TOWARDS AN EFFICIENT UNSUPERVISED FEATURE SELECTION METHODS FOR HIGH-DIMENSIONAL DATA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

NAIF YOUSEF A. ALMUSALLAM

M.Sc (Information Technology)
Monash University, Australia

B.Sc (Computer Science)
King Faisal University (KFU), Saudi Arabia

School of Science
College of Science, Engineering, and Health
RMIT University
Melbourne, Victoria, Australia

December, 2018
This work is dedicated to the secret behind my success
my parent (Yousef Almuslem and Layla Alshabibi), my wife Jana Alhawas, my
little daughter Layla, my sisters (Sarah, Haalah, Mariam and Ghand) and my
only brother Ahmed.
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; 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; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Naif Yousef A Almusallam

December 14, 2018
Acknowledgement

I thank and praise Almighty Allah for granting me the opportunity to undertake this research and giving me the strength, knowledge, and ability to persevere and complete it satisfactorily.

On my 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 for their remarkable guidance and help. Without these people, the completion of this thesis would have been more difficult.

I would like to express my sincere gratitude and appreciation to several people for their kind support and inspiration. First and foremost, I would like to thank my senior supervisor, Professor Zahir Tari, who guided me to accomplish my research goals. His effective supervision, encouragement and continuous support has always kept me motivated during my Ph.D. studies. I would also like to thank my associate supervisor, Dr. Jeffrey Chan, for his advice, support and valuable comments. I would also like to thank my family: my father, Yousef Almuslem, my mother, Layla Alshabibi, my brother and my sisters who always gave me mental strength and courage. I am grateful for the sacrifices they made for me. I will never forget their daily contact with me. Their prayers always motivated me to successfully
achieve my goal.

A special thanks goes to my wife, Jana, and my daughter, Layla, who have been my constant companions throughout this journey. I thank them for the support, encouragement, patient and care that I continue to receive. They were the joy of my learning journey. I would like to thank my father-in-law, Fahad Alhawas, and mother-in-law, Maryam Alhawas for their appreciation and prayers. I am thankful to my colleagues, Dr. Abdullatif Al-abdullatif, Dr. Adil Al-harthi and Dr. Abdulmohsen Al-malwawi who helped me with suggestions and materials for my research. I would also like to express my gratitude to all my friends within and outside of RMIT university who directly or indirectly supported me throughout this journey. My appreciation goes especially to Ahmed Al-nuaim. We have been supporting each other since our language course in 2011, during our Master studies and then our Ph.D. journey. Also, many thanks go to my friend Dr. Mohammed Al-nuaim who welcomed me to Melbourne on my first arrival. He helped me a lot, picking me from Melbourne’s airport, organising the hospitality and so on. He really made my adaptation to the lovely Melbourne easier.

Last but not least, I would like to thank my scholarship provider, the Imam Muhammad ibn Saud university for giving me the opportunity to do complete my degrees and further my career. I hope this work will benefit the research community in this area of interest.
Credits

Portions of the materials used in this thesis have previously appeared or under consideration in the following scientific publications:


• N. Almusallam, Z. Tari, J. Chan, A. Alharthi: An Efficient Unsupervised Feature Selection for Dynamic Features. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* - (SJR Rank: Q1)- Submitted in Oct 2018 - Chapter 4

Contents

Declaration iii

Acknowledgement iv

Credits vi

Contents viii

List of Figures xi

List of Tables xv

Abstract 2

1 Introduction 3
   1.1 Motivation ........................................ 4
   1.2 Existing Feature Selection Problems ................. 6
   1.3 Existing Solutions ................................ 11
   1.4 Research Questions ................................ 14
   1.5 Summary of Contributions ........................... 16
   1.6 Organisation of The Thesis ........................... 20

2 Background 22
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dimensionality Reduction</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Clustering-Based Methods</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>Data-Driven Intrusion Detection Systems (IDS)</td>
<td>30</td>
</tr>
<tr>
<td>2.4</td>
<td>Conclusion</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>AUFS - Towards an Efficient and Accurate Unsupervised Feature Selection</td>
<td>40</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>Related Work</td>
<td>45</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Supervised Feature Selection Methods</td>
<td>45</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Unsupervised Feature Selection Methods</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>Similarity Measures</td>
<td>48</td>
</tr>
<tr>
<td>3.4</td>
<td>AUFS - The Proposed an Accurate and Efficient Unsupervised Feature Selection Method</td>
<td>51</td>
</tr>
<tr>
<td>3.4.1</td>
<td>The AUFS Method</td>
<td>52</td>
</tr>
<tr>
<td>3.4.2</td>
<td>An Illustrative Example</td>
<td>55</td>
</tr>
<tr>
<td>3.5</td>
<td>Experimental Setup</td>
<td>56</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Datasets</td>
<td>57</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Evaluation Metrics</td>
<td>58</td>
</tr>
<tr>
<td>3.6</td>
<td>Experimental Results</td>
<td>60</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusion</td>
<td>67</td>
</tr>
<tr>
<td>4</td>
<td>UFSSF - An Efficient Unsupervised Feature Selection for Streaming Features</td>
<td>69</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>70</td>
</tr>
<tr>
<td>4.2</td>
<td>Related Work</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>Similarity Measures</td>
<td>75</td>
</tr>
</tbody>
</table>
## CONTENTS

4.4 The UFSSF Method .............................................. 79
  4.4.1 The Framework ........................................... 80
4.5 Experimental Evaluation ........................................ 83
  4.5.1 The Datasets ............................................... 86
  4.5.2 Evaluation Metrics ....................................... 87
4.6 Results and Analysis ........................................... 89
4.7 Conclusion .................................................... 101

5 OUDVFS: Online Unsupervised Feature Selection for Dynamic Multi-Views 102
5.1 Introduction .................................................... 104
5.2 Related Work .................................................. 108
5.3 The Proposed OUDVFS Method ................................ 113
  5.3.1 The Framework ............................................ 115
5.4 Experimental Evaluation ........................................ 121
  5.4.1 The Datasets ............................................... 124
  5.4.2 Evaluation Metrics ....................................... 125
5.5 Results and Analysis ........................................... 126
5.6 Conclusion .................................................... 136

6 Conclusion .................................................... 137
6.1 Future Work ................................................... 142

Bibliography ..................................................... 144
List of Figures

1.1  high-dimensional datasets in UCI repository (Bolon-Canedo et al. 2015)  4
1.2  Categorisation of feature selection problems from a data perspective
    (Li et al. 2017)  .................................................................  7
2.1  An overview of the feature selection process  .................................  23
2.2  Deviation of circle points (anomalies/outliers) from the normal triangle ones  .............................................................  31
3.1  Impact of non-representative features in classification accuracy  ........  42
3.2  Comparison of clusters produced by the method proposed in [1] and AUFS  .................................................................  44
3.3  Average time complexity of different methods  ..............................  67
4.1  Framework of UFSSF process  ..................................................  81
4.2 Comparison of classification accuracy metrics of different methods on Waveform dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.

4.3 Comparison of classification accuracy metrics of different methods on Spambase dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.

4.4 Comparison of classification accuracy metrics of different methods on Ionosphere dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.
4.5 Comparison of classification accuracy metrics of different methods on Waveform dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric. 

4.6 Comparison of classification accuracy metrics of different methods on Ionosphere dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric. 

4.7 Comparison of classification accuracy metrics of different methods on Spambase dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric.
4.8 Comparison of different methods’ running time for three datasets. The first, second and third rows of the figure show the running times on Waveform, Spambase and Ionosphere datasets respectively. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of arrived streaming features while the y-axis denotes the corresponding time (in seconds) taken by the methods to select a set of features .... 100

5.1 Healthcare application ................................................. 107
5.2 The framework of OUDVFS method ................................. 116
5.3 Illustration of dynamic views proposed structure where instances are rows and features are columns. Each new chunk consists of new instances and an additional view ........ 122
5.4 Time complexity of different methods on Handwritten dataset .... 134
5.5 Time complexity of different methods on Caltech-7 dataset ..... 135
5.6 Time complexity of different methods on Fox News dataset .... 135
List of Tables

2.1 Search strategies for subset generation ........................................... 24
2.2 Characteristics of clustering methods ............................................. 29

3.1 (a) PCC between the centroids and all the features vectors in the feature set. (b) Allocation of a feature to its most similar cluster centroid. ................................................................. 56
3.2 Standard Confusion Metrics for Evaluation of Normal/Anomaly Classiﬁcation ................................................................. 59
3.3 Classification accuracy using Water Treatment Plant dataset with Naive Bayes ................................................................. 61
3.4 Classification accuracy using Water Treatment Plant dataset with IB1 ................................................................. 61
3.5 Classification accuracy using Water Treatment Plant dataset with J48 Decision Tree ................................................................. 61
3.6 Classification accuracy using Spambase dataset with Naive Bayes . . . 62
3.7 Classification accuracy using Spambase dataset with IB1 .................. 62
3.8 Classification accuracy using Spambase dataset with J48 Decision Tree 62
3.9 Classification accuracy using PAMAP2 dataset with Naive Bayes . . . 62
3.10 Classification accuracy using PAMAP2 dataset with IB1 .................. 63
3.11 Classification accuracy using PAMAP2 dataset with J48 Decision Tree 63
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.12</td>
<td>The Average Accuracy of Different Similarity Measures Used for Each Method</td>
<td>65</td>
</tr>
<tr>
<td>3.13</td>
<td>The Average Classification Accuracy of Different Similarity Measures</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Characteristics of existing feature selection methods for streaming features</td>
<td>75</td>
</tr>
<tr>
<td>4.2</td>
<td>The comments on the provided pseudo code in UFSSF algorithm</td>
<td>83</td>
</tr>
<tr>
<td>4.3</td>
<td>Confusion matrix of the classification results</td>
<td>88</td>
</tr>
<tr>
<td>5.1</td>
<td>Characteristics of existing multi-view feature selection methods</td>
<td>109</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of the proposed and benchmarked methods</td>
<td>123</td>
</tr>
<tr>
<td>5.3</td>
<td>Classification accuracy (Recall) using Handwritten dataset with Naive Bayes</td>
<td>129</td>
</tr>
<tr>
<td>5.4</td>
<td>Classification accuracy (Precision) using Handwritten dataset with Naive Bayes</td>
<td>129</td>
</tr>
<tr>
<td>5.5</td>
<td>Classification accuracy (F-measure) using Handwritten dataset with Naive Bayes</td>
<td>129</td>
</tr>
<tr>
<td>5.6</td>
<td>Classification accuracy (Recall) using Handwritten dataset with IB1</td>
<td>129</td>
</tr>
<tr>
<td>5.7</td>
<td>Classification accuracy (Precision) using Handwritten dataset with IB1</td>
<td>130</td>
</tr>
<tr>
<td>5.8</td>
<td>Classification accuracy (F-measure) using Handwritten dataset with IB1</td>
<td>130</td>
</tr>
<tr>
<td>5.9</td>
<td>Classification accuracy (Recall) using Caltech dataset with Naive Bayes</td>
<td>130</td>
</tr>
<tr>
<td>5.10</td>
<td>Classification accuracy (Precision) using Caltech dataset with Naive Bayes</td>
<td>130</td>
</tr>
<tr>
<td>5.11</td>
<td>Classification accuracy (F-measure) using Caltech dataset with Naive Bayes</td>
<td>130</td>
</tr>
<tr>
<td>5.12</td>
<td>Classification accuracy (Recall) using Caltech dataset with IB1</td>
<td>131</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>5.13</td>
<td>Classification accuracy (Precision) using Caltech dataset with IB1</td>
<td>131</td>
</tr>
<tr>
<td>5.14</td>
<td>Classification accuracy (f-measure) using Caltech dataset with IB1</td>
<td>131</td>
</tr>
<tr>
<td>5.15</td>
<td>Classification accuracy (Recall) using Fox News dataset with Naive Bayes</td>
<td>131</td>
</tr>
<tr>
<td>5.16</td>
<td>Classification accuracy (Precision) using Fox News dataset with Naive Bayes</td>
<td>131</td>
</tr>
<tr>
<td>5.17</td>
<td>Classification accuracy (F-measure) using Fox News dataset with Naive Bayes</td>
<td>132</td>
</tr>
<tr>
<td>5.18</td>
<td>Classification accuracy (Recall) using Fox News dataset with IB1</td>
<td>132</td>
</tr>
<tr>
<td>5.19</td>
<td>Classification accuracy (Precision) using Fox News dataset with IB1</td>
<td>132</td>
</tr>
<tr>
<td>5.20</td>
<td>Classification accuracy (F-measure) using Fox News dataset with IB1</td>
<td>132</td>
</tr>
</tbody>
</table>
Abstract

With the proliferation of the data, the dimensions of data have increased significantly, producing what is known as high-dimensional data. This increase of data dimensions results in redundant and non-representative features, which pose challenges to existing machine learning algorithms. Firstly, they add extra processing time to the learning algorithms and therefore negatively affect their performance/running time. Secondly, they reduce the accuracy of the learning algorithms by overfitting the data with these redundant and non-representative features. Lastly, they require greater storage capacity. This thesis is concerned with reducing the data dimensions for machine learning algorithms in order to improve their accuracy and run-time efficiently. The reduction is carried out by selecting a reduced set of representative and non-redundant features from the original feature space so it approximates the original feature space. Three research issues have been addressed to achieve the main aim of this thesis. The first research task addresses the issue of accurate selection of representative features from high-dimensional data. An efficient and accurate similarity-based unsupervised feature selection method (called AUFS) is proposed to tackle the issue of the high-dimensionality of data by selecting representative features without the need to use data class labels. The proposed AUFS method extends the k-mean clustering algorithm to partition the features into k clusters based on different similarity measures in order to accu-
rately partition the features. Then, the proposed centroid-based feature selection method is used to accurately select those representative features.

The second research task is intended to select representative features from streaming features applications where the number of features increases while the number of instances remains fixed. Streaming features applications pose challenges for feature selection methods. These dynamic features applications have the following characteristics: a) features are sequentially generated and are processed one by one upon their arrival while the number of instances/points remains fixed; and b) the complete feature space is not known in advance. A new method known as Unsupervised Feature Selection for Streaming Features (UFSSF), is proposed to select representative features considering these characteristics of streaming features applications. UFSSF further extends the $k$-mean clustering algorithm to incrementally decide whether to add the newly arrived feature to the existing set of representative features. Those features that are not representative are discarded.

The last research task involves reducing the dimensionality of multi-view data where both the number of features and instances can increase over time. Multi-view learning provides complementary information for machine learning algorithms. However, it results in high-dimensionality as the data is being considered from different views. Indeed, extra views would definitely result in extra dimensions. In particular, existing solutions assume that the number of the views is static; however, this is not realistic when dealing with real applications as new views can be added. Therefore, an Online Unsupervised Feature Selection for Dynamic Views (OUDVFS) is proposed. As we are targeting unsupervised learning, we propose a new clustering-based feature selection method that incrementally clusters the views. The set of selected representative features is updated at each clustering step.
Chapter 1

Introduction

The number of dimensions (also called features) of the data has increased significantly in various real applications such as healthcare, social media and online learning [2]. Figure 1.1 depicts the maximum number of dimensions/features of different datasets from 2008 to 2014 in the UCI repository. It shows that the number of features varies from 857 to 3,231,961 which is a very high number. Machine Learning (ML) methods have been widely applied to high-dimensional datasets for various learning tasks such as classification, clustering, pattern recognition and recommendation [3]. However, this increase in data dimensions would result in non-representative/irrelevant and redundant features as not all features would be relevant for the machine learning tasks. For example, an application for weather forecasting will have a feature space consisting of a few features such as humidity, temperature, wind and sensor ID. In this case, the sensor ID would mislead the learning task (i.e. overfitting the data) and consequently result in poor learning as the sensor ID is not relevant to the learning task. Note that in all chapters, the terms relevant and representative are used based on whether the learning is supervised (i.e. requiring data class labels) or unsupervised (i.e. not requiring data class
Motivation

If the learning is supervised the term relevant/irrelevant is used. Otherwise, the term representative/non-representative is used.

Figure 1.1: high-dimensional datasets in UCI repository (Bolon-Canedo et al. 2015)

1.1 Motivation

The three major factors that play an essential role in ensuring that machine learning algorithms work efficiently in high-dimensional data are: classification accuracy, storage capacity and time complexity [4]. An efficient machine learning algorithm needs to learn from the data so that it is able to accurately classify it. It should store only representative features and not redundant ones. The learning process should be efficient in terms of running time. However, high-dimensional data is more likely to have redundant features as well as features that are not relevant to the learning task, resulting in extra memory usage (i.e. storage waste) and extra running time. Additionally, they would reduce the classification accuracy of the
learning algorithms when using these irrelevant/non-representative features during the learning task. Hence, these non-representative and redundant features need to be removed and not included during the learning task.

Dimensionality reduction techniques have been proposed in order to remove the redundant and non-representative features to approximate the original feature space well. Therefore, the reduced feature space is used as input for machine learning algorithms instead of using the entire feature space. The intuition is that the reduced selected set of features achieves better or similar results as those using the entire feature space [5]. There are two main approaches to dimensionality reduction, namely feature extraction and feature selection [6] [7]. The former projects the original feature space into a new reduced feature space [7]. As methods of this approach transforms the original features into new features; however, they are not efficient for applications where the original features should be identified prior to further analysis [8]. Text clustering is a real application where the original text should be maintained for further learning. Also, feature extraction is not as efficient as feature selection in solving the problem of redundancy as redundant features might be included in the transformation phase [9]. Representative examples of such methods include, but are not limited to, Singular Value Decomposition (SVD) [10], Linear Discriminant Analysis (LDA) [11] and Deep Learning as in [12].

On the other hand, feature selection methods select a subset from the original features [13]. Therefore, feature selection methods are preferable and more interpretable for real applications such as text mining because data loses its interpretability if it is transformed into a new text. Feature selection can be carried out either by ranking the features based on particular criteria and adopting the top \(N\) features or by selecting the minimum subset of features. This depends on a pre-
defined threshold to determine the number of features [14]. The selected features should represent the entire feature space well and not contain redundant features. As a result, dimensionality reduction methods help machine learning algorithms by providing: a) better learning (e.g. better classification accuracy), b) better storage capacity and c) better time complexity. Examples of feature selection methods include but are not limited to Fisher Score (FS) [15], Laplacian score [16] and SPEC [17].

1.2 Existing Feature Selection Problems

There are several ways to organise and categorise existing feature selection problems. For example, they can be categorised based on supervised or unsupervised problems. Another way is to categorise the problems based on the type of evaluation criteria to filter or wrapper problems. However, with the era of big data, new feature selection problems have emerged when selecting features from new data perspectives such as heterogeneous data, linked data and streaming features application data [18]. In fact, each of these problems has its own characteristics. Therefore, Li et al. (2017) proposed a new categorisation for feature selection problems that is based on the data perspective as shown in Figure 1.2. However, we further have extended that categorisation to also include multi-view data within the streaming environment, a problem which is addressed in Chapter 5.

There are various existing feature selection problems from different data perspectives. Below, we first describe these problems and then summarise existing solutions and their limitations.
Figure 1.2: Categorisation of feature selection problems from a data perspective (Li et al. 2017)
Existing Feature Selection Problems

Homogeneous Data

Homogeneous data suffers from redundant and non-representative features, which result in large volume and high-dimensional data. Therefore, it degrades the accuracy and the performance of machine learning algorithms [19]. In homogeneous data, both the number of instances and the number of features is fixed. Consequently, feature selection has been proposed to reduce the dimensionality of the data. Homogeneous data, in terms of data structure, can be grouped as flat features and structured features [20]. Flat features, which are the focus of Chapter 3, are the traditional features where every feature is a column vector and every row is an instance. On the other hand, structured features are a special case of flat features where features form a structure (e.g. tree or graph). Therefore, applying traditional feature selection methods [21, 22, 23, 24] which are designed for flat features might ignore the structure and therefore result in the selection of non-representative features.

Heterogeneous Data

Traditional feature selection methods are designed to reduce the number of dimensions generated from a single source data. However, data can come from multiple sources, which brings additional challenges due to the increase in the number of dimensions with redundant and non-representative features. Heterogeneous data faces three problems, namely linked data, multi-source data and multi-view data [25]. Feature selection methods designed for linked data are limited to applications where link information can be established [26]. An application example of linked data is Twitter where link information (e.g. hyperlink) can be established between the tweets. In multi-source data, the same set of data instances can come
from multiple sources while they have the same set of features. It is often used in gene applications [27].

In multi-view data, which is the focus of Chapter 5, data flows from heterogeneous sources, which are called views in the literature, and therefore it is more likely to provide complementary information than single-view data [28]. Unlike multi-source data, each instance in multi-view data is represented by different groups of features. In other words, different views are different representations of the same set of instances and each view is a group of features. For example, in medical applications, patients (i.e. instances) may obtain different groups (i.e views) of laboratory tests (i.e. features) at different clinics. Therefore, performing feature selection on the views can lead to more precise medical diagnostic results. However, not all features in different views are representative and redundancy might occur between the views. Therefore, they would result in the problem of high-dimensionality.

Streaming Data and Streaming Features

Data streaming has become ubiquitous in real applications. However, data streams are more likely to have more redundant and non-representative features than do homogeneous data. Therefore, this would produce high-dimensionality. High-dimensionality presents a major challenge to the efficient performance of machine learning algorithms in data stream environments, as non-representative and redundant features decrease the prediction/classification accuracy and the running time of the learning algorithms. Data streams can be broadly classified into streaming data and streaming features [20]. In streaming data, the number of features is fixed, while the instances arrive sequentially. Regarding streaming features, however, which is the focus of Chapter 4, the number of instances remains fixed, while the features arrive sequentially and are processed one by one. In real applications
such as Twitter, features such as slang words are dynamically created and therefore need to be processed upon their creation instead of waiting for all features to arrive, as required by traditional feature selection methods. It is impractical to wait for the arrival of all features before starting the selection process is impractical, as the number of streaming features is unknown in advance and new features appear over time.

Data streams bring new challenges to traditional feature selection methods, which are methods designed for static data. Traditional unsupervised feature selection methods [29, 30, 31] are not appropriate for streaming features as the number of features changes with time and is not fixed compared with static data. They require the full feature space to be known in advance, which can be impractical in streaming features. Technically, when applied to streaming features, they need to store large amounts of data. However, this can be infeasible due to the tremendous size resulting from the data streams. Traditional feature selection methods have greater computational complexity, which makes them inappropriate when working with high-dimensional streaming features as they require fast and real-time processing. Moreover, in streaming features applications, algorithms should read the data only once due to the finite amount of storage space, and then non-representative features should be removed to allow storage. Finally, traditional feature selection methods are static by nature, meaning that they do not dynamically update their selected representative features [32]. Therefore, this negatively affects the representativeness of the selected features. Hence, it is essential to consider the specific characteristics of streaming features when designing a feature selection method for such application.
1.3 Existing Solutions

The problems highlighted in red in Figure 1.2 are the ones that we are particularly interested to research as they are recent problems of research into feature selection methods. They introduce new challenges for traditional feature selection methods, which need to be addressed for real applications. Below, we summarise existing solutions to the highlighted problems.

Existing Feature Selection Methods for Homogeneous Data

Many feature selection methods [21, 22, 23, 24] have attempted to address feature selection in homogeneous data. Although they reduce the dimensions of the data by filtering out redundant and non-representative features, most of them require labeled data [26]. These are called supervised methods. Most of the high-dimensional data are not labeled, making existing methods unsuitable. The methods proposed in [1] and SPEC [17] are probably the two of the most well-known unsupervised methods (i.e. they do not require data class labels) used to select the representative features. However, they have accuracy and performance limitations. In terms of accuracy, the method proposed by Mitra et al. [1] partitions the feature space using $k$-NN clustering. However, $k$-NN is inefficient when data is not dense as it produces low quality clusters [33]. Therefore, it is not suitable for high-dimensional data because it is mostly non-dense data. Consequently, the classification accuracy will be low because of the badly selected representative features. On the other hand, SPEC [17] has not addressed the issue of feature redundancy because it evaluates individual features and does not consider redundancy, which would negatively affect the classification accuracy.

