.75</td>
<td>87.41</td>
<td>96.61</td>
<td>92.26</td>
<td>50.26</td>
<td>82.78</td>
<td>50.02</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>98.30</td>
<td>74.21</td>
<td>91.96</td>
<td>72.90</td>
<td>91.91</td>
<td>98.33</td>
<td>92.57</td>
<td>86.36</td>
<td>83.66</td>
<td>76.81</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>98.65</td>
<td>97.79</td>
<td>97.84</td>
<td>89.11</td>
<td>95.11</td>
<td>97.58</td>
<td>98.08</td>
<td>93.69</td>
<td>97.84</td>
<td>85.78</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>96.93</td>
<td>74.52</td>
<td>92.26</td>
<td>73.43</td>
<td>91.84</td>
<td>98.28</td>
<td>92.15</td>
<td>78.76</td>
<td>85.21</td>
<td>76.23</td>
</tr>
<tr>
<td>7K-NN</td>
<td>Original</td>
<td>94.98</td>
<td>93.31</td>
<td>97.91</td>
<td>91.49</td>
<td>96.11</td>
<td>96.81</td>
<td>92.42</td>
<td>87.80</td>
<td>78.56</td>
<td>76.14</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>94.84</td>
<td>94.21</td>
<td>96.32</td>
<td>92.74</td>
<td>94.01</td>
<td>97.62</td>
<td>92.41</td>
<td>86.71</td>
<td>47.81</td>
<td>54.89</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>96.08</td>
<td>93.32</td>
<td>97.96</td>
<td>91.48</td>
<td>95.59</td>
<td>98.47</td>
<td>91.75</td>
<td>88.20</td>
<td>79.68</td>
<td>76.98</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>98.76</td>
<td>97.91</td>
<td>96.39</td>
<td>47.91</td>
<td>94.41</td>
<td>97.61</td>
<td>98.03</td>
<td>93.77</td>
<td>84.40</td>
<td>70.55</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>94.84</td>
<td>93.30</td>
<td>98.56</td>
<td>90.71</td>
<td>94.98</td>
<td>98.56</td>
<td>92.69</td>
<td>88.95</td>
<td>79.27</td>
<td>76.98</td>
</tr>
<tr>
<td>SVM</td>
<td>Original</td>
<td>93.10</td>
<td>83.86</td>
<td>98.15</td>
<td>46.94</td>
<td>95.33</td>
<td>83.11</td>
<td>91.96</td>
<td>62.58</td>
<td>87.13</td>
<td>84.47</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>91.55</td>
<td>83.82</td>
<td>97.22</td>
<td>47.93</td>
<td>87.51</td>
<td>82.73</td>
<td>90.56</td>
<td>37.63</td>
<td>74.78</td>
<td>63.99</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>93.27</td>
<td>84.36</td>
<td>98.61</td>
<td>47.18</td>
<td>96.41</td>
<td>82.56</td>
<td>90.31</td>
<td>60.01</td>
<td>88.15</td>
<td>85.07</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>97.53</td>
<td>98.96</td>
<td>97.36</td>
<td>90.81</td>
<td>97.42</td>
<td>97.33</td>
<td>97.97</td>
<td>93.37</td>
<td>97.93</td>
<td>86.94</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>93.20</td>
<td>84.58</td>
<td>98.14</td>
<td>47.10</td>
<td>94.95</td>
<td>81.88</td>
<td>90.52</td>
<td>58.73</td>
<td>88.61</td>
<td>85.07</td>
</tr>
</tbody>
</table>
5.4.7 Computational Efficiency

Here we present some results of the experiments we have performed in order to analyse the efficiency of the proposed PrivTra framework and the baseline methods. In particular, we have performed two series of tests. First, we focus our attention on assessing the time requirement to perform the transformation for different datasets, and then compare the performance of classifiers before and after applying the transformation methods on the original datasets. The computational efficiency test is conducted in Matlab v12 on a PC with 3.16 GHz CPU and 3.49 GB memory. For each set of experiments, ten trials were executed and the average value has been computed.

Efficiency of Transformation Methods

The objective of this test is to evaluate the runtime taken for transformation of the original data. The time requirements (of the proposed PrivTra and baseline methods presented in Table 5.4) have been evaluated in terms of CPU time. In the testing phase, the same working load of the system was ensured and the transformation process is measured in terms of CPU time (in milliseconds). Table 5.4 shows the time that was required to perform the transformation. Lower values indicate better performance and are shown in bold. From Table 5.4, we observe that SDP and RDP efficiently transform the original data faster than transformed data in comparison to the PCA-DR method and PrivTra framework. On the

<table>
<thead>
<tr>
<th>Data</th>
<th>PCA-DR</th>
<th>PrivTra</th>
<th>SDP</th>
<th>RDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>4.90</td>
<td>3.67</td>
<td>1.70</td>
<td>1.97</td>
</tr>
<tr>
<td>MHORD</td>
<td>4.72</td>
<td>3.54</td>
<td>1.69</td>
<td>2.73</td>
</tr>
<tr>
<td>SPFDS</td>
<td>5.23</td>
<td>3.85</td>
<td>1.59</td>
<td>2.28</td>
</tr>
<tr>
<td>DOSDS</td>
<td>6.95</td>
<td>5.90</td>
<td>2.07</td>
<td>2.96</td>
</tr>
<tr>
<td>SPDOS</td>
<td>12.07</td>
<td>10.08</td>
<td>3.64</td>
<td>3.91</td>
</tr>
<tr>
<td>SHIRD</td>
<td>4.44</td>
<td>3.32</td>
<td>1.60</td>
<td>2.63</td>
</tr>
<tr>
<td>SHORD</td>
<td>5.13</td>
<td>4.85</td>
<td>2.10</td>
<td>3.03</td>
</tr>
<tr>
<td>ITD</td>
<td>1100.98</td>
<td>1020.06</td>
<td>258.96</td>
<td>364.29</td>
</tr>
<tr>
<td>WTP</td>
<td>2.80</td>
<td>3.10</td>
<td>1.71</td>
<td>2.67</td>
</tr>
<tr>
<td>DARPA</td>
<td>105.80</td>
<td>98.05</td>
<td>31.75</td>
<td>33.57</td>
</tr>
</tbody>
</table>
other hand, it can be seen that the \textit{PrivTra} framework often performs the transformation process better than the PCA-DR method on most of the datasets except on two, these being WTP and ITD datasets. This can be explained by the fact that the dimensionality of these two datasets is higher than the other datasets. For instance, the dimensionality of WTP and ITD is 39 and 249, respectively. An improvement of such criteria would be a valuable contribution in future work.

\textbf{Efficiency of Transformed Data on Classifiers}

The aim of this test is to evaluate the runtime of the classifiers using the transformed data of the framework and the baseline methods. This is an important task since the supervised techniques (discussed in 5.4.3) consist of two stages: a model-building stage and a classification steps. The former stage uses the training data to build the classifier model, while the latter step uses the testing data to evaluate the generated model. In this chapter, we focus on the model-building stage due to its extremely time-consuming computations, and an accurate model needs to be retrained frequently.

Table 5.5 collects the results obtained for the classifiers’ performance over the transformed data by the proposed framework and the baseline methods. Lower values indicate better performance and are shown in bold. The table clearly shows that the transformed data improves the runtime of the classifiers compared with the original data. It can be observed in Table 5.5 that the outcomes of PCA-DR often outperform other methods on the classifier’s speed as well as on the independent dataset. This can be explained by the fact that PCA-DR often produces transformed data with less dimensionality than the original data. Note also that \textit{PrivTra} often proves to be the second best performing transformed method compared to SDP and RDP. Nevertheless, according to Table 5.5, it can be seen that the output of \textit{PrivTra} helps the NB technique to perform better than PCA-DR; this is because NB tends to be more efficient with discretized attributes.
### Table 5.5: Comparison of the performance of different classifiers based on transformed data

<table>
<thead>
<tr>
<th>ML Tech</th>
<th>Method</th>
<th>MHIRD</th>
<th>MHORD</th>
<th>DOSDS</th>
<th>SPFDS</th>
<th>SPADOS</th>
<th>SHIRD</th>
<th>SHORD</th>
<th>ITD</th>
<th>WTP</th>
<th>DARPA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MLP</strong></td>
<td>Original</td>
<td>2.37</td>
<td>1.99</td>
<td>2.04</td>
<td>1.78</td>
<td>3.06</td>
<td>1.90</td>
<td>2.22</td>
<td>15.42</td>
<td>2.69</td>
<td>1040.49</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>1.88</td>
<td>1.88</td>
<td>1.52</td>
<td>1.23</td>
<td>2.21</td>
<td>1.81</td>
<td>2.01</td>
<td>14.10</td>
<td>1.34</td>
<td>1038.32</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>1.98</td>
<td>1.98</td>
<td>1.97</td>
<td>1.63</td>
<td>3.10</td>
<td>1.85</td>
<td>2.04</td>
<td>15.49</td>
<td>2.52</td>
<td>386.26</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>1.92</td>
<td>1.94</td>
<td>2.00</td>
<td>1.62</td>
<td>3.04</td>
<td>1.82</td>
<td>2.05</td>
<td>15.30</td>
<td>2.54</td>
<td>383.42</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>1.95</td>
<td>1.93</td>
<td>2.23</td>
<td>1.76</td>
<td>3.06</td>
<td>1.92</td>
<td>2.09</td>
<td>15.43</td>
<td>2.53</td>
<td>386.94</td>
</tr>
<tr>
<td><strong>J48</strong></td>
<td>Original</td>
<td>1.40</td>
<td>1.34</td>
<td>1.17</td>
<td>1.06</td>
<td>1.53</td>
<td>1.32</td>
<td>1.41</td>
<td>3.02</td>
<td>0.80</td>
<td>24.14</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>1.31</td>
<td>1.32</td>
<td>1.09</td>
<td>0.98</td>
<td>1.49</td>
<td>1.28</td>
<td>1.38</td>
<td>2.69</td>
<td>0.76</td>
<td>23.51</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>1.32</td>
<td>1.31</td>
<td>1.07</td>
<td>0.97</td>
<td>1.46</td>
<td>1.25</td>
<td>1.35</td>
<td>2.74</td>
<td>0.78</td>
<td>24.40</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>1.30</td>
<td>1.31</td>
<td>1.08</td>
<td>0.95</td>
<td>1.44</td>
<td>1.29</td>
<td>1.36</td>
<td>3.46</td>
<td>0.80</td>
<td>22.82</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>1.29</td>
<td>1.40</td>
<td>1.08</td>
<td>0.96</td>
<td>1.44</td>
<td>1.28</td>
<td>1.42</td>
<td>2.74</td>
<td>0.88</td>
<td>23.76</td>
</tr>
<tr>
<td><strong>NB</strong></td>
<td>Original</td>
<td>1.49</td>
<td>1.37</td>
<td>1.15</td>
<td>1.10</td>
<td>1.54</td>
<td>1.31</td>
<td>1.41</td>
<td>3.22</td>
<td>0.79</td>
<td>40.49</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>1.35</td>
<td>1.32</td>
<td>1.11</td>
<td>0.99</td>
<td>1.49</td>
<td>1.30</td>
<td>1.37</td>
<td>3.08</td>
<td>0.74</td>
<td>41.23</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>1.36</td>
<td>1.34</td>
<td>1.09</td>
<td>1.00</td>
<td>1.47</td>
<td>1.25</td>
<td>1.37</td>
<td>3.19</td>
<td>0.72</td>
<td>41.92</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>1.34</td>
<td>1.31</td>
<td>1.16</td>
<td>0.98</td>
<td>1.50</td>
<td>1.28</td>
<td>1.35</td>
<td>3.28</td>
<td>0.73</td>
<td>43.32</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>1.30</td>
<td>1.34</td>
<td>1.19</td>
<td>1.03</td>
<td>1.48</td>
<td>1.26</td>
<td>1.35</td>
<td>3.05</td>
<td>0.71</td>
<td>42.26</td>
</tr>
<tr>
<td><strong>7K-NN</strong></td>
<td>Original</td>
<td>2.38</td>
<td>2.25</td>
<td>1.85</td>
<td>1.42</td>
<td>3.29</td>
<td>2.18</td>
<td>2.51</td>
<td>7.96</td>
<td>0.86</td>
<td>989.21</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>1.91</td>
<td>1.88</td>
<td>1.40</td>
<td>1.18</td>
<td>2.25</td>
<td>1.80</td>
<td>2.04</td>
<td>5.69</td>
<td>0.74</td>
<td>1005.80</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>2.31</td>
<td>2.29</td>
<td>1.71</td>
<td>1.37</td>
<td>3.36</td>
<td>2.16</td>
<td>2.52</td>
<td>15.23</td>
<td>0.77</td>
<td>1112.28</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>3.05</td>
<td>2.68</td>
<td>1.90</td>
<td>1.47</td>
<td>4.00</td>
<td>2.87</td>
<td>2.84</td>
<td>7.19</td>
<td>0.78</td>
<td>1767.42</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>2.29</td>
<td>2.26</td>
<td>1.69</td>
<td>1.35</td>
<td>3.24</td>
<td>2.11</td>
<td>2.46</td>
<td>14.85</td>
<td>0.78</td>
<td>1117.52</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>Original</td>
<td>1.38</td>
<td>1.37</td>
<td>1.17</td>
<td>1.04</td>
<td>1.50</td>
<td>1.37</td>
<td>1.53</td>
<td>3.58</td>
<td>0.77</td>
<td>26.27</td>
</tr>
<tr>
<td></td>
<td>PCA-DR</td>
<td>1.31</td>
<td>1.32</td>
<td>1.34</td>
<td>1.05</td>
<td>1.46</td>
<td>1.33</td>
<td>1.48</td>
<td>3.15</td>
<td>0.74</td>
<td>27.20</td>
</tr>
<tr>
<td></td>
<td>RDP</td>
<td>1.30</td>
<td>1.32</td>
<td>1.15</td>
<td>0.99</td>
<td>1.46</td>
<td>1.28</td>
<td>1.38</td>
<td>3.64</td>
<td>0.69</td>
<td>24.33</td>
</tr>
<tr>
<td></td>
<td>PrivTra</td>
<td>1.42</td>
<td>1.34</td>
<td>1.10</td>
<td>0.96</td>
<td>1.45</td>
<td>1.27</td>
<td>1.41</td>
<td>3.49</td>
<td>0.74</td>
<td>24.94</td>
</tr>
<tr>
<td></td>
<td>SDP</td>
<td>1.34</td>
<td>1.31</td>
<td>1.11</td>
<td>0.98</td>
<td>1.49</td>
<td>1.27</td>
<td>1.42</td>
<td>3.18</td>
<td>0.77</td>
<td>24.36</td>
</tr>
</tbody>
</table>
5.4.8 Scalability Test

In order to test the scalability of our framework in comparison to the PCA-DR, SDP and RDP methods, we applied each method to samples of traffic of increasing size from the DARPA traces using all the flows’ attributes (41 attributes). We then measured the execution time taken by PrivTra, PCA-DR, SDP and RDP to transform the traffic samples on a time-shared dual 3.16 GHz CPU with 3.49 GB memory, running on Windows HP.

