Improving Classification and Attribute Clustering: An Iterative Semi-supervised Approach

by

Farid Seifi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
for the Doctorate in Philosophy degree in
Computer Science*

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

* The Doctoral Program in Computer Science is a joint program with Carleton University, administered by the Ottawa Carleton Institute for Computer Science

Abstract

This thesis proposes a novel approach to attribute clustering. It exploits the strength of semi-supervised learning to improve the quality of attribute clustering particularly when labeled data is limited. The significance of this work derives in part from the broad, and increasingly important, usage of attribute clustering to address outstanding problems within the machine learning community. This form of clustering has also been shown to have strong practical applications, being usable in heavyweight industrial applications.

Although researchers have focused on supervised and unsupervised attribute clustering in recent years, semi-supervised attribute clustering has not received substantial attention. In this research, we propose an innovative two step iterative semi-supervised attribute clustering framework. This new framework, in each iteration, uses the result of attribute clustering to improve a classifier. It then uses the classifier to augment the training data used by attribute clustering in next iteration. This iterative framework outputs an improved classifier and attribute clustering at the same time. It gives more accurate clusters of attributes which better fit the real relations between attributes.

In this study we proposed two new usages for attribute clustering to improve classification: solving the automatic view definition problem for multi-view learning and improving missing attribute-value handling at induction and prediction time. The application of these two new usages of attribute clustering in our proposed semi-supervised attribute clustering is evaluated using real world data sets from different domains.

Acknowledgements

I would like to express my special appreciation and thanks to my advisors, Professors Chris Drummond, Nathalie Japkowicz and Stan Matwin. I could never have done any of this, particularly the research and writing that went into this dissertation, without your continuous support and encouragement. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. You have been my friend, my mentor, my confidant, my colleague, and a never-ending fount of moral support.

I would especially like to thank my amazing family for the love, support, and constant encouragement I have gotten over the years. In particular, I would like to thank my wife, Sahar, who has been supporting me with love through this amazing journey, my parents whom I undoubtedly could not have been here without and my brothers who have encouraged me since day one.

I would also like to thank my fellow graduate students, research technicians and collaborators in the Text Analysis and Machine Learning Group (TAMALE) at University of Ottawa; your discussion, ideas, and feedback have been absolutely invaluable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis proposes a novel iterative approach to attribute clustering and semi-supervised learning. This iterative approach exploits the strength of semi-supervised classification to gradually improve the quality of attribute clustering particularly when labeled data is limited. It also applies the result of attribute clustering to improve the classification results iteratively. The significance of this work derives in part from the broad, and increasingly important, usage of attribute clustering to address outstanding problems within the machine learning community. For example, attribute clustering is commonly used with Big Data to extract relations between attributes, the key element for extracting information (Kim et al., 2013). This form of clustering has also been shown to have strong practical applications, being usable in heavyweight industrial applications; an example of which is provided in section 1.2.

Many knowledge representation methods have been proposed and used so far. Among the most popular in machine learning and data mining communities are semantic networks (Simmons, 1963; Quillian, 1963), frames (Winograd, 1975; Minsky, 1974), propositional logic (Mendelson, 1964) and attribute-value systems (Pawlak, 1991; Orłowska and Pawlak, 1984). All these knowledge representation systems describe objects and their properties. This is most clearly seen in an attribute-value system. This, perhaps, leads to it being the most popular knowledge representation model used in machine learning and knowledge discovery problems (Ziarko and Shan, 1996). As the Encyclopedia of Machine Learning describes it (Sammut and Webb, 2011): attributes are defined as different properties of things, in other words, ways that humans might describe things; a value is what we might assign to an attribute for a given thing that we try to describe in an instance. The data which we use in machine learning tasks is a set of such instances.

Label or class is an attribute which represents a common characteristic between some of the instances. It is more often the one we want to predict its value for new instances. If the label or class is included in the data as one of the attributes we call the data labeled and the learning task supervised.

Although an attribute-value system is the most common knowledge representation used in machine learning, there are many problems with this form of representation some of which this research addresses. A modern problem, generated by recent developments in digital and mobile communication techniques (Rainie and Wellman, 2012), is 'Big Data'. It refers to data sets so large and complex that they are beyond the ability of commonly used algorithms and softwares to process (Snijders et al., 2012). Laney (2001) formalized this challenging problem for the first time by defining three dimensions: high volume, high velocity and hight variety. Later, Douglas (2012) updated its definition as follows "Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization". Big Data may have a large number of attributes in a learning task, called the curse of dimensionality (Elder IV and Pregibon, 1996), which causes problems for learning algorithms. This is one of the major problems with Big Data (Matloff, 2013). Following are some famous quotes about curse of dimensionality:

- Elder IV and Pregibon (1996) said "methodological intuition gained from experience in low dimensions is thoroughly out of place in high-D spaces - a phenomenon known as the 'curse of dimensionality'".
- Yao et al. (2006) mentioned "It is a crucial issue to select the most suitable features or properties of the objects in a data set in the machine learning process".
- Mining (2008) said "Sometimes too much information can reduce the effectiveness of data mining. Some of the columns of data attributes assembled for building and testing a model may not contribute meaningful information to the model. Some may actually detract from the quality and accuracy of the model".
- Chizi and Maimon (2010) indicated that "Dimensionality (i.e., the number of data set attributes or groups of attributes) constitutes a serious obstacle to the efficiency of most Data Mining algorithms. This obstacle is sometimes known as the 'curse of dimensionality'".

Big Data includes raw information which is not self-explanatory and one of the ways to apply commonly used conventional algorithms is to use new methods to interpret or clean it using a filter. Bollier and Firestone (2010) said “As a large mass of raw information, Big Data is not self-explanatory. And yet the specific methodologies for interpreting the data are open to all sorts of philosophical debate”. An important factor in this interpretation or cleaning is to use relations between attributes. Boyd and Crawford (2011) mentioned “Big Data is notable not because of its size, but because of its relationality to other data. Due to efforts to mine and aggregate data, Big Data is fundamentally networked. Its value comes from the patterns that can be derived by making connections between pieces of data, about an individual, about individuals in relation to others, about groups of people, or simply about the structure of information itself”. Kim et al. (2013) indicated that “Primary goal of Big Data is extracting value from the large amount of data. Attribute relevance in Big Data is a key element for extracting information”. Many recent approaches attempt to handle the large number of attributes by finding the relations between them. Examples of the use of such relationships include feature selection (Tan et al., 2014; Hoi et al., 2012; Maji, 2012, 2011; Au et al., 2005; Li and Zha, 2002; Zhu et al., 2003), feature extraction (Jiang et al., 2011), feature replacement (Hong et al., 2009b), data summarization (Mampaey and Vreeken, 2010), subspace clustering (Niu et al., 2008) and topic identification (Blei et al., 2003). Many of these approaches use grouping of attributes in sets of similar or related ones, called attribute clustering, which is done using a similarity, correlation or information measure between attributes. Attribute clustering is mainly used in data preparation and filtering to refine the data prior to applying learning algorithms.

It is important to realize that attribute clustering is different from data clustering. The latter clusters similar instances and is an unsupervised process (there is no class definition). Attribute clustering, on the other hand, finds clusters of similar or related attributes. The data used in this process may be supervised or unsupervised (we may have the class label for each instance in the data set). The conventional data clustering methods and similarity measures are not useful for clustering attributes. Therefore, many alternative measures, like mutual information between attributes, have replaced them. Also conventional clustering algorithms could only be used for unsupervised attribute clustering. For supervised attribute clustering we need new approaches which consider the label of instances. To solve this problem many new approaches have been recently proposed.

In principle, attribute clustering could be achieved using

- supervised methods

- unsupervised methods
- semi-supervised methods

As we will see in the Related Work chapter, most of the work in this research area is concentrated on supervised and unsupervised attribute clustering. Supervised attribute clustering uses similarity measures which consider the class label of the training instances when looking for the relationship between attributes. These measures are more accurate compared to unsupervised ones and extract patterns which could not be recognized by the latter. Therefore, supervised attribute clustering will lead to better clusters of attributes which better fit the data. However, in many real world applications, labeled data are rare due to labeling costs. It is not possible to find clusters using a supervised method, because the small training set is insufficient to find relations between attributes. Semi-supervised attribute clustering is one answer, using the small labeled training set with a large unlabeled data set. It leads to finding clusters which better capture the important relationships between attributes. There exist three simple semi-supervised attribute clustering frameworks:

- One uses a semi-supervised classifier to add more labeled instances to the training set before performing attribute clustering. Problems will arise if the classifier is not accurate enough to add more labeled instances. Additionally, the classifier could be improved using the result of attribute clustering in an iterative framework.
- The second uses background knowledge in the form of some constraints on the clusters, together with unlabeled data. These constraints include attributes that must be clustered together and those that shouldn't. However, in most real world problems supervised information is in form of labeled data so this framework could not be applied.
- The third combines a supervised similarity measure with an unsupervised one used in the attribute clustering process. Yet the unsupervised similarity measure is not as accurate as supervised one and it will degrade performance.

None of these proposed semi-supervised attribute clustering methods is effective. We address this problem by proposing a new semi-supervised attribute clustering framework.

In the rest of this chapter we show how researchers have chosen to use various supervised and unsupervised attribute clustering methods to solve different problems. We show there has always been a need for more accurate attribute clustering methods, especially when we have a small training set of high dimensions. We address this need by proposing a new semi-supervised attribute clustering method. This work will not only be effective as a semi-supervised clustering approach but also as a novel semi-supervised classifier.

1.1 Attribute Clustering

In this section, we illustrate attribute clustering and its applications in machine learning literature using two real practical attribute clustering examples, applied to a text corpus. Then we use those examples to illustrate different applications of attribute clustering. An industrial application is also presented to indicate how useful attribute clustering could be in real world problems.

1.1.1 Attribute Clustering Applied to Improve Text Classification

In this section, a simple attribute clustering example is illustrated using a data set represented as a frequency based bag of words (la1s.wc.arff), available at the Weka (Hall et al., 2009) data set website[1]. The data has two classes, financial and sport.