In terms of performance, both of the methods proposed in [1] and SPEC [17]
Existing Solutions

experience high computational time complexity. Because the method proposed in [1] uses $k$-NN, it inherits the computational issues of such an approach because it calculates the distance between $k$ and all its neighbours. On the other hand, SPEC [17] also suffers from high time computational complexity as it is based on spectral graph theory, which is computationally expensive [34]. Although these two methods can be applied to high-dimensional data, they experience computational complexity.

**Existing Feature Selection Methods for Multi-view Data**

There are two ways to apply existing feature selection methods on multi-view data, namely *indirect* and *direct approaches*. In the *indirect approach*, all the views are concatenated into one single matrix. Then, traditional/single-view feature selection methods, which are not designed for heterogeneous data, can be applied on this single matrix. An example of such methods includes but is not limited to Fisher Score [35], sparse multi-output regression [36], Laplacian Score [16], SPEC [17] and Multi-Cluster Feature Selection [37]. However, this indirect approach of the aforementioned methods is inefficient for multi-view learning. This is because, by concatenating the views, these methods disregard the correlation among the data views (i.e. lacks of its physical meaning) which would result in an inaccurate representation of the features [38].

In the *direct approach*, feature selection methods are designed to select features from multi-views data. There are a few well-known unsupervised feature selection methods for multi-view applications such as AUMFS [39], OMVFS [40] and SRRS [41]. They were designed to tackle the problem of multi-view learning. In addition, none requires data class labels in order to select representative features. OMVFS is the only method that works in an online environment. However, these
Existing Solutions

methods assume that the number of the views is static. By static we mean that all the views are completely exist in advance and there are no new views that can be added. However, this assumption is not valid for real applications as new views can be added at any given time. Also, the number of instances can increase too (i.e. online). To the best of our knowledge, this is still an open issue that has not been addressed.

Existing Feature Selection Methods for Streaming Features

Several studies have been conducted on feature selection in streaming features applications. Perkins et al. [42] proposed a method, called *grafting*, which selects a subset of streaming features that have arrived so far as an integral part of a regularised learning process. It incrementally and gradually builds the selected subset of features in addition to training the predictive model using gradient descent. Because it works in an incremental way, this method can efficiently cope with the dynamic nature of the streams. However, in order to specify a good regulariser parameter value, this method requires knowledge about the complete feature space in advance. Therefore, it cannot process streaming features of an unknown size. Alpha-investing [43] evaluates the relevance of the arrived feature based on a dynamic threshold of error reduction (called $p$-value). In particular, the $p$-value is introduced to determine whether or not to add a feature to the selected set of features. Although Alpha-investing can process streaming features of an unknown size, no selected features can be removed. However, this can be an issue if the data stream experiences data drifts as the selected feature can no longer be representative. Finally, Online Streaming Feature Selection (OSFS) was proposed in [44] to select relevant features and remove redundant ones in real time. Whenever a feature arrives, OSFS measures its dependency on the available class labels and
Research Questions

then adds the feature to the best candidate feature if this meets a specific criterion. OSFS can dynamically remove redundant features using the Markov Blanket.

The methods discussed above require the class label as a guide to select representative features. However, in real applications most of the data is un-labeled and, moreover, labeling is time consuming. To the best of our knowledge, the only method that is unsupervised (i.e. no labels are needed) and is applicable for streaming features applications is proposed in [45]. Although this method has good performance, it is limited to scenarios where link information must be established (i.e. a friendship relationship between Twitter users). Also, the authors assume that the link information is stable, which obviously is not true as this could dynamically change.

1.4 Research Questions

This section introduces the addressed research questions based on the limitations discussed in Section 1.3. There are three core research questions in this thesis and they are as follows:

(A) **How to design an efficient and accurate feature selection for high-dimensional data without the need of data class labels?**

This research question addresses the issue of how to reduce the data dimensions for better learning. In particular, it is concerned with how to accurately selecting a reduced set of representative features such that it approximates the original feature space. This set of features should not have any redundant or non-representative features. The significance is that an accurate selection of representative features would help machine learning algorithms to perform more efficiently (i.e. time complexity) and achieve better classification accu-
Research Questions

1. How to design an efficient feature selection for streaming features applications without the need of data class labels?

This research question concerns the way to apply feature selection for streaming features applications. Streaming features applications have specific characteristics, which are different from those of homogeneous data. The number of features can increase and is not fixed compared to homogeneous data. The full feature space is not known in advance and features arrive and are processed one by one in real-time. Additionally, the selection of representative stream features should be done in an acceptable running time as it requires fast and real-time processing. Moreover, in streaming features applications, algorithms should read the data only once due to the finite amount of storage space, and then non-representative features should be removed to allow storage. Feature selection methods for streaming features should dynamically update their selected representative features when new features arrive. Due to the nature of data streaming, most data are not labeled [32]. Therefore, these characteristics need to be taken into account when designing a feature selection method for such applications.

2. How to design an online feature selection for multi-view data so that the views are dynamic and data class labels are not required?
This research question addresses the issue of designing an online feature selection for dynamic views in multi-view data. Although multi-view data provides complementary information for machine learning algorithms, it results in high-dimensional data. All existing multi-view feature selection methods assume that the number of views is fixed/static. However, this assumption is not true because in real applications, the same set of instances can be represented by new views at any given time. Also, the instances can increase at any time. The challenge here is to select features incrementally where both features and instances increase over time. In addition, the selection of the features should be done without the need for data class labels.

The above three research questions can be integrated as follows. The first research question addresses the problem of feature selection in a static setting. We mean by static is that both the number of features and the number of instances is fixed. In the second research question, we address the problem of feature selection for streaming features applications where the number of features increase while the number of instances is fixed. In the last research question, we address the problem of feature selection of multi-view data where both the number of features and instances can increase over time.

1.5 Summary of Contributions

This section summarises the main contributions of this work in addressing the research questions. There are three main contributions that focus on reducing the data dimensions from three different data perspectives. The adopted reduction approach is feature selection because we want to select a set of representative and non-redundant features from the original feature space. Unlike feature extraction,
feature selection approach does not project the features into new features and therefore it is more appropriate for real applications. In real applications, it is important to identify the features (i.e., not to be transformed) for further learning and analysis.

1. Designing an efficient and accurate feature selection for high-dimensional data

Chapter 3 proposes an unsupervised feature selection method for high-dimensional data (called AUFS). To overcome traditional unsupervised feature selection methods, we proposed a feature selection method for high-dimensional data that: a) does not require data class labels in order to select the representative features (i.e., unsupervised); b) is accurate in selecting representative features. This results in improved classification accuracy; and c) does not require any search for different subset of features. The result is a more efficient run-time. Technically speaking, we proposed a centroid-based methodology for selecting representative features from clusters. The features are partitioned into clusters based on different similarity measures. Then, a feature, which has a minimum distance to its cluster centroid, is selected from each cluster. AUFS was tested on real datasets. The following papers have been extracted from this contribution.


- N. Almusallam, Z. Tari, P. Bertok, A. Zomaya: An efficient and Accurate unsupervised Feature Selection From High-Dimensional Data. Fu-
Summary of Contributions

2. Designing an efficient feature selection for streaming features applications

Chapter 4 proposes an unsupervised feature selection method for streaming features applications called UFSSF. UFSSF overcomes feature selection methods for streaming features by: a) selecting representative features without the need for data class labels; b) selecting representative features in real time as they arrive (the entire feature space is not known in advance and they arrive sequentially); and c) it does not require link information in order to select representative features. Therefore, it is more applicable for a wider range of streaming features applications. The UFSSF method extends the $k$-mean algorithm to cluster a stream of features that are not known in advance. It integrates three linearly-dependent similarity measures, namely Pearson Correlation Coefficient (PCC), Least Square Regression Error (LSRE) and Maximal Information Compression Index (MICI), to incrementally measure the dependency of the newly-arrived streaming features to decide whether or not to add them to the existing set of representative features. The features arrive sequentially and they are processed upon their arrivals one by one in a real-time manner. Linearly dependent measures are used because they are not sensitive to the order and the scatter of the distribution of the features. Additionally, UFSSF incrementally updates the centroids to cope with concept drift in streaming features, as one feature might be representative only for a given time. After assigning a feature to its relevant cluster, the mean is updated and we compare the similarity of the arrived feature with the exist-
ing representative feature of the cluster. UFSSF was tested on real datasets. We experimentally simulated the streaming features environment where the number of the features increases while the number of the instances is fixed.

The following papers have been extracted from this contribution.


- N. Almusallam, Z. Tari, J. Chan, A. Alharthi: An Efficient Unsupervised Feature Selection for Dynamic Features. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* - (SJR Rank: Q1) - Submitted in Oct 2018 - Chapter 4

3. Designing an online unsupervised feature selection for dynamic views

Chapter 5 proposes an online unsupervised feature selection method for dynamic multi-views data (called OUDVFS). The proposed OUDVFS overcomes existing multi-view feature selection methods in the following ways: a) it does not require data class labels in order to select representative features from dynamic views; b) unlike existing feature selection methods for multi-view data, the OUDVFS is more appropriate for real applications as it selects features where the views can increase over time (i.e dynamic views); and c) the instances increase as well (i.e.online). The OUDVFS consists of two parts: clustering and feature selection. The chunk can have new instances or new views. If the chunk has only new instances, OUDVFS incrementally clusters the new instances with the clusters resulting from a previous chunk.
Similarly, when the chunk has a new view (i.e. set of features), the OUD-VFS incrementally clusters the new view with the clusters resulting from a previous chunk. This method relies on clustering to select features as it does not require the data class labels in order to group the data. Specifically, the OUDVFS relies on hierarchical clustering in order to merge the clusters to the required reduction of data dimensions. The selected set of representative features is updated at each clustering step. The OUDVFS was tested on real multi-view datasets and we experimentally simulated the increasing views and the increasing instances to simulate the real multi-view applications. The following paper has been extracted from this contribution.


## 1.6 Organisation of The Thesis

The contributions of our research are addressed in six chapters including this Introductory chapter. The remaining chapters of the thesis are structured as follows:

- **Chapter 2** describes three main background components of the thesis, namely dimensionality reduction, clustering-based methods and data-driven intrusion detection systems. The integration of these three components facilitate a better understanding of the thesis.

- **Chapter 3** proposes an efficient and accurate unsupervised feature selection method for high-dimensional data (AUFS). It extends the $k$-mean clustering algorithm with different similarity measures in order to partition
the feature space. Then, a centroid-based feature selection method is proposed to accurately select a reduced set of representative features. **In this chapter, both the number of instances and the number of features is fixed.**

- **Chapter 4** proposes an efficient *unsupervised* feature selection for streaming features applications (UFSSF). It processes streaming features where feature space in not known in advance. Rather, features arrive one by one in real-time. The features are clustered incrementally and the selected set of features is updated dynamically. **In this chapter, the number of instances is fixed while the number of features is dynamic.**

- **Chapter 5** proposes an online *unsupervised* feature selection method for dynamic views data. In multi-view data, the same set of instances is represented by multiple views allowing a comprehensive look at the data. In this chapter, the number of views is dynamic, which means that new views can be added incrementally. Also, the number of instance increases in the online mode. Therefore, the set of selected features is updated dynamically. **In this chapter, both the number of instances and the number of features is dynamic.**

- **Chapter 6** summarises the main contributions of this thesis. Also, suggestions are offered for future research work, which could be conducted for feature selection from various data perspectives.

The three core chapters (Chapters 3-5) are presented in a standalone and self-explanatory manner. Therefore, the relevant contexts including related work, description of method, experimental results and discussion are presented in each of these chapters separately.
Chapter 2

Background

This chapter provides the necessary background which will enable the reader to better understand the various chapters of this thesis. It briefly describes and reviews the progress that has been made in three fields, namely Dimensionality Reduction, Clustering-Based Methods and Data-Driven Intrusion Detection Systems (IDS). These three areas will hopefully provide the reader with a comprehensive background that will facilitate an understanding of the work carried out in this thesis.

2.1 Dimensionality Reduction

In the era of big data, the dimensions of data increases significantly. In particular, the number of features increases such that not all features are representative for the learning machines. In addition, feature redundancy is more likely to occur. There are various challenges resulting from the existence of non-representative and redundant features in the data. Firstly, they reduce the accuracy of the data mining algorithms by misdirecting the classification and clustering tasks [46]. Also, the existence of the redundant and non-representative features would negatively affect
the performance of the algorithms due to the large volume of data [47]. Moreover, they increase the processing time of the data mining algorithms, which would result in very expensive complexity [48]. Furthermore, a large storage capacity is required to store the large volume of data [49]. Finally, the curse of dimensionality is a challenge for feature selection algorithms due to the sparseness of the data, which would deceive the mining algorithms by appearing to be equal in terms of the distance between them [50]. Consequently, various researchers have proposed feature selection as an efficient technique which would help to address the aforementioned challenges.

The feature selection process comprises (i) subset generation, (ii) subset evaluation, (iii) stopping criterion and (iv) result validation [14]. This process is illustrated in Figure 2.1.

Figure 2.1: An overview of the feature selection process
Subset generation searches for a set of features based on a particular strategy in readiness for the evaluation at the next step. The three main types of search strategy, in addition to their strengths and weaknesses, are illustrated in Table 2.1.

Table 2.1: Search strategies for subset generation

<table>
<thead>
<tr>
<th>Complete Search [51]–[52]</th>
<th>Sequential Search [53]–[54]</th>
<th>Random Search [55]–[56]</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Starts with an empty feature set, and adds the features for the purpose of the evaluation and vice versa.</td>
<td>• Starts with an empty feature set, and adds one feature at a time until it reaches the stage when the features no longer enhance the quality of the subset features.</td>
<td>• Starts the search by selecting random subsets to be produced for the evaluation.</td>
</tr>
<tr>
<td>• Pros: guarantees the search for the optimal result based on the adopted evaluation criteria.</td>
<td>• Pros: it is simple to implement and obtains results quickly.</td>
<td>• Pros: ensures the global optimization of the selected subset.</td>
</tr>
<tr>
<td>• Cons: exhaustive search, which induces performance overheads.</td>
<td>• Cons: It does not produce optimal features set.</td>
<td></td>
</tr>
</tbody>
</table>

Subset evaluation is the second step of the feature selection process, where every generated candidate features is evaluated for its quality based on a specific evaluation criterion [57]. Evaluation criteria are broadly classified into filter and wrapper approaches whether or not the data mining algorithms are to be applied in the evaluation of the selected features [58]. The filter approach [59]–[60] relies on the general characteristics of the data to evaluate the quality of the generated candidate features without involving any data mining algorithm. This includes, but is
not limited to, distance, information, correlation and consistency measures. Filter-based algorithms have faster processing time than wrapper-based algorithms, as they do not include any data mining algorithm [61]. Conversely, the wrapper-based algorithms [62]–[63] require the use of specific data mining algorithms such as clustering in the evaluation process of the generated candidate features [64]. Despite the fact that the wrapper approach can discover better quality candidate features than does the filter approach, this incurs high computational overheads [49].

Subset generation and evaluation of the feature selection process is iteratively repeated until they meet the requirement of the stopping criterion. The stopping criterion is activated by the completeness of the search, a pre-set maximum iteration times or when the classification error rate is less than the pre-set threshold [65]. Then, the selected best candidate features are validated by conducting before and after experiment testing of different aspects such as classification error rate, number of selected features, the existence of redundant / non-representative features and the time complexity [14].

Based on the availability of the class labels, feature selection methods fall into two categories: supervised and unsupervised. The former e.g. [21, 22, 23, 24] assesses the significance of a feature by computing the correlation to its class label. It is often difficult to have the data class labels, especially for high-dimensional datasets, as it would take experts a long time to test and label the data. Therefore, unsupervised feature selection methods e.g. [66, 67, 68, 69, 70] have been introduced as a solution to this problem. Such methods are much harder to design due to the absence of data class labels, which guide them in the process of searching for finding the representative features. Initially, traditional unsupervised feature selection methods addressed the problem (of the absence of class labels) by ranking features independently based on certain scores. However, they are not able
to generate the best features set, as they do not compute the correlation between features [69]. The second of the unsupervised feature selection methods e.g. [71, 67] generally uses clustering to partition the features set into distinct clusters, where features in every cluster are similar to each other and dissimilar to the features of other clusters.

Feature selection methods, which are designed for static data, are not capable of efficiently working in data streams. This is because data streams have specific properties that do not exist in static data. A feature selection method should take into account the following properties in order to work efficiently in data streams. It should be restricted to read the data only once as it is impossible to store the entire stream. Also, it should take into account that many stream applications stream the features one-by-one and do not assume the existence of the entire feature space in advance (called dynamic feature space or streaming features). An feature selection method has to incrementally measure and update the representativeness of the features, as one feature might be representative in a time $t$ but not in $t+1$ (concept drift) [72] [73]. Furthermore, it is not enough to reduce the feature space from the stream; the instances must be reduced as well because they usually contain great amounts of noise, redundancy and non-representativeness. Finally, a feature selection method should not be limited to data class labels; instead, it should be (unsupervised), as the data class labels are not available for most applications.

There are very few feature selection methods that work in data stream applications. Every method contains some properties but not all of them. The OSFS [32] handles a stream of features one by one as they arrive. However, it requires the data to be labeled; it removes irrelevant/redundant features but not instances and only works for a single data stream. By contrast, Kankanhalli et al. [74] selects a subset of relevant features from multiple streams based on the Markovian decision
problem. However, it requires the full feature space to be known in advance and the
data to be labeled, and removes irrelevant/redundant features but not instances.
Toshniwal et al. [75] developed an unsupervised feature selection method that does
not require the data labels in order to select the non-representative features. It
is designed primarily for the purpose of outlier detection. However, it does not
handle stream features one by one as they arrive; it removes irrelevant/redundant
features but not instances, and works only for a single data stream. Finally, the
Zhang et al. [76] method incrementally measures and updates the relevance of the
features in order to accurately evaluates their relevance. On the other hand, it
requires the full feature space to be known in advance and is designed to work only
for a single data stream.

2.2 Clustering-Based Methods

In this section we provide a brief description and categorisation of clustering meth-
ods. Clustering is an approach whereby data points are grouped into different
clusters, so that points within a cluster are very similar to each other and different
from the data points of other clusters. Clustering methods do not require data
class labels in order to partition the feature space and therefore they are widely
used for unsupervised feature selection [77] [72]. As our proposed feature selection
methods, described in Chapters 3, 4 and 5, are intended for unsupervised learning,
clustering methods are used to select representative features without the need for
data class labels. Clustering methods can be broadly categorised into partition-
ing methods where data is portioned into groups based on similarity or distance,
density-based methods where data is partitioned into groups based on the density
of the data, hierarchal methods where groups data based on either agglomerative
or divisive strategy, and grid-based methods where data is assigned to cells and clustering is performed on each cell. Table 2.2 provides a categorisation of these methods as well as their characteristics.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>• Use mostly a distance-based, where the dataset is partitioned into $n$ parts, each representing a cluster with minimum data points.</td>
</tr>
<tr>
<td></td>
<td>• Each object is allocated to only one cluster.</td>
</tr>
<tr>
<td></td>
<td>• Does not maintain any hierarchal structure.</td>
</tr>
<tr>
<td></td>
<td>• Adopts iterative relocation mechanism in the partitioning to produce “optimal” results.</td>
</tr>
<tr>
<td></td>
<td>• Works efficiently with small to medium size datasets.</td>
</tr>
<tr>
<td></td>
<td>• $k$-means is an example clustering algorithm used as a partitioning method.</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>• Clustering is maintained based on hierarchal decomposition of the dataset.</td>
</tr>
<tr>
<td></td>
<td>• It is either agglomerative or divisive decomposition.</td>
</tr>
<tr>
<td></td>
<td>• Uses either distance-based or density-based.</td>
</tr>
<tr>
<td></td>
<td>• Clusters cannot be corrected when they have been merged or split.</td>
</tr>
<tr>
<td>Density-based</td>
<td>• Has been defined under proximity-based methods.</td>
</tr>
<tr>
<td></td>
<td>• Has good accuracy in detecting outliers.</td>
</tr>
<tr>
<td></td>
<td>• Capable of discovering clusters with arbitrary shape as it is based on density, not distance.</td>
</tr>
<tr>
<td></td>
<td>• DBSCAN clustering algorithm used as density-based algorithm.</td>
</tr>
<tr>
<td>Grid-based</td>
<td>• The feature space is divided into a limited number of cells to form the grid.</td>
</tr>
<tr>
<td></td>
<td>• Clustering operations are performed inside the cells.</td>
</tr>
<tr>
<td></td>
<td>• Has fast processing time, as complexity depends on the number of grid cells and not the number of instances.</td>
</tr>
</tbody>
</table>
2.3 Data-Driven Intrusion Detection Systems (IDS)

Feature selection is a pre-processing step that helps to optimise the performance of machine learning algorithms in achieving their tasks. For example, when grouping the data into normal and outlier groups as in intrusion detection applications, the existence of redundant and non-representative features would reduce the accuracy of classifying the data points and would also increase the processing time. Therefore, feature selection is applied as a pre-processing step for IDS in order to increase the classification accuracy and reduce the running time. This section provides an overview of IDS as we used IDS datasets in Chapters 3 and 4.

There are various security mechanisms (e.g. firewalls, cryptography or access controls), which have been designed mainly to protect computer or information systems from malicious attacks. In addition to those security mechanisms, IDS has been developed as a second-line defence to discover attacks after they have been successfully launched [87]. IDS can be host-based (e.g. to monitor the logs), network-based (e.g. to monitor the networks traffic flow) or data-driven (e.g. to detect any deviations from the normal pattern of the data), which is the focus of our interest.

Broadly, IDS is classified in terms of detecting intrusions into signature-based and anomaly-based [88]. The signature-based ID approach [89]–[90] discovers suspicious behaviours by comparing them with pre-defined signatures. Signatures are patterns associated with attacks, which are verified in advance by the human experts and used to trace any suspicious patterns. If the suspicious patterns and the signatures match, an alarm is activated to warn the administrators or to take a pre-defined actions in response to the alarm [84]. The algorithms that are signature-based ID are efficient in detecting known attacks with low false alarms and are
reasonably quick to do so. Despite the fact that most existing commercial IDs are signature-based, most of them cannot detect new types of attacks (also called un-known attacks), as their signatures are new and not known in advance [91].

Unlike the signature-based IDS algorithms, anomaly-based IDS algorithms [92]–[93] can identify new attacks because they “appropriately” model the ‘normal’ behaviour of a non-attacked system. They can therefore identify serious deviations from the normal profile to be considered as anomalies (also called outliers) [94]. Anomalies can emerge as a result of fraudulent behaviour, mechanical faults or attacks [95]. Figure 2.2 illustrates how the majority of the data points (triangle points) have a particular distribution, while the circle points have a significant deviation from the rest. The circle points are considered as outliers.

![Figure 2.2: Deviation of circle points (anomalies/outliers) from the normal triangle ones](image)

Anomaly-based IDS techniques can be categorised under three approaches based on the form of the input data they use: supervised anomaly detection [96][97],
semi-supervised anomaly detection \cite{98} \cite{99} and unsupervised anomaly detection \cite{100} \cite{101}. Supervised-based anomaly detection approaches require training data in advance along with their class labels for both normal and abnormal data, so as to accurately detect anomalies. The model is then trained with both classes and applied to unlabeled data to determine the class to which it belongs. Although there are plenty of classification methods that could be applied in this category, the classes of the data are un-balanced because the “normal class” is much bigger than the “anomaly class”, which therefore negatively affects the detection recall. Additionally, it is challenging to find accurate and representative data class labels, particularly for the anomalies, as they emerge periodically and they are uncountable \cite{102}. On the other hand, semi-supervised anomaly-based detection approaches require only one class label, which is either normal or outlier. The corresponding model is trained with the normal class only, and then any instance that does not belong to that class would be classified as an outlier. These approaches are much more applicable than supervised ones because they do not require the specification of anomalous behaviour. In addition, as the models for semi-supervised techniques could also be trained with anomaly class only, this provides substantial limitations because it is difficult to recognise all anomalies for the training of the data \cite{103}.