Figure 5.6 illustrates the performance of the PrivTra, the PCA-DR, SDP and the RDP methods with traffic samples varying from approximately $10 \times 10^3$ to $60 \times 10^3$ traffic instances. The test averaged over 10 trials for each sample to give more accurate average execution times. For all datasets, the $k$ value for the PrivTra was set to 2 (Normal and Attack), except for the Internet traffic dataset (ITD) where the value of $k$ was set to 10. This is due to the large number of applications that presented in such dataset. Figure 5.6a shows that the SDP and the RDP methods have better performance than the PrivTra and PCA-DR. This is due to the simplicity of these two methods. Furthermore, PrivTra shows significant reduction of computational time in comparison to PCA-DR. The running time of the PrivTra framework did not depend only upon the number of instances, but also upon the dimensionality of the data. Consequently, we need to measure the scalability of our framework with an increasing number of attributes and compare it with existing ones. This is particularly important when the number of attributes is large. To do so, we run our experiments on the (ITD) dataset, since this data has the largest number of attributes. In general, we applied each method to samples of traffic with the number of attributes varying from approximately 5 to 249 attributes. We then measure the execution time of each method. The execution time on each sample is repeated 10 times. Figure 5.6b shows that again the time taken by SDP and the RDP methods to transform the data is much less than PCA-DR and PrivTra. However, we observe from Figure 5.6 that the two enhancements that can further improve the speed-up factor of PrivTra without changing the way the data transforms includes (i) using GPU environment and/or (ii) parallel computing.
CHAPTER 5. PRIVTRA: PRIVACY-PRESERVING FRAMEWORK FOR TRAFFIC DATA PUBLISHING

![Comparison of the scalability of the four transformation methods](image)

Figure 5.6: Comparison of the scalability of the four transformation methods
5.4.9 Quantifying Privacy

Quantifying the privacy of the proposed framework is a key motivation to satisfy the requirements presented in Section 5.2. In particular, the quantity used to measure privacy should indicate how closely the original value of an attribute can be estimated from transformed datasets. This section presents a comparison of the proposed framework with some existing ones, RDP, SDP [Oliveira and Zaiane, 2003] and PCA-DR [Vidya Banu and Nagaveni, 2013], in terms of quantifying the privacy. The data security index measures the privacy preservation level with respect to the variance of data before and after perturbation as follows:

\[ S = \frac{\text{Var}(X - Y)}{\text{Var}(X)} \]  

(5.5)

where \( X \) and \( Y \) represent the original data and the transformed data respectively. \( \text{Var}(X) \) is the variance of \( X \). In particular, the above measures the level of privacy preservation; thus, the larger the index value, the better the protection level.

Table 5.6 compares the level of privacy(%) of the proposed PrivTra framework with the results of baseline methods. The privacy level is computed as Eq. (5.5).

Table 5.6: Quantifying privacy of geometric data transformation

<table>
<thead>
<tr>
<th>Data</th>
<th>PCA-DR</th>
<th>PrivTra</th>
<th>RDP</th>
<th>SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>1.311</td>
<td>1.630</td>
<td>0.426</td>
<td>0.655</td>
</tr>
<tr>
<td>MHORD</td>
<td>0.997</td>
<td>1.334</td>
<td>0.324</td>
<td>0.498</td>
</tr>
<tr>
<td>SPFDS</td>
<td>1.786</td>
<td>2.144</td>
<td>0.581</td>
<td>0.893</td>
</tr>
<tr>
<td>DOSDS</td>
<td>0.391</td>
<td>1.524</td>
<td>0.127</td>
<td>0.196</td>
</tr>
<tr>
<td>SPDOS</td>
<td>0.428</td>
<td>2.539</td>
<td>0.139</td>
<td>0.214</td>
</tr>
<tr>
<td>SHIRD</td>
<td>0.376</td>
<td>9.486</td>
<td>0.122</td>
<td>0.188</td>
</tr>
<tr>
<td>SHORD</td>
<td>0.955</td>
<td>2.277</td>
<td>0.310</td>
<td>0.477</td>
</tr>
<tr>
<td>ITD</td>
<td>1.799</td>
<td>3.798</td>
<td>0.585</td>
<td>0.900</td>
</tr>
<tr>
<td>WTP</td>
<td>1.080</td>
<td>2.645</td>
<td>0.351</td>
<td>0.540</td>
</tr>
<tr>
<td>DARPA</td>
<td>3.659</td>
<td>8.476</td>
<td>0.214</td>
<td>0.330</td>
</tr>
</tbody>
</table>

(167) (January 9, 2015)
This table shows that the *PrivTra* framework outperforms all other methods in terms of a high privacy level on all datasets. RDP and SDP obtain a similar percentage of privacy level, which is lower than that of PCA-DR. Apart from the problem of the low level of privacy, these methods (RDP and SDP) are invertible. Thus, one may be able to estimate the real value of data easily.

In order to further explore the privacy level of the proposed *PrivTra* framework against the baseline methods, we performed a Friedman test [Friedman, 1940] followed by Nemenyi post-hoc test [Hollander et al., 2013]. The former test compares the proposed framework and the baselines methods over *N* datasets by ranking each method on each dataset separately. In particular, the transformation method with the best performance is ranked 1, the second best ranks 2, and so on. In case of ties, average ranks are assigned, then the average ranks of all methods on all datasets are calculated and compared. The latter test compares the privacy level in a pairwise manner. In particular, this test determines which method perform statistically different if the average ranks exceeds the critical difference \(CD_{\alpha} = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}}\), where the \(q_{\alpha}\) is calculated based on the studentized range statistic [Newman, 1939] divided by \(\sqrt{2}\).

Figure 5.7 shows that the privacy level of the *PrivTra* framework (based on Friedman test) has average ranks significantly different from both RDP and SDP methods. However, the privacy level of *PrivTra* framework outperforms the PCA-DR method in practice, we reject the hypothesis that there is a significant difference. This is due to the 95% confidence interval for the difference between these methods (under the \(P\)-value of 0.05) is \([-0.052, 2.52]\) which contains zero.
Figure 5.7: Comparison of privacy level for the four preserving privacy methods based on Friedman test.

Figure 5.8 shows the result of a posthoc Nemenyi test with $\alpha = 0.1$ on the ten datasets. The result indicates that privacy level of the PrivTra framework is statistically better than those of RDP and SDP. There is no clear evidence to indicate statistical privacy differences between PrivTra framework and PCA-DR method. This is because the average rank of the PCA-DR method does not exceed the critical difference $CD_\alpha$. Note, we perform the posthoc Nemenyi test with ($\alpha = 0.1$, 0.05 and 0.01) and the calculated statistical significance of the privacy level gives almost the same critical difference value. In general, the observations from Figure 5.7 and Figure 5.8 are consistent with the original findings in Table 5.6.

Figure 5.8: Privacy level comparison of all preserving privacy methods against each other based on Nemenyi test.
5.4.10 Discussion and Summary

As shown in Table 5.7, the PrivTra framework and the baselines methods are validated against the desired requirements (presented in Section 5.2) that need to be met by any privacy-preserving method. These requirements include four properties: the ability to deal with multivariate data, the capability to improve data utility, the need for efficiency, and the capability of enhancing privacy level. Table 5.7 shows that the proposed PrivTra framework has advantages over the related baseline methods. First, the PrivTra framework can deal with various types of attributes, while the other related methods can deal only with numerical attributes. Second, extensive experiments show that the PrivTra framework also improves the quality of data. However, our PrivTra framework is significantly better than the baseline methods, but it suffers from the computational time problem. Therefore, a promising future research direction would be to reduce the execution time of the PrivTra framework by using parallel computing such as multi-cores CPU or Graphics Processing Units (GPU).

Table 5.7: Compliance of the proposed PrivTra framework and the related methods to desirable requirements.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dealing with Multivariate Data</th>
<th>Efficiency Problem</th>
<th>Improve Data Quality</th>
<th>Privacy Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivTra</td>
<td>Yes</td>
<td>Suffer from</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>PCA-DR</td>
<td>No</td>
<td>Partially address</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>SDP</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Low</td>
</tr>
<tr>
<td>RDP</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Low</td>
</tr>
</tbody>
</table>

5.5 Conclusion

Sharing traffic data is significantly desirable to create accurate and global collaborative classifiers between spatial-domain networks. However, due to various privacy, security and legal reasons, collaborators avoid to reveal and publish their private traffic data publicly. In this chapter, we proposed a privacy-preserving framework which ensures the utility of published data and also satisfies the privacy requirements of traffic data. The key contribution of our scheme is the development of a privacy framework (namely PrivTra) that allows automated permutation to be made to multivariate network traffic data and attributes, including nu-
CHAPTER 5. PRIVTRA: PRIVACY-PRESERVING FRAMEWORK FOR TRAFFIC DATA PUBLISHING

Numerical attributes with real values, categorical attributes with unranked nominal values, and attributes with a hierarchical structure. Through experiments, we have demonstrated that our $\text{PrivTra}$ framework can effectively and efficiently render a balance between data utility and data privacy in comparison to baseline approaches. For future research, we will further enhance the performance of the $\text{PrivTra}$ framework and reduce its computational time by using more efficient techniques.
SemTra: A Semi-supervised Approach for Network Traffic labelling

As discussed in the previous two chapters, the recent promising studies for network classification have relied on the analysis of the statistics of traffic flows and the use of machine learning techniques. However, due to the high cost of manual labelling, it is hard to obtain sufficient, reliable and up-to-date labelled data for effective IP traffic classification. This chapter proposes a novel semi-supervised approach, called SemTra, which automatically alleviates the shortage of labelled flows for machine learning by exploiting the advantages of both supervised and unsupervised models. In particular, SemTra involves the following: (i) generating multi-view representations of the original data based on dimensionality reduction methods to have strong discrimination ability, (ii) incorporating the generated representations into the ensemble clustering model to provide a combined clustering output with better quality and stability, (iii) adapting the concept of self-training to iteratively utilize the few labelled data along with unlabelled within local and global viewpoints; and (iii) obtaining the final class decision by combining the decisions of mapping strategy of clusters, the local self-training and global self-training approaches. Extensive experiments were carried out to compare the effectiveness of SemTra over representative semi-supervised methods using sixteen network traffic datasets. The results clearly show that SemTra is able to yield noticeable improvement.
in accuracy (as high as 94.96%) and stability (as high as 95.04%) in the labelling process.

6.1 Introduction

The previously proposed LOA and GOA approaches (in Chapters 3 and 4) are designed from the concept of supervised learning which requires the traffic flows to be labelled in advance to improve the quality of Transport Layer Statistics (TLS), and to identify the optimal and stable feature set from the TLS data. Nevertheless, the traditional setting of supervised techniques (used in Chapter 5) also required a large amount of training traffic flows to be available in advance to construct a classifier model with a good generalization ability. It is noteworthy that these training flows should be labelled. That means the labels of such training flows should be known in advance. However, in practice, only a limited amount of labelled data is available because obtaining the labels for network flows requires payload data and human efforts. For example, a common practice for labelling network flows requires the payload data and therefore it is time-consuming as well as expensive to manually label the data. Nevertheless, due to privacy concerns, it is hard to release any payload data, thereby making it difficult to have efficient supervised techniques for a traffic classification task. Thus, the lack of labelled traffic data has motivated the use of unsupervised clustering algorithms for traffic classification. The key idea of such algorithms is to discover the structure in traffic data by automatically grouping a set of unlabelled instances to construct application-oriented traffic classifiers using the clustering results. However, traffic classifiers based on the concept of clustering algorithms suffer from a number of limitations, including the following: (i) setting the optimal number of clusters; (ii) obtaining high-purity traffic clusters; and (iii) mapping a large number of traffic clusters to a small number of real applications without prior knowledge (class label).

Recently, with the availability of unlabelled data and the difficulty of obtaining labelled ones, a limited number of semi-supervised learning approaches for network traffic classification were proposed (e.g. Erman et al. [Erman et al., 2007a] and Moore et al. [Rotsos et al., 2010]). Unfortunately, these approaches suffer from force assignments problem (as unlabelled flows must be belong to fixed traffic classes), scalability and novel class detection problems.
This chapter overcomes such limitations with an alternative semi-supervised approach for traffic flow labelling, termed as SemTra, which is based on incorporating the predictions of multiple unsupervised and supervised models for better predictions. In particular, the prediction information for unlabelled instances is derived from diversified and heterogenous models. The strength of one usually complements the weakness of the other, and thus maximizing the agreement among them can boost the performance of the labelling process.

The contributions made in this chapter can be summarised as follows:

- A multi-view approach that creates multiple representations of the original traffic data (termed as single-view data). This is particularly important since many interesting patterns cannot be extracted from a single view and also each representation may reveal different structures or views of the data. Also we incorporated two different distance metrics (including Euclidean distance and point symmetry distance) in a clustering algorithm to reveal the underlying data space structure from multiple data representations by considering both spherical and arbitrary shapes.

- Explore the concept of evidence accumulation clustering (EAC) [Fred and Jain, 2005] to combine the results of each clustering output on different representations. In particular, the evidence accumulation clustering would not directly generate a class label prediction for unlabelled instances, but it would provide useful constraints on the joint prediction for such unlabelled instances.