In the first example, supervised mutual information is used as a similarity measure to find mutually related words and then to define clusters based on them. The aim of this example is to illustrate and interpret the relations between attributes extracted using such a measure. Figure 1.1 illustrates these relations. The first point of interest is that the measure has separated the attributes into two clusters based on the class: one includes all the attributes (words) which are related to sport and the other includes all those which are related to financial. As illustrated in figure 1.1 win, won, team, season, championship, victory, play, player, football, coach, game, night, sport, basketball, score and athlete are clustered together. Another cluster includes invest, company, market, inc, stock, corporate, secure, sale, acquisition, firm, president, chairman, industry, cooperate, story, financial, base, billion, million, invest. The former includes the words which are related to the concept of sport while the latter contains words which are more related to financial concept. For example win, won, victory and championship are related to the concept of competition in sport games while invest, stock and sale are related to

[1]This data is available at http://www.cs.waikato.ac.nz/ml/weka/datasets.html

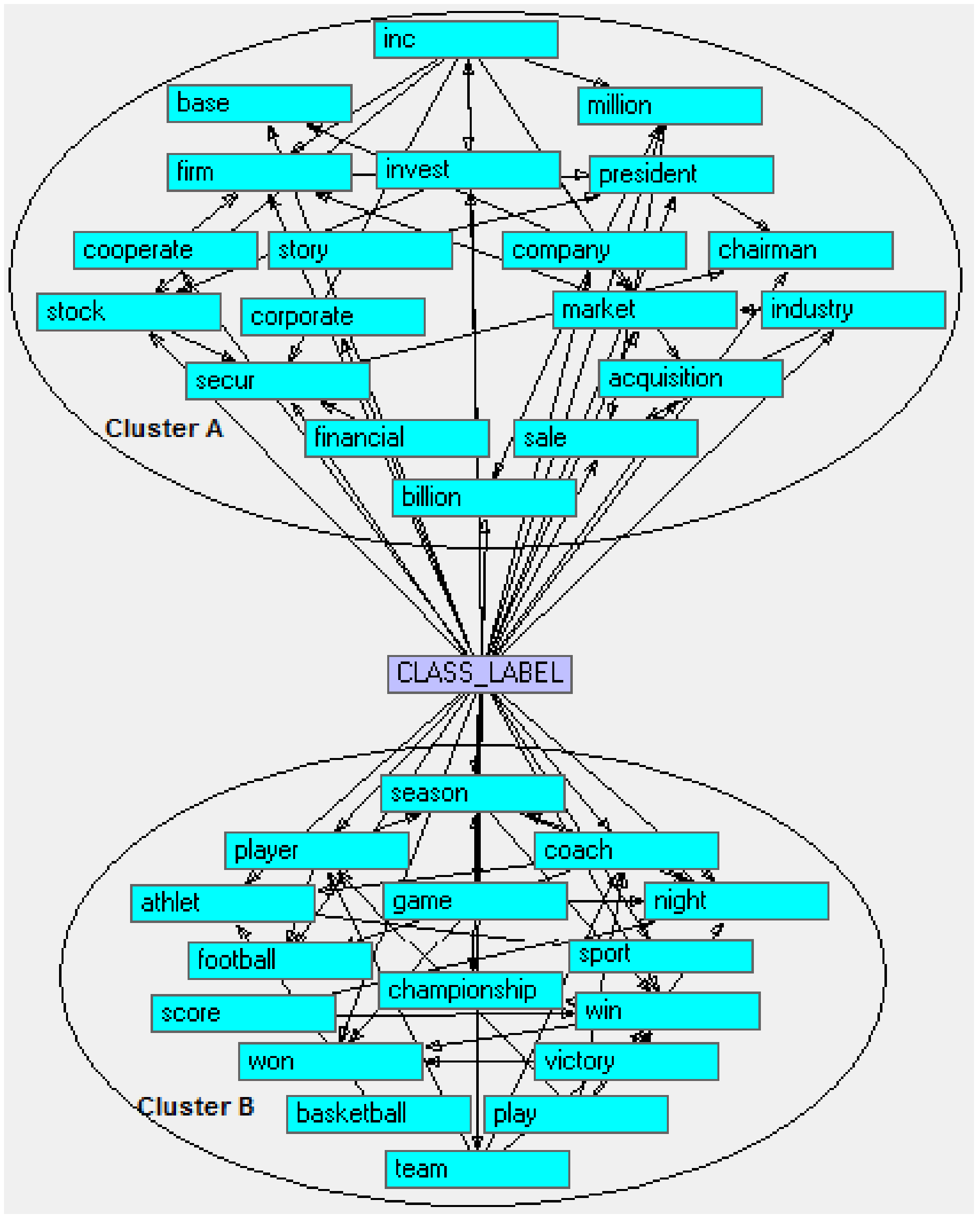


Figure 1.1: An attribute clustering example for a text corpus

the financial aspects of market. Each cluster contains meaningful words that are closely related to each other and the clusters seem to be conditionally independent given the class. No relation found between the words from the two groups. 14 words out of 50 are not clustered due to the low mutual information with other attributes. This may happen either when the word is not really related to any of the clusters or when there is not enough training instances available to obtain the mutual information.

In the second example the result of supervised and unsupervised attribute clustering are compared. Conditional correlation considers the class value of instances when calculating the correlation between every two attributes and it is a supervised similarity measure. We perform attribute clustering twice, on the same data set, using conditional correlation and standard correlation respectively as similarity measures. This will result in clusters of highly correlated attributes. Table 1.1 shows the result of supervised and unsupervised attribute clustering. Comparing these two tables reveals that the supervised attribute clustering clustered all the attributes while the unsupervised one was not able to find relations for 26 of the attributes and did not cluster them. As an example, the supervised attribute clustering found vice and president are highly correlated so they are clustered together while the unsupervised attribute clustering did not identify any relation between these two attributes. As another example, supervised attribute clustering clustered company, corporation and product together while the unsupervised attribute clustering did not cluster corporation and product.

Correlation between two attributes, two words in this example, could be interpreted as the following: if two words are highly correlated, then if one of them shows up in a document, it is more likely to see the other one in the same document. In other words, these words show up most of the time together in documents. The power of supervised attribute clustering comes from considering the class of documents when calculating the correlation between attributes. It is possible to have two attributes which are only correlated in instances of one of the classes. Supervised conditional correlation is able to find such relations while unsupervised correlation will consider such attributes not correlated.

1.1.2 Use of Attribute Clustering in Learning Tasks

Clustering attributes in machine learning tasks, specifically the ones which suffer from curse of dimensionality, has recently been of keen interest to researchers in machine learning. It is commonly used to extract relations between attributes, the key element

Table 1.1: An attribute clustering example for a text corpus, using A) conditional correlation B) standard correlation, as similarity measures.

A)

Cluster1	**Cluster2**	**Cluster3**	**Cluster4**	**Cluster5**	**Cluster6**	**Cluster7**
Win	President	Game	Team	Company	Industry	Inc
Won	Vice	Play	Coach	Corporation	Market	Invest
Beat		Score	Season	Product	Sale	Financial
Champion		Victory	Player		Price	Stock
Final		Basketball	League			Secure
		Story	Football			Acquisition
		Lead				Firm
		Night				
		Shot		**Cluster8**	**Cluster9**	**Cluster10**
		Minute		Million	Athlete	Chairman
		Left		Billion	Sport	Co
				Corporate	School	Amp
				Profit		

B)

Cluster1	**Cluster2**	**Cluster3**	**Cluster4**	**Cluster5**	**Cluster6**	**Cluster7**	**Cluster8**
Win	President	Game	Team	Company	Sport	Player	Acquisition
Won	Chairman	Score	Coach	Industry	Athlete	Play	Invest
Victory			Football	Firm		Season	Stock
Champion				Market			Inc

for extracting information from Big Data (Kim et al., 2013). Many attribute clustering methods have been developed and many usages of these methods have been studied and published. Most of these applications propose solutions for high-dimensional tasks. We will use the example proposed in the previous section to illustrate applications of attribute clustering in some learning tasks. These applications include

- **Attribute reduction (John et al., 1994; Kohavi and Frasca, 1994):** There are many high-dimensional learning tasks with thousands of attributes. In such data sets, some redundant attributes may exist. Attribute reduction is a solution for reducing the number of attributes. Feature selection and feature extraction are two alternatives. In text classification, attribute clustering is an effective method for attribute reduction (Jiang et al., 2011). The idea is to combine words into a set of clusters, whose attributes are semantically related to each other. Each cluster can be represented by a new single attribute thus reducing the dimensionality drastically. Each cluster could be represented by selecting an existing attribute or by creating a new one for each group like a mean value.

  - **Feature selection (Kira and Rendell, 1992; Liu and Motoda, 1998a):** Feature selection is a process of selecting relevant features and removing irrelevant and redundant ones. This represents the data in a simpler way which may not only reduce the execution time but also make the results easier to interpret. We can use the following hypothesis, "Good feature subsets contain features highly correlated with the class, yet uncorrelated with each other", presented in Hall (1999), to cluster attributes with high correlation together and then select an attributes or more from each cluster as new reduced attribute set.

    Applying feature selection using attribute clustering in the example given in section 1.1.1, we cluster words which have similar meanings or are closely related to each other in the same clusters. These are words which are more likely to appear in the same documents. For feature selection we would select one attribute from each cluster and that attribute will represent all the attributes in that cluster. For example, we would select *"president"* from the cluster containing *"president"* and *"chairman"* or from the cluster containing *"game"* and *"score"* the attribute *"game"* will be selected. The learning methods would be applied on the set of all selected attributes. Usually we use information measures to rank the attributes in each cluster and then we select

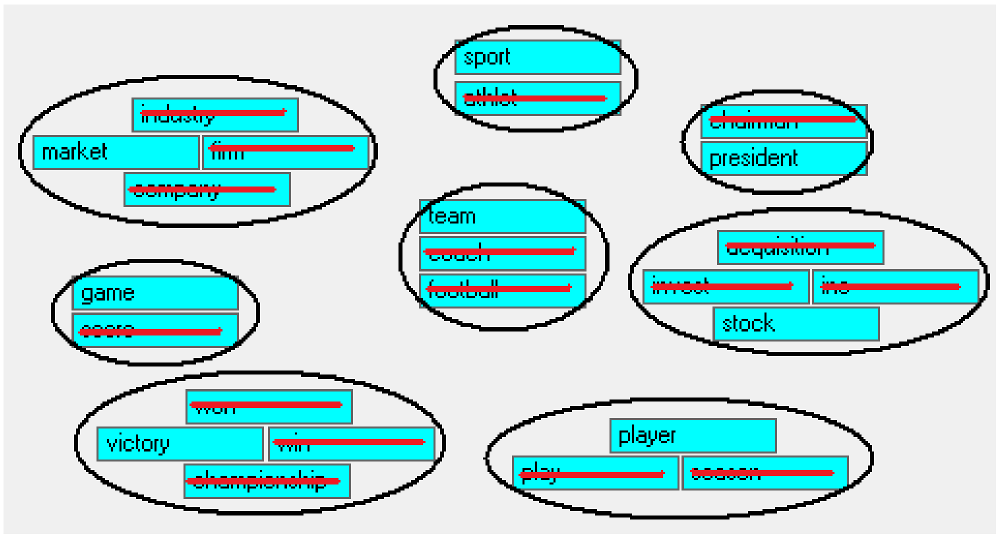


Figure 1.2: Selected attributes for the text classification example

some of those attributes in each cluster which give us much more information. A possible selected attribute set as the result would be the set of attributes not crossed with a red line in figure 1.2.

As a practical example feature selection is used in classification of microarray (Maji, 2012, 2011; Li and Zha, 2002) and gene expression (Au et al., 2005) data which suffer from curse of dimensionality. In such data sets genes are the attributes and there are a large number of genes but a small number of samples. An example is the Healthy Controls data set (Van der Pouw Kraan et al., 2007) which has almost 26000 genes as attributes but only 50 samples. In such tasks feature selection based on attribute clustering is used to identify a reduced set of the most relevant genes.

As another example Zhu et al. (2003) select features based on attribute clustering to provide a small feature set for indexing and representing cases in case-based reasoning. Retrieving similar cases using these representative features reduced the execution time.

– **Feature extraction (Liu and Motoda, 1998b):** For feature extraction, we would create a new feature for each cluster. These new features are extracted by summing up the features in each cluster, by adding the number of appearances of all the words in the same cluster, for each document.

  In our text classification example the process would be similar to feature selection but instead of selecting a representative attribute for each cluster we create a new attribute for each. For example for the cluster containing *"President"* and *"Chairman"* we create a new attribute called *"Boss"* which will represent that the word *"President"* or *"Chairman"* exists in the corresponding document.

As a practical example, Jiang et al. (2011) used attribute clustering to reduce features of a high-dimensional text classification problem by grouping similar words into the same clusters and then extracting one feature for each cluster.