Both of the aforementioned approaches are limited as they rely on the availability of labeled data. Hence, they are restricted for specific applications such as spacecraft fault detection and therefore they are not generic. On the other hand, the unsupervised anomaly detection approach is generic and widely applicable as it does not need the data to be labeled \cite{104}. This approach assumes that the normal data has a pattern that is significantly different from the pattern of the outliers. For instance, the normal data should form groups with instances that are very similar to each other and different from the outliers. Although this approach is widely
applicable, the related techniques experience a high rate of false alarms [105].

Anomaly-based ID can mainly be categorised into classification methods, statistical methods, proximity-based methods and clustering methods. Classification methods [106][107] are supervised by nature, and they are applicable only if there are class labels in the training data. The classifier is trained with the labeled data and then applied for the testing of un-labeled data. The test data is then classified as an outlier if it is not classified as normal by the classifier. Classification methods seem to provide good accuracy in distinguishing between data and their related classes. Although such methods demonstrate good performance during the testing phase in comparison to the other methods, their detection accuracy depends on the accuracy of the labeled data [88].

Statistical methods [108][109] are another type of approach, which observe the activity of the data so as to create profiles representing acceptable behaviour. There are two kinds of profiles: current and stored profiles. The former regularly logs and updates the distribution of the data as long as the data is processed. Additionally, the data is assigned with an anomaly score by comparing them with the stored profile. If any anomaly score exceeds a pre-defined threshold, it is labeled as an outlier. Statistical methods do not need knowledge about labeled data or attack patterns in advance. Hence, they seem to be efficient in detecting recent attacks. On the other hand, it is difficult to establish a threshold that balances the occurrence of false positives and false negatives [110].

Proximity-based methods use distance metrics to calculate the similarity between data. It assumes that the proximity between an outlier and its nearest neighbour is different from its proximity to the remaining data. Such methods can be either distance-based or density-based. Distance-based methods [111][112] search for a minimum pre-defined number of neighbours of a data point within a
specific range in order to decide its normality. The point is labeled as an outlier if the neighbours within the range are less than the pre-defined threshold. On the other hand, density-based methods \cite{113}\cite{114} compare the density of data with its neighbour densities so to decide its normality. The point is labeled as an outlier if its density is considerably less than the density of its neighbours. Generally, the effectiveness of proximity-based methods varies depending on the adopted measure as it is challenging to ensure effectiveness in particular situations. Furthermore, proximity-based methods seem to be inefficient in detecting outliers that form groups and are close to each other.

Lastly, clustering methods \cite{115}\cite{116} work in unsupervised mode to recognise patterns of un-labeled data by grouping similar instances into groups. They cluster data by examining their relationships with other clusters. Indeed, normal data are those data that belong to clusters that are dense as well as large. On the other hand, outliers can be identified based on the three assumptions \cite{95}: 1) outliers are objects which have not been allocated to any cluster. In fact, the initial goal of clustering is to find clusters in particular, not the outliers; 2) outliers are objects that are far, in terms of measured distance, from their closest cluster centroids. Indeed, every object is given a score based on its distance to its closest cluster centroid and it should not exceed a pre-defined distance in order to be considered as normal. The limitation of this assumption is that outliers cannot be found if they have already formed a cluster. The aforementioned assumptions have a common limitation in that they seem to detect only individual outliers but not groups of outliers, which form clusters by themselves \cite{102}. To overcome this limitation, 3) the last assumption defines the outliers as objects, which have been allocated to sparse or small clusters.

Generally, clustering methods do not require the data to be labeled so it can
handle zero-day attacks. Also, it can adapt to cluster “complex objects” by adopting existing clustering algorithms that can handle those particular types of objects. Furthermore, clustering methods are fast in the testing phase because every object is compared with the clusters only, which are relatively small in comparison with all the objects. On the other hand, the efficiency of clustering methods depends on the clustering algorithms in establishing the normal behaviour of the objects. Also, clustering methods work efficiently when the outliers are individuals but not when they form groups of clusters. Finally, clustering methods are still computationally expensive even with some recent work attempting to resolve the performance problem [117].

**Anomaly Detection for Multiple Data Streams**

Anomaly detection is no longer limited to statistical databases due to the emergence of very large data (Big Data) with specific characteristics: Volume, Variety, Velocity (3V). Volume relates to the huge amount of data generated. Such data can be found in different formats such as videos, music and large images. Velocity refers to the high speed at which data is generated, captured, and shared. Variety refers to the proliferation of new data types. The real world has produced big data in many different formats, posing a challenge that needs to be addressed. A data stream is an ideal example of big data because: a) a huge (Volume) of data is gathered from different sources (i.e sensors) to extract knowledge by mining and analysing the collected big data; b) a data stream arrives in a timely manner at different speed rates (Velocity); c) sensors can stream different data types (Variety).

Although anomaly detection for data streams has been investigated intensively, most of the recent research has focused only on single data stream. Therefore, we
believe it is crucial to investigate how to detect anomalies or launched attacks arriving from *multiple data streams*. In fact, attacks like Denial of Service (DoS) might cause severe damage to the systems if they have been flooded through multiple streams. Therefore, anomaly detection algorithms need to be improved and adapted to multiple data streams. A *data stream* is defined in [118] as a set of infinite data points that consist of attribute values along with an implicit or explicit timestamp. Anomaly-based ID methods are applied to detect outliers from not only a single stream but also from various data streams. This is often carried out by mining the relationships between those multiple streams, by: a) computing the correlations between multiple data streams and identifying points that have a high correlation; or b) computing the similarity by querying multiple data streams to determine high similarity points; or c) utilising clustering methods to discover the relationship between the streams in order to filter the outliers [119].

Traditional anomaly-based algorithms, which are not designed for data stream applications, might not be able to mine the data points in data streams for the following reasons [120]. *Firstly*, data arrives in the form of streams and should be tested for outlier-ness as long as they arrive which could result in wrong decisions due to the dynamic nature of the data streams [121]. *Secondly*, data streams produce a very high volume of data, which would be too expensive to store. In fact, it has been suggested in [122] that data stream algorithms should be executed in the main memory and not requisite secondary storage. *Thirdly*, unlike traditional methods for anomaly detection that assume the existence of the entire datasets in advance, the mining of data streams requires the consumption of a minimum amount of memory [123]. Therefore, the model should have only a single scan to access the data points in the storage for the purpose of detection.

In addition to the above-mentioned characteristics, it is challenging to deter-
mine whether or not the data streaming points are outliers as the characteristics of the data streams may change over time. This phenomenon is known as concept evolution [124], and it takes place when a new class emerges from streaming data over time. Therefore, clustering techniques in particular should adapt to the concept evolution in order to reflect the real characteristics of data points. Additionally, data streams do not form a unified distribution of the data points, which seems to increase the complexity of detecting outliers [125]. High-dimensionality is also a characteristic of data streams due to the sparseness of the data, which could degrade the efficiency of detecting outliers, as high-dimensional data appear to be equal in terms of distance between the data points due to the sparse data [126]. Moreover, in some situations, different data streams with different data types, such as categorical or numerical, need to be mined; hence, it becomes challenging to finding the relationship between them [127]. Finally, most data mining algorithms have high computational complexity when applied to data streams [128]. As a result, new algorithms should be designed, or improved from existing algorithms, to meet the requirements as well as the characteristics of multi-data streams so they can mine patterns efficiently and accurately.

There are a few existing solutions that specifically apply to anomaly detection in multi-data streams. The algorithm proposed in [129] attempts to solve the problem of judging the stream data points for outlier-ness as soon as they arrive due to limited memory capacity, which could result in wrong decisions. This is carried out by partitioning the data streams into chunks and later clustering each one by applying the $k$-means algorithm. Then, every point that deviates significantly from its cluster’s centroid would be saved temporarily as a candidate outliers, and the normal points are discarded after computing their mean values in order to free the memory. To decide whether or not they are outliers, the mean value of the
candidate clusters is then compared with the mean value of a pre-set $L$ number of previous chunks. Although this algorithm seems to be computationally efficient because it does not rely on distance measures, it has low detection accuracy. Additionally, several parameters need to be properly defined (e.g. number of clusters and $L$ number of chunks to compare the mean value and the chunk size as well), which makes the algorithm less attractive for multi-stream data.

Another clustering-based approach is proposed in [130] to detect anomalies for multi-data streams. It partitions a stream into windows or chunks, each of which is clustered and associated with a reference. Then, the numbers of adjacent clusters, along with representation degree references, are computed to find outlier references that contain potential anomalies. This model is believed to have better scalability and accuracy.

[95] proposed an incremental clustering algorithm that has two main phases to detect outliers in multi-data streams. In the online phase, the data in the windows is clustered using the $k$-mean algorithm, where clusters that are relatively small or quite far from other clusters are considered to be online outliers and therefore need further investigation. During the offline phase, the outlier from previous windows is added to the current window to be re-clustered by the $k$-mean algorithm. With higher confidence, it guarantees that any small or far clusters are real outliers as they have been given a survival chance. The work claims that the proposed algorithm is more accurate than existing techniques in discovering outliers; however, no evaluation results have been provided. Similar to other algorithms, many of its parameters need to be adjusted.
2.4 Conclusion

High-dimensional data is a big challenge for the machine learning algorithms due to the existence of redundant and non-representative features. Feature selection is an efficient dimension reduction technique used to reduce data dimensions by removing those redundant and non-representative features. In real applications, most data do not have class labels (i.e. unsupervised) and therefore clustering techniques are used to select features, as they do not require data class labels. An example of this application is a data-driven intrusion detection system. This chapter briefly described and reviewed the progress that has been made in three fields, namely dimensionality reduction, clustering-based methods as well as data-driven intrusion detection systems. These three areas will hopefully provide the reader with a comprehensive background enabling a better understanding of the work carried out in this thesis.
Chapter 3

AUFS - Towards an Efficient and Accurate Unsupervised Feature Selection

Both redundant and non-representative features result in large-volume and high-dimensional data, which degrade the accuracy and performance of classification as well as clustering algorithms. Most of the existing feature selection methods have limitations when dealing with high-dimensional data, as they search different subsets of features to find accurate representations of all features. Obviously, searching for different combinations of features is computationally very expensive, which makes existing work not efficient for high-dimensional data. The work carried out in this chapter, which relates to the design of an efficient and accurate similarity-based unsupervised feature selection (AUFS) method, tackles mainly the high-dimensionality issue of data by selecting a reduced set of representative and non-redundant features without the need for data class labels.

The proposed AUFS method extends the $k$-mean clustering algorithm to par-
partition the features into \( k \) clusters based on three similarity measures (i.e. PCC - Pearson Correlation Coefficient, LSRE - Least Square Regression Error and MICI - Maximal Information Compression Index) in order to accurately partition the features. Then, the proposed centroid-based feature selection method is used, where the feature with the closest similarity to its cluster centroid is selected as the representative feature while others are discarded. Extensive experimental work has shown that AUFS can generate a reduced representative and non-redundant feature set that achieves good classification accuracy in comparison with well-known unsupervised features selection methods.

3.1 Introduction

There has been extensive research in the field of feature selection because of the need to reduce the high-dimensionality of data. High-dimensional data suffers from redundant and non-representative features that result in the following challenges. Firstly, these features reduce the accuracy of the data mining algorithms by misdirecting the classification (supervised) and the clustering (unsupervised) processes [19]. Figure 3.1 illustrates the impact of non-representative features when they are used to classify the data. Figure 3.1(a) shows that there are two clusters when all used features are representative, which is correct. On the other hand, Figure 3.1(b) shows a “bad” data classification accuracy due to the use of the non-representative feature F3. This clearly shows that the features used in the classification cannot distinguish data after the inclusion of F3. Additionally, the existence of redundant and non-representative features negatively affects the processing time of the algorithms due to the large volume of data, which requires substantial storage space [131].
Figure 3.1: Impact of non-representative features in classification accuracy

Many feature selection methods [21, 22, 23, 24] have attempted to address the feature selection challenges. However, most of them are not efficient when applied to high-dimensional data because they require labeled data. These are called supervised methods. Therefore, they are outside the scope of this chapter. Most of the high-dimensional data are not labeled, making existing methods unsuitable. The methods proposed in [1] and SPEC [17] are probably the two most well-known unsupervised methods (i.e. they do not require data class labels) used to select the representative features. However, they have limitations in terms of accuracy and performance. In regard to accuracy, the method proposed by Mitra et al. [1] partitions the feature space using k-NN clustering. However, k-NN is inefficient when data is not dense as it produces low quality clusters [33]. Therefore, it is not suitable for high-dimensional data because, mostly, it is not dense. According to the method proposed in [1], and as illustrated in Figure 3.2(a), one of the three nearest features may be selected to represent the far features (i.e. features that are distant
Introduction

from others), which have different characteristics. Consequently, the classification accuracy will be low because of the badly selected representative features. On the other hand, SPEC [17] has not addressed the issue of feature redundancy because it individually evaluates features, which would negatively affect the classification accuracy.

In terms of performance, both of the methods proposed in [1] and SPEC [17] experience high computational complexity. Because the method proposed in [1] uses $k$-NN, it inherits the computational issues of such an approach because it calculates the distance between $k$ and all its neighbours. On the other hand, SPEC [17] also suffers from high time-computational complexity as it is based on spectral graph theory, which is computationally expensive [34]. Although these two methods can be applied to high-dimensional data, they experience computational complexity.

The proposed AUFS feature selection method overcomes the limitations of existing solutions with the following features: (i) it addresses the problem of high-dimensional data by designing an accurate method for selecting a reduced set of representative features; (ii) it has an efficient computational time by not requiring any search strategy for testing different subsets of features; and (iii) it works with unsupervised data (i.e. un-labeled data), which has more challenges than supervised data because of the absence of class labels.

AUFS adapts the $k$-mean algorithm to cluster the feature space as it is more powerful in clustering the features of high-dimensional data. The reason is that distant far features as in Figure 3.2(b), will form clusters and representative features will be selected from them. This results in more accurate feature clustering. In detail, AUFS partitions the feature space into $k$ clusters based on computing different similarity measures to assign the features into clusters. Then, from every cluster, only the feature that has the minimum dissimilarity to its cluster centroid
is selected, and this ensures that the selected feature represents all features within the cluster. This is done to enable the inclusion of only the representative features, and to remove redundant features.

Experimental results show that AUFS generates a reduced representative feature set that, when used by classifiers/evaluation models, achieves the best accuracy according to the evaluation metrics for the used datasets: it has the lowest FPR (False Positive Rate), the highest precision and F-measure in comparison with SPEC [17] and the method proposed in [1], whether the evaluation model is Naive Bayes, Lazy Nearest Neighbor or J48 Decision Tree. In addition, when compared...
with the benchmark methods, AUFS had the lowest running time when selecting the features.

3.2 Related Work

Based on the availability of the data class labels, feature selection methods fall into two categories: supervised and unsupervised. Below, we briefly describe the representative methods of both categories.

3.2.1 Supervised Feature Selection Methods

We start by introducing supervised feature selection methods below.

Information Gain

Information Gain [102] is a supervised feature selection method that requires data class labels to select a set of representative features. It builds a decision tree to measure the information in class prediction. This is done by observing the value of features. Any feature with high value of information is established as a splitting point. On the other hand, the features with low value of information indicate that the points are not ready to be partitioned. Generally, Information Gain can be defined as the difference between the original information obtained from the proportion of the class labels and the new information obtained after partitioning.

Fisher Score

Fisher Score [15] is a supervised feature selection method that requires data class labels in order to select a reduced set of representative features. The selected features have to meet a specific criterion: that the features values of samples
the same class are similar. Conversely, feature values of samples belonging to different classes are dissimilar. The $n$ number of features with highest fisher score are selected as the representative features.

**Chi-square**

Chi-square [132] is a statistical-based supervised feature selection method. It statistically measures the relevance of each feature to the class label individually. Specifically, it measures the association of each feature with the class label in order to evaluate its relevance. As it evaluates each feature individually, it is unable to discover and remove redundant features. Let’s say we have a set of features with a range of continuous values. Chi-square first transfers the continuous values into discrete intervals. The values of each feature are assigned to their own interval. Then, a merging step is applied to maintain the validity of the original feature space.

Supervised feature selection methods [21, 22, 23, 24] including the above mentioned representative ones, share a common limitation in that they are limited to the existence of data class labels. In other words, they evaluates the relevance/correlation of the features to the class labels. However, it is often difficult to have the data class labels, especially for high-dimensional datasets because, in real applications, most data do not have class labels, thereby making existing methods unsuitable. Although manual labeling can be a solution, it would take experts a long time to test and label the data, which is infeasible.

### 3.2.2 Unsupervised Feature Selection Methods

Unsupervised feature selection methods [66, 1, 17, 69, 70] have been proposed to overcome the need for data class labels. Such methods are much harder to
design due to the absence of data class labels, which guide them in the process of searching for the representative features. Initially, traditional unsupervised feature selection methods addressed the problem (of the absence of class labels) by ranking features independently based on certain scores. However, they are not able to generate the best features set, as they do not compute the correlation between features [69]. The second of the unsupervised feature selection methods e.g. [71, 1] generally uses clustering to partition the feature space into distinct clusters, where features in every cluster are similar to each other and different from the features of other clusters. Below we describe two well-known and representative unsupervised feature selection methods, which are selected for benchmarking.

**Spectral Feature Selection (SPEC)**

SPEC [17] is an unsupervised feature selection method that does not require data class labels in order to select representative features. It extends the Laplacian score [16] both to weight all the features and to select the top $n$ features as the subset of representative features. It finds the representativeness of a feature by estimating its consistency with the spectrum of a matrix that is derived from Radial-Base Function (RBF) similarity matrix. In particular, a specific graph is built based on the similarity matrix, which SPEC uses to weight features.

Given a dataset $D$, evaluation functions and number of samples, the following steps are taken by SPEC to weight the features.

- The similarity matrix $S$ is constructed from the dataset
- A graph $G$ is constructed based on $S$
- An adjacency matrix $W$ and diagonal matrix $D$ will be built from $G$.
- Evaluates each feature individually using the given evaluation functions.
Because SPEC evaluates the representativeness of different features individually, it does not handle any feature redundancy.

**Mitra’s Method**

Mitra et al. [1] proposed an unsupervised feature selection method that does not require data class labels to select representative features. It selects a reduced set of representative features from high-dimensional data using different similarity measures. They proposed a new similarity measure called Maximal Information Compression Index (MICI) (see Section 3.3) in order to compute the similarity between the features. Mitra et al. [1] use $k$-NN clustering algorithm to cluster the feature space with an MICI (Maximal Information Compression Index) similarity measure so that the features within a cluster are highly similar, while those in different clusters are dissimilar. Then, from every cluster, they use the compactness methodology to select the features. The only feature that is selected is the one that has the minimum dissimilarity to its NN.

The methods proposed in [1] and SPEC [17] are probably the two most well-known unsupervised approaches used to select the representative features from high-dimensional homogeneous data. Even though these two methods are unsupervised, they have classification accuracy and computational time complexity limitations when selecting representative features from high-dimensional data, as explained in the Introduction (see section 3.1).

### 3.3 Similarity Measures

Here we introduce the similarity measures used to measure the dependency of the features to: 1) allocate the feature to a relevant cluster; 2) decide which feature
Similarity Measures

of a cluster to be selected as a representative feature. The reason for adopting these linearly-dependent measures is their effectiveness for the purpose of feature selection as they are not sensitive to the location along with the scatters of the distribution of the features data [1]. Therefore, they are promising when working with high-dimensional data. These linearly dependent measures are illustrated below. For all similarity measures $x$ denotes a cluster centroid and $y$ denotes a feature.

**Pearson Correlation Coefficient (PCC)** [133]

PCC is a measure that computes the correlation between two random variables, and it determines whether they would have a linear dependency relationship. It can be computed by calculating either the correlation between a feature and a predicted class label or between a feature and a feature. Unlike the former, which measures the extent to which features are correlated to their class labels, the latter is adopted in our method to measure the correlation between the features and the clusters centroids to assign features to clusters. In fact, a feature and a feature correlation are better suited to our method as we are concerned with unsupervised learning, which does not assume the existence of the data class labels. Generally, correlation coefficient is fast and capable of identifying representative features without the need for pairwise correlation computation. Formally, it is computed as follows.

$$\text{PCC}(x, y) = \frac{n(\sum xy) - (\sum x)(\sum y))}{\sqrt{n \sum x^2 - (\sum x)^2} \sqrt{n \sum y^2 - (\sum y)^2}}$$

The result of the correlation between $x$ and $y$ is between 0, which indicates that the feature and the cluster centroid are completely uncorrelated or 1, which indicates their complete correlation.

**Least Square Regression Error (LSRE)** [134]

LSRE computes and analyses the degree of the correlation between a feature
and a cluster centroid by drawing a line that is best fitted to the data. It is computed based on linear model \( y = ax + b \), where \( a \) and \( b \) are given by minimising the mean square error and \( n \) denotes the number of features which is always 1 (one) as we process one feature at a time. The error is the distance between the actual data and the model data and is calculated based on the following equations:

\[
\text{LSRE}(x, y) = y_n - (ax_n + b) \quad (3.2)
\]

\( a \) is the slope of the \( x \) and is calculated by

\[
a = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \left(\frac{\sum x}{n}\right)^2} \quad (3.3)
\]

\( b \) is the \( y \)-intercept and is calculated by

\[
b = \frac{\sum y - (a \sum x)}{n} \quad (3.4)
\]

The final result of the former equations show the degree of the linear dependency correlation of a feature and a cluster centroid based on given value of equation 4.2. They are completely correlated when \( \text{LSRE} = 0 \).

**Maximal Information Compression Index (MICI) [1]**

MICI is an index technique for measuring the similarity between a feature and a cluster centroid. Let \( \Sigma \) be the covariance matrix of the random features. MICI is defined as \( \text{MICI}(x, y) = \) the smallest eigenvalue of \( \Sigma \), i.e.,

\[
\text{MICI}(x, y) = \text{var}(x) + \text{var}(y) - \sqrt{(\text{var}(x) + \text{var}(y))^2 - 4\text{var}(x)\text{var}(y)(1-\rho(x,y)^2))} \quad (3.5)
\]

A feature and a cluster centroid are linearly dependent when the value of \( \text{MICI} \) is zero and the value increases as much as the amount of dependency decreases.
3.4 AUFS - The Proposed an Accurate and Efficient Unsupervised Feature Selection Method

AUFS is primarily designed to select a reduced set of representative and non-redundant features from high-dimensional data without the need of the data class labels. The accurate selection of representative features would result in high classification accuracy. AUFS uses the three linearly-dependent similarity measures: PCC [133], LSRE [134] and MICI [1] to partition the feature space. Linearly-dependent measures are chosen because they are not sensitive to the order along with the distribution of features. Also, a single similarity measure might favour a specific model, and therefore will produce better selection of representative features for that model over other models. Finally, the three measures proved their effectiveness for feature selection experimentally as shown in [1]. Therefore, PCC, LSRE and MICI are used in the \( k \)-mean algorithm to compute the dependency between features and cluster centroids. Before giving the details of the proposed method, we first define what we mean by representative features and redundant features.

For the definitions below, let us assume that we have the following sets:

- \( F = \{ f_1, f_2, \ldots, f_n \} \): the set of all features in column vectors.
- \( C = \{ c_1, c_2, \ldots, c_n \} \): the set of all clusters centroids.
- \( \varepsilon \) is a subset of features (i.e. \( \varepsilon \subset F \)) in a cluster with centroid \( c_r \in C \).