- Propose a new mapping strategy based on both the internal-structure of the cluster, as well as a probability function to improve the mapping process from a cluster to a class label. A novel local self-training approach is proposed to address the overlapping issue within a cluster and to iteratively predict the class label for unlabelled data from the local viewpoint.

- Propose a meta-level learning approach that combines the output of the initial clustering process on the multiple representations and the original attributes to form a global view. The resultant global view is subsequently fed to a global self-training method to iteratively predict the class label from the global viewpoint. Cascading the decision of
the clustering process enables local and global self-training to make better class label predictions.

The performance of the SemTra approach was extensively evaluated on sixteen traffic datasets, which included binary classes (e.g. Normal or Attack) and multi-classes (e.g. WWW, FTP, P2P). We also compared the performance of SemTra with some of the well-known semi-supervised methods, namely SemiBoost [Mallapragada et al., 2009], PGM [Rotsos et al., 2010], BGCM [Gao et al., 2013] and ORTSC [Erman et al., 2007a], using various metrics including accuracy, F-measure, stability and runtime performance. The experiment studies also include a statistical analysis based on nonparametric tests. Experimental results show that the proposed SemTra approach is, in most cases, more effective than the selected semi-supervised methods, with an average accuracy of 94.96% on the binary class dataset and 93.84% on the multi-class datasets.

The rest of the chapter is organized as follows. SemTra approach is discussed in detail in Section 6.2. In Section 6.3, we evaluate the performance of SemTra on different network traffic datasets. Finally, we conclude the chapter and discuss future research in Section 6.4.

### 6.2 The Proposed Semi-supervised Traffic Flow labelling

A more recent class of network traffic classification approaches developed using machine learning techniques, including ([Erman et al., 2007a] [Rotsos et al., 2010] [Rrushi et al., 2009] [Portnoy et al., 2001] [Auld et al., 2007]) and others, have became popular because of their high detection accuracy and efficiency. However, accurate ground truth creation is mandatory for proper training and evaluation. To produce accurate results, a huge and accurately labelled dataset is necessary, involving a tremendous workload and cost. SemTra is a new semi-supervised labelling approach to produce such amount of data without involving tedious labelling processes and achieve that goal at reasonable cost. Figure 6.1 provides an overview of SemTra, consisting of five modules: (1) representation extraction, (2) initial clustering analysis, (3) ensemble clustering analysis, (4.1) label propagation, (4.2) local self-training, (5) global self-training and (6) function agreement.
In the representation extraction module, to achieve diversity among traffic data, different representations are extracted by transforming raw traffic data to feature vectors of fixed dimensionality. In the initial clustering analysis module, a clustering algorithm is applied to different representations received from the representation extraction module. As a result, a different partition for a given dataset is generated based on each representation and different distance measurements. In the ensemble clustering analysis, all partitions previously obtained from applying the clustering analysis on different representations using different distances measurements are fed to the clustering ensemble module for the reconciliation to a final partition. The label propagation module infers each cluster’s class label by making full use of both labelled traffic data and internal structure information in each cluster. In the local self-training, the labelled data in each cluster are iteratively utilized along with unlabelled data to predict the class label. In the global self-training step, an adapted classifier-based technique has been developed to iteratively manipulate the whole dataset along with the cluster-association matrix (which summarizes the results of the initial clustering analysis) to predict the class label from a global perspective. In the agreement function, the decisions of
the label propagation and the local and global self-training steps are combined to generate the final class label for the unlabelled data.

6.2.1 Multi-view Layer

Multi-view learning is a proven approach to improve classification performance [Wang and Chen, 2009][Muslea et al., 2002]. However, this approach is not applicable to the process of traffic labelling. This is due to the fact that traffic data is represented just by a single view, also meaning that the data is represented by only one set of attributes (and therefore not properly separated into several distinct sets of attributes). To improve the performance of the labelling process, we propose a multi-view layer that provides multiple representations of the original data. This is based on the assumption that many interesting patterns cannot be extracted from a single view, and also each representation may reveal different structures of the data.

In the proposed multi-view layer, traffic data is processed with different heuristics for dimensionality reduction to act as representation methods. Employing dimensionality reduction to the labelling process for traffic data can also bring several advantages: (1) it reduces noise and correlation within the data; (2) it obtains 2D / 3D visualization for exploratory data analysis; (3) it reduces the space and time computational complexity of the classifier and clustering methods; and (4) it alleviates the problem of over-fitting by constructing combinations of the variables. Dimensionality reduction methods fall into two categories, namely global and local [Van der Maaten et al., 2009]. The former derive embeddings in which all points satisfy a given criterion, whereas the latter methods construct embeddings in which only local neighbourhoods are required to meet a given criterion. Consequently, the global methods tend to give a more “faithful” representation of the data’s global structure, and their metrics-preserving properties are better theoretically understood. Thus, we have chosen the three common global methods as suitable candidates for the proposed multi-view layer. These techniques include: Isomap [Teng et al., 2005], random projections (RP) [Bourgain, 1985] and kernel principle component analysis (KPCA) [Shawe-Taylor and Cristianini, 2004].
• **Isomap**: This approach is a well-known manifold method that guarantees a globally optimal solution. It preserves the intrinsic geometry of the nonlinear data by utilizing the geodesic manifold distances between data points. In particular, the process of creating a new view of traffic data is briefly summarized as follows:

  – Construct a neighborhood graph. In particular, two flows are considered to be neighbors if they satisfy a predefined condition, which states that either their distance in the original datasets is shorter than a constant, or one of the flows belongs to the $k$ nearest neighbors (KNN) of the other flows. Based on this neighborhood information, a weighted graph containing all flows in the datasets is built.

  – Compute the shortest path distances in the neighborhood graph using Dijkstra’s algorithm [Dijkstra, 1959]. The output of calculating the shortest paths is a matrix expressing the geodesic distances of each pair of flows.

  – Apply a classical MDS to the output matrix (obtained in the previous step) to construct an embedding of the data that best preserves the manifold’s estimated intrinsic geometry.

• **Random Projections (RP)**: This approach is one of the most powerful dimension reduction techniques that uses random projection matrices to project the data into lower dimensional spaces. In the following, we formally describe the steps of using the random projection approach to create a new view of the traffic data.

  – Transform the instances of flows in the datasets $X \in \mathbb{R}^p$ into a lower dimensional space $S \in \mathbb{R}^q$, where $q \ll p$, via

    $$S = RX$$ (6.1)

    The output is a matrix $G$ of size $q \times N$, where $q$ is the new dimensionality of the flows and $N$ is the size of the traffic dataset.

    $$G = (x_1 \ | x_2 | \cdots | x_N)$$ (6.2)
– Generate $j$ random matrices $\{R_i\}_{i=1}^j$, where $j$ is the number of desirable views. The two common ways of generating the random entries are:

1. The vectors are uniformly distributed over the $q$ dimensional unit sphere.
2. The elements of vectors are chosen from a Bernoulli $+1/-1$ distribution and the vectors are normalized

– Normalize the columns so that their $l_2$ norm will be 1.
– Obtain $\{T_i\}_{i=1}^j$ views by projecting the matrix $G$ onto the random matrices $\{R_i\}_{i=1}^j$, i.e. $T_i = R_i \cdot G$, where $i = 1, ..., j$.

• Kernel Principle Component Analysis (KPCA): This approach is a generalization of PCA, which is one of the primary statistical techniques for feature extraction and data modeling. This method utilizes the kernel trick to model the non-linear structures that exist in network traffic data. In particular, to perform KPCA on the network traffic data, the following steps have to be carried out:

– Choose a kernel mapping $K(x_m, x_n)$ (e.g. Gaussian, Sigmoid, and Polynomial).
– Obtain a matrix of $N \times N$ kernel elements (referred to as $K$) from the original traffic data.
– Get the eigenvectors $a_i = \left[a_1^{(i)}, \cdots, a_N^{(i)}\right]^T$, and the eigenvalues $\lambda_i$ of the covariance matrix in the feature space by solving the eigenvalue problem of $K$.
– Obtain the principal components of each given instance $X$ in the feature space as follows:
  \[
  (f(x) \phi_i) = \sum_{n=1}^{N} a_n^{(i)} k(x, x_n)
  \]  
  \[ (6.3) \]
– Keep only a small number of principal components corresponding to the largest eigenvalues without losing much information by applying a regular PCA.

However, from the discussion on the design idea of the multi-view layer, the dimensionality reduction techniques which could be used to create multiple representations (of the original traffic data) should not be limited to what we discussed.
CHAPTER 6. SEMTRA: A SEMI-SUPERVISED APPROACH FOR NETWORK TRAFFIC LABELLING

The process of creating a multi-view of the original traffic data is summarized in Algorithm 9. Let $N_{ij} = \{(x_1, y_1), (x_2, y_2), (x_n, y_n)\}$ be a set of $n$ flow instances. Each $x_i = \{x_{ij}\}_1^m$, where $m$ is the number of features and $x_{ij}$ is the value of the $j$th feature for the $i$th flow instance. The process starts by discarding the label class from the original data before passing on to view type which is based on dimensionality reduction techniques. Then, project the original data into lower dimensional spaces ($\mathbb{R}^q$ where $q \ll n$) based on each technique. Their output is a set of column vectors in the lower dimensional space. Figure 6.2 illustrates how

\begin{algorithm}
\caption{Processing of generating multi-view layer}
\begin{algorithmic}[1]
\State \textbf{input :}
\State $Data(N_{ij}) \leftarrow \{f_1, f_2, \ldots, f_{n-1}\}$;
\State $ViewType \leftarrow \{Iso, RP, KPCA\}$;
\end{algorithmic}
\begin{algorithmic}[1]
\State \textbf{output:}
\State $View_{ij}$ // multi-view data \label{multi-view}
\If{Label is Removed}
\Switch{View Type}
\Case{Isomap}
\State $[NM_{Iso}] \leftarrow \text{IsomapView}([N_{ij}])$;
\EndCase
\Case{RP}
\State $[NM_{RP}] \leftarrow \text{RPView}([N_{ij}])$;
\EndCase
\Case{KPCA}
\State $[NM_{KPCA}] \leftarrow \text{KPCAView}([N_{ij}])$;
\EndCase
\EndSwitch
\State $[View_{ij}] \leftarrow ([NM_{Iso}],[NM_{RP}],[NM_{KPCA}])$;
\EndIf
\end{algorithmic}
\end{algorithm}

the proposed multi-view layer produces very distinct representations of the original DOSDS dataset [Almalawi et al., 2013b]. A single representation simply captures partial structural information and thus the joint use of the different representation is more likely to capture the intrinsic structure of the given traffic data. An open issue for research is how to discover a consensus partition of all representations that is superior to any single representation. To address this problem, our multi-view layer exploits the complementary information in different views of network data, as discussed in Section 6.2.3.
Figure 6.2: Different views of the traffic data using the multi-view layer
6.2.2 Initial Clustering Analysis

In order to achieve better class separation, we opted to bring in clustering to partitioned traffic data into multiple clusters. Clustering is the process of partitioning a dataset into multiple groups, where each group contains data patterns that are very similar in terms of specific distance measurements. Based on the different data representations with lower dimensionality, any existing clustering algorithm is applicable to efficiently cluster the data. Among the various clustering algorithms, we have chosen to use K-means clustering [Hartigan and Wong, 1979] for the following reasons: 1) it is a data-driven method with relatively a few assumptions on the distribution of the underlying data; and 2) it guarantees a local minimum of the criterion function, thereby accelerating the convergence of clusters on large datasets. The major limitation, however, is its inability to identify clusters with arbitrary shapes, ultimately imposing hyperspherical-shaped clusters on the data.

In the first stage, K-means clustering is performed on the training instances to obtain \( k \) disjoint clusters. In an ideal situation, each K-means cluster represents a region of similar instances, “similar” in terms of distances between the instances and their centroid. This basic K-means clusters works well when the flows (applications) conform to the assumptions and the procedure of K-means. The assumptions are that the flows are generated by a mixture model, and that there is a correspondence between mixture components and classes. However, these assumptions are often violated in practice and can result in poor performance due to high similarity between different patterns. Since these assumptions hold in traffic data, our approach achieves diversity among patterns by exploiting both different distance metrics as well as different number of clusters \( k \). In particular, the Euclidean distance [Su and Chou, 2001] and point symmetry distance [Su and Chou, 2001] are considered. The Euclidean distance is one of the most common forms of the general Minkowski distance metrics, which measures dissimilarity, and helps the conventional K-means algorithm to detect hyperspherical-shaped clusters. Given \( N \) patterns \( x_i = (x_{i1}, \cdots, x_{in})^T \), where \( i = \{1 \cdots N\} \), the Euclidean distance metric for measuring the dissimilarity between the \( j^{th} \) and \( k^{th} \) patterns
is defined by:

\[ d(j, k) = \left( \sum_{i=1}^{n} |x_{ji} - x_{ki}|^2 \right)^{\frac{1}{2}} \]  

(6.4)

noteworthy that the smaller value of \( d(j, k) \) is, the greater the similarity will be. By using Euclidean distance as a measure of similarity, hyperspherical-shaped clusters of equal size are usually detected [Su and Chou, 2000].

On the other hand, to take care of clusters with arbitrary shapes and size, and for situations where there is no a priori information about the geometric characteristics of a traffic dataset, it is necessary to consider another more flexible measure. A non-metric distance based on the concept of “point symmetry” is applied to the K-means algorithm. The point symmetry distance is defined as follows: Given \( N \) patterns \( x_j, i = \{1 \cdots N\} \), and reference vector \( c \) (e.g. a cluster centroid), the point symmetry distance between a pattern \( x_j \) and the reference vector \( c \) is defined as:

\[ d_s(x_j, c) = \min \left\| \frac{(x_j - c) + (x_i - c)}{(x_j - c) + (x_i - c)} \right\| \]  

(6.5)

The denominator normalizes the point symmetry distance and makes it insensitive to Euclidean distances. In particular, \( d_s(x_j, c) \) is minimized when the pattern \( x_i = 2c - x_j \) exists in the dataset (i.e. \( d_s(x_j; c) = 0 \)).