- **Feature replacement:** Attribute clustering clusters attributes with high similarity into the same cluster. In feature selection we select one of the attributes in a cluster to represent that cluster. Hong et al. (2009b) replaced any selected attribute with a missing value by an attribute within the same cluster and calls this approach feature replacement.

  In our text classification problem, suppose that *"president"* is selected as representative attribute from the cluster containing *"president"* and *"chairman"*. A classifier is trained using all the representative attributes. If a new instance to be classified has missing attribute value in the attribute "president", we would select *"chairman"* as another representative attribute from the corresponding cluster. Then we would train a new classifier using the new set of representative attributes and classify the instance.

- **Data summarization:** When exploring the data extracted from a new domain, it is quite hard to get a good first impression. Attribute labels will convey some information about the content. However, these do not provide a comprehensive overview of what is in the data set. Mampaey and Vreeken (2010) proposed a solution to this problem using attribute clustering that provides high-quality summary overviews for binary transaction data. The outcome provides insight into which attributes are most correlated and in what contexts these occur.

  In our text classification example, given in section 1.1.1, when mutual information is used as similarity measure, attribute clustering tells us that there exist two clusters representing two separate *"financial"* and *"sport"* views in our data set and how attributes within each cluster are related.

- **Subspace clustering:** Subspace clustering is a strategy that reduces the complexity of clustering high-dimensional data. It attempts to find clusters within subspace of the data sets. It is useful when dimensionality increases and the data become increasingly sparse and clustering of the data using all the feature space is not possible.

In our text example with two separate clusters (views) of attributes, to find the clusters in the data we would use one view as the attribute set to cluster the data instead of using the whole feature set.

As a practical example, Niu et al. (2008) used attribute clustering to propose a fast algorithm of subspace clustering which overcomes the limitations of existing subspace clustering methods. These limitations are

- the algorithms typically scale exponentially with the data dimensionality or the subspace dimensionality of clusters.
- the clustering results are often sensitive to input parameters.

- **Topic identification:** A topic model is a statistical model for discovering the implicit topic of a collection of documents in machine learning and natural language processing. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is one of the popular methods applied to textual data collections for topic identification. It uses attribute clustering to find the related words (attributes) and creates the topics (clusters). Then, based on the statistics of the words in each document it captures what the topics might be and what each document's balance of topics is.

  In our text example on page 5, attribute clustering separated the words into two clusters: one includes all the words which are related to sport and the other includes all those which are related to financial. For each document the statistics of the words related to a specific topic shows how much that document is related to that topic.

1.1.3 Definition

According to Au et al. (2005), Attribute clustering can be defined as the partitioning of a set A of attributes $A = \{A_1, A_2, ..., A_M\}$ in a machine learning task, into a collection $C_A = \{C_1, C_2, ..., C_K\}$ of mutually disjoint subsets of correlated features C_i of A, where k is the number of k clusters of attributes, such that $C_1 \cup C_2 \cup ... \cup C_k = A, C_i \neq \phi$, and $C_i \cap C_j = \phi$ for $i \neq j$. Many different correlation measures, capturing similarity or dissimilarity, measures of relatedness and information measure are used, based on the application, to form the clusters. Some of these measures are unsupervised but some are supervised. The supervised ones consider the class label of samples while they compute the similarity of attributes. These various measures and their applications are reviewed in the Related Work chapter.

1.2 Example of Attribute Clustering

There are many industrial applications and examples of attribute clustering in machine learning literature. Here we indicate an industrial example which shows application of attribute clustering to data preparation and attribute reduction for power system problems.

1.2.1 Data Preparation for Power System Problems

In this part we discuss work done by Louis Wehenkel in his PhD thesis (Wehenkel, 1994). He used attribute clustering for data preparation and more specifically attribute reduction. He applied it to the problem of preventive transient stability assessment applied to various power system problems like the 735kV system of Hydro-Quebec.

He was faced with a large number of attributes. Thus, there is a need to reduce this information. This is a typical attribute clustering problem, which he has solved using hierarchical agglomerative attribute clustering (Wehenkel, 1994).

In such clustering, each observation (here observations are attributes) starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. At each step of the algorithm it proceeds by identifying the two most similar clusters and merging them to form a single new cluster. This process continues until all attributes have been merged into a single cluster. The resulting hierarchy may be represented by a dendrogram like the example shown in figure 1.3 which shows graphically the hierarchical groupings of attributes along with the cluster (dis)similarities. This is particularly interesting for the analysis of attribute similarities, when the number of attributes to cluster is small.

To illustrate this problem, he has clustered 14 power flow and 3 power generation attributes. The similarity among the attributes was defined as their correlation coefficient. It is estimated for each pair of attributes. The similarity of two subsets of attributes was defined as the minimum similarity of pairs of attributes of the two subsets. It is interesting to know that, in results, there exist a contextual relation between the attributes inside each of the clusters. One cluster of attributes shows that all its attributes corre-

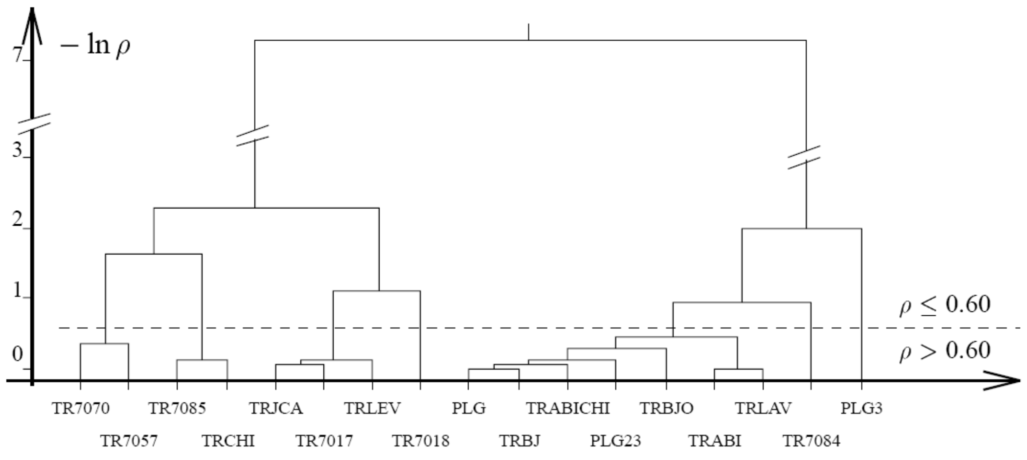


Figure 1.3: A hierarchical attribute clustering example

spond either to generations of the LaGrande power plant or to North to South power flows in the James Bay corridor. Similarly, attributes in another cluster are all related to lines within the Quebec to Montreal corridor, which are shared by the North to South and the West to East corridors of the Hydro-Quebec system. As we see in this example, the clustering analysis of attributes allows us to identify groups of attributes which share the same physical information. In the context of power system security problems this is very frequent for variables such as power flows or voltages. A simple dendrogram may then be drawn to suggest which groups of such attributes may be represented by a single prototype, e.g. a mean value or a representing attribute from the same group. In practice, this may lead to more efficient and more robust classification results. This method was found to provide a substantial reduction in dimensionality in several other power system security applications.

1.3 Proposed Work

Clustering attributes in tasks, specifically the ones which suffer from curse of dimensionality, has recently been of keen interest to researchers in machine learning. Many supervised and unsupervised attribute clustering methods have been developed and many applications of these methods have been studied and published so far. The lack of an effective semi-supervised attribute clustering method motivated us to propose a new two-step iterative semi-supervised attribute clustering approach.

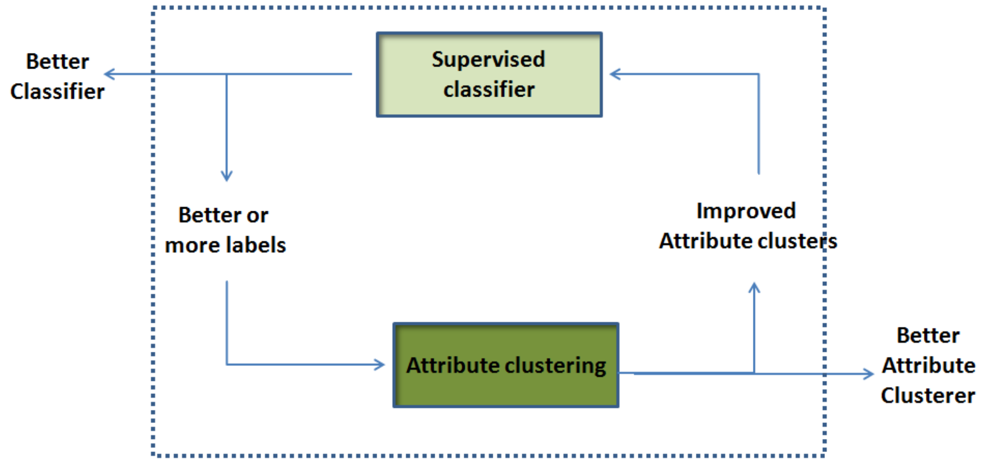


Figure 1.4: Iterative semi-supervised attribute clustering framework using supervised classifier and supervised distance measure.

1.3.1 Proposed Framework

This framework is illustrated in figure 1.4. In each iteration of this framework, the results of attribute clustering will be used to improve a classifier and then the improved classifier will be used to improve attribute clustering by augmenting its labeled training set, with the most confident new labeled instances. The improvement of the classifier could be done by using attribute clustering in different ways. For example, it could be done by using attribute clustering for defining views for a multi-view classifier or improving missing value handling methods in the classifier. All other usages of attribute clustering, for improving a classifier, could be applied to this framework.

1.3.2 Research Contributions

The proposed framework in this thesis has impact in two machine learning research areas: attribute clustering and data classification. It is a contribution to attribute clustering by proposing an effective semi-supervised attribute clustering framework. The proposed approach is able to extract hidden relations between attributes, using only small labeled training sets. It is a contribution to classification by proposing an improved semi-supervised classifier. The proposed framework applies attribute clustering results to improve the semi-supervised classifier in each iteration and the semi-supervised classifier used to improves the attribute clustering, by augmenting its training set, in each iteration.

The proposed framework is very broad. The result of attribute clustering can be used in different ways to improve a classifier. We look at two very particular ways of exploiting the framework. Unquestionably, there will be others worth exploring in future work. This research is a contribution to multi-view learning by proposing a solution to the problem of automatic view definition. It is also a contribution to missing attribute handling by proposing an improved missing attribute value handling method.

The specific contributions of this thesis are:

- An effective semi-supervised attribute clustering approach
- An improved semi-supervised classifier
- A solution for the problem of automatic view definition, for multi-view learning classifiers
- An improved missing attribute value handling method

We have submitted a journal paper on this thesis to the Journal of Machine Learning Research (Seifi et al.).

1.4 Summary

Variants of attribute clustering are encountered in various machine learning tasks such as attribute reduction, feature replacement, data summarization, subspace clustering and topic identification. Though there is a strong appeal for more efficient and effective attribute clustering methods, little has been done to improve attribute clustering when labeled data is limited. Most of the work in this domain is concentrated on supervised and unsupervised attribute clustering. In real world learning tasks it is more likely to have a small labeled training set together with a large number of unlabeled instances. In such cases, supervised attribute clustering is not reliable. On the other hand, the unsupervised attribute clustering is not able to reveal and use all the relationships between attributes.