**Definition:** Representative Feature \( f_i \in \varepsilon \) is a representative feature in \( c_r \) if and only if:

\[
PCC (f_i, c_r) > PCC (\varepsilon, c_r)
\] (3.6)
Any feature $f_i$ that is not representative is said to be non-representative.

**Definition:** Redundant Feature A feature $f_i$ is redundant in a given cluster if it exists another feature $f_j$ in the same cluster such that:

\[
PCC(f_i, f_j) = 1
\]  
\[
LSRE(f_i, f_j) = 0
\]  
\[
MICI(f_i, f_j) = 0
\]

### 3.4.1 The AUFS Method

After defining the various concepts, we now provide details of the various steps of the proposed AUFS. Unlike the wrapper approach, where related algorithms [135, 136, 137] experience performance degradation due to the use of complex data-mining algorithms, the filter approach is used in AUFS as it has better performance (i.e. processing time) because it does not use any data-mining algorithm to evaluate the generated set of features.

In addition to adopting the filter approach, AUFS is categorised as an unsupervised method as it does not depend on the availability of the data class labels. It is therefore based on various existing clustering techniques that process data without requiring their class labels, such as $k$-mean [138]. However, the $k$-mean algorithm needs to be extended to integrate the three above mentioned similarity measures (i.e. PCC, LSRE and MICI) in order to properly cluster the features, as
the use of a single similarity measure will be biased towards specific models and therefore will not produce accurate classification. The three similarity measures will cover most of the possible linear dependent correlations between features, and therefore will reflect the real accuracy of AUFS as well other methods used in the benchmark.

Algorithm 1 shows that AUFS has two main stages: (1) the original feature space is partitioned into a pre-defined number of clusters using the extended $k$-mean algorithm (with three similarity measures); (2) using each similarity measure during the clustering stage, AUFS computes the similarity between the centroid vector of each cluster and all features in that cluster to find the representative features and removes the rest (i.e. non-representative and redundant features). Then, the representative features from every cluster will form the reduced feature set. Following are the details of the various steps carried out by AUFS to select a reduced set of representative features:

- Firstly, AUFS partitions the feature space by applying the $k$-mean into $k$ clusters using every similarity measure, namely PCC, LSRE and MICI. Each similarity measure is computed individually.

- Secondly, the centroids are initialised to be the first feature vectors from the feature space based on the $k$ value. For example, if $k=10$ then the first ten features vectors are the initial clusters centroids. $k$ value is determined based on the required reduction of the feature space.

- Thirdly, AUFS assigns every feature to a cluster (i.e. hard-clustering). To do so, the similarity between every centroid and all the features in the feature space is computed. Every feature is therefore assigned to its relevant cluster. This process is repeated until the re-assigning of features no longer
changes the centroids, meaning that the set of centroids is stable (i.e. does not change).

- Fourthly, AUFS finds the representative feature of every cluster. The feature of a cluster that has the highest similarity (i.e. highest PCC or lowest LSRE and MICI) to its centroid (mean) is selected as the representative feature for the cluster.

- Lastly, AUFS ignores all the remaining features of every cluster (and therefore retains only the representative features). This guarantees the removal of redundant as well as non-representative features, and produces the set of all representative features.

The AUFS is a novel method because: i) the way it selects representative features ensures that the selected feature accurately represents all the features of a cluster, as the feature with the closest similarity to its cluster centroid is going to be selected; ii) it uses only one parameter and is not overwhelmed by having to find the best parameters since AUFS namely has $k$ (the # of clusters) which is the number of features to be selected, as one feature is selected to represent every cluster; iii) AUFS has low computational time complexity as it does not require the search as well as the evaluation of different subsets of features to find representative features, thereby reducing the computational complexity of AUFS. In addition, by removing all features other than representative ones, redundant features will be definitely removed because they will be a part of the non-representative features, which would reduce computational time.
Algorithm 1: The AUFS Method

1. **Input:**
   - \( F \): \( \{f_1, f_1, ..., f_n\} \), is a set of features;
   - \( \text{SM}_j \): \( \{1 = \text{PCC}, 2 = \text{LSRE}, 3 = \text{MICI}\} \), is the similarity measure;
   - \( k \): the number of clusters, \( n-1 > k > 1 \);

2. **Output:**
   - \( R \): the representative and non-redundant feature set;

3. **foreach** \( j \in [\text{SM}_j] \) **do**
   - // \( F \) is partitioned into \( k \) clusters based on \( j \) measure
   - \([\text{idxbest}, \text{Cbest}] \leftarrow \text{kmean}\left(F, k, \text{SM}_j\right)\);
   - // get a list of clusters’ ids
   - \( \text{clusIds} = \text{unique}(\text{idxbest}) \);
   - // get the index for each feature
   - \( \text{featuresIndex} = [1:\text{size}(F,2)] \);
   - // go through each cluster and find the representative features
   - **for** \( i = 1 : \text{size(clusIds,1)} \) **do**
     - \( \text{clusterFeatures} = \text{featuresIndex}(1, [\text{idxbest}(:,1) == \text{clusIds}(i)])' \);
     - \( \text{clusterData} = \text{data}(:, \text{clusterFeatures}) \);
     - \( \text{clusterMean} = \text{Cbest}(:, i) \);
     - \( \text{distances} = \text{zeros} (\text{size(clusIds,1)},1) \);
     - **for** \( k = 1 : \text{size} (\text{clusterData},2) \) **do**
       - \( \text{distances}(k,1) = \text{calcDistance} (\text{clusterMean}, \text{clusterData}(:,k), \text{SM}_j) \);
     - **end**
   - [\( \text{dis}, \text{indx} \)] = \( \text{min} (\text{distances}) \);
   - \( \text{FeatureSelected} = [\text{FeatureSelected}, \text{clusterFeatures}(1, \text{indx})] \);
   **end**

4. **return** \( R \);

3.4.2 An Illustrative Example

A simple example is given here to illustrate the way that AUFS algorithm works to select representative features from the feature space and removes redundant ones. This example covers one measure, as the example is also applicable with the two others. Let us make up the feature set such as \( F = \{F_1, F_2, F_3, ..., F_9\} \) be the
feature vectors, \( k=3 \) be the number of clusters and \( j = \text{PCC} \) be the similarity measure. Firstly, the feature set is partitioned into three clusters based on computing PCC between every feature from the feature set and every centroid \( c_1 \), \( c_2 \) and \( c_3 \) so every feature is assigned to its relevant cluster centroid (see Table 3.1(a)).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Cluster#</th>
<th>PCC</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 )</td>
<td>1</td>
<td>0.85</td>
<td>0.32</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>( F_2 )</td>
<td>1</td>
<td>0.28</td>
<td>0.98</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>( F_3 )</td>
<td>1</td>
<td>0.88</td>
<td>0.44</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>( F_4 )</td>
<td>2</td>
<td>0.15</td>
<td>0.37</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>( F_5 )</td>
<td>2</td>
<td>0.96</td>
<td>0.42</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>( F_6 )</td>
<td>2</td>
<td>0.65</td>
<td>0.60</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>( F_7 )</td>
<td>3</td>
<td>0.26</td>
<td>0.58</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>( F_8 )</td>
<td>3</td>
<td>0.56</td>
<td>0.93</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>( F_9 )</td>
<td>3</td>
<td>0.33</td>
<td>0.75</td>
<td>0.42</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: (a) PCC between the centroids and all the features vectors in the feature set. (b) Allocation of a feature to its most similar cluster centroid.

Table 3.1(b) provides an allocation of every feature from the feature set to its cluster based on PCC. Then from every cluster, a feature that has the highest similarity to its centroid is selected to be the relevant feature for the cluster and discards the rest features. For example, in Table 3.1(b), cluster #1 has three features assigned to it namely, \( F_1 \), \( F_3 \) and \( F_5 \). \( F_5 \) is only selected from cluster #1 as representative feature because it has higher PCC to \( c_1 \) than \( F_1 \) and \( F_3 \). On the other hand, \( F_1 \) and \( F_3 \) are discarded. Consequently, the reduced, representative and non-redundant subset of features from the three clusters is \( \{ F_5, F_2, F_4 \} \).

### 3.5 Experimental Setup

This section describes the performance of the proposed AUFS method with different datasets. In order to properly investigate the accuracy and the time complex-
ity of AUFS, two well-known algorithms were used for the comparison: Mitra’s method [1] and SPEC [17]. These two methods were selected as they are well-known unsupervised feature selection methods that do not require the data class labels for selecting features. These two algorithms and AUFS were evaluated using three different families of classifiers, namely Naive Bayes [139], J48 Decision Tree [140] and the Lazy Nearest Neighbor [141] (also called IB1). In addition to the classifiers, $k$-fold-cross validation was applied on all datasets to efficiently evaluate the accuracy of the benchmark methods. The entire dataset was first divided into subsets of equal size depending on the selected $k$ folds. Then, only one $k$ was used as the testing subset and the rests were the training subsets. Finally, the average value of all folds was set as the average result. In the evaluation, $k$ was set to 10 as suggested in [142] to demonstrate the efficiency of our proposed method along with the benchmark methods. All the three algorithms were implemented in Matlab programming language. They were executed under Mac operating system OS X Yosemite with 2.4 GHz Intel Core 2 Duo and 8 GB RAM.

### 3.5.1 Datasets

Three data sets were used in the experiments, namely Spambase, Water Treatment Plant and Physical Activity Monitoring (PAMAP2). The preference of the selection of those datasets is because they are commonly used for the aim of data mining algorithms as well as they are from diverse domains. They are found in UCI Machine Learning Repository website. The three data sets were mainly collected for the purpose of classification and clustering as clustering is a part of the proposed method to filter out redundant and non-representative features. Here is a brief description of each of each dataset:

- **Spambase**: is a multivariate data set that contains spam and non-spam
email classes where each email is described by 57 real data type features. The total number of emails (records) is 4601.  https://archive.ics.uci.edu/ml/datasets/Spambase

- **Water treatment plant**: the collected multivariate data are measures of the daily sensor readings in urban wastewater treatment plant. The goal of this data to train any learning model to classify the operational state of the plant to predict the occurrence of faults at any stage of the treatment process. It has 527 objects where each object is described by 38 real data type features.  https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant.

- **Physical activity monitoring data set (PAMAP2)**: is a multivariate time-series data set that is collected by monitoring 18 different physical activities of 9 people such as walking, lying and cycling. PAMAP2 consists of 3,850,505 objects that are described by 54 features including the class label of the activities ID. http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring

### 3.5.2 Evaluation Metrics

The major concern of the feature selection method is to select a reduced set of features. The accurate selection of representative features should increase the classification accuracy of the classifiers as the redundant and non-representative features should be removed. The selection of representative features should be done within an acceptable running time. Therefore, two groups of evaluation metrics were selected namely, classification accuracy and running time as shown below.
Classification accuracy

The primary aim of this experiment was to investigate whether the reduced representative set of features competitively improved the accuracy of the classifiers in terms of data classification. We first applied AUFS on three given datasets; then the results (representative features set) were provided to the data mining classifiers (i.e. Naive Bayes, J48 and IB1) to test the efficiency of the reduced features in term of classification accuracy. The evaluation metrics used for classification accuracy were False Positive Rate (FPR), Precision and F-measure. These metrics were appropriate given the main aim of the AUFS method, which is the accurate selection of representative features to increases the classification accuracy of the data. They are calculated based on Table 3.2 and are provided below.

Table 3.2: Standard Confusion Metrics for Evaluation of Normal/Anomaly Classification

<table>
<thead>
<tr>
<th>Actual label of flows</th>
<th>Predicted label of flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Anomaly</td>
</tr>
<tr>
<td>True Negative (TN)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>False Negative (FN)</td>
<td>True Positive (TP)</td>
</tr>
</tbody>
</table>

- FPR: the percentage of normal instances that are detected as anomalies over all normal instances, which is defined as follows in terms of the metrics defined in Table 3.2:

\[ FPR = \frac{FP}{FP + TN} \]  \hspace{1cm} (3.12)

- Precision: the percentage of correctly detected anomaly instances over all the detected anomaly instances. This is defined as follows in terms of the metrics defined in Table 3.2:
\[
Precision = \frac{TP}{TP + FP}
\]  

(3.13)

- F-measure is the equally-weighted (harmonic) mean of precision and recall. This is defined as follows:

\[
F-measure = \frac{Recall \times Precision}{Recall + Precision}
\]

(3.14)

Running time

High-dimensional data requires more processing time particularly when there are redundant and non-representative features. Therefore, the major purpose of AUFS is to remove those redundant and non-representative features in order to improve the accuracy of the classification task within a short running time.

3.6 Experimental Results

This section presents and analyses the experimental results. For every feature selection method, Naive Bayes, IB1 and J48 Decision Tree classifiers were used to evaluate the classification accuracy of the generated representative features sets with different similarity measures. All the experiments were carried out on three datasets. For every data set, every method was run with all possible similarity measures developed for that method. The method proposed in [1] already includes the three similarity measures (i.e. PCC, LSRE and MICI), while SPEC works with the RBF Kernel similarity measure. To investigate their classification accuracy, we compared these methods by considering all of their similarity measures individually. The experimental results are presented in Tables 3.3 - 3.11, and they clearly show that AUFS has consistently the lowest FPR as well as the highest precision and F-measure for the three similarity measures.
Experimental Results

For example, the water treatment plant dataset shows that the proposed AUFS method achieved a higher accuracy result compared to the two other methods based on the F-measure. The accuracy range varies between 5% to 94% and 15% to 29% compared with [1] and SPEC respectively when using the Naive Bayes classifier as shown in Table 3.3. Moreover, the Spambase dataset shows that AUFS also achieves a higher accuracy result compared to the two other methods based on the F-measure. The accuracy range varies from 4% to 9% and 23% to 26% compared to [1] and SPEC respectively when using the J48 Decision Tree classifier as shown in Table 3.8.

Table 3.3: Classification accuracy using Water Treatment Plant dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCC</td>
<td>0.021</td>
<td>0.45</td>
<td>0.5294</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.039</td>
<td>0.2593</td>
<td>0.3415</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.012</td>
<td>0.6471</td>
<td>0.7097</td>
</tr>
</tbody>
</table>

| AUFS                     | PCC                | 0.018| 0.55      | 0.6471    |
|                          | LSRE               | 0.02 | 0.5455    | 0.6667    |
|                          | MICI               | 0.006| 0.7692    | 0.7407    |

| SPEC [17]                | RBF Kernal         | 0.031| 0.4286    | 0.5714    |

Table 3.4: Classification accuracy using Water Treatment Plant dataset with IB1

<table>
<thead>
<tr>
<th>Feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCC</td>
<td>0.0058</td>
<td>0.625</td>
<td>0.4545</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0039</td>
<td>0.7778</td>
<td>0.6087</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0039</td>
<td>0.7143</td>
<td>0.4702</td>
</tr>
</tbody>
</table>

| AUFS                     | PCC                | 0.0019| 0.8889    | 0.6957    |
|                          | LSRE               | 0.0019| 0.875     | 0.6764    |
|                          | MICI               | 0.0019| 0.8899    | 0.6957    |

| SPEC [17]                | RBF Kernal         | 0.0058| 0.7273    | 0.64      |

Table 3.5: Classification accuracy using Water Treatment Plant dataset with J48 Decision Tree

<table>
<thead>
<tr>
<th>Feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCC</td>
<td>0.0117</td>
<td>0.6</td>
<td>0.6207</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0078</td>
<td>0.6923</td>
<td>0.6667</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0058</td>
<td>0.7692</td>
<td>0.7407</td>
</tr>
</tbody>
</table>

| AUFS                     | PCC                | 0.0019| 0.9       | 0.75      |
|                          | LSRE               | 0.0039| 0.8333    | 0.7692    |
|                          | MICI               | 0.0039| 0.8462    | 0.8148    |

| SPEC [17]                | RBF Kernal         | 0.0078| 0.7143    | 0.7143    |
Experimental Results

Table 3.6: Classification accuracy using Spambase dataset with Naive Bayes

<table>
<thead>
<tr>
<th>feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>method in [1]</td>
<td>PCC</td>
<td>0.4258</td>
<td>0.5915</td>
<td>0.7285</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.3917</td>
<td>0.6135</td>
<td>0.7473</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.4659</td>
<td>0.5758</td>
<td>0.7233</td>
</tr>
<tr>
<td>UFSDA</td>
<td>PCC</td>
<td>0.1191</td>
<td>0.8146</td>
<td>0.8907</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.1202</td>
<td>0.8268</td>
<td>0.8535</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.425</td>
<td>0.5986</td>
<td>0.7328</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernal</td>
<td>0.4283</td>
<td>0.5923</td>
<td>0.7312</td>
</tr>
</tbody>
</table>

Table 3.7: Classification accuracy using Spambase dataset with IB1

<table>
<thead>
<tr>
<th>feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>PCC</td>
<td>0.1313</td>
<td>0.8002</td>
<td>0.8044</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.1065</td>
<td>0.8382</td>
<td>0.8435</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0911</td>
<td>0.8495</td>
<td>0.8129</td>
</tr>
<tr>
<td>UFSDA</td>
<td>PCC</td>
<td>0.08</td>
<td>0.8721</td>
<td>0.8549</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0789</td>
<td>0.8777</td>
<td>0.8743</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0721</td>
<td>0.8886</td>
<td>0.8621</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernal</td>
<td>0.4763</td>
<td>0.5688</td>
<td>0.7161</td>
</tr>
</tbody>
</table>

Table 3.8: Classification accuracy using Spambase dataset with J48 Decision Tree

<table>
<thead>
<tr>
<th>feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>PCC</td>
<td>0.1022</td>
<td>0.8341</td>
<td>0.8117</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0681</td>
<td>0.8887</td>
<td>0.8619</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0886</td>
<td>0.8476</td>
<td>0.8002</td>
</tr>
<tr>
<td>AUFS</td>
<td>PCC</td>
<td>0.0552</td>
<td>0.9</td>
<td>0.883</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0567</td>
<td>0.9122</td>
<td>0.9086</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0624</td>
<td>0.9003</td>
<td>0.8831</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernal</td>
<td>0.4788</td>
<td>0.5678</td>
<td>0.7156</td>
</tr>
</tbody>
</table>

Table 3.9: Classification accuracy using PAMAP2 dataset with Naive Bayes

<table>
<thead>
<tr>
<th>feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>PCC</td>
<td>0.0801</td>
<td>0.5241</td>
<td>0.5314</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0421</td>
<td>0.8494</td>
<td>0.8467</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0394</td>
<td>0.8015</td>
<td>0.7913</td>
</tr>
<tr>
<td>AUFS</td>
<td>PCC</td>
<td>0.0801</td>
<td>0.5464</td>
<td>0.5601</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0306</td>
<td>0.8763</td>
<td>0.8705</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0366</td>
<td>0.8325</td>
<td>0.8217</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernal</td>
<td>0.0721</td>
<td>0.5172</td>
<td>0.5284</td>
</tr>
</tbody>
</table>
Experimental Results

Table 3.10: Classification accuracy using PAMAP2 dataset with IB1

<table>
<thead>
<tr>
<th>Feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>PCC</td>
<td>0.0681</td>
<td>0.6889</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0015</td>
<td>0.9935</td>
<td>0.9935</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.003</td>
<td>0.9901</td>
<td>0.9901</td>
</tr>
<tr>
<td>AUFS</td>
<td>PCC</td>
<td>0.0652</td>
<td>0.6985</td>
<td>0.6957</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0002</td>
<td>0.9992</td>
<td>0.9992</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0027</td>
<td>0.9908</td>
<td>0.9908</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernel</td>
<td>0.0821</td>
<td>0.6678</td>
<td>0.6597</td>
</tr>
</tbody>
</table>

Table 3.11: Classification accuracy using PAMAP2 dataset with J48 Decision Tree

<table>
<thead>
<tr>
<th>Feature selection method</th>
<th>Similarity Measure</th>
<th>FPR</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>PCC</td>
<td>0.0711</td>
<td>0.7185</td>
<td>0.7159</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0017</td>
<td>0.9937</td>
<td>0.9937</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0034</td>
<td>0.9883</td>
<td>0.9883</td>
</tr>
<tr>
<td>AUFS</td>
<td>PCC</td>
<td>0.069</td>
<td>0.7291</td>
<td>0.7258</td>
</tr>
<tr>
<td></td>
<td>LSRE</td>
<td>0.0009</td>
<td>0.9967</td>
<td>0.9967</td>
</tr>
<tr>
<td></td>
<td>MICI</td>
<td>0.0032</td>
<td>0.9934</td>
<td>0.9934</td>
</tr>
<tr>
<td>SPEC [17]</td>
<td>RBF Kernel</td>
<td>0.0874</td>
<td>0.6974</td>
<td>0.6833</td>
</tr>
</tbody>
</table>

The advantage of AUFS over existing feature selection methods is clearly the accurate selection of representative features and therefore the improvement of the classification accuracy, and this for the following reasons. This can be explained by the following. Firstly, K-mean works best in clustering the features with high-dimensional data as it is not sensitive to the non-dense data. Therefore, this would result in better partitioning of the features and consequently an accurate selection of representative features. Secondly, the way in which representative features are selected contributes to improving the classification accuracy by guaranteeing the representativeness of the selected features. Actually it is not enough to assume that all features grouped in a given cluster are representative; this is not an adequate method of properly measuring the representativeness of features. Conversely, AUFS strictly limits the representativeness of features to only those features that have the highest similarity to the cluster centroids, and this applies to every similarity measure. This way of selecting representative features helps AUFS to obtaining better classification accuracy by guaranteeing that the selected features will repre-
sent all the features in the clusters. Thirdly, all the features in a cluster other than representative features are discarded to ensure the removal of any redundant features. As a result, this method ensures the generation of a reduced representative feature set that helps the classifiers to accurately classify the data.