In general, by using different distance metrics, the vicinities identified for a given flow can be different, even when the same \( k \) is used; whereas by using different \( k \) values, the predictions for a given flow also can be different, even when the same distance metric is used. Thus, K-means may be able to somewhat diversify the network flows by partitioning them with different distance metrics and/or different \( k \) values. Such a setting can also bring another advantage, that is, since it is usually difficult to decide which distance metric and \( k \) value are better for the labelling task. Inspired by the work in sensor fusion and classifier combination [Gao et al., 2013][Li et al., 2012][Verma and Rahman, 2012], the functions of such distance metrics can be combined to explore distinct views of flow relationships. Therefore, the K-means clustering algorithm takes each data representation \( X \) as input and organizes the \( n \) flows into \( k \) clusters according to different distance metrics and may produce different
partitions for the same dataset, either in terms of cluster membership and/or the number of clusters produced.

An open issue for research is to find the final consensus partition (from the different K-means runs on different representations) in a computationally efficient manner. To address this problem, we consider the use of a consensus function in Section 6.2.3.

6.2.3 Ensemble Clustering

In the area of machine learning, combining several self-contained predicting algorithms (into an ensemble to yield better performance in terms of accuracy than any of the base predictors) is backed by a sound theoretical background [Zhou and Li, 2007][Iam-On et al., 2012][Strehl and Ghosh, 2003]. Thus, with the aim of achieving accurate results superior to that of any individual clustering, in this section, we propose a cluster ensemble approach to combine the partitions of different clustering algorithms based on each representation to produce a consensus partition.

Figure 6.3 shows a general picture of the cluster ensemble framework. Essentially, all segment representations received from the representation extracting module (discussed in Section 6.2.1) are fed to the ensemble clustering-based module. As a result, multiple partitions for a given dataset are generated based on each representation as well as on the various clustering settings. Then, producing the final solution can simply be achieved by aggregating all partitions of the base clusterings according to a consensus function. In the following subsections, we discuss in detail the various steps of the proposed framework of the cluster ensemble.
The consensus function is concerned with finding a consensus partition that improves the performance of weak clustering algorithms. Although there is a wide variety of consensus functions in the literature [Ayad and Kamel, 2007][Vega-Pons and Ruiz-Shulcloper, 2011], we have explored the EAC (Evidence Accumulation Clustering) [Fred and Jain, 2005] method as the optimal consensus function to extract the final partition by combining the results of the different data representations and multiple-distance metrics. This method combines the results of multiple-distance metrics, which are obtained from the initial clustering step, into a single data partition by viewing each distance metric result as independent evidence of data organization. In particular, each result of K-means clustering is mapped to co-occurrences, where entries can be interpreted as votes on the pairwise co-occurrences of flows and are computed as the number of times each pair of flows co-occurs in the same cluster. The underlying assumption is that flows belonging to a “natural” cluster are very likely to be allocated to the same cluster regardless of what data representation or distance measure are used. Formally, the value of each co-occurrence matrix entry can calculated as follows:

$$C(x_i, x_j) = \frac{1}{m} \sum_{t=1}^{m} \delta(P_t(x_i), P_t(x_j))$$  \hspace{1cm} (6.6)
CHAPTER 6. SEMTRA: A SEMI-SUPERVISED APPROACH FOR NETWORK TRAFFIC LABELLING

\[
\delta(P_t(x_i), P_t(x_j)) = \begin{cases} 
1 & \text{if } P(x_i) = P(x_j) \\
0 & \text{otherwise}
\end{cases}
\]  

(6.7)

where \(C(x_i, x_j)\) donates the number of times the pairs of flows \((x_i, x_j)\) appear in the same cluster among \(m\) partitions, and \(\delta(P_t(x_i), P_t(x_j)) = 1 \text{ if } x_i = x_j\), and \(0\) otherwise. The final consensus clustering is extracted by applying the graph-based METIS algorithm [Karypis and Kumar, 1999] on the co-association matrix. The main principle of this algorithm is to minimize the edge cutting by making the weight of vertex distributed evenly among different regions. Thus, it is expected that obtaining the final partition in such a way will better explain the natural grouping of the traffic data.

Cluster Mapping Process

The final step of our ensemble clustering approach is the mapping phase. We aim to map descriptive labels to clusters that reflect their contents. This step is very important to determine the label associated with each flow in the final partition. Unfortunately, estimating the labelling confidence in ensemble clustering is not as straightforward as in classification. This is because the labelling confidence for classification can be estimated directly by checking the probability of the unlabelled flows being labelled to a finite number of classes; whereas the process of ensemble clustering tends to group highly correlated traffic flows to geometrically remain very close. Consequently, the underlying challenge is to classify such flows, especially when patterns from multiple classes overlap within a cluster.

We propose here a mapping mechanism to estimate the genuine class label of unlabelled flows. In general, the performance of the proposed mapping process depends on the content of the cluster. If all the flows in a cluster belong to the same class, then the mapping is unique and is simply mapped to the correspondent class. We refer to these clusters as atomic clusters [Verma and Rahman, 2012]. Nonatomic clusters, however, are comprised of different classes, and these are labelled according to the proposed mapping strategy, which depends on the internal structure of cluster and the probabilistic value of the labelled flows in the cluster.

In particular, for the internal structure of a cluster, we assume an important property
of each class: that instances belonging to the same class should be closer to each other (referred to as *cohesion*) and also should be apart from the instances that belong to other classes (referred to as *separation*) [Masud et al., 2013]. We generalized this assumption by introducing the concept of cohesion of cluster.

**Definition 8 (Cohesion of Cluster)** Instances within class \( y_i \) are said to be cohesive if they have minimum mean distance \( \overline{D}_{i_{\min}} \) among all \( \overline{D}_i \). The mean distance \( \overline{D}_i \) is defined as:

\[
\overline{D}_i = \frac{1}{|\Omega_{i_{y_i}}^C| |\Gamma_{u_i}^C|} \sum_{l=1}^{|\Omega_{i_{y_i}}^C|} \sum_{u=1}^{|\Gamma_{u_i}^C|} D(x_l, x_u)
\]  

(6.8)

where \( \Omega_{i_{y_i}}^C \) is a set of label instances of class \( y_i \) in cluster \( C \), and \( \Gamma_{u_i}^C \) is a set of unlabelled instances in cluster \( C \). \( D(x_l, x_u) \) is the distance between the labelled instance \( x_l \) and unlabelled instance \( x_u \) in some appropriate metric. For a fair comparison between cohesion of different classes with unlabelled instances within a cluster, all cohesion factors should have the same scale. To do so, we define its weighted factor.

**Definition 9 (Weighted Cohesion Factor - WCF)** The WCF of class \( y_i \) in cluster \( C \) is defined as

\[
WCF = 1 - \frac{\overline{D}_i}{\sum_{i=1}^{q} \overline{D}_i}
\]

(6.9)

It is obvious that the value of WCF close to 1 implies a high level of cohesion between instances labelled with class \( y_i \) and unlabelled instances \( \Gamma_{u_i}^C \), and a value close to 0 implies a low level of cohesion. However, knowing the WCF value does not necessarily reflect the actual class of a cluster. Therefore, we need to utilize a probabilistic measurement \( P(\omega_{lk}) \) to find a mapping from a cluster to an actual class label.

Let \( \Omega_{i_{y_i}}^k \) be the subset of labelled instances representing all labelled instances that appear in cluster \( C_k \):

\[
\Omega_{i_{y_i}}^k = \left\{ y | y \in \Omega_{i_{y_i}}^k, F_{y_i} > 0 \right\} = \bigcup_{i=1}^{n} y_i^k, \Omega_{i_{y_i}}^k \neq 0
\]

(6.10)

where \( F_{y_i} \) is the number of assurances (frequency) of class \( y_i \) in cluster \( C_k \). Let \( N \) denote
the total number of observed labelled instances in cluster $C^k$:

$$N = \sum_{y \in \Omega^k} F_y = \sum_{i=1}^{n} |y^k_i|, N \in \mathbb{N}, N \geq n. \quad (6.11)$$

**Definition 10 (Probability measurement)**  The probability of the labelled instances for a given class $y_i$ in cluster $k$ is defined as:

$$P(\omega_{1k}) = \frac{F_{y_i}}{N} \quad (6.12)$$

Finally, the decision function for assigning a class label to a cluster is based on the maximum decision function of both the weighted cohesion factor (WCF) and the probabilistic value $P_{\omega_{1k}}$. This is particularly important since reliance on any single measurement may lead to misleading conclusions. The decision function for assigning class label to a cluster is defined as follows:

$$C^k_i = \arg \max \left( P(\omega_{1k}) \times \left[ 1 - \frac{D_i}{\sum_{l=1}^{s[k]} D_l} \right] \right) \quad (6.13)$$

WCF penalizes the probability of instances $\Omega_l$ labelled to a particular class $y_i$ in cluster $s$ with its mean distance from the unlabelled instances $\Gamma_u$, i.e. a high value of $D_i$ yields a low MP$_i$ score and vice versa.

To this end, if a cluster $C^k$ is given a class label with respect to the decision function defined in (6.13), each of its unlabelled instances can automatically inherit the same class label. To complete the mapping, clusters that do not have any labelled instances assigned to them should have a different class label from any of the existing classes and be named *novel class*.

Figure 6.4 illustrates the type and the proportion of the clusters with respect to the number of clusters on both binary class and multi-class traffic datasets. A general observation from Figure 6.4, indicates that even though setting the number of cluster to a large $k$ can increase the portion of high-purity traffic clusters (referred to as atomic clusters), there is still a large proportion of non-atomic and novel clusters requiring further investigation.
Figure 6.4: Portion of clusters types (e.g. atomic, non-atomic and novel clusters) based on the number of clusters
CHAPTER 6. SEMTRA: A SEMI-SUPERVISED APPROACH FOR NETWORK TRAFFIC LABELLING

6.2.4 Local Self-training

In the previous phase, the proposed mapping strategy ensures that each training flow is associated with only one cluster. However, the traffic datasets may contain overlapping patterns from different classes, resulting in subgroups or overlap within clusters. However, identifying the class boundaries between overlapping class patterns within clusters is a difficult problem. To handle such a challenge, the second phase of SemTra refines the decision boundaries for each class by utilizing the available labelled and unlabelled flows in each cluster. In particular, the proposed approach learns the accurate decision boundaries and iteratively improves the labelling process. At each iteration, the labelled flows in each cluster are selected to train the given classification learning algorithm. The trained classification model will produce $k$ scores, indicating the likelihood that the unlabelled flow belongs to each of the $k$ traffic classes. Unlabelled flow then must be assigned to the traffic class with the highest score. The assigned labels are hereafter referred to as pseudo-labels. The labelled flows along with the selected pseudo-labels flows are utilized in the next iteration to re-train the classifier, and the process is repeated.

Choice of Supervised Learning Algorithms

In order to strike a good balance between accuracy and scalability, our approach finds a discriminative decision boundary in the nonatomic clusters [Verma and Rahman, 2012] (that builds on the idea of the Support Vector Machine SVM algorithm [Vapnik, 2000]). We have chosen the SVM algorithm as a suitable candidate for the following reasons:

- It is scalable to very large date sets.
- It is highly accurate, owing to its ability to model complex nonlinear decision boundaries, which can be generalized to overlapping and fuzzy patterns.
- It has a small number of user-tunable parameters.

In practice, the SVM algorithm transforms training data into a higher dimension. Within this new dimension, it then searches for the linear optimal decision boundary (which is referred to...
as a hyperplane). The SVM uses both the support vector and the margin to find a hyperplane that separates objects of one class from objects of other classes at the maximal margin. To this end, the SVM finds the maximum marginal hyperplane using the following objective function:

$$\min_{w,b,\xi} = C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|w\|^2$$

(6.14)

subject to

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1 \cdots N$$

where \(x_i\) belongs to the training instances \(D_l\), \(\xi_i\) is a slack variable for labelled data, and \(C\) is a constant parameter used as a trade-off between the two goals of learning minimization of error and maximization of margin. Note that the SVM algorithm is used for classification and prediction, which means that the training flows need to be labelled first. Consequently, we need to incorporate pseudo-labels flows with the labelled flows. To do so, we modify the objective functions of SVM algorithm as follows:

$$\min_{w,b,\xi} = C \left( \sum_{i=1}^{n} \xi_i + \lambda_1 \sum_{j=1}^{m} \xi_j \right) + \frac{1}{2} \|w\|^2$$

(6.15)

subject to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad (labelled \ data)$$

$$\lambda_1 y_j(w \cdot x_j + b) \geq 1 - \xi_j \quad (labelled \ data)$$

$$\xi_i \geq 0, i = 1, \cdots, n, \ \xi_j \geq 0, j = 1, \cdots, m$$

where \(x_j\) belongs to the pseudo-labels flows \(D_{p2}\) \(D_{p2} \subset D_u\); \(\xi_j\) is a slack variable for pseudo-labels flows, and \(\lambda_1 \in [0, 1]\) is a weighting parameter to regulate the impact of labelled flows and pseudo-labels flows. Hence, the modified objective functions are still convex, avoiding the problem of local maxima (minima).

However, to efficiently solve the optimization problem of the modified objective function, we used the Gauss-Seidel/smo method introduced in [Collobert et al., 2006] to calculate its dual. In particular, the iterative process of the self-training utilized the modified objective
function to gradually increase the amount of pseudo-labels flows, which is expected to play an important role in the labelling of new data. Therefore, we do not re-weight the labelled and pseudo-labels data in the objective functions. Furthermore, we define a hybrid objective function to estimate the optimum values of $\lambda_1$ during local self-training.

The following steps are taken by SemTra in the second phase to assign class labels to unlabelled flows in each nonatomic cluster. Let $X = (x_1, \cdots, x_n) \in \mathbb{R}^{n \times d}$ denote the entire dataset, which consists of $n$ flow instances in $d$-dimensional space. The dataset includes both the labelled data $X_l = (x_l, y_l)_{l=1}^{L} \in X \times Y$, (where $y_l \in 0, 1$, representing a binary classification decision of “0” to indicate normal and “1” to indicate anomaly class assignments) and unlabelled data $X_u = (x_u)_{u=L+1}^{L+U} \in X$.