Very few simple semi-supervised attribute clustering approaches have been proposed yet. One of these approaches uses background knowledge as supervised information, in the form of some restrictions on what attributes should be clustered together or not. This is not applicable for the common cases when we only have labeled and unlabeled data. Another approach combines labeled and unlabeled data by using a hybrid distance measure. The combination of supervised and unsupervised distance measures is not accurate, specifically when the labeled training set is very small. The supervised distance measure is not reliable and also the unsupervised measure is not able to extract all the relationships between attributes. A third approach simply performs semi-supervised learning to increase the size of the labeled training set in the first step and then performs supervised attribute clustering in the second step. The problem in this approach is that the first step which uses a classifier could not be improved using attribute clustering.

To address these problems we propose a new iterative semi-supervised attribute clustering framework by combining attribute clustering and supervised learning. Our research not only provides an effective semi-supervised attribute clustering approach but it also creates an improved semi-supervised classifier. When attribute clustering is used for view definition in this framework, it provides a solution to the problem of automatic view definition, for multi-view learning classifiers. We also propose the use of attribute clustering to provide an improved missing attribute value handling method at induction time, which looks for the missing value of an attribute considering related attributes from the same cluster. Our framework, when used for automatic view definition for multi-view learning, improves missing attribute value handling at prediction time by grouping attributes and reducing the number of reduced feature models. In our experimental results, we evaluate this framework and its usages, using real data sets from two different domains including text and microarray gene expression data.

Chapter 2

Related Work

This chapter addresses the justification, positioning and novelty of this research. The justification motivates this research by emphasizing the importance of attribute clustering and showing the need for an effective semi-supervised attribute clustering approach. The positioning takes a broad view of the machine learning area, including supervised, unsupervised and semi-supervised learning, and positions the proposed research. The novelty shows how this work differs from prior research, by overcoming weaknesses of those methods.

2.1 Justification

When the dimensionality of the data space increases, the volume of the space increases dramatically and consequently the available data become sparse. This sparsity of the data degrades the performance of the learning methods and is problematic for those which require statistical significance. Elder IV and Pregibon (1996) called this phenomenon 'Curse of Dimensionality' and Mining (2008) has justified it as the result of some attributes which may not contribute meaningful information to the model or even detract from the accuracy of it. According to Chizi and Maimon (2010), large increase in dimensionality, the number of data set attributes, causes efficiency problems for most of the machine learning algorithms, in terms of the time need to build the model.

Clustering attributes, specifically the ones which suffer from the curse of dimensionality, has recently been of keen interest to researchers in Machine Learning. Many attribute clustering methods have been developed and published. These methods have been used to apply machine learning techniques in many practical situations. These new techniques include

- Attribute reduction
  - Feature selection
  - Feature extraction
- Feature replacement
- Data summarization
- Subspace clustering
- Topic identification

A topic area that has grown rapidly in recent years is Gene Expression and Microarray Analysis. It has the potential to radically facilitate how medicine discovers totally novel and unexpected functional roles of genes (Slonim and Yanai, 2009). Attribute clustering is critical to find predominant patterns in the Gene Expression and Microarray data. It is an inherently high dimensional domain, so is one of the major places where attribute clustering techniques are needed. Such high-dimensional data contains large number of genes (attributes) but a small number of samples. Attribute clustering is used to reduce the number of dimensions searched by data mining algorithms. This is done for example by attribute selection or reduction using clusters of correlated attributes.

Over the last decade many well-known conferences, workshops or journals concentrated on applications of attribute clustering on gene expression data. DIMACS Workshop on Analysis of Gene Expression Data, which was held by Rutgers University at Feb 2001, was one of the earliest workshops which completely focused on this topic. International Workshops on Practical Applications of Computational Biology and Bioinformatics, which started at 2007, and MultiClust Workshops: Discovering, Summarizing and Using Multiple Clusterings, which started at 2010, are the two most recent ones that included major parts on applications of attribute clustering on gene expression data. Two comprehensive surveys have also been published on this domain collecting and summarizing most of the proposed techniques and methods. Jiang et al. (2004) divided the cluster analysis for gene expression data to Gene-based clustering (attribute clustering) and Sample-based clustering. They devoted the first part to the papers related to application of attribute clustering to gene expression data. Nagi et al. (2011) showed how this domain has been active for the last decade and addressed various approaches to gene expression data analysis using clustering techniques. More than one hundred related papers on attribute clustering are referenced in these surveys.

These publications, workshops, surveys and usages of attribute clustering, show the great appeal for more accurate attribute clustering approaches but most of the research in this domain focused on unsupervised and supervised attribute clustering. An effective semi-supervised attribute clustering method has not been proposed yet.

When enough labeled data is available, supervised attribute clustering can provide more accurate results than unsupervised one because it uses supervised similarity or distance measures. Such measures consider the class label of instances, while measuring the relationship between attributes, therefore, they are capable of extracting class-based relationships between attributes. However, empirical experiments (e.g. (Du et al., 2011; Palanikkumar and Scholar, 2013; Maji, 2012, 2011)) showed that the result of supervised attribute clustering is not reliable when a small labeled training set is used. The number of samples is not enough to distinguish the real relationship between attributes. This is where we need semi-supervised attribute clustering to get benefit from both labeled and unlabeled instances.

As an example, attribute clustering could be used to split attributes for Co-training. Standard co-training (Blum and Mitchell, 1998) uses two disjoint sets of attributes or views to train two separate classifiers on initial labeled data. Then, alternately each classifier classifies the unlabeled data and adds a few of the most confidently labeled samples to the training set. Then the classifiers are trained again and the process repeated for a number of times until a stop criterion is met.

Du et al. (2011) raised and answered the following three crucial questions for applying standard co-training:

1. Is it possible to verify that the sufficiency and independence assumptions of co-training are satisfied, for a given large labeled dataset with two views?
2. Can we split a given large labeled dataset with single view, into two sufficient and independent views and apply the two view co-training?
3. Is it possible to split a given small labeled set with single view, into two views and apply co-training? Can we verify co-training assumptions for a given small labeled dataset (training set)?

The answer to the first two questions is positive. Du et al. (2011) described a novel approach to verify the two assumptions empirically. They also proposed four different methods to split single views to make standard co-training work. The first two questions are based on a large labeled set while the third a small labeled training set. This is actually an attribute clustering issue when the labeled data are rare. Du et al. (2011) answered this question by an emphatic no. Experimental results on both real world and synthetic datasets revealed that in the latter case, not only the verification of the sufficiency and independence assumptions are not accurate but also splitting single view into two is not reliable. This is an empirical example of supervised attribute clustering, not working reliably when labeled samples are rare.

Very few semi-supervised attribute clustering approaches have been proposed so far, which solve this problem by exploiting a small labeled dataset together with a large unlabeled dataset. All that have been proposed suffer from major weaknesses. We address them by proposing an effective semi-supervised attribute clustering approach which combines the attribute clustering with semi-supervised learning in an iterative manner.

2.2 Positioning

Depending on the availability and usage of labeled and unlabeled data, machine learning and data mining techniques can be divided into three categories:

- **Supervised Learning:** using completely labeled training data.
- **Unsupervised Learning:** using unlabeled data for learning and data mining.
- **Semi-supervised Learning:** using both labeled and unlabeled data for learning.

Attribute clustering is a machine learning technique, which uses a similarity or distance measure between attributes to cluster similar or related attributes. It can be supervised, unsupervised or semi-supervised, based on the type of similarity or distance measure used. Semi-supervised attribute clustering could be considered as a conjunction of attribute clustering and semi-supervised learning.

In the rest of this section we briefly review the history of attribute clustering methods and their weaknesses in these three categories. As we will see, most of the work done for attribute clustering is concentrated on supervised and unsupervised attribute clustering and there seems to be a big niche in improving attribute clustering using large unlabeled data sets, available in most learning tasks.

2.2.1 Unsupervised Attribute Clustering

In this section, we briefly outline the history of unsupervised attribute clustering, to show how it has been an important topic for researchers over the last decade with a great amount of effort dedicated to it.

In early research into unsupervised attribute clustering, Tavazoie et al. (1999) used *K-means* as a clustering method to group genes, the attributes in gene expression data. Heyer et al. (1999) pointed out two main problems. First, the conventional measure used, Euclidean distance, is not meaningful when we talk about clustering attributes. Second, the number of clusters in gene expression data is unknown so we may need to run the algorithm with different number of clusters to find the most suitable one.

Many other researchers proposed new approaches to address the problems with applying *K-means*. Heyer et al. (1999)'s algorithm clusters similar portions of an organism's genom using a similarity score between these portions. Mitra et al. (2002) used *k-means* with a new similarity measure, based on linear dependency between two random variables, called maximal information compression. Au et al. (2005) replaced the concept of mean with the concept of mode and also used the interdependence redundancy measure between attributes. Hong and Liou (2007) proposed a new distance measure between attributes, based on the relative dependency. Covões et al. (2009) improved the method described by Mitra et al. (2002) in a new method, capable of selecting features automatically.

The work to propose a more efficient and effective unsupervised attribute clustering has not been limited to the application of k-means and its variants. Tjhi and Chen (2006) developed a new fuzzy co-clustering algorithm which considers feature-cluster weighting. An innovative attribute clustering approach based on genetic algorithms is proposed in Hong et al. (2009a), Hong et al. (2009b) and Hong et al. (2010). Zeng et al. (2009) generated a new attribute clustering method based on spanning trees. Minimum Description Length (MDL) is used by Mampaey and Vreeken (2010) to develop an attribute clustering approach, which is then used for data summarization. A novel unsupervised feature selection method, based on hierarchical attribute clustering, is proposed by Li et al. (2008). Maximal Information Compression Index (MICI) is used in this work as similarity measure. Niu et al. (2008) used a new overlapping attribute clustering method in a new subspace clustering algorithm.

All this research shows the great interest in finding a better attribute clustering approach. However, with labeled instances we could get better results by applying a supervised approach.

2.2.2 Supervised Attribute Clustering

Since supervised attribute clustering methods apply supervised measures with labeled data, they can extract and use the class-based relationship between attributes. Therefore, if there is enough labeled instances available, they provide better clusters of attributes which better fit the real patterns in the data. Here we review supervised attribute clustering approaches to highlight the amount of work done on this topic.

Li and Zha (2002) proposed a simultaneous classification and attribute clustering method using Discriminant Vector Quantization (DVQ) (Li, 2002). This iterative approach gradually fits the model to the data by applying attribute clustering for feature selection and then recalculating the parameters of the model considering the result of testing the classifier. Jiang et al. (2007) invented a new confidence-based hierarchical word clustering method where word groups represent new features for document classification. Occurrences of words, given the class, form clusters, so this is a supervised attribute clustering method. Jiang et al. (2011) groups similar words into the same cluster using a supervised similarity test. Maji (2012) used a new supervised similarity measure for grouping co-regulated genes which has strong association to the class labels. This work is extended in Maji (2011) by using a new supervised similarity measure based on fuzzy-rough sets.

Although supervised attribute clustering tends to be more reliable than unsupervised one, one reason that has limited its popularity is the requirement for labeled data, a resource that is often scarce.

2.2.3 Semi-supervised Attribute Clustering

The previous sections show the large amount of work directed towards finding an effective attribute clustering method. They also show that most of the work done in this domain is concentrated on unsupervised attribute clustering. Although supervised attribute clustering gives better clusters of attributes, in most real world problems we do not have large labeled data sets. So, we are not able to find accurate relationships between attributes. In many of such real world problems we also have large number of unlabeled data available. This is where semi-supervised attribute clustering approaches would be most effective.