As SPEC uses one similarity measure, the average of the similarity measures of every method is then computed for every dataset with the three evaluation classifiers in order to further investigate their classification accuracy, as shown in Table 3.12. After computing the average of the similarity measures of every method, AUFS continues to achieve the best results for all the accuracy metrics as well as for all datasets, whether the evaluation model is Naive Bayes, IB1 or J48 Decision Tree.
Further experiments were also carried out to investigate which one of the similarity measures works the best for the proposed method for all the tested datasets in terms of the classification accuracy. The average classification accuracy of the three evaluation models was computed. The results are presented in Table 3.13, which indicates that AUFS achieved the best results with the LSRE similarity measure for all tested metrics. AUFS has the lowest FPR, highest precision and f-measure compared to the other similarity measures. Actually, regression similarity measures are very efficient for classification and prediction tasks [143].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>method</th>
<th>Evaluation Model</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spambase</td>
<td>method in [1]</td>
<td>Naive Bayes</td>
<td>0.4278</td>
<td>0.5936</td>
<td>0.73303</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.1996</td>
<td>0.8293</td>
<td>0.8223</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0863</td>
<td>0.8568</td>
<td>0.8246</td>
</tr>
<tr>
<td></td>
<td>AUFS</td>
<td>Naive Bayes</td>
<td>0.2214</td>
<td>0.7466</td>
<td>0.7986</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.077</td>
<td>0.8778</td>
<td>0.8637</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0581</td>
<td>0.9041</td>
<td>0.8915</td>
</tr>
<tr>
<td></td>
<td>SPEC</td>
<td>Naive Bayes</td>
<td>0.4283</td>
<td>0.5921</td>
<td>0.7312</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.4763</td>
<td>0.5688</td>
<td>0.7161</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.4788</td>
<td>0.5678</td>
<td>0.7156</td>
</tr>
<tr>
<td>Water treatment plant</td>
<td>method in [1]</td>
<td>Naive Bayes</td>
<td>0.024</td>
<td>0.4521</td>
<td>0.5268</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0415</td>
<td>0.7057</td>
<td>0.5131</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0084</td>
<td>0.6871</td>
<td>0.676</td>
</tr>
<tr>
<td></td>
<td>AUFS</td>
<td>Naive Bayes</td>
<td>0.0142</td>
<td>0.6215</td>
<td>0.6848</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0019</td>
<td>0.8486</td>
<td>0.6892</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0032</td>
<td>0.8598</td>
<td>0.778</td>
</tr>
<tr>
<td></td>
<td>SPEC</td>
<td>Naive Bayes</td>
<td>0.0312</td>
<td>0.4256</td>
<td>0.5714</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0058</td>
<td>0.7273</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0078</td>
<td>0.7143</td>
<td>0.7143</td>
</tr>
<tr>
<td>PAMAP2</td>
<td>method in [1]</td>
<td>Naive Bayes</td>
<td>0.0538</td>
<td>0.731</td>
<td>0.7225</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0242</td>
<td>0.8908</td>
<td>0.8898</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0254</td>
<td>0.9001</td>
<td>0.8993</td>
</tr>
<tr>
<td></td>
<td>AUFS</td>
<td>Naive Bayes</td>
<td>0.01243</td>
<td>0.7516</td>
<td>0.7474</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0227</td>
<td>0.8961</td>
<td>0.8952</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0243</td>
<td>0.9064</td>
<td>0.9053</td>
</tr>
<tr>
<td></td>
<td>SPEC</td>
<td>Naive Bayes</td>
<td>0.0721</td>
<td>0.5172</td>
<td>0.5284</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB1</td>
<td>0.0821</td>
<td>0.6678</td>
<td>0.6597</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0874</td>
<td>0.6974</td>
<td>0.6833</td>
</tr>
</tbody>
</table>
Table 3.13: The Average Classification Accuracy of Different Similarity Measures

<table>
<thead>
<tr>
<th>Similarity Measure</th>
<th>Evaluation Model</th>
<th>FPR</th>
<th>Precision</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCC</td>
<td>Naive Bayes</td>
<td>0.0655</td>
<td>0.6369</td>
<td>0.6689</td>
</tr>
<tr>
<td></td>
<td>IB1</td>
<td>0.049</td>
<td>0.8198</td>
<td>0.7487</td>
</tr>
<tr>
<td></td>
<td>J48 Decision Tree</td>
<td>0.042</td>
<td>0.843</td>
<td>0.7862</td>
</tr>
<tr>
<td>LSRE</td>
<td>Naive Bayes</td>
<td>0.0567</td>
<td>0.7495</td>
<td>0.7969</td>
</tr>
<tr>
<td></td>
<td>IB1</td>
<td>0.027</td>
<td>0.9173</td>
<td>0.8499</td>
</tr>
<tr>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0205</td>
<td>0.914</td>
<td>0.8915</td>
</tr>
<tr>
<td>MICI</td>
<td>Naive Bayes</td>
<td>0.1558</td>
<td>0.7334</td>
<td>0.765</td>
</tr>
<tr>
<td></td>
<td>IB1</td>
<td>0.0255</td>
<td>0.9214</td>
<td>0.8495</td>
</tr>
<tr>
<td></td>
<td>J48 Decision Tree</td>
<td>0.0231</td>
<td>0.9133</td>
<td>0.8971</td>
</tr>
</tbody>
</table>

In addition to considering the classification accuracy, we also investigated the computational time complexity of AUFS in producing the representative feature set. The three methods were tested on three datasets and their average time complexity was determined, as shown in Figure 3.3. AUFS has the lowest time complexity followed by the method proposed by Mitra et al. [1] and SPEC, respectively. The reason that AUFS has lower computational time complexity than SPEC and Mitra’s method [1] is that it does not require any search strategy to examine different subsets of features in order to find the representative feature set. Instead, AUFS selects the feature that has the highest similarity to its cluster centroid as the representative feature of that cluster. Furthermore, AUFS has a smart process of selecting representative features and removing redundant ones, which results in reducing the time complexity. Indeed, AUFS removes all features other than representative ones from clusters; and therefore, redundant features will definitely be removed, as they will be considered as being non-representative. Finally, Mitra’s [1] has higher running time due to the complexity of $k$-NN unlike the adapted k-mean proposed in our method.
3.7 Conclusion

Redundant and non-representative features, which result from high-dimensional data, have negative consequences on any applied classification algorithms essentially in terms of high computational time complexity and low classification accuracy. Selecting a reduced feature set, which only has representative and non-redundant features is critical, particularly when targeting high-dimensional datasets. In this chapter, a filter-based approach unsupervised feature selection method is proposed. The challenge is to accurately select a reduced feature set that represents the majority of features in a cluster with high-dimensional data. The selection of a reduced feature set would definitely enhance the classifiers to accurately classify the data. Also, the features are selected without applying any search strategy for selecting the best subset of features. In the proposed experiments using three
Conclusion

datasets, AUFS is compared with two unsupervised feature selection methods. We clearly showed that AUFS outperforms the benchmarked methods in selecting a reduced feature set that helps the selected evaluation classifiers to accurately classify the data with lowest computational time complexity. Future work will focus on improving AUFS so to produce the representative feature set in the online mode.
Chapter 4

UFSSF - An Efficient
Unsupervised Feature Selection
for Streaming Features

In Chapter 3 we proposed a feature selection method that does not require data class labels (i.e. unsupervised feature selection). Both the number of features and the number of instances is fixed. This chapter proposes a feature selection method for streaming features applications where the number of features increase while the number of instances is fixed. Streaming features applications pose challenges for dimensionality reduction techniques, particularly for feature selection. These dynamic features applications have the following characteristics: a) features are sequentially generated and are processed one by one upon their arrival while the number of instances/points remains fixed; and b) the complete feature space is not known in advance. For example, in a text classification task for spam detection, new features (e.g. words) are dynamically generated and therefore need to be mined to filter out the spams rather than waiting for all features to be collected in order to
do so. Traditional feature selection methods, which are not designed for streaming features applications, cannot be used in such an environment, as they do require the full feature space in advance in order to statistically determine the representative features. Although several of the exists methods reported in the literature address feature selection in streaming features applications, most of them require the data class labels as a guide to selecting the representative features. However, in real-world applications most data are not labeled and, moreover, manual labeling is costly.

This chapter proposes a new method, called Unsupervised Feature Selection for Streaming Features (UFSSF), to select representative features in streaming features applications without the need to know the features or class labels in advance. UFSSF extends the $k$-mean clustering algorithm to incrementally decide whether to add the newly arrived feature to the existing set of representative features. Those features that are not representative are discarded. Experimental results indicate that UFSSF significantly has a better prediction accuracy and running time compared to the baseline methods.

4.1 Introduction

High-dimensionality presents a major challenge to the efficient performance of machine learning algorithms in data stream environments. Non-representative features decrease the prediction accuracy and the running time of such algorithms. Feature selection has been widely used as a pre-processing technique to select representative features from data streams in order to tackle the dimensionality issue. However, existing feature selection methods (see Section 4.2) assume that features are *static* because they need to be known in advance so as to accurately select
a set of representative features. Therefore these methods are not appropriate for streaming features applications, where features are not static but arrive sequentially.

Data streams can be broadly classified into streaming data and streaming features [20]. In streaming data, the number of features is fixed while the instances arrive sequentially. Regarding streaming features, however, which is the focus of this chapter, the number of instances remains fixed while the features arrive sequentially and are processed one by one. In real-world applications such as Twitter, features such as slang words are dynamically created and therefore need to be processed upon their creation instead of waiting for all features to arrive, as required by traditional feature selection methods. Actually waiting for all features to arrive before starting the selection process is impractical, as the number of streaming features is unknown in advance and new features appear over time. The process of feature selection in streaming features applications comprises two tasks [45]: 1) the evaluation of the new feature to check whether this is a representative one based on a specific criterion (e.g. dependency of the features); and 2) the evaluation of the selected set of features to check whether they remain representative. The non-representative features are discarded. By following this process, we ensure that only representative features are included in the set of selected features. Additionally, we ensure that features that tend to be no longer representative over time are removed from the selected set of features, as new and more representative features will be added.

Traditional unsupervised feature selection methods [29, 30, 31] are inefficient to be applied to streaming features applications. This is because they require the full feature space to be known in advance, which is impractical in streaming features. Additionally, we would need to store large windows of the data streams, which is
infeasible due to the tremendous size resulting from the streams. Traditional feature selection methods have greater computational complexity, which makes them inappropriate for high-dimensional streaming features as they require fast and real-time processing. Moreover, in streaming features applications, algorithms should read the data only once due to the finite amount of storage space, and then non-representative features should be removed to allow storage. Finally, traditional feature selection methods are static by nature, meaning that they do not dynamically update their selected predictive features. Therefore this negatively affects the representativeness of the selected features. Hence, it is essential to consider the specific characteristics of streaming features when designing a feature selection method.

There are few methods [42, 43, 44] that have addressed feature selection for streaming features applications (see Section 4.2 for more details about these methods). These require data class labels so to guide the selection of representative features. To the best of our knowledge, the only unsupervised feature selection method for streaming features is the one proposed by Li et al. [45]. Although this method does not require class labels, their model is limited to the scenarios where link information should be established (i.e. a friendship relationship between Twitter users). Although a trick can be used to replace the link information by computing the similarity of the data, their model is then no different from traditional feature selection methods because it relies on the link information to evaluate the representativeness of the features. Also, the authors assume that the link information is stable, which obviously is not true as this could dynamically change.

The proposed UFSSF method extends the \( k \)-mean algorithm to cluster a stream of features that are not known in advance. It integrates three linearly dependent
Introduction

similarity measures, namely PCC (Pearson Correlation Coefficient), LSRE (Least Square Regression Error) and MICI (Maximal Information Compression Index), to incrementally measure the dependency of the newly arrived streaming features to decide whether or not to add them to the existing set of representative features. The features arrive sequentially and, upon arrival, they are processed in a real-time one by one. Linearly-dependent measures are used because they are not sensitive to the order and the scatter of the distribution of the features. Additionally, UFSSF incrementally updates the centroids to cope with concept drift in streaming features, as one feature might be representative only for a given time. After assigning a feature to its relevant cluster, the mean is updated and we compare the similarity of the arrived feature with the existing representative feature of the cluster.

Extensive experiments have been carried out to benchmark the proposed UFSSF against two well-known unsupervised methods, namely SPEC [17] and the one proposed in [1]. These methods are evaluated in terms of their prediction accuracy and the running time. The evaluation work is carried out in two parts. In the first part, we simulated the streaming features environment such that: a) features are not completely known in advance; and b) they are processed in real time. In the second part of the evaluation, we assume the existence of the entire stream in order to test the stability of the results. Therefore, we vary the number of features selected from the whole stream (i.e. select 10, 15, 30, etc from the entire stream). In both experiments, UFSSF outperforms these two selected methods in terms of prediction accuracy and running time.

In summary, our contributions are twofolds:

- Proposing unsupervised feature selection method for streaming features applications by working without the requirement of data class labels, features size or information about the link between the users, e.g., in Twitter.
- Adapting k-mean clustering algorithm to work in streaming features applications where features are not known in advance, and considering data stream properties such as one pass over data.

4.2 Related Work

To the best of our knowledge, few studies have been conducted on feature selection for streaming features applications. Perkins et al. [42] proposed a method, called grafting, which selects a subset of streaming features that have arrived so far as an integral part of a regularised learning process. It incrementally and gradually builds the selected subset of features in addition to training the predictive model using gradient descent. Because it works in an incremental way, this method can efficiently cope with the dynamic nature of the streams. However, in order to specify a good regulariser parameter value, this method requires an insight into the complete feature space in advance. Therefore, it cannot process streaming features of an unknown size. Alpha-investing [43] evaluates the representativeness of the arrived feature based on a dynamic threshold of error reduction (called $p$-value). In particular, the $p$-value is introduced to determine whether or not to add a feature to the selected set of features. Although Alpha-investing can process the unknown size of streaming features, no selected features can be removed from the set. Finally, Online Streaming Feature Selection (OSFS) was proposed in [44] to select representative features and remove redundant ones in real time. Whenever a feature arrives, OSFS measures its dependency on the available class labels and then adds the feature to the best candidate feature if this meets a specific criterion. OSFS can dynamically remove redundant features using the Markov Blanket. The characteristics of these methods are summarised in Table 4.1.
Table 4.1: Characteristics of existing feature selection methods for streaming features

<table>
<thead>
<tr>
<th>Method/characteristic</th>
<th>Streaming features</th>
<th>Unsupervised (i.e. no class labels)</th>
<th>Constrains (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grafting</td>
<td>Yes</td>
<td>No</td>
<td>NA</td>
</tr>
<tr>
<td>Alpha-investing</td>
<td>Yes</td>
<td>No</td>
<td>NA</td>
</tr>
<tr>
<td>OSFS</td>
<td>Yes</td>
<td>No</td>
<td>NA</td>
</tr>
<tr>
<td>Li et al.</td>
<td>Yes</td>
<td>Yes</td>
<td>limited to applications where link information must be established</td>
</tr>
<tr>
<td>UFSSF (proposed)</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
</tr>
</tbody>
</table>

The methods discussed above require the data class label as a guide to selecting representative features. However, in real-world applications most of the data is unlabeled and, moreover, labeling is time consuming. To the best of our knowledge, the only method that is unsupervised (i.e. no data class labels are required) and is applicable for streaming features applications as proposed in [45]. Although this method has good performance, it is limited to scenarios where link information must be established such as a friendship relationship between Twitter users. Also, the authors assume that the link information is stable, which obviously is not true as this could dynamically change.

4.3 Similarity Measures

Here we introduce the similarity measures used to measure the dependency of newly-arrived streaming features in order: 1) to allocate the feature to a relevant cluster; 2) to decide whether to add the feature to the selected set of representative features; and 3) to dynamically update the set of selected features by removing the no longer representative ones. The reason for adopting these linearly-dependent measures is their effectiveness for the purpose of feature selection as they are not sensitive to the location or to the scatters of the distribution of the features data [1]. Therefore, they are promising to work in data stream applications that have a dynamic rather than static nature. These linearly-dependent measures are illustrated
below. For all similarity measures $x$ denotes a cluster centroid and $y$ denotes a feature arriving from a stream.

**Pearson Correlation Coefficient (PCC)** [133]

PCC is a measure that computes the correlation between two random variables, and determines whether they have a linear dependency relationship. It can be computed by calculating the correlation either between a feature and a predicted class label or between a feature and a feature. Unlike the former, which measures the extent to which the features are correlated to their class labels, the latter is adopted in our method to measure the correlation between the streaming features and the clusters centroids in order to assign features to clusters. In fact, a feature and a feature correlation is better suited to our method as we are targeting unsupervised learning, which does not assume the existence of data class labels. Generally, correlation coefficient is fast and capable of identifying representative features without the need for pairwise correlation computation. Formally, it is computed as follows.

$$PCC(x, y) = \frac{n(\sum xy) - (\sum x)(\sum y))}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$  \hspace{1cm} (4.1)

The result of the correlation between $x$ and $y$ is between 0, which indicates that the feature and the cluster centroid are completely uncorrelated or 1, which indicates their complete correlation.

**Least Square Regression Error (LSRE)** [134]

LSRE computes and analyses the degree of the correlation between a feature and a cluster centroid by drawing a line that best fits the data. It is computed based on linear model $y = ax + b$, where $a$ and $b$ are given by minimising the mean square error and $n$ denotes the number of features which is always one (1) as we process one feature at a time. The error is the distance between the actual data
Similarity Measures

and the model data and is calculated using the following equations:

\[
\text{LSRE}(x, y) = y_n - (ax_n + b) \tag{4.2}
\]

\(a\) is the slope of the x and is calculated by

\[
a = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \left(\frac{\sum x}{n}\right)^2} \tag{4.3}
\]

\(b\) is the y-intercept and is calculated by

\[
b = \frac{\sum y - (a \sum x)}{n} \tag{4.4}
\]

The final result of the former equations show the degree of the linear dependency correlation of features from a stream and a cluster centroid based on given value of equation 4.2. They are completely correlated when \(\text{LSRE} = 0\).

Maximal Information Compression Index (MICI) \cite{1}

MICI is an index technique for measuring the similarity between a feature and a cluster centroid. Let \(\Sigma\) be the covariance matrix of the random features. MICI is defined as \(MICI(x, y) = \) the smallest eigenvalue of \(\Sigma\), i.e.,

\[
MICI(x, y) = \frac{\text{var}(x) + \text{var}(y) - \sqrt{(\text{var}(x) + \text{var}(y))^2 - 4 \text{var}(x) \text{var}(y)(1 - \rho(x, y)^2)}}}{2} \tag{4.5}
\]

A feature and a cluster centroid are linearly-dependent when the value of \(MICI\) is zero and the value increases as much as the amount of dependency decreases.

Preliminaries and Problem Statement

We formally introduce the notation used in this chapter and in this subsection we describe the problem of \textit{unsupervised feature selection for streaming features}. We assume a stream of feature vectors, \(F = \{f_1, f_2, \ldots\}\) (possibly infinite in their number), where each \(f_i\) is a vector of the feature values for \(n\) instances. Let \(F_t\)
be the features observed up to time $t$. E.g., if $F$ represents a stream of tweets from Twitter, then the features are individual words, and each post is an instance, and a feature vector would represent the frequency with which that word (feature) appears in each of the tweets. $F_t$ is the feature/word vectors observed up to time $t$. Each feature vector in $F$ arrives one by one; there are no restrictions on the order in which they arrive, and they do not have class labels.

We wish to maintain a representative set of features that approximates the feature stream seen so far. As the feature stream is potentially infinite in length and the relevant set of features could change with time due to concept drift, it is not efficient to wait for all the features to be collected. Let $R_t = \{f^R_1, f^R_2, \ldots, f^R_k\}$, $f^R_i \subset F_t$, $1 \leq i \leq k$, denote the set of $k$ representative features at time $t$. $k$ can range from 1 to $k_{max}$, the maximum number of representative features.

As features arrive one by one, the problem of unsupervised feature selection for streaming features is to maintain a set of representative features $R_t$, such that $R_t$ approximates the features $F_t$ observed up to time $t$. Each representative feature $f^R_j$ of $R_t$ represents a subset/cluster of features in $F_t$.

For each incoming feature $f_i$, the problem we are addressing in this chapter is related to two issues:

1. How to determine which existing representative feature and associated feature cluster $f_i$ must be assigned?

2. How to update the feature cluster and representative feature?

For both (1) and (2) above, the following three similarity measures have been selected: PCC [133], LSRE [134], and MICI [1]. We have chosen these linearly dependent measures because they are known not to be sensitive to the order and scatter of the features [1]. These similarity measures will measure the dependency
of streaming features in order to 1) allocate a feature to a relevant cluster; 2) decide whether to add a feature to a set representative features; and 3) dynamically update a set of selected features by removing those that are no longer representative.

**Problem Statement**

Given a streaming data $F = \{f_1, f_2, ..., f_n\}$ where $F$ is a stream of features vectors that arrive one by one, fixed in the number of instances and have no set of order nor the class labels, we want to maintain a representative subset of them that approximates the representative features. The selected subset of representative features is used to train the data mining algorithms for the purpose of classification.

The difference between feature selection from a data stream and traditional static feature selection is that the entire feature space is not known in advance and it is inefficient to wait for all the features to be collected. Therefore, a feature must be evaluated for its representativeness upon its arrival in a real-time. Also, the dimensionality of the stream increases dramatically because of the large size of the streaming features. As a result, this requires a long processing time and large capacity storage to select the representative features.

### 4.4 The UFSSF Method

This section provides details of the proposed method. We first define the concepts of *cluster centroid* and *representative feature*.

**Definition 1 (Cluster Centroid).** We represent each feature cluster by a centroid, which is a weighted mean of all the features assigned to it. The weights are largest for recently arrived features, and smallest for features that arrived in the distant past.
Definition 2 (Representative Feature). A feature assigned to a cluster is considered to be representative if it has the maximum similarity to the cluster’s centroid amongst all other stream features assigned to the same cluster. Given a centroid $c_r$, $f_r \in F_t$ is a representative feature in $c_r$, namely $f_r \in R_t$, if and only if we have one of the following properties:

\[ PCC(f_r, c_r) > PCC(f_j, c_r) \] (4.6)
\[ LSRE(f_r, c_r) < LSRE(f_j, c_r) \] (4.7)
\[ MICI(f_r, c_r) < MICI(f_j, c_r) \] (4.8)

where $R_t$ is the set of current representative features and $f_j$ is any feature of $c_r$.

Therefore, any feature that is not representative is therefore discarded. This will lead to less usage of space and will also allow UFSSF to rapidly filter out non-representative features in a dynamic feature space.

4.4.1 The Framework

This section explains how the proposed UFSSF method computes the set of representative features. This method comprises two parts: 1) adding features to the set of representative features; and 2) updating the set of representative features by removing the ones that are no longer representative. To do so, we employ the similarity measures mentioned in Section 4.3. We rely on clustering approaches that are capable of selecting the representative features without requiring data class labels. The $k$-mean algorithm [144] works well with multi-dimensional datasets, and is therefore well suited for streaming features.

Linearly-dependent measures are more efficient for the purpose of feature selection as they are not sensitive to the order and the scatter of the distribution
A new feature arrives → Assign the new feature to the most similar cluster → Update the cluster centroid

Update the set of selected features → Replace representative feature → No → Discard the feature

Yes → Update the set of selected features

Figure 4.1: Framework of UFSSF process

of the features. Three well-known linearly-dependent measures (i.e. PCC, LSRE, MICI) are used for the following reasons. Firstly, a single similarity measure might produce bias towards a specific method, and therefore will produce a better selection of representative features compared with the other methods. Secondly, the three measures proved their effectiveness for feature selection as experimentally shown in [1]. Therefore, PCC, LSRE and MICI are used in the $k$-mean algorithm to compute the dependency between features and cluster centroids.

The following steps show how UFSSF selects a set of representative features from a stream of features. These steps are illustrated in Figure 4.1. Features are processed one by one upon their arrival according to a first-in-first-out strategy as they are not known in advance. The first step is the initialisation of the clusters and the representative features:

- UFSSF assigns the first newly-arrived $k$ features from a stream as centroids of $k$ number of clusters. For example, if $k=10$ then the first ten features
collected from a stream are the initial centroids of 10 clusters.

- UFSSF sets the initial centroid of every cluster as the initial representative feature of that cluster.

**Whenever a feature \( f_j \) arrives:**

For every similarity measure PCC/LSRE/ MICI, the following steps are carried out to update the representative feature set:

- The similarity between \( f_j \) and the centroid of every cluster is computed. \( f_j \) is assigned to a cluster \( C_{lus} \) if \( f_j \) has the maximum similarity to \( C_{lus} \)'s centroid amongst all other clusters centroids. The mean of \( C_{lus} \) is then incrementally updated and \( f_j \) is assigned to \( C_{lus} \).

- In \( C_{lus} \), we compare the similarity (say \( S \)) between \( f_j \) and the representative feature (i.e. \( f_r \)) with \( C_{lus} \)'s centroid \( c_r \). If \( S(f_j,c_r) > S(f_r,c_r) \), \( f_j \) is set as the representative feature and \( f_r \) is removed.

- The representative feature from every cluster comprises the set of representative features.

UFSSF has one pass over the data, as it reads the stream of the data only once. Additionally, UFSSF incrementally updates the mean of the clusters: i) to accurately measure the representativeness of the features (as a feature \( f_r \) might be representative at time \( t \) but not in \( t+1 \)); and ii) to tackle the concept drift in clusters that could result from the dynamic nature of the stream. This helps to improve the prediction accuracy of the classifiers.

Finally, UFSSF requires only a reasonable storage capacity as it stores only the representative feature and the centroid of every cluster. Because UFSSF is able to meet the requirements of major streaming applications, we believe that
it is capable of efficiently working in streaming features applications as shown by the experimental results. The pseudo code of UFSSF is given in Algorithm 1. Comments on the pseudo code are provided in Table 4.2 so to make it more readable and understandable.