• **Step 1.** Train the standard SVM algorithm using $X_l$, and perform prediction on $X_u$. The output of this step would be the predicted labels of the unlabelled flows being labelled to different classes $[y^{(1)}(1), \ldots, y^{(1)}(L_n)]$

• **Step 2.** Estimate the initial labelling confidence by consulting the probabilities of the predicted labels, obtained in the previous step. For example, suppose the probability of the flow $a$ being classified to class $l_1$ and $l_2$ is 0.95 and 0.05, respectively, whereas that of the flow $b$ is 0.60 and 0.40, respectively. Then, the flow $a$ is more likely to be labelled to class $l_1$. The assigned labels are hereafter referred to as the pseudo-labels and denoted as $X_s$.

• **Step 3.** Utilize the labelled flows along with pseudo-labels flows to produce a new augmented training data $X^k_l = X_l + X_s$, and remove the pseudo-labels from the unlabelled set $X^k_u = X_u - X_s$.

• **Step 4.** Use the augmented training data $X^k_l = X_l + X_s$ for training the SVM in the next iteration $(k)$, resulting in a new classifier model which can be used to perform prediction on $X^k_u = X_u - X_s$.

• **Step 5.** Calculate the modified objective function value in Eq. 6.16 after using the
augmented training data $X_l^k$ as follows:

$$f(w^{(k)}, \xi^{(k)}) = C \left( \sum_{i=1}^{n} \xi_i^{(k)} + \lambda_1 \sum_{j=1}^{m} \xi_j^{(k)} \right) + \frac{1}{2} \left\| w^{(k)} \right\|^2$$  \hspace{1cm} (6.16)

- **Step 6.** Repeat the steps (2-5) until the stopping criterion determines when the process should stop. Here, a predefined stopping criterion $\delta_0$ terminates the process if the value of the objective function in the current iteration $k$ is worse than the value of the objective function in the previous iteration $k-1$; otherwise, the algorithm continues to expand the labelled data at the next iteration. Consequently, at the end of the $(k+1)$ iteration, the output would be the predicted labelled $[y^{(k+1)}(1), ..., y^{(k+1)}(l_n)]$.

From the local point of view, data instances in the same cluster can be very similar. However, due to the overlapping issue existing within a cluster, unlabelled instances may have different class labels with respect to the global point of view (whole datasets). Thus, this issue will be addressed in the following section.

### 6.2.5 Global Self-training on Meta-level Features

In the previous section, it was explained how the local self-training phase can predict the class label based on the local information in each cluster. However, this phase might not fully exploit the distribution of unlabelled instances in the context of the whole dataset (i.e. by building the classification model only from instances in each cluster, while ignoring instances in other clusters). With the aim of improving the labelling process, this section proposes new extended meta-level features to represent the global information as well as global self-training to learn at the meta level. The extended meta-level features are comprised of two parts: the first part represents the original features, whereas the second part represents the output of each cluster on each representation that is subsequently considered as a new attribute. Consequently, the labelled instances in the extended meta-level set would be represented as follows:

$$\Gamma_l = \{[F_1(X_l), \ldots, F_N(X_l), C_1(X_l), \ldots, C_M(X_l), Y_l]\}_{i=1}^L$$  \hspace{1cm} (6.17)
where $L$ stands for the number of instances, $F_1(X_l) \cdots F_N$ represents the original features of labelled instances $X_l$, and $C_1(X_l) \cdots C_M(X_l)$ is the output of each cluster on different data representations of labelled instances $X_l$. The unlabelled instances would be represented as follows:

$$\Gamma_u = \{[F_1(X_u), \cdots, F_N(X_u), C_1(X_u), \cdots, C_M(X_u)]\}^{L+U}_{u=L+1}$$

where $L + U$ stands for the number of unlabelled instances, $F_1(X_u) \cdots F_N(X_u)$ represents original features of unlabelled instances $X_u$, and $C_1(X_u) \cdots C_M(X_u)$ is the output of each cluster on different data representations of unlabelled instances $X_u$.

The extended meta-level features are applied as a training set for the global self-training process. Given that the global self-training step tries to use unlabelled instances in the meta-level to adjust the decision boundary learned from the small number of labelled instances, such that it goes through the less dense region while keeping the labelled data correctly classified. In particular, the global self-training initially initiates an SVM using labelled instances and assigns potential labels to unlabelled instances. Then, it iteratively maximizes the margin for both labelled and unlabelled instances with their potential labels by flipping the labels of the unlabelled instances on different sides of the decision boundary. An optimal prediction is reached when the decision boundary not only classifies the labelled instances as accurately as possible, but also avoids going through the high-density region.

### 6.2.6 Function Agreement and Labelling

Algorithm 10 sketches the labelling process of SemTra. During this process, the unlabelled flows must be assigned labels following two main criteria: (i) unlabelled flows with high similarity must share the same label; and (ii) those unlabelled flows which are highly similar to a labelled flow must share its label. Thus, to follow these two criteria, we assume that all the unlabelled instances (in atomic clusters) belong to the same class, so the algorithm (see Lines 5-11) immediately assigns the class label of instances with its corresponding cluster’s label, and then updates the labelled and unlabelled instances ($D_L$, $D_{UL}$). Lines 9-11 of Algorithm 10 check whether clusters have only unlabelled instances and do not have any
labelled instances assigned to them. If so, the algorithm defines these unlabelled instances in such clusters as **novel class**, and then updates the labelled and unlabelled instances \((D_L, D_{UL})\). However, the remaining unlabelled instances do not necessarily match any of previously mentioned criteria. This implies that we may need additional guidance to improve the labelling process by combining the decision of the first phase (Cluster Mapping Process) along with the second phase (Local self-training) and the third phase (Global self-training) to give a final decision on the class membership of unlabelled flows. Since the prediction of these three phases is derived from diversified and heterogenous sources, the consensus combi-

---

**Algorithm 10: Function of Agreement and Labelling.**

```
input:
1. \(D_L = \{(x_1, y_1), \ldots, (x_l, y_l)\}\), \(D_{UL} = \{x_{l+1}, \ldots, x_n\}\);
2. \(\text{Iter} = 50, \beta = 2, \vartheta = \text{Iter} - 1\);
output:
3. \([D_L] = [x_{ij}] : \text{class}_i\); // Augmented labelled data

foreach \(i \in [1, \text{Iter}]\) do
  5. \(\text{MP} \leftarrow \text{ApplyMP}(D_{UL});\) // mapping function
  6. if \(D_{UL} \in \text{atomic}\) then
     7. \(D_L \leftarrow D_L \cup \text{atomic}(X);\)
     8. \(D_{UL} \leftarrow D_{UL} \setminus \text{atomic}(X);\)
   9. if \(D_{UL} \in \text{Novel}\) then
      10. \(D_L \leftarrow D_L \cup \text{Novel}(X);\)
      11. \(D_{UL} \leftarrow D_{UL} \setminus \text{Novel}(X);\)
   12. \(\text{LST} \leftarrow \text{ApplyLST}(D_{UL});\) // Local self-training
   13. \(\text{GST} \leftarrow \text{ApplyLST}(D_{UL});\) // Global self-training
   14. for \(x \in D_{UL}\) do
      15. \(\text{majority-voting} \leftarrow \text{ComputeVote(MP, LST, GST, x);}\)
      16. if \(\text{majority-voting} \geq \beta\) then
         17. \(D_L \leftarrow D_L \cup x;\)
      18. else
         19. \(\text{tempList} \leftarrow \text{tempList} \cup x;\)
         20. \(\text{tempCount} \leftarrow \text{Count(tempList);}\)
         21. if \(\text{tempCount}(x) \geq \vartheta\) then
            22. \(\text{outlierList} \leftarrow \text{outlierList} \cup x;\)
               // Add to outlier List
         23. \(\text{tempList} \leftarrow \text{tempList} \setminus \text{outlierList};\)
      24. \(D_{UL} \leftarrow \text{tempList;}\)
```
nation can significantly boost the performance by producing more accurate results. In Lines 12-24 of Algorithm 10, a class label is assigned to an unlabelled instance by taking majority votes among the three different phases. Otherwise, if all phases did not have agreement on the predicted class, Algorithm 10 would consider such instances as outliers and remove them from the dataset.

6.3 Experimental Evaluation

This section demonstrates the effectiveness of the proposed SemTra approach by conducting extensive experiments on a number of benchmark datasets covering traditional computer network traffic datasets and industrial network (SCADA) datasets. In particular, the objective of this section is to investigate the effectiveness of the proposed SemTra in generating accurate and stable class labels for unlabelled flows on both binary-class multi-class datasets.

6.3.1 Datasets Used in Experiments

To illustrate the broad applicability of the proposed SemTra approach, sixteen traffic datasets are used in the experiments, including: Multi-hop Outdoor Real Data (MHORD) [Suthaharan et al., 2010], Multi-hop Indoor Real Data (MHIRD) [Suthaharan et al., 2010], Single-hop Outdoor Real Data (SHORD) [Suthaharan et al., 2010], Single-hop Indoor Real Data (SHIRD) [Suthaharan et al., 2010], simulated spoofing attack for SCADA system (detonated as SPFDS) [Almalawi et al., 2013b], simulated denial of service attack DOS for SCADA system (detonated as DOSDS) [Almalawi et al., 2013b], simulated both spoofing and attacks for SCADA system (detonated as SPDOS) [Almalawi et al., 2013b], and the operational state of a water treatment plant (WTP). We experimented also with two other publicly-available datasets, namely DARPA [Stolfo et al., 2000] and internet traffic data (ITD) [Moore et al., 2003]. For ITD datasets, different number of datasets have been collected from different periods of time and from different networks, and thus we refer to them as ITD1, ITD2, etc. Table 6.1 summarizes the proportion of normal and anomalous flows, the number of attributes and the number of classes for each dataset. This chapter does not collect the descriptions of the datasets due to space restrictions. Thus, for more complete details, we
recommend that readers consult the original references [Suthaharan et al., 2010; Almalawi et al., 2013b].

Table 6.1: Summary of datasets used in the experiments

<table>
<thead>
<tr>
<th>data</th>
<th># instances</th>
<th># attributes</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>699</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>MHORD</td>
<td>2500</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>SPFDS</td>
<td>1000</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>DOSDS</td>
<td>4000</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>SPDOS</td>
<td>2500</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>SHIRD</td>
<td>1800</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>SHORD</td>
<td>400</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>WTP</td>
<td>512</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>ITD 1</td>
<td>21000</td>
<td>149</td>
<td>13</td>
</tr>
<tr>
<td>ITD 2</td>
<td>23000</td>
<td>149</td>
<td>11</td>
</tr>
<tr>
<td>ITD 3</td>
<td>22000</td>
<td>149</td>
<td>12</td>
</tr>
<tr>
<td>ITD 4</td>
<td>23000</td>
<td>149</td>
<td>12</td>
</tr>
<tr>
<td>ITD 5</td>
<td>21500</td>
<td>149</td>
<td>10</td>
</tr>
<tr>
<td>ITD 6</td>
<td>20500</td>
<td>149</td>
<td>11</td>
</tr>
<tr>
<td>ITD 7</td>
<td>21000</td>
<td>149</td>
<td>10</td>
</tr>
<tr>
<td>DARPA</td>
<td>60000</td>
<td>42</td>
<td>5</td>
</tr>
</tbody>
</table>

6.3.2 The Baseline Methods

To understand the common advantages/disadvantages of using semi-supervised learning to label network flows, the performance of SemTra is evaluated against four of the most current and relevant semi-supervised approaches.

- **Probabilistic graphical model (PGM) [Rotsos et al., 2010]:** This approach uses a self-training algorithm, and probabilistic graphical models are proposed to assign a label to unlabelled flows. This approach extends Naive Bayes to learn from both labelled and unlabelled data. The first stage employs Naive Bayes to define a probability distribution over all known and unknown variables in labelled flows. In the second stage, this approach specifies a rule which states how the probability distribution relates to their decision rule. In particular, it introduces two algorithms, called *hard assignment* and *soft assignment*, to approximate the traffic labels.
• Offline/real-time semi-supervised classification (ORTSC) [Erman et al., 2007a]:
  This is a flexible mathematical approach that leverages both labelled and unlabelled flows. This approach involves two steps. Firstly, it employs a K-means clustering algorithm to partition a training dataset that consists of scarce labelled flows combined with abundant unlabelled flows. Secondly, the available labelled flows are used to obtain a mapping from the clusters to the different known \( q \) classes (\( Y \)) based on a probabilistic assignment. To estimate these probabilities, the authors uses the set of flows in the training data that are labelled to map clusters to different applications. This approach is fast and can handle both previously unseen applications and changed behaviour of existing applications.

• Bipartite graph-based maximization (BGCM) [Gao et al., 2013]: This approach combines the outputs of multiple supervised and unsupervised models. In particular, it is assumed that each model partitions a given dataset into groups so that the instances in the same group share either the same predicted class label or the same cluster label. Thus, the outputs of models are summarized by a bipartite graph with connections only between group and instance nodes. A group node and an instance node are connected if the instance is assigned to the group. To obtain the final consensus labelling, the authors introduced the Bipartite Graph-based Consensus Maximization (BGCM) algorithm, which is essentially a block coordinate descent-based algorithm that performs an iterative propagation of probability estimates among neighboring nodes.

• Boosting for semi-supervised learning (SemiBoost) [Mallapragada et al., 2009]: This approach is similar to most boosting algorithms. SemiBoost improves the accuracy of classification iteratively. At each iteration, a number of unlabelled instances will be selected and used to train a new classification model. In particular, the authors used a pairwise similarity measurements to guide the selection of unlabelled instances at each iteration, and to assign class labels to them.
6.3.3 The Experimental Setup

Algorithm 11 shows the process details of the experiment. For each dataset, each semi-supervised method (M=5 ∈ N=10) cross-validation strategy is used. The 10-fold cross-validation is repeated M=5 times, with the order of the instances of the dataset being randomized each time. This is because many of the algorithms are biased by the data order; that is, certain orderings dramatically improve or degrade performance. For each semi-supervised approach, its corresponding runtime, overall accuracy, F-measure value and stability value are obtained for each dataset.