Although such a semi-supervised method can give us much more accurate attribute clustering results, only a little work (Niu et al., 2005; Frigui and Mahdi, 2007; Quinzán et al., 2009) has been already done in this area. The lack of an effective semi-supervised attribute clustering approach motivated our research and we are addressing this problem by proposing a new iterative semi-supervised attribute clustering approach to fill this gap. The very few existing semi-supervised attribute clustering approaches and their weaknesses are mentioned later in the novelty section. They are explained in more detail in chapter 3.

2.3 Novelty

This section shows how the related work differs from proposed research, its weaknesses and how they are overcome by the approach put forward in this thesis.

2.3.1 Improving Semi-supervised Attribute Clustering

The simplest way to use labeled and unlabeled data in attribute clustering would be the use of a semi-supervised classifier to increase the size of labeled set and then the use of the enlarged labeled set to do supervised attribute clustering. This framework, which is shown in figure 2.1, performs supervised classification at the first stage and once the labeled training set is augmented, applies supervised attribute clustering. The main difference between this approach and ours is that it is not using the result of attribute clustering to improve the classifier. It is a one pass process while our approach is an iterative one.

Some supervised algorithms perform attribute clustering using a matrix which includes the number of times each feature co-occurs within each class. Those supervised algorithms can not cluster attributes which only appear in unlabeled data because the distribution of class labels over those attributes is not known (instances which include those attributes are not labeled). Niu et al. (2005) proposed a semi-supervised attribute clustering approach which uses the distribution of class labels over features in label data to derive the distribution unavailable in training data. This is based on the similarity between attributes in labeled data and those which only exist in unlabeled data. Therefore, this algorithm could be used not only on the attributes which exist in training set but also on those attributes which only appear in unlabeled data set. When this method is used for dimensionality reduction, it outperforms other techniques. This means using

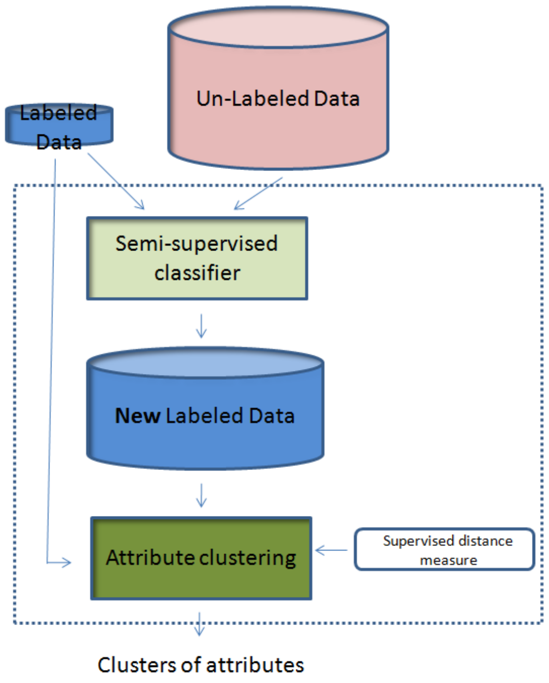


Figure 2.1: One pass semi-supervised attribute clustering framework using a semi-supervised classifier.

unlabeled data which includes more attributes can improve the performance of attribute clustering. This algorithm is different from our approach in that it is not using the unlabeled data to improve the attribute clustering but using it to cluster attributes not in labeled data. It is performing one pass and therefore it will not work well when the number of labeled instances is small.

Frigui and Mahdi (2007) developed another semi-supervised attribute clustering approach, which is used for feature weighing. This framework is illustrated in figure 2.2. It uses a set of unlabeled data together with supervised information, in the form of background knowledge. This consists of a small set of constraints on which attributes should or should not be in the same cluster. The attribute clustering process includes minimization of an objective function which has three parts. One part is responsible for minimizing the distance between attributes in clusters while the two other parts are taking care of must-link and can-not-link constraints. Minimization of this objective function lead to compact clusters, learned feature weights and the incorporation of partial supervision information. Experimental results showed that the new algorithm outperforms similar algorithms and the use of supervision constraints with unlabeled data set improves the

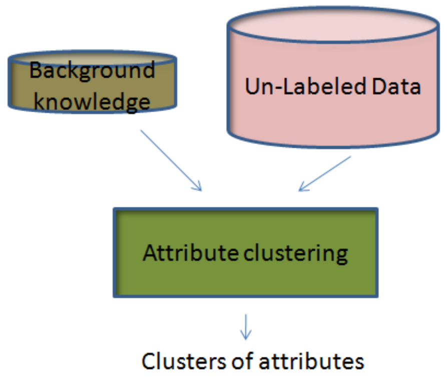


Figure 2.2: One pass semi-supervised attribute clustering framework using background knowledge

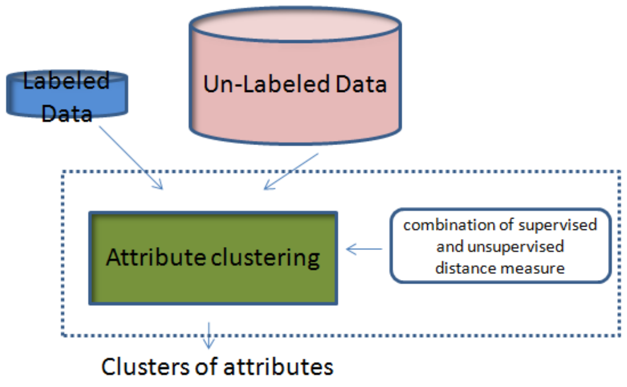


Figure 2.3: One pass semi-supervised attribute clustering framework using semi-supervised distance measure

attribute clustering result. The main difference between this approach and ours, is the difference between supervised information used in learning process. This approach is using background knowledge in form of must-link and can-not-link constraints, however, in many real world learning tasks we only have a small labeled training set with large number of unlabeled instances. Our approach could be applied on such tasks for attribute clustering.

Quinzán et al. (2009) proposed a new semi-supervised attribute clustering which is used for feature selection. This algorithm uses a combination of supervised and unsupervised distance measure to use both labeled and unlabeled data for attribute clustering. The distance between attributes is calculated based on conditional mutual information and conditional entropy. This approach is shown in figure 2.3.

The difference between this approach and ours is in the way the unlabeled data is used. It is used together with labeled data in a semi-supervised distance measure. This measure has two parts. One part calculates the distance between two attributes based on unlabeled data. Other part measures this distance using labeled data and by considering the class label. Combining these two in a single measure degrades the accuracy of the supervised measure. In our approach, only a supervised distance measure is used and the unlabeled data is used to augment the labeled set, using a classifier iteratively.

As we saw in this section, a few semi-supervised attribute clustering methods have been proposed so far, which all have weaknesses. We addressed these by proposing a new iterative semi-supervised attribute clustering approach. This new approach is applicable on many real world learning tasks which include a small labeled training set and a large unlabeled data set.

2.3.2 Solving the Automatic View Definition Problem

This section shows the value of a new approach which solves the problem of automatic view definition for multi-view learning especially for cases which we do not have enough labeled data.

There usually exist several views, from different angles, to a learning problem. Conventional machine learning algorithms concatenate all these multiple views into a single one for simplicity. However, this concatenation causes overfitting, specifically in case of small labeled training sets and is not meaningful because it removes the specific statistical property of each view (Di and Crawford, 2012). Alternatively, multi-view learning is used to train multiple learners on different views and then concatenate the learning results using an aggregation function.

Blum and Mitchell (1998) formalized multi-view learning for the first time by proposing co-training as a semi-supervised approach to multi-view learning. They proposed two assumptions under which it is guaranteed to work well. The first assumption is that the views are sufficient; each view could be used to perfectly predict the class. The second assumption is that the two views are conditionally independent, independent given the class. Their theoretical results have shown that if these two assumptions are satisfied then co-training is guaranteed to work well.

Unfortunately in many real world problems we do not have a priori views which meet these assumptions. This problem motivated researchers to relax these assumptions. Of the work done, some completely removed the need for different views. Wang and Zhou

(2007) derived a theorem showing why co-training can work with a data set that does not have two views. Goldman and Zhou (2000) proposed a new approach which used two different predictors that are each trained using the whole feature space. Zhou and Li (2005) used two different parameter configurations for the two predictors which are from the same base learner. These two methods are actually ensembles of two different learners, trained on the whole feature space, and not multi-view learners, because they are not following its base assumption, aggregation between learners which are trained on different sets of attributes. Raskutti et al. (2002a) presented another variation of multi-view learning in which the first predictor is trained using the original feature space and the second with new features that are derived by clustering both the labeled and unlabeled instances. The clusters are considered as higher level concepts in the data, and the derived features indicate the similarity of each instance to these concepts. This approach is limited to two separate views: the original and the derived feature set.

Other work aims to weaken or replace the assumptions. Wang and Zhou (2010), Abney (2002) and Wang and Zhou (2013) showed that weak dependence can result in successful co-training. Balcan et al. (2004) analyzed co-training and proved that the strong assumptions needed by previous theoretical analyses, guarantee the success of one-shot co-training and a much weaker 'expansion' assumption is sufficient given the iterative nature of standard co-training. Based on this analysis, Chen et al. (2011) and Ando and Zhang (2007) proposed new feature decomposition algorithms for single-view co-training which was only able to divide the features of a single-view data set into two mutually exclusive subsets. Zhu (2006), in his literature survey of semi-supervised learning, mentioned that the co-training assumptions are strong and, in general, are not required by multi-view learning models.

Another group of work concentrated on construction of multiple views from the single-view to make multi-view learning applicable on single-view data sets. A simple way to generate multiple views is to randomly split the attributes (Bickel and Scheffer, 2004; Brefeld and Scheffer, 2004; Brefeld et al., 2005). However, the view generation process must ensure that the views sufficiently represent the data and satisfy the assumptions required for learning (Xu et al., 2013). Du et al. (2011) designed several methods to split feature space into two views. Their empirical results showed that, given a large labeled training set, the splitting methods are quite effective. However, given small labeled training sets the view splitting methods are unreliable. Semi-supervised learning, is precisely used when the labeled training set is small. Therefore, their proposed splitting methods could not be applied to the problem of automatic view definition for semi-supervised multi-view learning. Di and Crawford (2012) proposed active learning strategies to generate views for Hyperspectral Image Data when training set is limited. Their proposed active learning methods differ from semi-supervised learning in the sense that it requires the true labels for the samples that the learner is the least confident about. Their empirical results showed that increasing the number of views increases the diversity and consequently improves the performance.

The proposed supervised multiple view generation methods are not reliable when labeled training set is small Du et al. (2011). However, in many real world learning tasks, there usually exist a large number of unlabeled instances available which could be used to enhance learning performance in a semi-supervised manner. This problem motivated us to propose a semi-supervised approach to generate multiple views for a single-view data set. This is addressed in our research by combining attribute clustering with classification, in an iterative process, when using multi-view learner as classifier and view generation as the way attribute clustering results are used to improve the classifier. In each iteration, the results of attribute clustering will be used to improve the views. Then, the improved classifier will be used to improve attribute clustering by augmenting its labeled training set, with the most confident new labeled instances. This iterative process will not only improve the attribute clustering results gradually, but it will also improve the views, and consequently, the multi-view classifier.