Table 4.2: The comments on the provided pseudo code in UFSSF algorithm

<table>
<thead>
<tr>
<th>Comments</th>
<th>Lines#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization of centroids matrix and representative features matrix</td>
<td>1,2</td>
</tr>
<tr>
<td>Assigning the first $n$ features to be both the centroids and the</td>
<td>3-6</td>
</tr>
<tr>
<td>representative features of the first n clusters</td>
<td></td>
</tr>
<tr>
<td>Initialization of the indexes of the centroid features</td>
<td>7-10</td>
</tr>
<tr>
<td>Looping over the remaining stream of features</td>
<td>11</td>
</tr>
<tr>
<td>Compute the similarities between the arriving feature and every centroid</td>
<td>12-14</td>
</tr>
<tr>
<td>Finding the most similar cluster centroid to the arriving feature</td>
<td>15-20</td>
</tr>
<tr>
<td>Incremental mean computation</td>
<td>21</td>
</tr>
<tr>
<td>Computing the similarity of arriving feature $f_j$ and representative</td>
<td>22,23</td>
</tr>
<tr>
<td>feature $f_r$ to the cluster centroid</td>
<td></td>
</tr>
<tr>
<td>Evaluating the arrived feature for its representativeness in a cluster</td>
<td>24-34</td>
</tr>
<tr>
<td>and finding the corresponding index</td>
<td></td>
</tr>
<tr>
<td>Return the matrix of the selected representative features</td>
<td>35</td>
</tr>
</tbody>
</table>

4.5 Experimental Evaluation

This section describes the experimental setup of the proposed UFSSF method. The evaluation of UFSSF will provide answers to the following questions:

- How accurate is UFSSF in selecting a set of representative features?
- How efficient is UFSSF in terms of running time?

Following the experimental settings given in [45, 44], the experiments are conducted in two phases:
Algorithm 2: UFSSF

Input: $D = \{f_1, f_2, ..., f_n\}$, a stream of features vectors
Input: $j = \{1 = PCC, 2 = LSRE, 3 = MICI\}$, similarity measure
Input: $n$, number of clusters centroids
Output: representative features

// Initialization of centroids matrix and representative features matrix
cluster_centroids=NaN(size(D,1),n);
representative_features(:,q)=D(:,q);

// Assigning the first $n$ features to be both the centroids and the representative features of the first $n$ clusters
for $q=1:n$ do
    cluster_centroids(:,q)=D(:,q);
    representative_features(:,q)=D(:,q);
end

feature_indexes=zeros(1,n);
for $u=1:n$ do
    feature_indexes(1,u)=u;
end

// Looping over the remaining stream of features
for $w=n+1:size(D,2)$ do
    for $r=1:n$ do
        // Compute the similarities between the arriving feature and every centroid
        similarity(r,1)=calcDistance(D(:,r),D(:,w),j);
    end

    // Finding the most similar cluster centroid to the arriving feature
    if $j==1$ then
        cluster_most_similar=find(similarity==max(similarity));
    else
        cluster_most_similar=find(similarity==min(similarity));
    end

    // Incremental mean computation
    cluster_centroids(:,cluster_most_similar)=mean([cluster_centroids(:,cluster_most_similar),D(:,w)],2);

    // Computing the similarity of arriving feature $f_j$ and representative feature $f_r$ to the cluster centroid
    $f_j$=calcDistance(cluster_centroids(:,cluster_most_similar),D(:,w),j);
    $f_r$=calcDistance(cluster_centroids(:,cluster_most_similar),representative_features(:,cluster_most_similar),j);

    // Checking the representativeness of feature
    if $j==1$ then
        if $f_j>f_r$ then
            representative_features(:,cluster_most_similar)=D(:,w);
            feature_indexes(1,cluster_most_similar)=w;
        end
    else if $f_j<f_r$ then
        representative_features(:,cluster_most_similar)=D(:,w);
        feature_indexes(1,cluster_most_similar)=w;
    end
end

Return representative_features;

- In the first part of the evaluation, we simulated the streaming features environment such that: a) features are not completely known in advance; and b) they are processed in real time [44]. The feature space is split into five
subsets: 20%, 40%, 60%, 80% and 100%. First, we pick the 20% subset of streaming features and then 40% and so on to sequentially simulate the arrival of the features. In each subset of streaming features, we apply UFSSF, SPEC [17] and [1] to select representative features. We ensure that all the methods select the same number of features for a fair comparison.

- In the second part of the evaluation, we vary the number of features selected from the full feature stream, i.e., 100% of features. In this case, we assume the existence of the entire space. The purpose of this is to test the stability of the results and to avoid randomness.

UFSSF is compared with two well-known traditional unsupervised feature selection methods, namely the one proposed in [1] and SPEC [17]. To the best of our knowledge, no other unsupervised feature selection method has been developed for streaming features applications without requiring link information. Therefore, these two methods have been selected as they are the most common traditional unsupervised feature selection methods (e.g. batch applications). Although the benchmarked methods are not designed for streaming features applications, the way we conducted the experiments ensures the fairness of the comparison. We apply the UFSSF, SPEC [17] and [1] to every subset individually and select the same number of features. Also, the entire dataset is tested with different numbers of selected representative features for every method in order to compare the non-streaming features benchmark methods.

The respective computed representative feature sets of these three methods are evaluated by taking the average of three well-known classifiers, namely Naive Bayes [139], J48 Decision Tree [140] and Lazy Nearest Neighbour [141] (also called IB1). In addition to these classifiers, the $k$-fold-cross validation is applied on all
selected features to produce better results by avoiding the problem of over fitting data. The selected feature set is first divided into subsets of equal size depending on the selected \( k \) folds. Then, only one \( k \) is used as a testing subset and the rest are used as training subsets. Finally, the average value of all folds is set to be the average result. In the evaluation, \( k \) is set to 10 to demonstrate the efficiency of our proposed algorithm, as suggested in [145]. All the three algorithms are implemented in Matlab programming language. They are executed under Mac operating system OS X EI Capitan with 2.4 GHz Intel Core 2 Duo and 8 GB RAM.

4.5.1 The Datasets

Three datasets with various dimensionality were used to evaluate the performance of the proposed UFSSF and the benchmark methods. We selected these datasets because they are commonly used for the aim of data mining algorithms as well as they are from diverse domains. The three adopted datasets have been acquired mainly for the purpose of classification and clustering as clustering is a part of the proposed method to select the representative features. They are found in the UCI Machine Learning Repository website as provided below. Here is a brief description of each data set:

- **Spambase**: is a multivariate data set that contains spam and non-spam email classes where each email is described by 57 real data type features. The total number of emails (records) is 4601. [https://archive.ics.uci.edu/ml/datasets/Spambase](https://archive.ics.uci.edu/ml/datasets/Spambase)

- **Waveform**: is a multivariate data set that consists of 40 continuous features some of which are noise. These features have 5000 instances. [https://archive.](https://archive.2c2c)
ics.uci.edu/ml/datasets/Waveform+Database+Generator+(Version+2)

- Ionosphere: This dataset is collected from a radar that is provided by a system in Goose Bay, Labrador. The system consists of 16 antennas with 6.4 kilowatts power transmission. The aim is to investigate whether electrons exist in the ionosphere. The radar returns two classes either good or bad. The dataset consists of 351 instances that are described by 34 features. https://archive.ics.uci.edu/ml/datasets/Ionosphere

4.5.2 Evaluation Metrics

The aim of the proposed UFSSS method is to select a representative subset of streaming features. The selected features should improve the accuracy of the classifiers in their classification tasks. The process of selecting the streaming features should be done in an acceptable running time. Therefore, the metrics considered for the evaluation are grouped according to classification accuracy metrics and running time.

(A) Classification accuracy metrics

Three classification accuracy metrics are adopted, namely False Positive Rate (FPR), Precision and F-measure. These metrics investigate whether the selected subset of stream features is competitively enhancing the accuracy of the classifiers. These metrics are computed as follows. Assume that we have a group of 27 people who suffer three types of diseases (cold) 8 people, (diabetes) 6 people and (blood pressure) 13 people. They are classified as shown in Table 4.3.
Table 4.3: Confusion matrix of the classification results

<table>
<thead>
<tr>
<th>Actual class label</th>
<th>Predicted class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>cold</td>
<td>cold 5 diabetes 3 pressure 0</td>
</tr>
<tr>
<td>diabetes</td>
<td>cold 2 diabetes 3 pressure 1</td>
</tr>
<tr>
<td>pressure</td>
<td>cold 0 diabetes 2 pressure 11</td>
</tr>
</tbody>
</table>

Considering the class cold as an example, we have the following:

- 5 True Positive (TP): the number of people who actually have cold and are correctly classified as having cold.
- 3 False Negative (FN): the number of people who actually have cold and are incorrectly classified as having diabetes.
- 2 False Positive (FP): the number of people who actually have diabetes and are incorrectly classified as having cold.
- 17 True Negative (TN): the number of all remaining people who are correctly classified as non-having cold.

Now the following equations are computed in order to calculate FPR, Precision and F-measure.

- False Positive Rate (FPR): how many times people are classified as having cold while they are not.
  \[
  FPR = \frac{FP}{FP + TN}
  \]
  \[ (4.9) \]

- Precision: how many times people are classified as having cold and they actually have cold.
  \[
  Precision = \frac{TP}{TP + FP}
  \]
  \[ (4.10) \]
• F-measure: is the harmonic mean of precision and recall, which precisely demonstrates the accuracy of the classification task [146].

\[ F - \text{measure} = \frac{2TP}{2TP + FP + FN} \]  

(B) Running time

UFSSF’s running time is compared with Mitra’s method [1] and SPEC [17] to evaluate their efficiency. To precisely measure the running time, we apply these methods to different percentages of arrived streaming features and count the time taken by each method to select a subset of features.

4.6 Results and Analysis

This section discusses the experimental results carried out for UFSSF and the benchmark methods. Three evaluation metrics are used to evaluate the accuracy / prediction of the selected features, namely FPR, Precision and F-measure. Also, the running time is measured in seconds. First, we present the results related to the stream of features, where features are not known in advance and arrive sequentially. Then, we present the results relating to the one that considers the existence of the entire data stream, where we investigate the selection of different numbers of features to show the stability of the provided results. Finally, we present the results relating to the efficiency of UFSSF and those for the two other methods in terms of the running time. For each dataset, every method runs its own similarity measure/s to investigate its prediction accuracy. Mitra’s method [1] already includes these three similarity measures (i.e. PCC, LSRE and MICI), while SPEC works with the RBF Kernel similarity measure.

Figures 4.2, 4.3 and 4.4 show the results of the streaming features where they are not completely known in advance but arrive sequentially. For the Waveform
dataset as in Figure 4.2, UFSSF outperforms the method in [1] and SPEC [17] with all different percentage of streaming features for all similarity measures. Generally, it has the lowest FPR and the highest precision and F-measure. Although UFSSF and the method proposed in [1] have very similar levels of accuracy at the early stage of arrived streaming features 20% and 40%, the accuracy of UFSSF distinctly increases for all other percentage of arrived streaming features 60% - 100%.

Similarly, for the Spambase dataset, as shown in Figure 4.3, UFSSF tends to consistently have the lowest FPR and the highest precision and F-measure from 60% until the arrival of all the features. Again, the accuracy of UFSSF is similar to that achieved by the two baseline methods in the early stages of arriving features (i.e. 20% and 40%). UFSSF waits for the arrival of more features from these two datasets to significantly perform well. Indeed, UFSSF gradually builds the model due to the incremental updating of the clusters, which affects the selection of representative features. UFSSF processes a stream of features one by one and incrementally selects the representative feature seen so far from a cluster. Therefore, in a few scenarios where we do not really have good representative features, UFSSF is forced to select the maximum representative features that have just arrived. Therefore, the accuracy gradually improves with the arrival of more features. Conversely, the other two methods statistically search the complete subset of the streaming features that have arrived and select the best of them.

For the Ionosphere dataset, as shown in Figure 4.4, UFSSF has significantly the lowest FPR and the highest precision and F-measure with all different percentages of streaming features compared to the two other methods. This is valid when using either PCC, LSRE or MICI as the similarity measure for UFSSF and [1].

SPEC [17] and Mitra’s method [1] do not incrementally update their models to cope with the dynamic nature of the streams. A feature arriving from the
stream can be representative at only a specific time due to dynamic nature of the stream. Conversely, UFSSF incrementally updates its clusters to check whether the selected representative features are still representative for every arriving new feature. Therefore, the features selected by UFSSF show that it outperforms the other two methods in terms of classification.
Figure 4.2: Comparison of classification accuracy metrics of different methods on Waveform dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.
Results and Analysis

Figure 4.3: Comparison of classification accuracy metrics of different methods on Spambase dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.
Figure 4.4: Comparison of classification accuracy metrics of different methods on Ionosphere dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of streaming features while the y-axis denotes the corresponding accuracy metric.
Results and Analysis

Figures 4.5, 4.6 and 4.7 demonstrate the accuracy of UFSSF and that of the baseline methods when selecting different numbers of features by considering the entire feature space as a stream (i.e. 100%). For the Waveform dataset, as depicted in Figure 4.5, UFSSF significantly outperforms (lowest FPR and highest precision and F-measure) the method in [1] followed by SPEC [17], respectively. This holds for all the different numbers of selected features and for all the different similarity measures. However, when only 10 features are selected, UFSSF and [1] are quite similar in terms of prediction accuracy when LSRE and MICI are used as similarity measures.

For the Ionosphere and Spambase datasets, as illustrated in Figures 4.6 and 4.7 respectively, UFSSF has either a slightly better or a competitive prediction accuracy compared to [1] and SPEC [17]. This is valid for all different numbers of selected features and for all different similarity measures. Baseline methods are indeed designed to work with statistical datasets. In contrast, UFSSF is designed to work in a stream environment where features are not completely known in advance but arrive sequentially. Although UFSSF has a lower accuracy than the baseline methods for a few selected features, the difference in accuracy is negligible.

It worth pointing out that the UFSSF is not sensitive to the order of the features. This is due to the method used for selecting the representative features. Indeed, every cluster retains only the feature that has the maximum similarity to the cluster centroid. Therefore, it does not matter which feature arrives first as the similarity is computed based on its values.
Figure 4.5: Comparison of classification accuracy metrics of different methods on Waveform dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric.
Figure 4.6: Comparison of classification accuracy metrics of different methods on Ionosphere dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric.
Figure 4.7: Comparison of classification accuracy metrics of different methods on Spambase dataset. The average results of Naive Bayes, IB1 and J48 decision tree classifiers are computed. The first, second and third rows of the figure show respectively the results of the FPR, Precision and F-measure evaluation metrics. The columns of the figure show the results when considering different similarity measures. The x-axis denotes different numbers of selected features while the y-axis denotes the corresponding accuracy metric.
The running time for the three methods is depicted in Figure 4.8. UFSSF has the lowest running time for all different similarity measures on the three datasets. It consistently outperforms the baseline methods on all the different percentages of the streaming features. The method in [1] is competitive with UFSSF while SPEC [17] has a higher running time. [1] relies on K-Nearest Neighbour (K-NN) search strategy to partition the subset of arrived features. As a result, it has a higher running time due to the computation of the similarity between features. The performance of SPEC [17] is the worst in terms of running time due to the time required to build the Laplacian matrix, which is computationally expensive. The reason for UFSSF having the best running time is that it does not have to search the entire subset of newly arrived features as the other two methods do when selecting features. Instead, UFSSF processes the arriving features one by one by computing their dependency on only the cluster’s centroids, which are very few compared to the number of streaming features.
Figure 4.8: Comparison of different methods’ running time for three datasets. The first, second and third rows of the figure show the running times on Waveform, Spambase and Ionosphere datasets respectively. The columns of the figure show the results when considering different similarity measures. The x-axis denotes the percentage of arrived streaming features while the y-axis denotes the corresponding time (in seconds) taken by the methods to select a set of features.
4.7 Conclusion

This chapter proposed an unsupervised feature selection method to reduce the dimensionality of a stream in streaming features applications, known as the dynamic feature space. In such applications, traditional features selection methods are inefficient as the entire features are not available in advance; rather, they arrive sequentially one by one for the learning machines. Unlike existing streaming features methods that require class labels, UFSSF can efficiently select a set of representative features without requiring class labels or information such as the link between users. Therefore, it is widely applicable for various applications. With UFSSF, a \( k \)-mean clustering algorithm is extended to work in streaming features applications. It clusters a stream of features that are not known in advance. It uses three similarity measures namely, PCC, LSRE and MICI, in order to: a) allocate a feature to a relevant cluster; b) decide whether to add the arrived feature to the set of representative features; and c) decide whether to dynamically update a set of selected features by removing those that are no longer representative. Our experiments considered: 1) the streaming features settings where features are not completely known in advance and rather arrive in different percentage. 2) the entire features space as a stream with different numbers of selected features, to investigate the stability of our findings. 3) the time taken by every method to generate its selected features. Experimental results show that UFSSF generates a representative feature set with the lowest running time. The selected set of representative features has mostly achieved the best prediction accuracy according to FPR, Precision and F-measure evaluation metrics.
Chapter 5

OUDVFS: Online Unsupervised Feature Selection for Dynamic Multi-Views

In the previous chapter we addressed the problem of dimensionality reduction for streaming features. In streaming features applications, the number of features increases while the number of instances remains fixed. In this chapter, we reduce the dimensionality of multi-view applications/data where both the number of features and instances can increase over time. In real-world applications, such as Web clustering, data arrives from diverse groups (i.e. sets of features) and therefore exhibit heterogeneous properties. Each feature group is referred to as a particular view [147]. For example, a Web page can have a group of features related to image, a group of features related to text and a group of features related to hyperlinks. Each group of these features is called a view.

Multi-view learning provides complementary information for machine learning algorithms. However, it results in high-dimensionality because, if data is looked at
data from different views, the extra views would definitely result in extra dimensions. Therefore, feature selection has been widely used as an efficient method to select only representative features from the views so to reduce the dimensionality of the data. Traditional/single-view feature selection methods, which are not designed for multi-view learning, can be indirectly used for multi-view applications. However, they do not exploit the similarity/correlation between the views as all the views must be concatenated into a single view matrix. As the views are concatenated into a single matrix, it loses its actual meaning, which would result in an inaccurate representation of the selected features [38]. Recently, several feature selection methods [39, 148, 41] have been designed for multi-view applications. However, they all assume that the number of views is static, which is not valid for real applications as existing data can be updated in terms of views with additional views at any given time. Also, new features can be added to the existing views if needed.

An Online Unsupervised Feature Selection for Dynamic Views (OUDVFS) is proposed to address the aforementioned limitations. It includes a clustering-based feature selection method that incrementally clusters the views; hence, the set of selected representative features is updated at each clustering step. Specifically, the selected set of representative features is dynamic so that those features that are no longer representative are removed. Both the number of views and the number of instances can increase over time. Experimental results demonstrate that the selected set of features selected by OUDVFS has the best classification accuracy compared with well-known single-view and multi-view unsupervised feature selection methods.
5.1 Introduction

In machine learning, data that arrive from heterogeneous views (i.e. multiple heterogeneous sources of data) are more likely to provide complementary information than does a single view; hence, these are known as multi-view data [25]. Each instance in multi-view data has different groups of features. In other words, different views are different representations of the same set of instances and each view comprises a group of features. For example, in medical applications, it is better to look at patients (i.e. instances) from different groups (i.e views) of laboratory tests (i.e. features) for precise medical diagnostic. However, not all features in different views are representative, thereby producing the problem of high-dimensionality. Indeed, non-representative features not only increase the time complexity of the learning algorithms but also reduce the classification accuracy [149]. In addition, non-representative features waste the storage capacity [150]. Therefore, feature selection is an efficient method that can reduce the dimensionality of the original feature set. This can be achieved by selecting only a reduced set of features from the original feature space that can represent the entire feature set.

Traditional unsupervised feature selection methods [17, 1, 36, 151] assume that data is independent and identically distributed (i.i.d.). However, this assumption is not valid when there are heterogeneous families/views of features. Also, these traditional methods are unable to exploit the correlation among the views if they are concatenated. For these two reasons, they are inappropriate for multi-view learning [38].

There are three main approaches for clustering the features in multi-view data: concatenation, distribution and centralisation [152]. With the concatenation methods suggested in [153, 154, 17, 1, 36, 151], all the data views are combined into a
single matrix ignoring the heterogeneous nature of the multi-view data. However, data loses its actual meaning if it is combined, resulting in poor feature selection. With the distribution methods as suggested in [155, 156], features are selected from each view independently. However, this method focuses on local feature selection at each view and does not correlate the features of all the views; thereby, possibly resulting in redundant features. Finally, the centralisation method as proposed in [157] simultaneously considers the correlation of the features of all the views, thereby producing a better selection of representative features.

There are few unsupervised feature selection methods for multi-view learning (see Section 5.2). However, they all assume that the number of the views is static, which means that the views are complete and exist in advance and no new views can be added. However, this assumption is not valid for real applications as new views and features can be added at any given time. Also, the number of instances can increase. To the best of our knowledge, this is still an open issue that has not been addressed, which motivated us to investigate it in this chapter. Therefore, our proposed algorithm overcomes the existing solutions of unsupervised feature selection methods for multi-view applications by developing a feature selection method for dynamic multi-views applications and in online manner so that both instances and views can increase over time and be clustered incrementally.

**Problem Statement**

The problem we aim to solve is to select a set of representative features that approximates the original feature space. The challenge is how to incrementally update the set of selected representative features among all the views clustered to date so that: case 1) the views, which are different sets of features, increase over time so that existing instances can be represented by new additional views; case
2) the number of instances can increase also (i.e. online learning) and the new instances are shown in all the current views; case 3) different instances are shown by different combinations of views. In this chapter we addressed case 1 and 2. Case 3 will be investigated in the future.

There are two applications that can illustrate the problem. Health care application is a good example where we have a group of patients (i.e. instances) who have different groups/views of medical tests (i.e. features) related to a prediction of early symptoms of heart attack. Let us say that we initially have two groups or views of medical tests namely, medical tests belonging to a heart view and medical tests belonging to a physical therapy view as shown in Figure 5.1(a). There are three possible scenarios: 1) in Figure 5.1(a), new patients do all the medical tests of existing views; 2) in Figure 5.1(b), current patients have conducted additional medical tests, which relate to diabetes (additional view); 3) different patients can do different medical tests presented in different views, and not all of them. In this chapter we have addressed the first and second scenarios while the third one will be addressed in future work. This application example shows that both the number of views and the number of instances can increase at any time.

Another application is a Web page clustering where a set of pages (i.e. instances) is presented in text view and image view. This set of pages might be updated later on, in terms of views, with additional video view. Also, the number of the Web pages can increase.

Formally, lets \( \{ X^{(v)}, v = 1, 2, \ldots \} \) is a given dataset of different views such that views can increase over time. Following the online learning, \( N = \{ inst_1, inst_2, \ldots \} \) is a group of instances in \( n_v \) views where \( X^{(v)} \in R^{(N_{inst} \times D_v)} \) and \( D_v \) is the dimension of the instances in the \( v^{th} \) view.

This chapter proposes a novel method, namely Online Unsupervised Feature
Selection for Dynamic Multi-Views (OUDVFS). It consists of two parts, namely clustering and feature selection. These two steps are technically challenging as both features and instances can increase over time. Whenever new instances or views arrive, OUDVFS performs the proposed clustering step. However, the resulting clusters might be very large as the views and the instances are not static. Therefore, the merging step is applied in order to reduce the number of the clusters to the required number of features. A single feature is selected as the representative feature of each of the merged clusters. The representative feature of each cluster is the one that has the minimum distance to its cluster centroid. These representative features of all the clusters comprise the set of selected features and they are updated at each step that clusters either new instances or new views. The proposed
Related Work

OUDVFS is evaluated and compared with three well-known unsupervised feature selection methods, namely OMVFS [40], UFSSF [158], and SPEC [17]. In this evaluation, three real multi-view datasets were used. OUDVFS significantly outperformed the benchmark methods when selecting a set of representative features that was able to classify the data accurately.