Algorithm 11: Experimental Procedure

1 Input:
2 $M=5$;
3 $\text{SemiTech}(\text{ST})=\{\text{PGM, ORTSC, BGCM, SemiBoost, SemTra}\}$;
4 $\text{DATA}=\{\text{DARPA, WTP, \cdots ,D}_n\}$;
5 Output:
6 $\text{PerfoMetrics}=\{\text{Accuracy, F-measure, Stability, Runtime}\}$;
7 foreach $\text{SemiTech}_i \in [1, \text{ST}]$ do
8   foreach $\text{D}_i \in \text{DATA}$ do
9     foreach $\text{times} \in [1, M]$ do
10        randomise instance-order for $\text{D}_i$;
11        generate $N$ bins from the randomised $\text{D}_i$;
12        foreach $\text{fold} \in [1, N]$ do
13           $\text{LabelledData} = \text{bin}[\text{fold}]$;
14           $\text{UnlabelledData} = \text{D}_i - \text{LabelledData}$;
15           $\text{Train'}_\text{Data} = \text{LabelledData} + \text{Subset from UnlabelledData}$;
16           $\text{Test'}_\text{Data} = \text{UnlabelledData} - \text{Train'}_\text{Data}$;
17           foreach $\text{ST}_i \in \text{SemiTech}$ do
18              $\text{ST}_i^\text{Model} = \text{learning}(\text{Train'}_\text{Data}, \text{ST}_i)$;
19              $\text{Results}_i = \text{testing}(\text{Test'}_\text{Data}, \text{ST}_i^\text{Model})$;
20              $\text{OverallResult} = \text{OverallResult} \cup \text{Results}_i$;
21     PerfoMetrics = average ($\text{OverallResult}$);
6.3.4 Performance Metrics

We adopt the overall accuracy and F-measure as the evaluation metrics for the performance of SemTra, as well as the baseline approaches outlined in the previous section. F-measure is defined as the harmonic mean of recall and precision as follow:

\[
F - \text{Measure} = \frac{(1 + \beta^2).\text{Recall}.\text{Precision}}{(\beta)^2.\text{Recall} + \text{Precision}}
\]

(6.19)

where a coefficient, \( \beta \), is set to 1 to adjust the relative importance of precision versus recall. In particular, recall is defined as the number of traffic flows that are correctly classified divided by the actual number of flows in each class. Precision is defined as the number of flows that are correctly classified divided by the number of all the flows predicted as the same class. Since different semi-supervised methods may produce different class labels for specific instances for the different runs, therefore, the stability of the results across different runs is considered to be important for assessing the semi-supervised methods. This chapter carries out an experimental study to examine the stability of the semi-supervised methods. In doing so, we consider a pairwise approach to measuring the stability of the semi-supervised methods. In particular, the match between each of the \( n(n-1)/2 \) runs of each semi-supervised method is calculated and the stability index is obtained as the averaged degree of match across different runs. Let \( S_r(R_i, R_j) \) be the degree of match between runs \( R_i \) and \( R_j \). The semi-supervised pairwise stability index \( S_k \) is:

\[
S_k = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} S_r(R_i, R_j).
\]

(6.20)

where

\[
S_r(R_i, R_j) = \begin{cases} 
1 & \text{if } R_i(x_i) = R_j(x_j) \\
0 & \text{otherwise} 
\end{cases}
\]

(6.21)

Clearly, it can be seen that \( S_k \) is the average stability measure for the final predicted class labels across different runs. It takes values from [0, 1], with 0 indicating the results between predicted class labels of \( R_i \) and \( R_j \) are totally different, and 1 indicating that the results of
predicted class labels across different runs are identical.

### 6.3.5 Analysis of the Experimental Results

This section provides a summary of the experimental results relating to the accuracy, performance and stability of the SemTra approach as well as the baseline methods. Also, following the approach of García et al. [García et al., 2010] to compare algorithms on multiple datasets, we performed a nonparametric Friedman test followed by a Nemenyi post-hoc test to further explore the statistical significance of the experimental results.

To further explore the accuracy and F-measure of SemTra against the baseline methods, we performed a Friedman test [Friedman, 1940] followed by a Nemenyi post-hoc test [Hollander et al., 2013]. The former test compares the proposed framework and the baseline methods over N datasets by ranking each method on each dataset separately. In particular, the transformation method with the best performance is given rank 1, the second best is given rank 2, and so on. In case of ties, average ranks are assigned, then the average ranks of all methods for all datasets are calculated and compared. The latter test compares the performance of the semi-supervised methods in a pairwise manner. In particular, this test determines which method performs statistically differently if the average rank exceeds the critical difference \( CD_\alpha = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \), where \( q_\alpha \) is calculated based on the studentized range statistic [Newman, 1939] divided by \( \sqrt{2} \).

### Results on the Two-Classes Problem

Here, we first use the binary class dataset to demonstrate the role that different semi-supervised methods play in terms of the accuracy and F-measure values. Table 6.2 is dedicated to the two-class problem, and it shows the 10-fold cross-validation results of the five different semi-supervised methods: SemiBoost, SemTra, PGM, BGCM and ORTSC. The two main dimensions of this table are overall accuracy (value one) and F-measure (value two). The means and standard deviations of the results are obtained by five different methods.
Table 6.2: Comparing overall accuracy and F-measure values of semi-supervised methods on eight binary-class traffic datasets

<table>
<thead>
<tr>
<th>dataset</th>
<th>SemiBoost</th>
<th>SemTra</th>
<th>PGM</th>
<th>BGCM</th>
<th>ORTSC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>F-measure</td>
<td>Accuracy</td>
<td>F-measure</td>
<td>Accuracy</td>
</tr>
<tr>
<td>MHIRD</td>
<td>96.78</td>
<td>2.00</td>
<td>93.92</td>
<td>2.56</td>
<td>95.53</td>
</tr>
<tr>
<td>MHORD</td>
<td>97.08</td>
<td>0.84</td>
<td>95.80</td>
<td>1.56</td>
<td>95.82</td>
</tr>
<tr>
<td>SPFDS</td>
<td>96.74</td>
<td>1.42</td>
<td>94.62</td>
<td>1.89</td>
<td>94.59</td>
</tr>
<tr>
<td>DOSDS</td>
<td>98.48</td>
<td>1.94</td>
<td>93.40</td>
<td>0.50</td>
<td>94.64</td>
</tr>
<tr>
<td>SPDOS</td>
<td>96.96</td>
<td>1.21</td>
<td>94.30</td>
<td>2.05</td>
<td>93.84</td>
</tr>
<tr>
<td>SHIRD</td>
<td>95.86</td>
<td>0.29</td>
<td>94.98</td>
<td>2.08</td>
<td>94.69</td>
</tr>
<tr>
<td>SHORD</td>
<td>97.33</td>
<td>0.91</td>
<td>92.51</td>
<td>2.57</td>
<td>95.82</td>
</tr>
<tr>
<td>WTR</td>
<td>96.43</td>
<td>1.04</td>
<td>94.63</td>
<td>1.96</td>
<td>94.79</td>
</tr>
</tbody>
</table>
By looking at the results of the five methods, we observe that:

- Generally, all five semi-supervised methods achieve significant accuracy scores on the binary class traffic datasets. In particular, SemiBoost obtains the best mean accuracy values, with more than one value being above 95 percent. **SemTra** has the second highest accuracy, with an average of 94.96 percent that has a tiny margin of 2 percent to the SemiBoost method. PGM ranks 3 with average accuracy of 93.32 percent. BGCM and ORTSC rank 4 and 5 respectively with average accuracies of 93.33 percent and 93.02 percent.

- As for standard deviations, Semi-Boost ranges from 0 to 2 on the SD range, while **SemTra** does much the same thing except for a few values above 2. PGM method ranges from 2 to 5 as does BGCM with the largest BGCM value being a slightly over five standard deviations. ORTSC method ranges from 3 to 6 standard deviations.

- On an extremely imbalanced dataset, the accuracy rates cannot provide information about a minority class (e.g. Attack). Therefore, in this section, we also compare the performance of **SemTra** with the baseline methods according to the very popular F-measure that equally weights precision and recall. Semi-Boost and **SemTra** produce the best results among the five semi-supervised methods. The average F-measure scores of Semi-Boost and **SemTra** are always higher than the PGM, BGCM and ORTSC methods by about 0.72-4.23 percent on all binary class datasets.

To further explore whether the overall accuracy on binary class datasets is significantly different, we performed a Friedman test, followed by a Nemenyi potshot test. The result of the Friedman test (at \( \alpha = 0.1 \)) is \( p = 0 \), which indicates that the overall accuracy of all semi-supervised methods is equivalent.

Thus, in order to further explore semi-supervised methods whose overall accuracy is statistically significant difference, we performed a Nemenyi test. Figure 6.5 shows the results with \( \alpha = 0.1 \) on the binary datasets. These indicate that overall accuracy of SemiBoost had the highest value compared to the PGCM, PGM and ORTSC methods. However, there is
no consistent evidence to indicate statistical differences between the SemiBoost and *SemTra* approaches.

**Figure 6.5:** Overall accuracy comparison of all semi-supervised methods with each other on the binary class datasets

The null hypothesis of the Friedman test is that all the semi-supervised methods are equivalent.

To further explore whether the F-measure on binary class datasets is significantly different, we performed a Friedman test, followed by a Nemenyi potshot test. The result of the Friedman test (at $\alpha = 0.1$) is $p = 0$, and this result indicates that the overall accuracy of all semi-supervised methods is equivalent. Thus, to further explore semi-supervised methods whose F-measure had statistically significant differences, we performed a Nemenyi test. Figure 6.6 shows the results with $\alpha = 0.1$ on the binary datasets, and the results indicate that overall accuracy of SemiBoost had the highest value compared to the PGCM, PGM and ORTSC.

However, there is no consistent evidence to indicate statistical differences between SemiBoost and *SemTra* methods. From these results, we can conclude that the *SemTra* performs as well as SemiBoost in binary class traffic datasets.

**Figure 6.6:** F-measure comparison of all semi-supervised methods with each other on the binary class traffic datasets
CHAPTER 6. SEMTRA: A SEMI-SUPERVISED APPROACH FOR NETWORK TRAFFIC LABELLING

Results on Multi-Classes Problem

This section provides the performance evaluation of SemTra against the baseline methods over eight multi-class traffic datasets. We consider the cases where the traffic dataset contains different types of traffic applications (e.g. WWW, FTP, P2P, MAIL etc.) rather than only normal and attack classes.

The overall performance is evaluated in terms of overall accuracy, and we also compare SemTra and the baseline methods according to the so popular F-measure. This is because on an extremely imbalanced dataset, the overall accuracy rate cannot provide information over imbalanced data. Thus, we need to consider such cases where the number of WWW flow instances is much greater than the number of instances in any other class. Table 6.3 shows the overall accuracy of the five semi-supervised methods on eight multi-class traffic datasets.

First, a general observation is that the proposed SemTra approach significantly outperforms all of the baseline methods. Its average overall accuracy of 93.84 percent is 9.23 percent higher than the second best (PGM) and 52.48 percent better than the worst (SemiBoost’s 41.36 percent). This is because SemTra can combine the decision of multiple labelling processes to obtain accurate labels and discard unsupported ones. Surprisingly, SemiBoost performs poorly on all eight datasets, as it is designed to work well only with binary class datasets.

Second, in terms of F-measure, SemTra is the best among the four semi-supervised baseline methods, with an average of 90.02 percent. The average F-measure score of SemTra is always higher than other baseline methods by approximately 23.03 percent to 52.60 percent. It can be seen also that the average F-measure score of PGM is ranked 2, with an average F-measure score of 66.99 percent. We note that there is a large gap between the scores of PGM with respect to overall accuracy and F-measure. This is because the PGM method was biased toward the majority class (e.g. WWW, P2P etc.).
### Table 6.3: Comparing overall accuracy and F-measure values of semi-supervised methods on eight multi-class traffic datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SemiBoost Accuracy</th>
<th>SemiBoost F-measure</th>
<th>SemTra Accuracy</th>
<th>SemTra F-measure</th>
<th>PGM Accuracy</th>
<th>PGM F-measure</th>
<th>BGCM Accuracy</th>
<th>BGCM F-measure</th>
<th>ORTSC Accuracy</th>
<th>ORTSC F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITD 1</td>
<td>39.85 (1.79)</td>
<td>35.92 (2.21)</td>
<td>93.80 (1.15)</td>
<td>89.84 (1.56)</td>
<td>85.26 (2.70)</td>
<td>69.65 (4.48)</td>
<td>53.61 (5.30)</td>
<td>51.27 (6.36)</td>
<td>46.01 (6.86)</td>
<td>39.65 (7.94)</td>
</tr>
<tr>
<td>ITD 2</td>
<td>42.27 (2.56)</td>
<td>35.96 (3.06)</td>
<td>93.76 (1.89)</td>
<td>90.11 (2.61)</td>
<td>81.14 (4.02)</td>
<td>70.11 (4.51)</td>
<td>54.77 (5.99)</td>
<td>49.76 (6.42)</td>
<td>45.89 (7.31)</td>
<td>41.87 (7.15)</td>
</tr>
<tr>
<td>ITD 3</td>
<td>41.84 (2.37)</td>
<td>38.73 (4.58)</td>
<td>93.36 (1.70)</td>
<td>90.39 (2.34)</td>
<td>81.69 (3.63)</td>
<td>66.88 (6.16)</td>
<td>54.06 (5.40)</td>
<td>51.85 (5.65)</td>
<td>46.04 (6.45)</td>
<td>42.00 (7.90)</td>
</tr>
<tr>
<td>ITD 4</td>
<td>41.54 (1.58)</td>
<td>41.00 (4.65)</td>
<td>93.90 (1.34)</td>
<td>89.69 (2.36)</td>
<td>83.80 (3.58)</td>
<td>63.05 (5.13)</td>
<td>55.77 (4.67)</td>
<td>50.36 (5.30)</td>
<td>46.02 (7.86)</td>
<td>39.72 (7.84)</td>
</tr>
<tr>
<td>ITD 5</td>
<td>40.70 (2.58)</td>
<td>37.66 (3.52)</td>
<td>93.64 (1.90)</td>
<td>89.70 (2.65)</td>
<td>85.31 (2.67)</td>
<td>67.11 (2.42)</td>
<td>53.66 (3.42)</td>
<td>49.23 (5.48)</td>
<td>46.06 (6.71)</td>
<td>41.40 (7.62)</td>
</tr>
<tr>
<td>ITD 6</td>
<td>42.79 (1.92)</td>
<td>35.22 (2.60)</td>
<td>93.76 (1.91)</td>
<td>90.38 (1.14)</td>
<td>85.04 (3.65)</td>
<td>67.32 (3.85)</td>
<td>56.44 (5.32)</td>
<td>51.37 (6.62)</td>
<td>46.04 (6.33)</td>
<td>39.89 (7.34)</td>
</tr>
<tr>
<td>ITD 7</td>
<td>41.50 (1.79)</td>
<td>38.85 (2.23)</td>
<td>94.07 (1.27)</td>
<td>90.10 (2.08)</td>
<td>85.12 (3.33)</td>
<td>66.24 (3.71)</td>
<td>55.62 (4.68)</td>
<td>50.83 (6.34)</td>
<td>45.99 (6.82)</td>
<td>41.11 (7.91)</td>
</tr>
<tr>
<td>DARPA</td>
<td>40.40 (2.44)</td>
<td>36.03 (3.23)</td>
<td>94.42 (1.07)</td>
<td>89.95 (1.97)</td>
<td>84.53 (3.49)</td>
<td>65.33 (4.31)</td>
<td>55.27 (4.87)</td>
<td>52.34 (5.91)</td>
<td>45.85 (6.97)</td>
<td>40.84 (8.31)</td>
</tr>
</tbody>
</table>
To further explore whether the overall accuracy and F-measure of semi-supervised methods on multi-classes datasets are significantly different, we performed a Friedman test, followed by a Nemenyi potshot test. The results of the Friedman test indicate that the overall accuracy and F-measure of all semi-supervised methods are equivalent. Thus, to further explore semi-supervised methods whose F-measures have statistically significant differences, we performed a Nemenyi test. Figure 6.7 shows the overall accuracy results with $\alpha = 0.1$ on multi-class datasets. The results indicate that overall accuracy of SemTra is scored the highest value compared to the PGM, PGCM, SemiBoost and ORTSC.