2.3.3 Improving Missing Attribute-value Handling

Missing attribute-values are very common in real data sets. An attribute-value might be missing because of various reasons, i.e., it is lost, it is considered irrelevant or it is expensive to be calculated for all instances. The absence of these values would degrade the performance of machine learning techniques. This motivated researchers to propose missing attribute-value handling methods. When talking about missing attribute-value handling, we should distinguish between missing attribute-values at induction time in the training data (Grzymała-Busse and Grzymala-Busse, 2005), and at prediction time in test data or the upcoming new instances (Saar-Tsechansky and Provost, 2007). In this section we mention existing problems in missing attribute-value handling and we indicate how these problems are addressed in our research using attribute clustering.

Missing Attribute-value Handling at Induction Time

Various approaches have been proposed for handling missing attribute-values at induction time. A group of well-known missing attribute-value handling methods are designed based on the idea of closest fit. The closest fit of a particular instance is the closest instance, in terms of Manhattan distance. This idea originated in Grzymała-Busse et al. (1999) and it has two different variations: The Global and Concept Closest Fit.

The **Global Closest Fit** method (Grzymała-Busse et al., 1999) looks into the pool of instances with known values and replaces a missing value by the value of another instance which is the closest fit to the one with missing value. To find the closest fit, Manhattan distance measure is computed between the vector for the case with missing value and the vector of attribute-values for another case which is a candidate of closest fit. The distance measure is calculated for all cases with a known value for the corresponding attribute and the instance with the smallest distance is selected. This method considers the class label as simply one of the attributes. Thus, it may find the closest fit from other classes. This might be problematic because the difference can come from the value of an attribute with a missing value.

To solve this problem, the **Concept Closest Fit** method (Grzymała-Busse et al., 1999) is proposed. It first splits the data sets into subsets based on the class label of instances. Then, for replacing each missing attribute-value, it finds the closest fit in the subset of the instances with the same class label. The only difference, comparing to the global closest fit method, is that the search space is limited to the subset of instances with the same class label. This decreases the search time and also improves the accuracy of the value found for the missing attribute-value.

According to Suguna and Thanushkodi (2011), although several approaches are proposed for missing attribute value handling at induction time, many based on the idea of closest fit (Gaur and Dulawat, 2011; Gaur, 2012, 2014), the one used more frequently is deleting instances containing at least one missing value. If the number of instances with missing attribute-value is small, this approach is applicable, otherwise it is not. Alternatively, some methods, such as assigning an average or the most common value, make good use of all the available data. However these methods do not consider the existing attribute-values of the instance when calculating its missing attribute-values. Grzymala-Busse et al. (2002) and Grzymala-Busse et al. (2005) compared the closest fit methods with different variations of the common value method. They concluded that even though the closet fit methods, unlike common value methods, consider the attribute-values of each instance when looking for its missing values, they are not better than common value methods. The computational time of closest fit methods is higher because in order to find the missing value of an instance, they need to calculate its distance to all other instances. When looking for the closest fit, the vector of attribute-values includes all the attributes. However, the feature space can include attributes which are completely unrelated to the attribute with missing value and can mislead the process of finding the closest fit. To overcome this, Common Value methods remove all the attributes except the one with missing attribute value from consideration. This removes the misleading attributes but it also removes useful attributes.

When looking for the closest fit in such cases, using a subset of the feature space, reduces the computational time. If the subset includes all the related attributes, it will improve the accuracy of the value found for replacing the missing attribute-value. Li and Cercone (2006) used rough set theory to limit the attributes to the core or one of the reducts. Each reduct is a subset of the attributes which is sufficient enough to represent the whole data. The core is the intersection of all the reducts. This approach reduces the computational time but it is not effective in terms of removing irrelevant attributes from the search. The reducts are generated considering that each one should be able to represent the whole data, so it is not necessarily gathering related attributes in each reduct. There also may be more than one reduct which contain the attribute with missing value so it is not clear which one to use. According to Du et al. (2011), when labeled data is limited it is impossible to find the underlying relationships between attributes, therefore the reducts and the core would not be reliable.

We address these problems in this research by proposing a new approach, which uses attribute clustering to limit the feature space to the related attributes. In other words, we are changing the definition of closest fit to be the closest instance from the pool, considering the subset of related attributes to the one with missing attribute-value. In first step, attribute clustering is used to create clusters of related attributes. Then, in order to find a missing attribute-value, the cluster which the attribute belongs to will be used as the feature set to find the closest fit. This will remove the irrelevant attributes while drastically decreasing the computational time. When there is not enough labeled instances available to perform supervised attribute clustering, the proposed semi-supervised framework in chapter 3 would be used. The result of attribute clustering would be used in each iteration to improve the classifier by improving the missing attribute value handling and the classifier will be used in each iteration to improve the attribute clustering by augmenting its training set. Therefore the result would be a better missing attribute value handling and also an improved semi-supervised classifier.

Missing Attribute-value Handling at Prediction Time

There exist four approaches for missing attribute-value handling at prediction time. The simplest missing attribute-value handling method at prediction time is to **discard the instances with missing attribute-values**. This means that we decline to provide prediction for some of the instances which may be problematic. Another approach is to **obtain the missing value by paying the cost** to run some complimentary tests or to get it from a third party. This method may not be applicable in some cases or the cost may not be acceptable. **Imputation-based methods** are a group of methods which handle missing attribute-values, at prediction time, by replacing it with an estimation of its value or its distribution from the data. According to Hastie et al. (2001), imputation-based methods are the most common approaches used for handling missing attribute-values. Another approach is to apply a specific prediction model for each instance based on the available attributes, its pattern of missing values. Saar-Tsechansky and Provost (2007) called such approaches as **reduced-feature models**.

Their empirical evaluation showed that, although imputation methods are the most common, reduced-feature models have substantially better predictive performance. The difficulty of reduced-feature models is that for each instance with a particular pattern of missing values, a different model should be applied. The models can be created on-line, which involves computation time, or precomputed and stored, which involves storage of different models which is exponential in the number of attributes.

Saar-Tsechansky and Provost (2007) tried to address this problem by proposing a hybrid method that uses the reduced-feature models for frequent patterns of missing values, and imputation for other cases. It allows the user to manage the tradeoff between predictive performance and storage/computation cost. Their empirical results show that when reduced-feature models are used, even for a few patterns, it improves the predictive performance substantially. This method still does not offer the use of reduced-feature model for all the cases and also the maximum number of precomputed models is still exponential in the number of attributes.

To reduce the number of attributes and consequently the number of precomputed models for reduced-feature models, Hong et al. (2009b) proposed an unsupervised attribute clustering approach for feature selection. The number of possible models, in this approach, is still exponential to the number of all selected attributes. Therefore, if the number of selected attributes is large, it is still problematic. If the number of selected attributes is small and the dataset has many missing attribute-values, it is problematic because in some cases very few attributes, of the selected ones, may contain values.

To address this problem, Hong et al. (2009b) suggested replacing selected features with missing attribute-values, with another from the same cluster. This replacement will help to create effective feature sets. However, the feature replacement increases the number of possible models. The number of possible models with replacement is the n-combination of attribute set. The size of each model is also based on the size of all selected attributes. If the number of selected attributes is high this model will still have high storage/computation cost. The attribute clustering method in this approach is unsupervised reducing its effectiveness, in terms of placing attributes into related clusters.

We address these problems in our proposed semi-supervised attribute clustering framework, when multi-view learner is used as classifier and attribute clustering is used to create the views for multi-view learner automatically. Each attribute of a view comes from a cluster of related attributes. If there is a missing value for such an attribute, it could be replaced with an attribute from the same cluster in that view. This replacement will only affect the learner corresponding to that view and that is the only part of the model which will be changed. The splitting of the selected attributes into views, in our approach, significantly reduces the number of possible patterns and consequently the number of models which need to be precomputed and stored for reduced-feature model.

A more detailed explanation of the performance of existing approaches and our new solution is provided in Improving Missing Attribute-value Handling chapter.

2.4 Summary

Although the various publications, workshops, surveys and usages of attribute clustering, show the great appeal for more accurate attribute clustering approaches, most of the research in this domain focused on unsupervised and supervised attribute clustering. When the labeled data is limited, supervised attribute clustering methods are unreliable. Unsupervised methods, as an alternative, are not able to extract the class-based relationships between attributes and existing semi-supervised methods have significant limitations. In this research we address them by proposing a new improved semi-supervised attribute clustering method. We use it to solve the automatic view definition problem for multi-view learning and to improve missing attribute value handling at induction and prediction time.

Chapter 3

Semi-supervised 'Attribute Clustering'/Classification Framework

This chapter starts with the vocabulary helpful in defining attribute clustering. Then, a definition of attribute clustering and its different types is provided and its geometric interpretation is discussed using a simple example. The next section discusses the weaknesses of existing semi-supervised attribute clustering methods. Finally, we describe the semi-supervised attribute clustering framework, and its underlying algorithm, and how it addresses these weaknesses.

Data, fundamental in machine learning tasks, could be represented in several ways. One of the most common is as a matrix. Each row represents an **instance** and each column an **attribute** or a **feature**. Each cell in the matrix represents an **attribute value**. The **label** or **class** attribute represents a common characteristic between some of the instances and more often is the one we want to predict its value for new instances. If the label is included in the data we call the data **labeled** and the learning tasks **supervised**.

Data classification is the assignment of labels to the upcoming new instance, based on the relations found in a labeled training set. It is used in many problems like diagnosis of a disease based on information that we have about previously observed patients and healthy people. Disease symptoms are the attributes in this problem.

In some learning problems we do not have any information about the class and the label set is unknown. An interesting task in such problems is to find out different subsets of similar instances called clustering. Clustering algorithms are used in many applications. They could be used to cluster patients to find out different unknown types of a disease. For example Paykel (1971) used data clustering to cluster a heterogeneous sample of 165 depressed patients with 35 rating variables into 4 clusters. The clusters are then specified to contain 4 types of depressed patients as anxious, hostile, retarded psychotic and young depressive. Many distance/similarity measures have been proposed for clustering. Using appropriate distance/similarity measure is very important in clustering results and critical to accurate data analysis (Cha, 2007; Choi et al., 2010).

In this thesis the focus is on clustering attributes rather than instances. **Attribute clustering** is the assignment of attributes in a learning task to some subsets in a way that similar attributes are gathered in the same cluster. Clustering attributes in machine learning tasks, specifically the ones which suffer from curse of dimensionality, has recently been of keen interest to researchers. Many different methods, applied to many different problems, have been studied and published so far.

Attribute clustering methods could be categorized as

- supervised attribute clustering
- unsupervised attribute clustering
- semi-supervised attribute clustering

The difference between **supervised and unsupervised attribute** clustering is whether the instances, used to measure the distance between attributes, are labeled or not. Saying attribute clustering is supervised may seem at first a little counterintuitive. We typically think of clustering as the archetype of unsupervised learning algorithms. However, it is important to notice that the class or label which we are talking about is the label of instances not attributes. We do not have any information about label that could be assigned to attributes. So, supervised attribute clustering is a clustering approach, which clusters attributes, using labeled instances to find out dissimilarity between attributes.

Semi-supervised attribute clustering could be an improvement of supervised attribute clustering, enriching the small labeled training sets to better fit the inherent structure. Most of the work in this field are concentrated on supervised and unsupervised attribute clustering. The need for more effective semi-supervised attribute clustering methods motivated us to propose a new iterative semi-supervised attribute clustering method in this study.