5.2 Related Work

Feature selection is an efficient method for reducing the high-dimensionality of multi-view data. There are two ways to apply existing feature selection methods to multi-view data, namely indirect and direct approaches. In the indirect approach, all the views are concatenated into one single matrix. Then, traditional/single-view feature selection methods (e.g. Fisher Score [35], sparse multi-output regression [36], Laplacian Score [16], SPEC [17] and Multi-Cluster Feature Selection [37]), which are designed for homogeneous data, can be applied on this single matrix.

UFSSF [158] addressed the problem of feature selection with streaming features. It dynamically updates the set of selected features. However, the indirect approach of aforementioned methods is inefficient for multi-view learning. By concatenating the views, these methods cannot exploit the correlation among the views as the data lose their actual meaning due to the concatenation. Therefore, these methods would result in an inaccurate representation of the selected features.

In the direct approach, feature selection methods are designed to select features from multi-view data. There are few well-known unsupervised feature selection methods for multi-views data; their characteristics are shown in Table 5.1. Table 5.1 demonstrates that all the methods, which are included in the table, were designed to tackle the problem of multi-view learning. In addition, none requires
Related Work

Table 5.1: Characteristics of existing multi-view feature selection methods

<table>
<thead>
<tr>
<th>Method/characteristic</th>
<th>Multi-view</th>
<th>Dynamic views</th>
<th>Online</th>
<th>Streaming features</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUMFS</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MVFS</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Wang et al.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>LUFS</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SRRS</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ASVW</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OMVFS</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OUDVFS (proposed)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

data class labels in order to select features. Also, OMVFS and OUDVFS are the only methods that work in an online environment. Moreover, UFSSF and OUD-FVS are the only methods that addressed the problem of streaming features where complete features do not exist in advance; rather, they arrive sequentially. However, no current feature selection method has attempted to introduce and solve the problem of dynamic views, other than our proposed OUDVFS method.

Below, we demonstrate the workings of the methods shown in Table 5.1.

Adaptive Unsupervised Multi-View Feature Selection (AUMFS)

AUMFS [39] selects representative features across all the views simultaneously. It was proposed in order to tackle the problem of multi-view learning without the need for data class labels. To select representative features, AUMFS benefits from three information resources: the correlation, the data cluster structure and the similarity between the views. A regression model was improved to predict cluster labels based on $l_{2,1}$-norm as it imposes joint sparsity on all the views. To do so, AUMFS relies on data cluster structure. Also, a graph regularization is built based on the data similarity across all the views. Then, spectral clustering is performed to produce the pseudo labels. To form the objective function, the learned graphs from each view are united with the weight vector of all the views. Tang et al. proposed similar method, namely Multi-view Feature Selection (MVFS) [159]. However, the
main difference is that MVFS learns only one feature weight matrix of each view and not those of all the views. Then, it utilises the feature weight matrix to fit the pseudo class labels by the least square and the norm regularizer.

**Multi-view Clustering and Feature Learning via Structured Sparsity**

Wang et al. [148] assume that previous studies assign the same weight for all data in a single source. However, they believe that some views in some applications have more representative features than others. They demonstrated this with an example of an image processing application. In such an application, they believe that colour features are more discriminative in recognising stop signs than are other features. Therefore, they addressed this problem by developing an unsupervised feature selection method that assigns weight to features. To do so, they adopted structured-sparsity regularizer to select representative features. They first apply $l_1$ norm regularizer on the feature weight matrix in order to select more useful views. In addition, they apply $l_2$ norm regularizer to the features within the selected views in order to find the representative features. Therefore, the selected features should be able to discriminate the cluster structure.

**Unsupervised Feature Selection for Linked Social Media Data (LUFS)**

LUFS [26] targets applications in which linked data can be established. In particular, it selects representative feature in social media where the data are linked. The algorithm does not require class labels to select features. It extracts information from the linked data in the form of link information and attribute-value informa-
Online Unsupervised Multi-view Feature Selection (OMVFS)

OMVFS [40] addressed the problem of feature selection for multi-view online data. Specifically, the number of instances increases while the number of the views is fixed. It deploys feature selection into a clustering objective function via non-negative matrix factorization with sparse learning. It processes the streaming data in the form of chunks and aggregates into small matrices all the previous data to date. These matrices are updated as new data arrives in order to learn the feature selection matrices. Therefore, the selected features are updated correspondingly.

For each view, OMVFS selects a subset of features to represent that view. Moreover, OMVFS combines the data of all the views into a consensus cluster indicator matrix. The derived objective function is as follows:

$$\min_{U, \{V^{(v)}\}} \sum_{v=1}^{n_v} \left( \| X^{(v)} - UV^{(v)T} \| + \beta_v \| V^v \|_{2,1} \right)$$

such that $U^T U = I$, $U \geq 0$, $V^{(v)} \geq 0$, $v = 1, 2, ..., n_v$. $X^{(v)}$ is the dataset with different views. $U$ is the cluster indicator, $V^{(v)}$ is the features matrix and $\beta_v$ is the sparsity parameter. OMVFS applies $l_{2,1}$ to $V$ in order to sort the features and select the ones with higher weight.

Sparse Low-Rank Representation through Multi-view Subspace Learning (SRRS)

Many existing dimension reduction methods assume that the data (i.e. instances) is complete (i.e. there is no missing values). However, it is more likely in real applications that data samples are not complete for various reasons. For example,
there may be restricted access to data or sensor failure. Therefore, SRRS [41] was proposed to address the problem of incomplete or missing multi-view data (i.e. instances or samples) in one or multiple views. To impute the missing values, SRRS jointly computes: 1) the intra-view relation by the sparse low-rank representation; 2) interview relations by global subspace representation. Then, a sparse feature selection via rank minimisation was proposed to find a set of representative features.

Multi-view Unsupervised Feature Selection with Adaptive Similarity and View Weight (ASVW)

MVFS and AUMFS measure the similarity of each view independently. They build a fixed Laplacian graph for each view individually. Therefore, they ignore the correlation across the views. ASVW [38] addressed this problem by exploiting the correlation across the view adaptively. The objective function is formulated by a global Laplacian graph across all the views. Also, the objective function is united with sparse norm constrain in order to select representative features. The objective function is given in Equation 5.2

$$\min L(W_1,\ldots,W_V,\alpha,S) = \sum_{v=1}^{V} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_v ||W_v x_i^{(v)} - W_v x_j^{(v)} ||^2 (S_{ij})^{r_2} + \lambda \sum_{v=1}^{V} ||W_v||^p_{2,p}$$

(5.2)

such that $W_v^T W_v = I$

$$\sum_{v=1}^{V} \alpha_v = 1, \alpha_v \geq 0$$

$$\sum_{j=1}^{n} S_{ij} = 1, S_{ij} \geq 0, || S_i ||_0 \leq k$$
$W_v$ is the projection matrix of each view. $S_{ij}$ is the similarity matrix. $k$ is the number of close neighbours. $||W_v||_{2,p}$ is the $L_{2,p}$ norm. $\lambda$ is the non-negative control parameter. $r_1$ is a balance parameter in order to avoid trivial solution. $X^{(v)}$ is the data in different views.

All the direct approach methods discussed above are designed for multi-view learning. However, they assume that the number of the views is static. We mean by static that the views are completely exist in advance and there are no new views can be added. However, this assumption is not valid for real applications as new views can be added at any given time. Also, the number of instances can increase (i.e. online). To the best of our knowledge, this is still an open issue that has not been addressed, which motivated us to investigate it in this chapter. Therefore, our proposed algorithm overcomes the limitations of existing solutions for unsupervised feature selection methods for multi-view applications by developing a feature selection method for dynamic views applications so that both instances and views can increase overtime and be clustered incrementally.

## 5.3 The Proposed OUDVFS Method

This section provides details of the proposed method. We first define the concepts of the dynamic views, the online learning and the representative features.

For the definitions below, let us assume that we have the following notations:

- $F = \{f_1, f_2, ..., f_n\}$ is the set of features. Every feature is a column vector.
- $C = \{c_1, c_2, ..., c_n\}$ is the set of the clusters centroids.
• $f_i$ be a feature vector (i.e. $f_i \subset F$) in a cluster with the centroid $c_i \in C$

• $\epsilon$ be a subset of features in a cluster

• $R_i$ be a representative feature vector so far (i.e. $R_i \subset F$) in a cluster with the centroid $c_i \in C$

Definitions

The definitions are given below:

• **Definition 1 (Dynamic multi-views):** the complete views, which are heterogeneous sets of features, do not exist in advance. The views arrive sequentially one by one and are incrementally processed as they arrive. It is inefficient to wait for all the views to be collected before starting the clustering process as they are not static.

• **Definition 2 (The Online learning):** the complete instances do not exist in advance. They arrive sequentially and are incrementally processed in chunks upon their arrival.

• **Definition 3 (Representative feature):** a feature assigned to a cluster is considered to be representative if it has the minimum distance to its cluster’s centroid amongst all other features assigned to the same cluster. Formally, a feature $f_i \subset \epsilon$ is a representative feature in a cluster with centroid $c_i$ if and only if:

$$\text{dist}(f_i, c_i) < \text{dist}(\epsilon, c_i)$$

Any feature $f_i$ that is not representative is simply considered as non-representative.
5.3.1 The Framework

Figure 5.2 broadly shows how the proposed method works. OUDVFS consists of two parts, namely clustering and feature selection. The chunk can have new instances or new views. If the chunk has new instances only, OUDVFS incrementally clusters the new instances with the clusters resulting from the previous chunk. Similarly, when the chunk has a new view (i.e. set of features), OUDVFS incrementally clusters the new view with the clusters resulting from the previous chunk. This method relies on clustering to select features as it does not require the data class labels in order to group the data. Specifically, the OUDVFS relies on hierarchical clustering in order to merge the clusters as detailed below. The selected set of representative features is updated at each clustering step.
The following steps are those taken by OUDVFS to select a set of representative features from dynamic views in online mode.

(A) Initialising phase

For the first chunk of data, OUDVFS initialises a cluster by assigning its first feature as a centroid and a member of that cluster. Then, OUDVFS clusters
all remaining features in the chunk as illustrated in the subsequent clustering phase. Every feature is strictly assigned to one cluster such that there is no overlapping of clusters (i.e. hard-clustering).

(B) **Clustering phase**

For each feature, OUDVFS either assigns a feature \( f_j \) to a cluster or creates a new cluster with \( f_j \) as a centroid and a member of it. To do so, a) OUDVFS finds the closest cluster to \( f_j \) by computing the Euclidean distance \( \sqrt{\sum_{i=1}^{n}(f_j-c_i)^2} \) between the feature \( f_j \) and each cluster centroid; b) OUDVFS decides the inclusion of \( f_j \) in the closest cluster by computing the new radius of the closest cluster including the feature \( f_j \). If the new radius does not exceed the predefined input threshold \( T \), OUDVFS confirms the inclusion of \( f_j \) in its closest cluster. Otherwise, OUDVFS creates a new cluster with \( f_j \) as a centroid and a member of it. The radius of a cluster is the sum of the squares of the distances of all its features to the cluster centroid. Formally, \( \text{cluster-radius} = \sum \text{dist}(\epsilon-c_i)^2 \). The clustering phase does not initially force the features to be grouped into limited \( k \) number of clusters. Instead, it optimises the features to be clustered naturally based on their distribution by allowing the number of clusters to increase. Then, the merging phase limits the clusters by merging them to the predefined reduction size \( k \) as provided in the next phase.

(C) **Merging phase**

As a result of the clustering phase, we might end up with too many clusters which exceed the required reduction size \( k \) number of clusters. Therefore, in this phase, OUDVFS hierarchically merges clusters until the number of clusters = \( k \). OUDVFS merges (lets say cluster 1 and cluster 2) if the distance
between the centroid of cluster 1 and the centroid of cluster 2 is the minimum compared to all other clusters centroids. Indeed, clusters with close distance centroids are more likely to share the same characteristics and therefore they are merged. Every time that we merge clusters, we compute a new centroid for the new cluster.

(D) **Whenever another chunk arrives**

The new chunk either has new values (i.e. new instances) for exiting clustered features or additionally has new features in the new view. In the first case, OUDVFS first finds the corresponding features in the $k$ clusters resulting from the merging phase. This is done based on indexing the features. Then, OUDVFS adds the new values to corresponding features in the $k$ clusters. For each $k$ cluster, OUDVFS calculates the new radius after updating the centroid. If the new radius exceeds the threshold $T$, the clustering and merging phases are repeated for only those clusters. By limiting the clustering and merging phases to the clusters whose radius is more than the parameter $T$, the computational complexity of the OUDVFS is reduced. For example, if $k=5$ and we have two clusters whose radius exceeds the predefined threshold $T$, then we repeat the clustering and merging phases for $k-3$ clusters so that the total number of clusters is five. Otherwise, we go to the finding representative feature phase. In the second case, OUDVFS performs the same process as the first case in addition to doing the clustering and merging phases with the new features of the new view.

(E) **Finding representative features**

OUDVFS selects a single feature of each cluster as a representative feature. The selected features are those which have minimum distances to their clus-
ters’ centroids. The selected features from each cluster comprise the set of representative features.

The OUDVFS algorithm is given below. It can work in three scenarios based on the chunk structure: 1) streaming features where every new chunk only comes with new features and the number of instances is fixed as in lines 2 - 7; 2) online learning where every new chunk arrives with new instances while the number of features is fixed as in lines 9 - 14; 3) dynamic views with online learning where every new chunk comes with both new views or set of features and new instances as in lines 9 - 16. This makes the algorithm widely applicable for different application requirements. In this chapter, we experimentally investigate the dynamic views with online learning as it covers all the former scenarios and flexible for real applications.
such as health care.

Algorithm 3: OUDVFS

1. **Input:** $X^{(v)}$, data matrices from different views
2. **Input:** $T$, the radius threshold
3. **Input:** $K$, the number of desired clusters
4. **Output:** $R$, the set of representative features

1. Initialise the chunks following the required scenario (streaming features, online learning or online learning with dynamic views);
2. Initialise a cluster with a single feature $f_i \subset chunk_1$ as a centroid and a member;

// for all remaining features in $chunk_1$
3. for $f=1:n$
do
4. $[clusters] =$clustering(features);
5. $[desired\ clusters] =$merging(clusters);
6. end

7. resulted_clusters=desired_clusters;
8. for $chunk=2:n$
do

// for new instances
9. idx=find(features(resulted_clusters));
10. Add the new values to the corresponding features(resulted_clusters);
11. updated_clusters=(update radius(resulted_clusters), update centroid(resulted_clusters));
12. clusters exceed $T=$(find(updated_clusters)== radius $T));
13. $[clusters] =$clustering(clusters exceed $T)$;

// merge clusters such that clusters exceed $T + updated \ clusters = k$
14. $[desired\ clusters] =$merging(clusters);

// for new view
15. $[clusters] =$clustering(new_view,desired_clusters);
16. $[desired\ clusters] =$merging(clusters);

// select a representative feature of each cluster
17. for $desired\ clusters=1:n$
do
18. for $j=1:n$
do
19. selected_feature=min(distance(feature(j), centroid(desired_cluster)));
20. end
21. set_of_selected_features(1,desired_clusters)=(selected_feature);
22. end
23. $R=\ set\ of\ selected\ features ;$
24. resulted_clusters=desired_clusters;
25. end
26. Return $R$;
5.4 Experimental Evaluation

This section describes the experimental setup of the proposed OUDVFS and the benchmark methods for real datasets. The experiments were conducted to investigate the following:

- How accurate is OUDVFS in selecting a reduced set of representative features incrementally such that the views increase over time in online manner.
- How fast is OUDVFS in selecting a set of representative features in such an environment.

To simulate the online learning and the dynamic views, the datasets are structured as shown in Figure 5.3 where every new chunk has new instances and a view. The number of the chunks is set to be equal to the number of the views, and the chunk size is the number of data instances divided by the number of the views. The chunks are structured this way as it matches the settings of real applications where both views and instances can increase over time. For a hospital, new patients can do existing medical tests of different groups. Also, existing patients may undergo new group of medical tests. (see Section 5.1 for detailed example of the health care application).
Figure 5.3: Illustration of dynamic views proposed structure where instances are rows and features are columns. Each new chunk consists of new instances and an additional view.

In order to properly answer the above two questions, OUDVFS is benchmarked against three well-known unsupervised feature selection methods, namely OMVFS [40], UFSSF [158], and SPEC [17]. To the best of our knowledge, there is no unsupervised feature selection method developed for dynamic views applications. Therefore, these three methods have been selected as they comprise a combination of streaming features, batch, online and multiple views applications. Summary of these methods are provided in Table 5.2. SPEC and UFSSF cannot be applied directly to dynamic views as SPEC only designed for batch datasets and UFSSF for streaming features. Therefore, the views of each dataset were concatenated in a single matrix. Then, these two methods were applied on that data matrix.
Table 5.2: Summary of the proposed and benchmarked methods

<table>
<thead>
<tr>
<th>Method/category</th>
<th>Multi-view</th>
<th>Dynamic views</th>
<th>Online</th>
<th>Streaming features</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEC</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>UFSSF</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OMVFS</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OUDVFS (proposed)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For OMVFS, we followed the settings given in OMVFS paper [40] and we set the chunk size of each view to be equal to the chunk size used in OUDVFS for a fair comparison. An exception is the dynamic views as OMVFS works with multiple but not dynamic views. $\alpha_v$ and $\beta_v$ are all set equally for all different views. We conducted a grid search in $\{10^{-2}, 10^{-1}, 10^1, 10^2\}$ and selected the one with better results. $\gamma$ was set to $10^7$. We used the code given in \(^1\) as directed in the OMVFS paper.

For OUDVFS, we reported the selected features of the last chunk for a fair comparison with the benchmark methods. The radius was set such that we can get $k$ number of clusters or selected features. $K$ is set to the required number of representative features to be selected. For all the evaluated methods, the number of selected features were varied to ensure the reliability of the results. The selected representative features by OUDVFS and the other benchmark methods were evaluated using two well-known classifiers: Naive Bayes [139] and Lazy Nearest Neighbor [141] (also called IB1). In addition to the classifiers, $k$-fold-cross validation was applied on all selected features to produce accurate results by avoiding the problem of over fitting data. The selected features set was first divided into subsets of equal size depending on the selected $k$ folds. Then, one subset was retained as the testing subset while the rest were used as training subsets. Finally, the average value of all folds was set to be the average result. In the evaluation, $k$ was set

\(^1\)https://github.com/software-shao/Online-Unsupervised-Multiview-Feature-Selection/blob/master/OMVFS.m
to ten (10) as suggested in [145]. All the four algorithms were implemented in the Matlab programming language. They are executed using the Mac operating system macOS High Sierra with 2.9 GHz Intel Core i7 and 16 GB RAM.

5.4.1 The Datasets

Three datasets with various dimensionality were used to evaluate the proposed OUDVFS and the benchmark methods. We selected these datasets because they are commonly used for multi-view learning. The three adopted datasets are collected mainly for the purpose of classification and clustering as clustering is a part of the proposed approach to selecting the representative features. Data are randomly shuffled to avoid order-dependency between the instances and the features in order to accurately evaluate the OUDVFS and the benchmark methods. A brief description of each dataset is given below:

- **Fox News**: This is a news articles dataset. Each article is represented in two views, namely text view and image view. Text or words in titles, abstracts, and body comprise the text view data. The images associated with each article comprise the image view data. The image view has 996 features and the text view has 27072. The total number of the instances is 1523 and the data fall into four classes. [https://sites.google.com/site/qianmingjie/home/datasets/cnn-and-fox-news](https://sites.google.com/site/qianmingjie/home/datasets/cnn-and-fox-news)

- **Caltech-7**: This is an image dataset where pictures of objects belong to seven classes. It has six views/group of features, namely Gabor (48 features), Wavelet moments (40 features), CENTRIST (254 features), Histogram of oriented gradients (1984 features), GIST (512 features) and Local binary
patterns (928). In total, there are 1474 instances. https://github.com/yeqinglee/mvdata

- **Handwritten/Multiple features**: This dataset consists of features of written numerals (‘0’ to ‘9’). This numerals were written manually by hands. There are six groups/views of features, namely Fourier coefficients of the character shapes (76 features), pixel averages (240 features), Profile correlations (216 features), Zernike moment (46 features), Karhunen-Loeve coefficients (64 features) and Morphological (6 features). The total number of instances is 2000. https://archive.ics.uci.edu/ml/datasets/Multiple+Features

### 5.4.2 Evaluation Metrics

The aim of the proposed OUDVFS is to select a set of representative features from dynamic views in online mode. The selected features should improve the accuracy of the machine learning algorithms when performing their tasks. We selected classifiers as the datasets had class labels. The classifiers were trained with the class labels. Then the test data, which did not have class labels, was tested by the trained classifiers. The process of selecting the features from the dynamic views should be done within an acceptable running time. Therefore, the metrics used for the evaluation are grouped into classification accuracy metrics and running time.

**A) Classification accuracy metrics**

Three metrics are adopted to determine the classification accuracy: recall, precision and F-measure and they are computed as below:

\[
Recall = \frac{\text{TruePositive}(TP)}{\text{TruePositive}(TP) + \text{FalseNegative}(FN)}
\]  

(5.3)
Results and Analysis

\[ Precision = \frac{TruePositive(\text{TP})}{TruePositive(\text{TP}) + FalsePositive(\text{FP})} \]  

\[ F - measure = \frac{Recall \times Precision}{Recall + Precision} \]

These metrics determine whether the selected features competitively enhance the classification accuracy of the classifiers. These particular metrics were adopted as they are widely used for measuring the classification accuracy.

(B) Running time

The major motivation for OUDVFS is to remove those non-representative features from dynamic views applications for better data representation. Therefore, this enhances the classification accuracy of the classifiers when using this set of representative features as input for the classifiers. However, this target should be achieved within an acceptable running time. To do so, the time taken by OUDVFS, UFSSF, SPEC and OMVFS to select representative features were computed in seconds.

5.5 Results and Analysis

This section introduces and discusses the results of the conducted experiments. We first present the results relating to the classification accuracy of the selected features. The classification accuracy was measured based on three evaluation metrics: recall, precision and F-measure. Two classifiers were used, namely Naive Bayes and IB1. Then, we present the results relating to the running time. The time taken by each benchmark method to select features is reported in seconds.
The experimental results regarding the classification accuracy of OUDVFS and the benchmark methods on the three datasets are presented in Tables 5.3 - 5.20. The following observations can be made from the results:

- With all methods, the accuracy (recall, precision and f-measure) increases as the number of the selected features increases, thus indicating that all methods selected good representative features.

- OUDVFS significantly outperformed the traditional/single view feature selection methods as it exploited the correlation among the views incrementally. Therefore, it selected better representative features.

- OUDVFS was also compared with a multi-view feature selection method, namely OMVFS. However, OUDVFS consistently has the highest recall, precision and f-measure with all different numbers of selected features. This is because OUDVFS incrementally updates the means of the clusters to evaluate the representativeness of the selected features as these might lose their representativeness over time due to the arrival of new features.

- For example, Tables 5.3 to 5.8, show the classification accuracy of OUDVFS and the benchmark methods on Handwritten dataset. For both Naive Bayes and IB1 classifiers, OUDVFS has around 5-7% better accuracy compared to the traditional/single view methods. Also, it has around 50% better accuracy compared to OMVFS regardless of whether the classification metric is recall, precision or f-measure. Tables 5.15 - 5.17 show the classification accuracy of OUDVFS on the Fox news dataset with the Naive Bayes classifier. OUDVFS has around 15-20% better accuracy when compared with traditional/single view methods. Also, it has around 40-60% better accuracy compared to OMVFS.
• The method with second best accuracy is UFSSF. This is valid for almost all datasets with all evaluation classifiers. Indeed, although UFSSF has not been developed for multi-view data, it incrementally clusters the features in a streaming features manner.