![Figure 6.7: Overall accuracy comparison of all semi-supervised methods with each other on the multi-class traffic datasets](image)

Figure 6.8 shows Nemenyi test of F-measure for all semi-supervised methods, the results indicate that the SemTra method is statistically better than those of PGCM, SemiBoost and ORTSC, and there is clear evidence of a statistical difference between SemTra and PGM in terms of the F-measure.

![Figure 6.8: F-measure comparison of all semi-supervised methods on the multi-class traffic datasets](image)

**Running times and Scalability**

Table 6.4 shows the runtime of the five semi-supervised methods. It can be observed the individual semi-supervised methods of ORTSC, PGM and BGCM are much faster than
SemiBoost and SemTra, respectively. ORTSC is consistently faster than all other semi-supervised methods. Several observations can be made from Table 6.4. First, it can be seen that the runtime of ORTSC is only 36.35 percent of the runtime of PGM, 1.06 percent of the runtime of BGCM, 0.30 percent of the runtime of SemiBoost and 0.24 percent of the runtime of SemTra. Second, it can be seen that both SemTra and SemiBoost have the worst runtime in comparison with other three semi-supervised methods. Thus, future work should be devoted to reducing the computational time of the SemTra approach.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ORTSC</th>
<th>PGM</th>
<th>BGCM</th>
<th>SemiBoost</th>
<th>SemTra</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>0.39</td>
<td>1.24</td>
<td>1.87</td>
<td>64.85</td>
<td>79.75</td>
</tr>
<tr>
<td>MHORD</td>
<td>0.39</td>
<td>1.25</td>
<td>1.90</td>
<td>64.11</td>
<td>78.85</td>
</tr>
<tr>
<td>SPFDS</td>
<td>0.28</td>
<td>0.86</td>
<td>1.77</td>
<td>30.48</td>
<td>37.49</td>
</tr>
<tr>
<td>DOSDS</td>
<td>0.21</td>
<td>0.67</td>
<td>1.26</td>
<td>54.79</td>
<td>67.38</td>
</tr>
<tr>
<td>SPDOS</td>
<td>0.50</td>
<td>1.53</td>
<td>3.08</td>
<td>91.41</td>
<td>112.41</td>
</tr>
<tr>
<td>SHIRD</td>
<td>0.40</td>
<td>1.20</td>
<td>1.83</td>
<td>57.56</td>
<td>70.79</td>
</tr>
<tr>
<td>SHORD</td>
<td>0.51</td>
<td>1.36</td>
<td>2.28</td>
<td>73.67</td>
<td>90.60</td>
</tr>
<tr>
<td>WTP</td>
<td>0.08</td>
<td>0.16</td>
<td>1.98</td>
<td>2.17</td>
<td>2.67</td>
</tr>
<tr>
<td>ITD 1</td>
<td>1.17</td>
<td>3.19</td>
<td>220.89</td>
<td>892.13</td>
<td>1097.12</td>
</tr>
<tr>
<td>ITD 2</td>
<td>1.28</td>
<td>3.49</td>
<td>222.88</td>
<td>977.09</td>
<td>1201.61</td>
</tr>
<tr>
<td>ITD 3</td>
<td>1.23</td>
<td>3.34</td>
<td>221.89</td>
<td>934.61</td>
<td>1149.36</td>
</tr>
<tr>
<td>ITD 4</td>
<td>1.29</td>
<td>3.50</td>
<td>222.98</td>
<td>978.55</td>
<td>1203.39</td>
</tr>
<tr>
<td>ITD 5</td>
<td>1.20</td>
<td>3.26</td>
<td>221.39</td>
<td>913.37</td>
<td>1123.24</td>
</tr>
<tr>
<td>ITD 6</td>
<td>1.14</td>
<td>3.11</td>
<td>220.39</td>
<td>870.89</td>
<td>1071.00</td>
</tr>
<tr>
<td>ITD 7</td>
<td>1.17</td>
<td>3.19</td>
<td>220.89</td>
<td>892.13</td>
<td>1097.12</td>
</tr>
<tr>
<td>DARPA</td>
<td>13.75</td>
<td>37.4</td>
<td>786.57</td>
<td>1559.7</td>
<td>1795.10</td>
</tr>
</tbody>
</table>

In order to further explore whether the runtimes of the five semi-supervised methods are significantly different, we performed a Friedman test. The null hypothesis of the Friedman test indicates that all semi-supervised methods are equivalent in terms of runtime. The result of the test is $p=0$, which means that, at $a=0.1$; hence, there is evidence to reject the null hypothesis and all the five semi-supervised methods are different in terms of runtime. Thus, we have conducted a post-hoc Nemenyi test. Figure 6.9 shows the results with $a=0.1$ on the 16 datasets. The results indicate that the runtime of ORTSC method is statically better than those of BGCM, SemiBoost and SemTra respectively, and there is no consistent evidence to
indicate a statistical runtime difference between ORTSC and PGM methods.

![Figure 6.9: Scalability of semi-supervised methods on DARPA dataset](image)

Figure 6.9: Scalability of semi-supervised methods on DARPA dataset

Figure 6.10 shows the scalability of SemTra method and the other five semi-supervised methods with a varying number of instances in the dataset. Considering the page limitations, we use only the DARPA dataset. We vary the number of flows from 1000 to 50000 and plot them on the graph. It is clear from the trend that all semi-supervised methods scale linearly with respect to the number of instances.

![Figure 6.10: Runtime comparison of all semi-supervised methods with the Nemenyi test](image)

Figure 6.10: Runtime comparison of all semi-supervised methods with the Nemenyi test

**Stability**

The aim here is to evaluate the stability of the proposed SemTra approach and the baselines methods in generating accurate class labels for unlabelled flows across different runs. An
important property of a semi-supervised process is the ability to indicate randomness in assigning class labels.

Table 6.5 shows the stability indices for all semi-supervised methods on the 16 datasets. Equation 6.20 is used to measure the stability. The following setup was used in the experiment to evaluate the stability. The number of trials was set to \( n = 100 \). From each dataset, 30 percent of data in each class was reserved for testing, and thus was excluded from the training dataset. In each trial, 90 percent of the remaining data was randomly sampled to form a trial local dataset. Note, the stability indices have been computed from the training dataset only.

**Table 6.5: Comparison of the stability the SemTra approach and the baseline methods**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SemTra</th>
<th>SemiBoost</th>
<th>PGM</th>
<th>BGCM</th>
<th>ORTSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHIRD</td>
<td>97.35</td>
<td>97.44</td>
<td>94.05</td>
<td>90.96</td>
<td>90.84</td>
</tr>
<tr>
<td>MHORD</td>
<td>97.91</td>
<td>98.44</td>
<td>94.97</td>
<td>91.55</td>
<td>90.60</td>
</tr>
<tr>
<td>SPFDS</td>
<td>98.71</td>
<td>98.11</td>
<td>94.90</td>
<td>92.73</td>
<td>92.93</td>
</tr>
<tr>
<td>DOSDS</td>
<td>98.98</td>
<td>99.50</td>
<td>93.77</td>
<td>91.49</td>
<td>91.55</td>
</tr>
<tr>
<td>SPDOS</td>
<td>98.10</td>
<td>97.95</td>
<td>93.64</td>
<td>91.81</td>
<td>91.84</td>
</tr>
<tr>
<td>SHIRD</td>
<td>98.98</td>
<td>97.92</td>
<td>93.94</td>
<td>91.74</td>
<td>92.29</td>
</tr>
<tr>
<td>SHORD</td>
<td>98.58</td>
<td>97.43</td>
<td>96.17</td>
<td>92.57</td>
<td>90.59</td>
</tr>
<tr>
<td>WTR</td>
<td>96.42</td>
<td>93.04</td>
<td>91.96</td>
<td>85.64</td>
<td>85.64</td>
</tr>
<tr>
<td>ITD 1</td>
<td>95.99</td>
<td>92.86</td>
<td>90.35</td>
<td>84.53</td>
<td>76.80</td>
</tr>
<tr>
<td>ITD 2</td>
<td>96.94</td>
<td>91.79</td>
<td>91.52</td>
<td>83.64</td>
<td>77.06</td>
</tr>
<tr>
<td>ITD 3</td>
<td>95.89</td>
<td>91.94</td>
<td>90.49</td>
<td>83.58</td>
<td>77.85</td>
</tr>
<tr>
<td>ITD 4</td>
<td>96.16</td>
<td>90.42</td>
<td>88.84</td>
<td>84.35</td>
<td>77.10</td>
</tr>
<tr>
<td>ITD 5</td>
<td>96.14</td>
<td>90.35</td>
<td>89.87</td>
<td>84.70</td>
<td>77.16</td>
</tr>
<tr>
<td>ITD 6</td>
<td>95.85</td>
<td>91.08</td>
<td>92.58</td>
<td>84.52</td>
<td>77.98</td>
</tr>
<tr>
<td>ITD 7</td>
<td>97.36</td>
<td>92.40</td>
<td>91.15</td>
<td>83.38</td>
<td>77.66</td>
</tr>
<tr>
<td>DARPA</td>
<td>97.92</td>
<td>97.77</td>
<td>95.29</td>
<td>88.66</td>
<td>85.09</td>
</tr>
</tbody>
</table>

It can be seen from Table 6.5 that *SemTra* ranks 1 not only on the binary class dataset but also on the multi-class datasets, with margin of 2.38 percent over the second most stable semi-supervised method (SemiBoost) and 12.3 percent over the least stable method (ORTSC). This is explained by the fact that *SemTra* is designed to assign an accurate class label based on a fusion of multiple decisions. The SemiBoost method achieves good stability results on all binary class datasets, but performs badly on the multi-class datasets. This is
because it was designed to deal with only binary class datasets. Notably, smaller differences in terms of stability indices can be observed among SemiBoost and PGM methods with a margin of 2.17 percent. It can be seen from Table 6.5 that the ORTSC method has the lowest stability indices on both binary and multi-class traffic data. This is because ORTSC method is based on only K-means clustering algorithm, which is considered to be sensitive to random initialization. The second worse stability result was produced by the BGCM method, especially on multi-class traffic data.

To further explore whether the stability indices of SemTra with the four semi-supervised are significantly difference, we perform two tests including Friedman tests and post-hoc Nemenyi test. The result of the former test is $p = 0$ which means that $a = 0.1$, this result indicates that the stability indices for all five semi-supervised methods is not equivalent. Thus, post-hoc Nemenyi tests were conducted to explore further difference in such methods. From Figure 6.11, we observe that SemTra is statistically better than PGM, BGCM and ORTSC methods. However, there is no consistent evidence to indicate a significant different between SemTra and SemiBoost.

![Figure 6.11: Stability comparison of all semi-supervised methods with the Nemenyi test](image)

**Figure 6.11:** Stability comparison of all semi-supervised methods with the Nemenyi test

**Discussion and Summary**

As shown in Table 6.6, SemTra and the baseline methods are validated against the desired requirements for the labelling process. These requirements include five properties to deal with (i) binary-class and multi-class datasets, (ii) detection of novel classes, (iii) detection of outliers, (iv) prediction of stable class assignments and (v) coping with efficiency. Table 6.6 shows that SemTra has advantages over the related baseline methods. First, SemTra can deal with various types of classes, while the related methods can deal only with binary-class (except for PGM, which can deal partially with multi-class). Second, extensive experiments
Table 6.6: Compliance summary of the semi-supervised performance based on empirical evaluation metrics

<table>
<thead>
<tr>
<th>Semi. Methods</th>
<th>Binary-class</th>
<th>Multi-class</th>
<th>Novel Class Detection</th>
<th>Outlier Detection</th>
<th>Stability</th>
<th>Efficiency</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemTra</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Partially</td>
<td>High</td>
<td>Suffer from</td>
<td></td>
</tr>
<tr>
<td>SemiBoost</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>High/Low</td>
<td>Suffer from</td>
<td></td>
</tr>
<tr>
<td>PGM</td>
<td>Yes</td>
<td>Partially</td>
<td>No</td>
<td>No</td>
<td>High/Mid</td>
<td>Partially</td>
<td></td>
</tr>
<tr>
<td>BGCM</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Low</td>
<td>Suffer from</td>
<td></td>
</tr>
<tr>
<td>ORTSC</td>
<td>Yes</td>
<td>No</td>
<td>Partially</td>
<td>No</td>
<td>Low</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

show that *SemTra* is also capable of detecting novel classes as well as partially detecting outliers, while the other related methods are not able to detect either novel classes or outliers (except ORTSC, which can partially deal with detection of novel classes and these are termed as “known class”). Regarding the stability value, it can be seen that *SemTra* assigns a stable class to the same instances. The baseline methods have high stability on binary class datasets and low stability on multi-class datasets. However, *SemTra* performs significantly better than the baseline methods on number of metrics, although it has the highest computational time. A possible solution is to rely on specific technology (HPC) to enable such algorithms to be executed more efficiently.