3.1 Geometric Interpretation of Attribute Clustering

Let us reiterate that attribute clustering is different from data clustering. Data clustering is an unsupervised process (there is no class definition) which clusters similar data instances, based on a similarity measure between different instances. Attribute clustering, on the other hand, finds clusters of similar or related attributes. Attribute clustering methods are similar to conventional data clustering but they are not the same. One difference is that these methods cluster attributes based on instances instead of clustering instances based on attributes. Another difference is that they usually do not use conventional measures like Euclidean or Manhattan distance. Attributes may be numerical or categorical and it is hard to compare them if they have different formats. Even if the attributes are normalized such distance measures may not provide enough information to capture the dissimilarity between attributes. When we have more instances, the dimensionality of attribute clustering increases and data become increasingly sparse. Distance measures become decreasingly informative. In fact, at very high dimensions data points are so spread out that they are almost equidistant from each other (Niu et al., 2008). This is the reason why many new similarity measures have been proposed. Such measures are correlation-based measures like correlation co-efficient (Domany, 2003), information based measures like mutual information (Au et al., 2005), relative dependency (Hong and Liou, 2007) and Kullback-leibler divergence (Vinh and Phuong, 2008) and many other similar measures as alternatives. In this section we illustrate the geometric interpretation of attribute clustering with a simple example.

Let us consider a data set in which each instance captures the marks for a particular student on different courses. So attributes are courses and attribute values are marks. Suppose that we want to find some factors which predict the academic ability of students. Suppose that the courses are from the two categories of art and science but we do not know this a priori. So considering that some students are good at art and some at science we use attribute clustering to find two categories of courses. Using these two clusters of attributes we can determine if a student is good at art or is good at science. This is an example of attribute clustering.

We have two students Mary and Jane and we have the information given in table 3.1 about the marks which they have received for different courses. Do not forget that we consider that we do not have information about course categories and we want to find two categories of courses using attribute clustering. As we see in this table Mary is good

Table 3.1: Marks received by two students for 3 art courses and 3 science courses.

	Art1	**Art2**	**Art3**	**Science1**	**Sceince2**	**Science3**
Jane	90	85	80	20	35	25
Mary	50	45	60	45	50	60

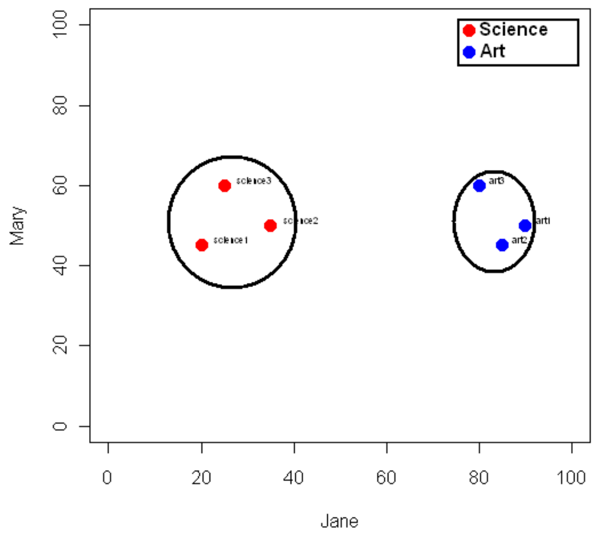


Figure 3.1: Illustration of attribute clustering for two instances of Jane and Mary and 6 attributes (courses)

at Science and Jane is good at Art. But let us consider, for simplicity, that we do not have this information and we want to do unsupervised attribute clustering. For clustering courses we need to use a distance/similarity measure to find out what courses should be clustered together. Let us for simplicity use Euclidean distance as the distance measure. In the new space each course will be represented as a point in the cartesian coordinate system while Jane will be one axis and Mary would be another. This is illustrated in figure 3.1.

If we use Euclidean distance as distance measure we could find the two clusters of courses, art and science, in this simple example. It is important to mention that in real attribute clustering problems this measure or similar measures do not work well and some similarity measures like correlation coefficient are used.

Now lets see a more complicated example with 2 new courses one from art and one from science category. In this example we show how using correlation coefficient could

Table 3.2: Marks received by two students for 4 art courses and 4 science courses.

	Art1	**Art2**	**Art3**	**Art4**	**Science1**	**Sceince2**	**Science3**	**Science4**
Jane	90	85	80	30	20	35	25	70
Mary	50	45	60	25	45	50	60	80

outperform the use of Euclidean distance. Consider we have the information given in table 3.2. We have 4 courses from art category and 4 courses from science category. Again Jane is good at art and Mary is good at Science. The attributes are illustrated in figure 3.2 and 3.3 while Euclidean distance and Correlation are used as distance and similarity measure respectively. Art courses are shown in these figures in blue and science courses are shown in red. As shown in figure 3.2, considering Euclidean distance as distance measure, we would cluster the new art course with science courses and new science course with art ones. The Euclidean distance of a new art course with all other courses is shown in figure 3.4. Although Jane is better than Mary in this course, Jane got 30 while Mary got 25, this course is closer to science courses than art courses. Figure 3.5 also shows the correlation between new art course and all other courses. In this two dimensional space correlation between any two science courses is 1, correlation between any two art courses is 1 and correlation between any art and any science course is -1. So using correlation in this example will give us better results.

It is very easy to split the space in this two dimensional space problem into two subspaces of courses which Jane is better than Mary and the one in which Mary is better than Jane by drawing the diagonal. For all points below Jane got better marks; for all points above Mary got better marks. So, as illustrated in figure 3.3, we may have two courses one at the very bottom left and one at the top right both below the diagonal and should be clustered together. However, if Euclidean distance is used as distance measure we will find them surprisingly far from each other and they will not be clustered together. Considering correlation as similarity measure they have correlation of 1 and they are perfectly correlated and will be clustered together.

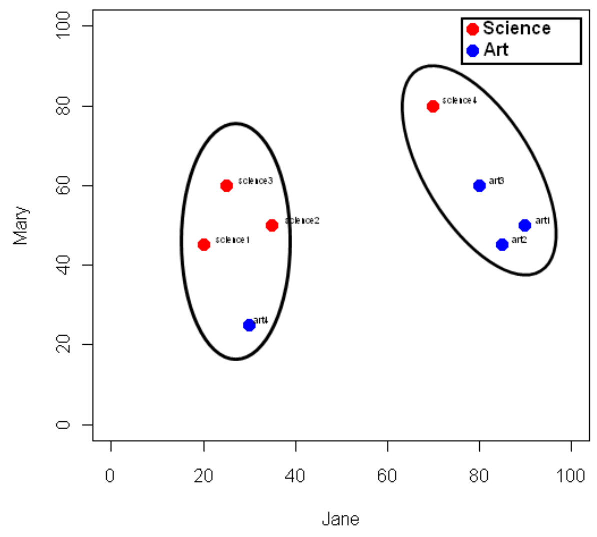


Figure 3.2: Illustration of attribute clustering, using Euclidean distance, for two instances of Jane and Mary and 8 attributes (courses)

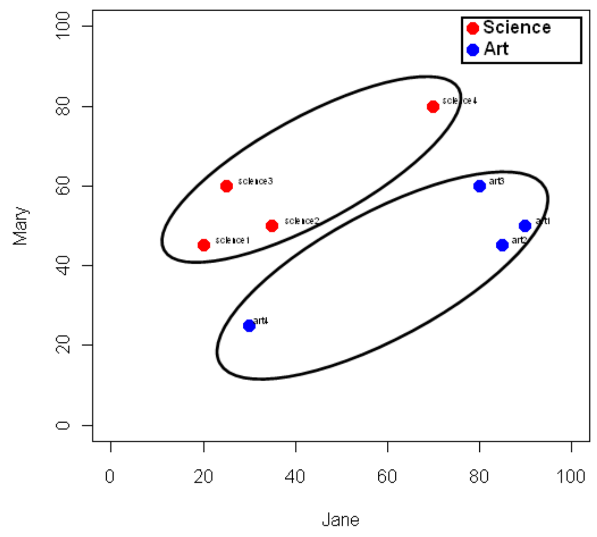


Figure 3.3: Illustration of attribute clustering, using correlation, for two instances of Jane and Mary and 8 attributes (courses)

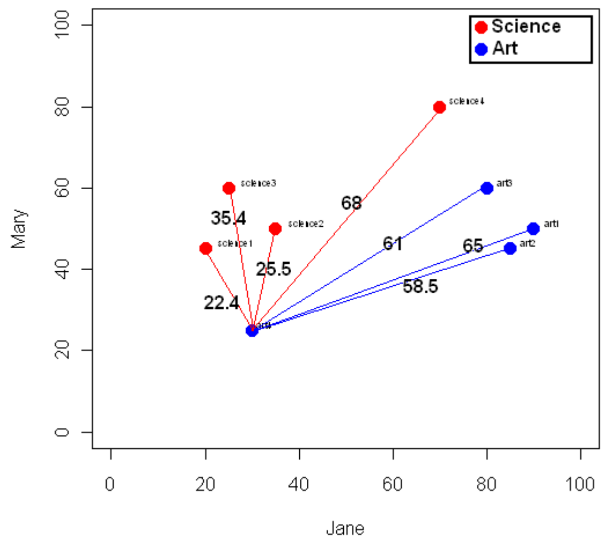


Figure 3.4: Euclidean distance, illustrated for two instances of Jane and Mary and 8 attributes (courses)

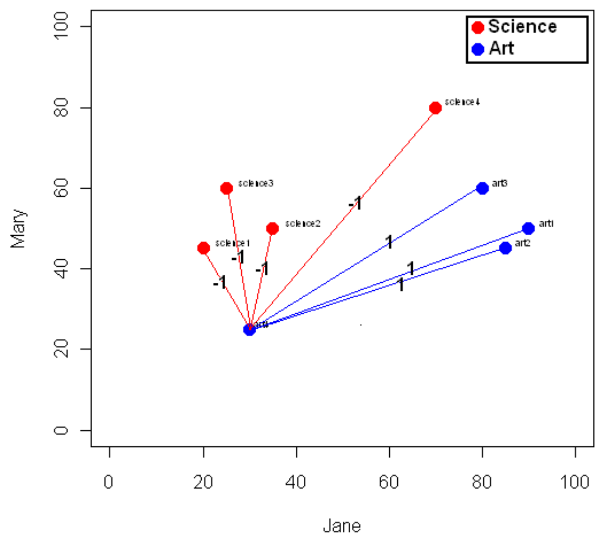


Figure 3.5: Correlation, illustrated as dissimilarity measure for two instances of Jane and Mary and 8 attributes (courses)

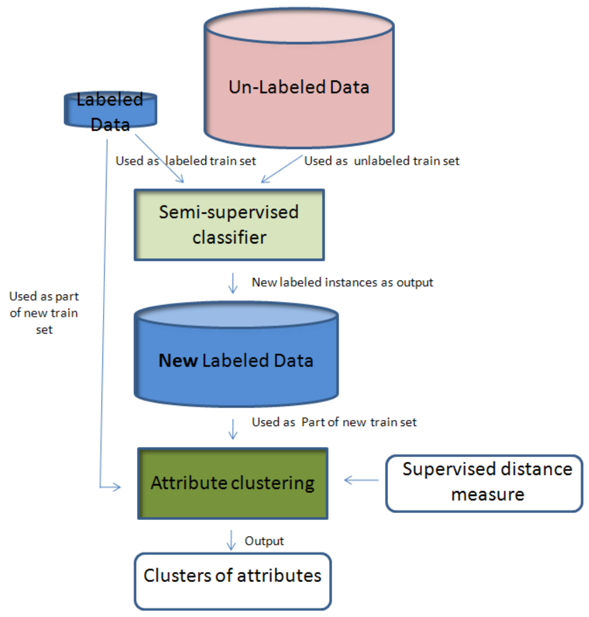

Un-Labeled Data
Labeled Data
Used as labeled train set
Used as unlabeled train set
Semi-supervised classifier
New labeled instances as output
Used as part of new train set
New Labeled Data
Used as Part of new train set
Attribute clustering
Supervised distance measure
Output
Clusters of attributes

Figure 3.6: One pass semi-supervised attribute clustering framework using a semi-supervised classifier.