• For the Fox News dataset, all the methods have an overall reduction in accuracy when they are evaluated using the IB1 classifier. This might be due to the Fox News data being sparse, since the IB1 classifier is efficient when classifying dense data.

OUDVFS has better accuracy compared with the benchmark methods. This is due to the fact that OUDVFS has an efficient clustering methodology. In fact, when partitioning the clusters, OUDVFS allows the features to be clustered naturally based on their similarities and later on it merges the clusters to required clusters. This clustering methodology ensures that features are well partitioned or grouped and good clustering will definitely result in a good representation of the features. Also, OUDVFS limited the representative feature of each cluster as the one with minimum distance to its cluster’s centroid. This step ensures that the selected feature of each cluster represents well the other features in that cluster. Finally, because views are dynamic, the set of features is updated at each clustering step as some features might be representative for only a specific time.

Although in a few cases UFSSF achieved better accuracy than the proposed OUDVFS, the latter addressed the problem of increasing instance (i.e. online learning) and increasing views (dynamic views). UFSSF only addressed the problem of dynamic features (i.e. increasing features) and therefore we may compromise the tiny accuracy difference in few cases while addressing the problem of both increasing features and increasing instances. Also, when OUDVFS is compared with the
online multi-view method (OMVFS), it significantly has better recall, precision and f-measure for all the used datasets.

Table 5.3: Classification accuracy (Recall) using Handwritten dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method \ Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td>0.861</td>
<td>0.8855</td>
<td>0.965</td>
<td>0.9605</td>
</tr>
<tr>
<td>OMVFS</td>
<td>0.28829</td>
<td>0.2993</td>
<td>0.33433</td>
<td>0.34935</td>
</tr>
<tr>
<td>SPEC</td>
<td>0.804</td>
<td>0.8655</td>
<td>0.8905</td>
<td>0.9065</td>
</tr>
<tr>
<td>UFSSF</td>
<td>0.81</td>
<td>0.867</td>
<td>0.9124</td>
<td>0.9225</td>
</tr>
</tbody>
</table>

Table 5.4: Classification accuracy (Precision) using Handwritten dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method \ Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td>0.86242</td>
<td>0.88653</td>
<td>0.95596</td>
<td>0.9716</td>
</tr>
<tr>
<td>OMVFS</td>
<td>0.28419</td>
<td>0.32927</td>
<td>0.34222</td>
<td>0.34863</td>
</tr>
<tr>
<td>SPEC</td>
<td>0.8036</td>
<td>0.8692</td>
<td>0.89251</td>
<td>0.90779</td>
</tr>
<tr>
<td>UFSSF</td>
<td>0.82301</td>
<td>0.86965</td>
<td>0.91373</td>
<td>0.94537</td>
</tr>
</tbody>
</table>

Table 5.5: Classification accuracy (F-measure) using Handwritten dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method \ Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td>0.86099</td>
<td>0.88562</td>
<td>0.95519</td>
<td>0.96074</td>
</tr>
<tr>
<td>OMVFS</td>
<td>0.26864</td>
<td>0.27003</td>
<td>0.30736</td>
<td>0.34288</td>
</tr>
<tr>
<td>SPEC</td>
<td>0.80297</td>
<td>0.86674</td>
<td>0.89066</td>
<td>0.90647</td>
</tr>
<tr>
<td>UFSSF</td>
<td>0.82148</td>
<td>0.86642</td>
<td>0.91</td>
<td>0.92279</td>
</tr>
</tbody>
</table>

Table 5.6: Classification accuracy (Recall) using Handwritten dataset with IB1

<table>
<thead>
<tr>
<th>Method \ Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td>0.926</td>
<td>0.936</td>
<td>0.976</td>
<td>0.979</td>
</tr>
<tr>
<td>OMVFS</td>
<td>0.11562</td>
<td>0.11812</td>
<td>0.12713</td>
<td>0.14715</td>
</tr>
<tr>
<td>SPEC</td>
<td>0.8545</td>
<td>0.864</td>
<td>0.909</td>
<td>0.895</td>
</tr>
<tr>
<td>UFSSF</td>
<td>0.8565</td>
<td>0.8835</td>
<td>0.9155</td>
<td>0.9375</td>
</tr>
</tbody>
</table>
### Results and Analysis

#### Table 5.7: Classification accuracy (Precision) using Handwritten dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td><strong>0.92854</strong></td>
<td><strong>0.93918</strong></td>
<td><strong>0.9761</strong></td>
<td><strong>0.97903</strong></td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.11934</td>
<td>0.12271</td>
<td>0.13369</td>
<td>0.15232</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.85691</td>
<td>0.85556</td>
<td>0.90952</td>
<td>0.89644</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.85816</td>
<td>0.87477</td>
<td>0.91577</td>
<td>0.93763</td>
</tr>
</tbody>
</table>

#### Table 5.8: Classification accuracy (F-measure) using Handwritten dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td><strong>0.92551</strong></td>
<td><strong>0.93581</strong></td>
<td><strong>0.97598</strong></td>
<td><strong>0.97899</strong></td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.11626</td>
<td>0.11736</td>
<td>0.12829</td>
<td>0.14823</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.85322</td>
<td>0.85423</td>
<td>0.90888</td>
<td>0.89471</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.85674</td>
<td>0.86323</td>
<td>0.912</td>
<td>0.93742</td>
</tr>
</tbody>
</table>

#### Table 5.9: Classification accuracy (Recall) using Caltech dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td><strong>0.81479</strong></td>
<td><strong>0.87517</strong></td>
<td><strong>0.87924</strong></td>
<td><strong>0.90502</strong></td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.038776</td>
<td>0.040136</td>
<td>0.043537</td>
<td>0.48367</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.72049</td>
<td>0.7863</td>
<td>0.81072</td>
<td>0.827</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.76323</td>
<td>0.83039</td>
<td>0.85346</td>
<td>0.87313</td>
</tr>
</tbody>
</table>

#### Table 5.10: Classification accuracy (Precision) using Caltech dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td><strong>0.84761</strong></td>
<td><strong>0.90035</strong></td>
<td><strong>0.90571</strong></td>
<td><strong>0.91499</strong></td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.38052</td>
<td>0.38305</td>
<td>0.38499</td>
<td>0.41796</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.72074</td>
<td>0.80979</td>
<td>0.83467</td>
<td>0.85094</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.77469</td>
<td>0.85361</td>
<td>0.87983</td>
<td>0.89843</td>
</tr>
</tbody>
</table>

#### Table 5.11: Classification accuracy (F-measure) using Caltech dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td><strong>0.82357</strong></td>
<td><strong>0.88197</strong></td>
<td><strong>0.88671</strong></td>
<td><strong>0.90746</strong></td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.022646</td>
<td>0.029528</td>
<td>0.043047</td>
<td>0.40547</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.71728</td>
<td>0.79533</td>
<td>0.82074</td>
<td>0.8358</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.76541</td>
<td>0.83694</td>
<td>0.86236</td>
<td>0.88101</td>
</tr>
</tbody>
</table>
### Results and Analysis

Table 5.12: Classification accuracy (Recall) using Caltech dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.89077</td>
<td>0.92741</td>
<td>0.93487</td>
<td>0.93894</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.37075</td>
<td>0.37959</td>
<td>0.38163</td>
<td>0.38707</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.73881</td>
<td>0.77001</td>
<td>0.77883</td>
<td>0.78019</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.87856</td>
<td>0.8806</td>
<td>0.91248</td>
<td>0.91995</td>
</tr>
</tbody>
</table>

Table 5.13: Classification accuracy (Precision) using Caltech dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.88025</td>
<td>0.92642</td>
<td>0.93104</td>
<td>0.93651</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.37437</td>
<td>0.37728</td>
<td>0.39254</td>
<td>0.39263</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.73624</td>
<td>0.79069</td>
<td>0.8152</td>
<td>0.81981</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.87172</td>
<td>0.87335</td>
<td>0.90624</td>
<td>0.91237</td>
</tr>
</tbody>
</table>

Table 5.14: Classification accuracy (f-measure) using Caltech dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.8825</td>
<td>0.92338</td>
<td>0.93056</td>
<td>0.93516</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.37367</td>
<td>0.37739</td>
<td>0.38563</td>
<td>0.38972</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.73407</td>
<td>0.7714</td>
<td>0.78109</td>
<td>0.78226</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.86839</td>
<td>0.86953</td>
<td>0.90598</td>
<td>0.91234</td>
</tr>
</tbody>
</table>

Table 5.15: Classification accuracy (Recall) using Fox News dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.56139</td>
<td>0.68746</td>
<td>0.7459</td>
<td>0.76362</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.35441</td>
<td>0.16996</td>
<td>0.14954</td>
<td>0.15679</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.3786</td>
<td>0.3991</td>
<td>0.42</td>
<td>0.4421</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.38936</td>
<td>0.43336</td>
<td>0.51018</td>
<td>0.53972</td>
</tr>
</tbody>
</table>

Table 5.16: Classification accuracy (Precision) using Fox News dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.64122</td>
<td>0.72122</td>
<td>0.76302</td>
<td>0.77728</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.3044</td>
<td>0.25421</td>
<td>0.35887</td>
<td>0.32373</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.4281</td>
<td>0.462</td>
<td>0.4725</td>
<td>0.5143</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.51797</td>
<td>0.59833</td>
<td>0.6278</td>
<td>0.62402</td>
</tr>
</tbody>
</table>
Table 5.17: Classification accuracy (F-measure) using Fox News dataset with Naive Bayes

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.56283</td>
<td>0.69398</td>
<td>0.75042</td>
<td>0.7684</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.31582</td>
<td>0.16321</td>
<td>0.12214</td>
<td>0.12289</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.4179</td>
<td>0.4505</td>
<td>0.4688</td>
<td>0.5142</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.5165</td>
<td>0.5921</td>
<td>0.6176</td>
<td>0.61346</td>
</tr>
</tbody>
</table>

Table 5.18: Classification accuracy (Recall) using Fox News dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.58634</td>
<td>0.65069</td>
<td>0.66185</td>
<td>0.6763</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.28656</td>
<td>0.30237</td>
<td>0.30171</td>
<td>0.30896</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.50287</td>
<td>0.5734</td>
<td>0.5992</td>
<td>0.6122</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.50821</td>
<td>0.58634</td>
<td>0.6021</td>
<td>0.62508</td>
</tr>
</tbody>
</table>

Table 5.19: Classification accuracy (Precision) using Fox News dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.59753</td>
<td>0.66436</td>
<td>0.6769</td>
<td>0.68838</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.28843</td>
<td>0.29636</td>
<td>0.30725</td>
<td>0.31058</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.5011</td>
<td>0.581</td>
<td>0.5841</td>
<td>0.5995</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.5075</td>
<td>0.58463</td>
<td>0.61733</td>
<td>0.62744</td>
</tr>
</tbody>
</table>

Table 5.20: Classification accuracy (F-measure) using Fox News dataset with IB1

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of selected features</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUDVFS (proposed)</td>
<td></td>
<td>0.59106</td>
<td>0.6545</td>
<td>0.66566</td>
<td>0.67571</td>
</tr>
<tr>
<td>OMVFS</td>
<td></td>
<td>0.28724</td>
<td>0.29882</td>
<td>0.30435</td>
<td>0.30966</td>
</tr>
<tr>
<td>SPEC</td>
<td></td>
<td>0.5001</td>
<td>0.5793</td>
<td>0.5838</td>
<td>0.5991</td>
</tr>
<tr>
<td>UFSSF</td>
<td></td>
<td>0.50759</td>
<td>0.58454</td>
<td>0.60561</td>
<td>0.6248</td>
</tr>
</tbody>
</table>

Figures 5.4 to 5.6 show the results of the running time of OUDVFS and the benchmark methods. The running time of all methods increases as the data is sparse. For example, all methods applied to the Fox News dataset had higher running time than when they were applied to Caltech-7 and Handwritten datasets. Also, the running time of all the methods except for SPEC increased as the number
of the selected features increased. The reason is that SPEC returns a weighted vector of all features. Therefore for each dataset, it has a fixed running time for all different numbers of selected features. For the Handwritten dataset as shown in Figure 5.4, OUDVFS has the best running time compared with OMVFS, UFSSF and SPEC. OUDVFS selected different number of features with excellent running time. It selected 25, 50 and 75 features in approximately 0.023 seconds. Also, OUDVFS took 0.875 seconds to select 100 features. This is because of two factors: 1) when a new chunk arrives, OUDVFS re-partitions only those clusters whose features exceeds the predefined threshold T, not all the clusters; 2) in the merging step, distance is computed between centroids so as to merge closest clusters. This is unlike other clustering methods that merge clusters based on the distance of their features. SPEC took the highest running time compared to the benchmarked methods. It returned a weight vector of features in approximately 19.56 seconds. OMVFS and UFSSF were the second and third fastest methods respectively. They were very competitive when 25 features were selected. However, they consistently had a difference of around 0.52% for all other numbers of selected features.

However, OUDVFS is not as fast as the benchmark methods with Caltech and Fox News datasets as shown in Figures 5.5 and 5.6, respectively. This is due to two factors: 1) the OUDVFS algorithm has many for-loops function so to incrementally cluster the instances following clustering the views. Actually, it is well-known that the Matlab platform works very slowly with loops, which is why a cloud platform was developed for the loop function; 2) If a new chunk arrives, OUDVFS assigns the new values of existing features into their corresponding clusters. Then OUDVFS re-cluster current clusters and merge them if they exceeds a pre-set radius threshold. However, for sparse datasets such as those of Caltech-7 and Fox News, we need to set a small radius value in order to obtain the required number of selected features.
However, the negative side is that the smaller the radius, the more clusters need to be clustered and sequentially merged which in turn requires more processing time. SPEC is the second highest method in terms of running time for both datasets. it needed 32.68 and 36.04 seconds for Caltech-7 and Fox datasets respectively. OMVFS is the fastest in terms of running time for both datasets. This is because of its reliance on matrix factorisation as it can process sparse data efficiently. UFSSF is the second method with lowest running time for both datasets.

Figure 5.4: Time complexity of different methods on Handwritten dataset
Results and Analysis

Overall, OUDVFS is not the best in terms of running time and further investigation will be done to measure the running time of OUDVFS using a different
programming language. Although OUDVFS did not achieve the best running time, it had the best classification accuracy. This is considering the challenge of solving unsupervised feature selection for dynamic multi-view and online learning.

5.6 Conclusion

In this chapter, an online unsupervised feature selection algorithm is developed for dynamic views. In real applications, new views as well as new instances can be added over time. OUDVFS is different from existing single-view feature selection methods as it exploits incrementally any similarity of all the views presented so far. This results in better representation of selected features. Also, it is different from existing multi-view feature selection methods as it addresses the problem of dynamic views and online learning. OUDVFS, proposed a clustering based algorithm so to cluster the views and the instances sequentially. Then from each cluster, it selects the feature with minimum distance to its cluster centroid as representative feature. The set of selected features is updated at each clustering step. OUDVFS is benched against well-known single-view and multi-view feature selection methods on three multi-view datasets. OUDVFS has better accuracy when tested with several evaluation metrics, namely recall, precision and F-measure. However, OUDVFS is not the best in terms of running time and this due to the loop function complexity of clustering the views and the instances incrementally. Further investigation will be undertaken to consider implementing the code with a programming language other than the Matlab which is very slow with for loops.
Chapter 6

Conclusion

This chapter presents an overview of the research problem and the questions formulated in this thesis. The contributions made by this work are summarised along with the core findings. Moreover, we suggest future research directions pursued subsequent to the work presented in this thesis.

Machine learning has been used for various applications such as Internet of Things (IOT), early cancer prediction and classifying network data for intrusion detection systems. However, due to the proliferation of such data, the dimensions of data have increased significantly, resulting in what is known as high-dimensional data. This increase in data dimensions results in redundant and non-representative features, which raise the following issues for machine learning algorithms. Firstly, they add extra processing time to the machine learning algorithms and therefore negatively affect on their performance/running time. Secondly, they reduce the accuracy of the machine learning algorithms by overfitting the data with these redundant and non-representative features. Lastly, they result in more storage utilisation which is a waste of energy and money.

The research presented in this thesis has, therefore, focused on the development
feature selection methods in order to reduce the dimensions of data for machine learning algorithms. They reduce the dimensions of the data by selecting a reduced set of representative and non-redundant features that approximate the original feature space. Therefore, the selected set of features can help machine learning algorithms to improve their classification accuracy, running time and storage utilisation efficiently. Three distinct feature selection methods have been developed and adapted to select features from three data perspectives, namely homogeneous data, streaming features and multi-view data. Each of these data perspectives has its own challenges and characteristics when a reduced set of representative and non-redundant features is being selected from it. Therefore, these challenges have been addressed while considering the characteristics of each of these data perspectives.

In the first research question, we accurately selected a reduced set of representative and non-redundant features from homogeneous data, which experienced high-dimensionality. The intuition is that an accurate selection of representative features results in better learning (i.e. classification accuracy). To address the second research question, a feature selection method was developed to incrementally selecting features in streaming features data perspective. For the third research question, we addressed feature selection from the perspective of heterogeneous data (specifically multi-view data). Unlike existing problems, we investigated the selection of features while new views can be added, which is more realistic. Also, the selection is done in online mode. Specifically, the research questions addressed in this thesis are:

(A) How to design an efficient and accurate feature selection for high-dimensional data without the need for data class labels?

(B) How to design an efficient feature selection for streaming features applications
without the need for data class labels?

(C) How to design an online feature selection for multi-view data such that the views are dynamic and data class labels are not required?

Below we outline the innovations and advantages of the work conducted in this thesis to well address the research questions. We have not used any search strategy to evaluate different subset of features for all these three proposed features selections methods, which made them efficient in terms of run-time. Moreover, none of these methods requires data class labels in order to select features and therefore they suit real-world applications.

1. AUFS - Towards An Efficient and Accurate Unsupervised Feature Selection

We started by tackling the problem of high-dimensional data as it reduces the accuracy and the time complexity of machine learning algorithms. However, in high-dimensional data, it is a challenge to accurately selecting representative features so that they approximate the original feature space. Therefore an accurate and unsupervised feature selection method is proposed for high-dimensional data (called AUFS). The proposed method is novel as it does not require data class labels in order to select the representative features (i.e. unsupervised). Also, AUFS is accurate in selecting representative features, thereby resulting in better classification accuracy. Finally, it does not require any search for different subset of features, which make it efficient in terms of run-time. A $k$-mean clustering algorithm was extended to work with more similarity measures for better clustering of the features. Also, a centroid-based methodology was proposed to identify and select the representative features from the clusters. AUFS was tested on real datasets with two well-
known unsupervised feature selection methods. Three evaluation classifiers were used with different similarity measures. AUFS accurately selected representative features, which was reflected by the classification accuracy of the classifiers. Also, AUFS outperformed the other two benchmark methods in terms of the running time.

2. **UFSSF - An Efficient Unsupervised Feature Selection for Streaming Features**

For the second contribution, we investigated the issue of designing feature selection method for streaming features applications. Streaming features applications have specific characteristics than static data. The number of features can increase and is not fixed compared to those of homogeneous data. The full feature space is not known in advance and features arrive and are processed one by one in real-time. Additionally, the selection of representative features should be done within an acceptable running time as streaming features require fast and real-time processing. Moreover, in streaming features applications, algorithms should read the data only once due to the finite amount of storage space, and then non-representative and redundant features should be removed to allow storage. Feature selection methods for streaming features should dynamically update their selected representative features when new features arrive. Due to the streaming nature, most data is not labeled [32]. Therefore, a feature selection method has been developed that takes these characteristics into consideration. We proposed an unsupervised feature selection method for streaming features applications (called UFSSF). A $k$-mean is adapted to work in streaming features applications. It incrementally clusters the arriving features in real-time. It dynamically updates
the selected set of representative features. We experimentally simulated the streaming features environment where the number of features increases while the number of instances is fixed. UFSSF was tested on real datasets with three similarity measures. The average of three evaluation classifiers was reported and UFSSF achieved the best accuracy and most efficient running time compared with the benchmark methods.

3. **OUDVFS: Online Unsupervised Feature Selection for Dynamic Views**

For the third contribution, we investigated the issue of designing an online feature selection for dynamic multi-view data. Although multi-view data provides complementary information for machine learning algorithms, it results in high-dimensional data. All existing multi-view feature selection methods assume that the number of the views is fixed/static. However, this assumption is not valid because in real-world applications the same set of instances can be represented by new additional views at any given time. Also, the instance can increase at any time. The challenge here is to select features incrementally since both features and instances can increase over time. In addition, the selection of the features should be done without the need for data class labels. The proposed OUDVFS method does not require data class labels in order to select representative features from dynamic views. Unlike existing feature selection methods for multi-view data, OUDVFS is more appropriate for real applications as it selects features where the views can increase over time (i.e. dynamic views) as do the instances (i.e. online). Technically, we proposed an incremental clustering method where the selected set of features are clustered whenever a new view or a new chunk of instances arrives. OUDVFS was tested on three real multi-view datasets and we experi-
mentally simulated an increasing number of views and instances. Experimental results demonstrated that when selecting a set of features, OUDVFS has the best classification accuracy compared with well-known single-view and multi-view unsupervised feature selection methods. Also, OUDVFS achieved the most efficient running time.

6.1 Future Work

Although this thesis has proposed a set of innovative feature selection methods to reduce the data dimensions from different data perspectives, there are further improvements which can be carried out to optimise them.

- In Chapter 3, AUFS showed that it can reduce the dimensions of data by accurately selecting a limited set of representative and non-redundant features. However, this method does not address the data streaming aspects. In recent applications, data arrives from sensors and therefore feature selection methods should consider the streaming characteristics such as real-time applications. Therefore, AUFS can be improved to work from multiple streams. Actually, AUFS can be extended to work in multiple streams with two versions, namely, centralised version and distributed version. In the centralised version, AUFS waits for chunks of data from every stream, and later clusters them to find the representative features. Conversely, in the distributed version, AUFS clusters every stream individually and then aggregates its selected representative features.

- In Chapter 4, UFSSF select features in streaming features applications. It showed an efficient selection of representative features and therefore achieved high classification accuracy when used by the classifiers. Also, it had an
efficient running time when compared with well-known unsupervised feature selection methods. However, UFSSF cannot select features from multiple streaming features or correlate them to filter out any redundant features. Therefore, we are motivated to improve this work for multiple data streams. This can be developed by applying UFSSF to each stream individually and then having a global feature selection method for all the selected features from all the streams. This is to ensure that there are no selected redundant features intra and inter the multi-streaming features.

- In Chapter 5, OUDVFS selects features from dynamic views in online mode. This method had to deal with the challenging problem that both the number of features and the instances can increase over time. Although OUDVFS has selected a set of features that achieves high accuracy, it does not remove the non-representative and redundant features from the clusters, which would result in high storage utilisation due to the non-limited nature of data streams. Therefore, OUDVFS can be improved by statistically summarising the features of the clusters so that we can remove those non-representative and redundant features for efficient storage utilisation. By summarising the clusters with main statistical information (e.g. number of features, standard deviation and the mean), we can select representative features based on this statistical information and not the features themselves. As a result, we can remove the old and non-representative features from the clusters to allow more storage.
Bibliography


[54] Roberto Ruiz, José C Riquelme, Jesús S Aguilar-Ruiz, and Miguel García-Torres. Fast feature selection aimed at high-dimensional data via hybrid-


[73] Jiawei Han, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.


[102] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.


[120] Naif Y Almusallam, Zahir Tari, Peter Bertok, and Albert Y Zomaya. Dimensionality reduction for intrusion detection systems in multi-data streams—a