### 6.4 Conclusion

The capability of accurate traffic classification ML-based is very important for the management, monitoring and provision of networks. In this chapter, we investigated the problem of lacking labelled traffic data to achieve an improvement in network traffic classification task, and a novel semi-supervised method for automatically labelling traffic data with a minimum human effort was proposed. In particular, the proposed *SemTra* approach incorporates the advantages of the complementary predictive powers of supervised and unsupervised algorithms over multiple representations of the original traffic dataset.

An extensive study using a publicly-available traffic data benchmark has proved the strength of the proposed *SemTra* approach in achieving an improvement in the network traffic classification task in comparison to the state-of-art semi-supervised learning approaches.

For future work, we plan to enhance the performance of the *SemTra* approach to be
applicable online by providing an efficient and scalable mechanism to incrementally update the generated model.
Chapter 7

Conclusion

Network traffic classification has the potential to resolve key issues for network operators, including network management problems, QoS provisioning, Internet accounting and charging, and lawful interception [Nguyen, 2009]. The traditional network classification techniques that rely mostly on well-known port numbers have been used to identify Internet traffic. Such an approach was successful because traditional applications used fixed port numbers; however [Moore and Papagiannaki, 2005; Karagiannis et al., 2004] show that the current generations of P2P applications try to hide their traffic by using dynamic port numbers. Consequently, applications whose port numbers are unknown cannot be identified in advance.

Another approach relies on the inspection of packet contents, and analyses the packets’ payload content to check if they contain signatures of well-known or anomalous applications. Features are extracted from the traffic data, and later compared to well-known signatures of applications provided by human experts. This works well for Internet traffic; however, several studies [Auld et al., 2007; Yuan et al., 2008; Moore and Papagiannaki, 2005] have shown that this approach has a number of drawbacks and limitations. Firstly, it cannot identify a new or unknown attack for which signatures are not available, so there is a need to maintain an up-to-date list of signatures. This is a problem because new applications and attacks emerge everyday; hence, it is not practical and sometimes impossible to keep up with the latest signatures. Secondly, deep packet inspection is a difficult task as this
requires significant processing time and memory. Finally, if the application uses encryption, this approach no longer works. Significant recent research has attracted some attention is based on TLS (Transport Layer Statistics) data and efficient data mining algorithms. This assumes that applications typically send data in some sort of pattern, which can be used as a means of classification of connections by different traffic classes. To extract such patterns, only TCP/IP headers are needed to observe flow statistics such as mean packet size, flow length and total number of packets.

In this thesis, we have investigated several key issues related to the problem of accurate and effective network traffic classification. We were particularly motivated by the specific problems associated with network traffic classification based on machine learning and Transport Layer Statistics (TLS). Most past research has applied different categories of machine learning, including: supervised learning and unsupervised learning algorithms on the TLS of the traffic data to address the problem of network traffic analysis.

Significant recent research has revealed that TLS data allows the machine learning classification-based techniques to rely on sufficient information. In light of these findings, we focused our efforts on the modelling and improvement of network traffic classification based on the concept of machine learning and TLS data.

Our first two research questions were focused on devising more effective preprocessing approaches to improve the quality of selected features to train network classifiers by discarding the irrelevant and redundant features from the original features of the Transport Layer Statistics (TLS). We then considered a framework to preserve the privacy of network traffic data to help network collaborators to publish their traffic data and make them publicly-available for the common good. In particular, such a framework dealt with the unique characteristics of network traffic data to preserve the data privacy, and improve their utility for building an accurate traffic classifiers.

Finally, a semi-supervised approach was presented to construct a traffic classification model with a good generalization ability, and to evaluate the generated traffic classification. The semi-supervised approach allowed us to discover the emergence of a new class and eliminate outlier instances.
Specifically, the following research questions have been addressed in this thesis:

A) How to optimize various feature-selection methods and improve the quality of transport-layer statistics data for accurate and effective network traffic classification?

B) How to identify the optimal and stable feature set in the temporal-domain and the spatial-domain for accurate and effective network traffic classification?

C) How to preserve the privacy of traffic data publishing for accurate intrusion detection systems and network traffic classification?

D) How to “automatically” label raw traffic data for evaluating and building an accurate network traffic classification?

7.1 Contribution

Here we outline the innovations and advantages of the work conducted in this thesis:

1. Improve the quality of transport-layer statistics data for accurate and effective network traffic classification

Transport Layer Statistics (TLS) was specifically introduced to address the problems caused by traditional classification techniques which use port number and payload. However, the presence of irrelevant and redundant features in the TLS data can affect the accuracy and efficiency of network traffic classification. To achieve better overall classification performance, several feature selection techniques can be used to filter out such irrelevant and redundant features from the original features of the Transport Layer Statistics (TLS). A key issue with these feature selection techniques is that they are designed with different evaluation criteria (e.g. information-based measure, dependence-based measure, consistency-based measure and distance-based measure). Therefore, it is a challenge to choose a suitable one for identifying the best features (that minimize redundancy and maximize relevance) for network traffic classification. To address this issue, new metrics were presented, allowing us to extensively evaluate
and compare such techniques from different perspectives including goodness, stability and similarity. Several issues associated with each feature selection technique were recorded. Consequently, we continue our efforts toward developing an integrated FS technique that is built on the key strengths of existing FS techniques. In particular, a Local Optimization Approach (LOA) is proposed to identify efficiently and accurately select the “best” features by first combining the results of some well-known FS techniques to find consistent features, and then use the proposed concept of support to select a smallest set of features and cover data optimality. The empirical study over number of high-dimensional network traffic datasets demonstrates significant gain in accuracy and improved runtime performance of a network classifier compared to individual results produced by some well-known FS techniques.

2. Identify the optimal and stable feature in the temporal-domain and the spatial-domain for accurate network traffic classification

Obtaining an optimal and stable feature set across the temporal-domain and the spatial-domain is crucial in enhancing the confidence of network operators. This is because an optimal feature selection set does not necessarily imply high stability and vice versa. Thus, with the aim of discovering more-valuable features for the description of traffic flows with respect to both stability and optimality criteria, a Global Optimisation Approach (GOA) was proposed. In particular, to avoid a situation where the dependence between a pair of features is weak, but the total inter-correlation of one attribute to the others is strong, the proposed GOA first select informative features for each feature selection technique from a global perspective. Second, the outputs of multiple well-known FS techniques are combined to obtain a possible optimal feature subsets across different traffic datasets. Third, an adaptive threshold based on entropy is proposed to extract the stable features. Finally, a new goodness measure is proposed within a Random Forest framework to estimate the final optimum feature subset. Experimental studies on network traffic data in spatial and temporal domains showed that the proposed GOA approach outperforms the commonly-used feature selection techniques in identifying both optimal and stable features for traffic classification. Nevertheless.
to enhance the performance of the GOA approach an efficient discretisation method is used to significantly improve the accuracy of different ML algorithms which suffer from the presence of continuous-valued features in the temporal-domain and the spatial-domain traffic data.

3. **Preserve the privacy for traffic data publishing for accurate network traffic classification**

Sharing traffic data between multiple organizations/sites has become vital requirements for such organizations to create a collaborative anomaly detection, an accurate and a global predictive traffic classification model. However, inappropriate sharing and usage of traffic data could threaten the privacy and security of the data providers resulting in preventing the sharing of data. Consequently, a privacy-preserving framework was proposed to enhance the trust between data providers for publishing network traffic data by maintaining provable privacy of the original traffic data and providing masked data equivalent to the original ones. In particular, our proposed privacy-preserving framework involves: (i) vertically partitioning the original dataset to improve the performance of perturbation, (ii) developing a framework to deal with various types of network traffic data including numerical, categorical and hierarchical attributes: (iii) grouping the portioned sets into a number of clusters based on the proposed framework; and (iii) the perturbation process is accomplished by the alteration of the original attribute value by a new value (clusters centroid). Extensive experimental analysis on a number of traffic datasets showed that the proposed privacy framework can effectively deal with multivariate traffic attributes, produces compatible results as the original data, improves the performances of the five supervised approaches and provides high level of privacy protection compared to the traditional privacy-preserving approaches that are still frequently used in many real-world applications.
4. **Automatically label raw traffic data for accurate network traffic classification**

To maintain good traffic classification performance, sufficient, reliable and up-to-date labelled data is needed. However, due to the high cost and time of manual labelling, it is hard or impossible to obtain such labelled data. As such, a new a novel semi-supervised approach, called *SemTra* was proposed to automatically alleviates the shortage of labelled flows for machine learning by exploiting the advantages of both supervised and unsupervised models. In particular, *SemTra* involves the following: (i) generating multi-view representations of the original data based on dimensionality reduction methods to have strong discrimination ability, (ii) incorporating the generated representations into the ensemble clustering model to provide a combined clustering output with better quality and stability, (iii) adapting the concept of self-training to iteratively utilize the few labelled data along with unlabelled within local and global viewpoints; and (iii) obtaining the final class decision by combining the decisions of mapping strategy of clusters, the local self-training and global self-training approaches. Extensive experimental analysis on a large number of traffic datasets showed that the *SemTra* approach addressed several limitations of the existing semi-supervised approaches, including accurate prediction information for unlabelled instances on multiclass traffic data (e.g. WWW, FTP, P2P, MAIL etc.) rather than only on normal and attack classes, high stability in predicting actual classes, better novel class detection in the presence of concept-drift, and efficient elimination of outlier instances.

### 7.2 Future Work

This thesis has proposed a set of innovative approaches to improve the effectiveness and accuracy of network traffic classification. However, as highlighted in Chapter 7, there is still room for improvement to these approaches, and also to address other relevant issues in the area of network traffic classification which have not been covered in the thesis. Based on our current analysis, in this section we highlight some of these key areas and issues for future exploration.
In Chapter 3, the Local Optimization Approach (LOA) is proposed for identifying the “best” features in the Transport Layer Statistics (TLS) by utilizing the label information. In particular, the LOA approach is designed for supervised learning which requires the traffic flows to be labelled in advance to eliminate these features from the TLS data. However, unlabelled traffic data with extremely high dimensionality faces the serious challenge of defining the concept of irrelevant and redundant features in such data. Therefore, it would be valuable to extend the LOA approach to make it applicable for unsupervised learning. Also, reducing the computational time and enabling the LOA approach to process large-scale problems efficiently would be a significant achievement by adopting high-performance distributed computing frameworks and protocols, such as MapReduce and MPI.

In Chapter 4, the Global Optimization Approach (GOA) has been proposed to identify both an optimal and stable feature set for network classification. GOA approach has achieved significant results in accurately classifying traffic from temporal and spatial domains compared to existing feature selection algorithms. However, the generation of the candidate feature set by this approach depends on a multi-criterion fusion which is computationally expensive. Future work would be devoted to enhance the performance of the GOA approach and to reduce its pre-processing time by using parallel computing such as multi-core CPUs or Graphics Processing Units (GPU). Nevertheless, theoretical justification is significantly important and should be considered to explain why and how the GOA approach in fact has found the trade-off between stability and accuracy.

In Chapter 5, a PrivTra framework is proposed for preserving privacy in traffic data publishing and to enhance data mining utility. In particular, we considered different supervised techniques to evaluate the data utility of this privacy framework. However, evaluating the effectiveness of transformed data of such a framework on unsupervised learning points to an interesting direction for future work. Nevertheless, the proposed framework could be used in the area of distributed network classification. Therefore, it is potentially useful to consider the communication cost and amount of data when
CHAPTER 7. CONCLUSION

exchanging the knowledge (based on the processed data) between collaborative learning for effective detection of new patterns and attacks. Furthermore, an investigation into ways of providing probabilistic privacy and accuracy guarantees by using the proposed framework would be extremely useful.

- In Chapter 6, a SemTra semi-supervised approach for automatically labelling traffic flow is proposed. However, we would like to explore whether any theoretical performance guarantees can be offered by this approach. Although the proposed semi-supervised approach has shown promising results, there is a need to minimize the size of the labelled and unlabelled data by selecting only the most-representative instances of each class rather than the whole traffic dataset to improve the quality and computational time of the labelling process. Furthermore, distinguishing among multiple novel classes is important. Nevertheless, enabling a dynamic number of cluster settings rather than a manual setting to provide a better drift detection mechanism is significantly important and should be considered in the future. Also, within the scope of our current and future work, we would extend the proposed semi-supervised approach to handle class imbalance and to maintain high accuracy (recall) on minority classes without sacrificing the performance of majority classes.


Mawi working group traffic archive. [ONLINE]. 2009.

URL: http://mawi.wide.ad.jp/mawi/.


URL: http://www.iana.org/assignments/port-numbers;[DateLastaccessed:2014, 22February].


H. Cramér et al. The elements of probability theory and some of its applications. 1955.


BIBLIOGRAPHY


BIBLIOGRAPHY


H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


URL: http://www.theregister.co.uk/2003/08/20/slammer_worm_crashed_ohio_nuke/.


BIBLIOGRAPHY


URL: http://www.modbustools.com; [Date last accessed: 15.08.2013].


S. Valenti and D. Rossi. Identifying key features for P2P traffic classification. In IEEE International Conference on Communications (ICC), pages 1–6. IEEE, 2011.


BIBLIOGRAPHY