3.2 Semi-supervised 'Attribute Clustering'/Classification Framework

In this section we propose our new framework to improve attribute clustering by performing iterative semi-supervised attribute clustering. First, let us review the existing semi-supervised attribute clustering methods in more detail.

The simplest way is to use unlabeled data with a semi-supervised classifier to increase the size of labeled set and then the use of new enlarged labeled set to do supervised attribute clustering. This framework is illustrated in figure 3.6. As we see in this figure, labeled data and unlabeled data are given as input to the semi-supervised classifier. Unlabeled data is labeled augmenting the training set used to cluster attributes. This framework is one pass. However, the classifier used in first step could be improved by the result of attribute clustering. Then if we improve the classifier we can get more accurate labeled instances and consequently we can improve the attribute clustering. This is one of the strengths of our new approach, it improves the classifier and the attribute clustering iteratively.

Another semi-supervised attribute clustering framework, proposed by Quinzán et al. (2009), uses a semi-supervised distance measure. This is a combination of supervised and

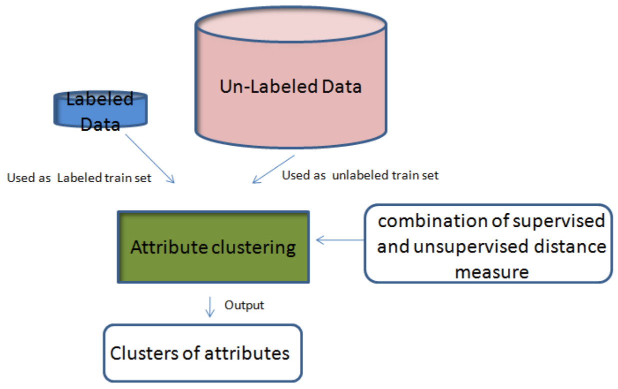

Un-Labeled Data
Labeled Data
Used as Labeled train set
Used as unlabeled train set
Attribute clustering
combination of supervised and unsupervised distance measure
Output
Clusters of attributes

Figure 3.7: One pass semi-supervised attribute clustering framework using semi-supervised distance measure

unsupervised distance measures applied on labeled and unlabeled data sets, respectively. This approach, illustrated in figure 3.7, is also a one pass method. As we see in this figure labeled and unlabeled data are given as input to the attribute clustering method. A supervised distance measure calculates the distance between every two attributes using the labeled data and considering the class label. An unsupervised distance measure uses the unlabeled data and calculates the distance between every two attributes without considering the class label. The attribute clustering method combines the two distances. The problem with this framework is that the distance calculated by supervised measure is much more accurate than the one calculated by unsupervised measure and therefore the unsupervised measure negatively affects the combination.

Another semi-supervised attribute clustering method is the one proposed by Frigui and Mahdi (2007). This method applies background knowledge as supervision information in form of a small set of constraints, on which attributes should or should not group in the same cluster. This framework is illustrated in figure 3.8. This framework could only be applied on problems for which we have both unlabeled data and background knowledge. If we have a few labeled instances this method is not able to exploit them.

All of these proposed approaches are one pass methods. In machine learning literature, it has been considered as a fact that using more labeled instances will lead to improvement of the results of learning tasks (Valiant, 1984; Lanquillon, 2000; Wu and Srihari, 2004; Ling et al., 2008; Wan et al., 2011). So the first framework, which labels the unlabeled samples first and increases the size of labeled training set for attribute clustering, is a good first step. But this method could be improved by using the result of attribute clustering, to improve the classifier used to add labeled data. We propose

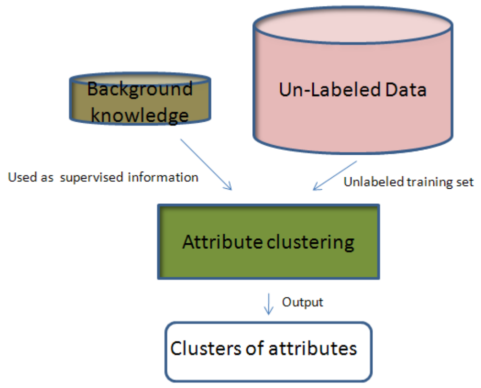


Figure 3.8: One pass semi-supervised attribute clustering framework using background knowledge

an iterative framework for attribute clustering in this study. This framework is shown in figure 3.9. As we see in this figure this framework has two main steps: Supervised classification and supervised attribute clustering. The combination of these two in an iterative process provides a semi-supervised classifier and also a semi-supervised attribute clustering. The labeled data is used to train a classifier in the first step. The classifier could be any of the existing classifiers like neural network (multi-layer perceptron), support vector machine, k-nearest neighbor, Gaussian mixture model, Naïve Bayes and decision tree. The classifier is used to label unlabeled data. The output is a set of new labeled instances. Further for each one we have a confidence value which tells how sure the classifier is about the class assigned to the instance. This is used to add a few most confident new labeled instances to the labeled training set.

Then, in second step the new labeled training set is used with a supervised attribute clustering algorithm to find more accurate clusters of attributes. This attribute clustering algorithm could be any of the existing ones. Any of the supervised similarity/distance measures could be selected for this step. It would be selected based on the data and the way we use its output to improve the classifier. The result of attribute clustering could be used in different ways to improve the supervised classifier. For example we could use these results to

- define multiple views if we are using a multi-view classifier
- perform feature reduction: selection or extraction
- handle missing attribute values, using related attributes from the same cluster, not the whole attribute space

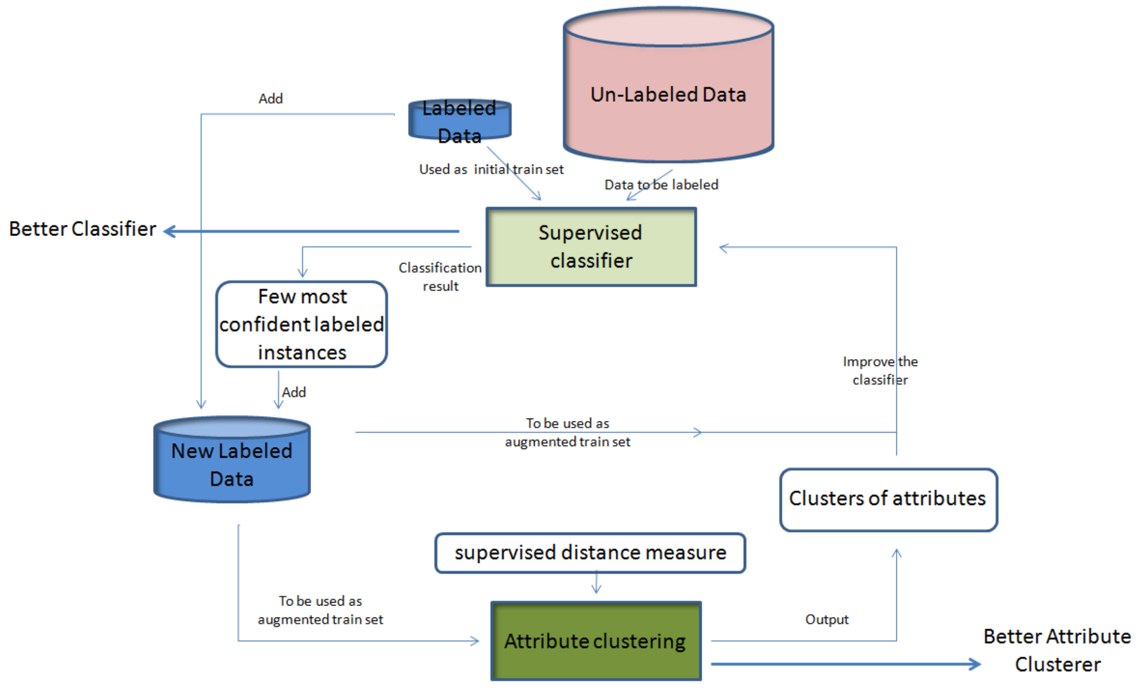


Figure 3.9: Iterative semi-supervised attribute clustering framework using supervised classifier and supervised distance measure.

The general algorithm for this semi-supervised attribute clustering framework is defined in algorithm 1. The **Input** includes labeled training set L with M instances and A attributes, Unlabeled training set U with N instances and A attributes and also Number of clusters of attributes, P. P is optional based on attribute clustering approach used in step 5 of the algorithm. **Output** of this framework includes both an improved attribute clustering and an improved classifier.

Algorithm 1 Proposed semi-supervised attribute clustering approach

```
1: Input Labeled training set $L$ with $M$ instances and $A$ attributes, Unlabeled training set $U$ with $N$ instances and $A$ attributes, Number of Clusters of attributes $P$ (optional)
2: repeat
3:     if First Iteration then
4:         Initialize the framework: If a multi- view Classifier would be used define the initial views of attributes, perform initial attribute selection, perform initial missing attribute value handling
5:     else
6:         Use new clusters of attributes $F$ to improve the classifier used in step 2 by Updating views of multi-view classifier or performing attribute selection or performing missing attribute value handling
7:     end if
8:     Train a classifier $C$ using the Labeled training set $L$ and the result of previous step.
9:     Classify unlabeled instances $U$
10:    Add $K$ most confident new labeled instances to the labeled set ($M = M + K, L = L \cup \{K$ most confident new labeled instance$\}$)
11:    Use new labeled set with an attribute clustering method to provide clusters of attributes $F$
12: until No new instances added
13: Output: The set of clusters of attributes and the classifier trained in the final iteration
```

In the first iteration of the algorithm, the framework is initialized in **line 4**. This step can include initialization of the classifier, attribute selection or missing attribute value handling. All these processes try to improve the result of classification. Since the labeled training set, used for this purpose, is very small at this step we augment the training set and run these processes iteratively in next steps of the framework. If this is not the first iteration of the algorithm the **line 6** uses the clusters of attributes F, obtained in previous iteration, to improve the classifier to be trained in next iteration by updating views of multi-view classifier or performing attribute selection or performing missing attribute value handling. In next step, in **line 8**, a classifier C is trained using the available labeled training set L and the result of previous part of the algorithm. The trained classifier is then used in **line 9** to classify unlabeled instances in U. The classifier provides a confidence for each new labeled instance which shows how much the classifier was confident about the label assigned to each instance. All the information provided in previous step is used in **line 10** to assign K most confident new labeled instances to the labeled set. After this step the size of labeled training set will increase to $M + K$ and the size of unlabeled training set will decrease to N - K. In **line 11** the augmented labeled training set is used by a supervised attribute clustering approach to provide clusters of attributes F. **Line 12** checks the stop condition of the iterative process. If the termination criterion is satisfied then the algorithm terminates and outputs the final set of clusters of attributes and also the improved classifier. Otherwise it will go to next iteration.

3.3 Summary

This chapter discussed the differences and similarities between the attribute and data clustering. The geometric interpretation of attribute clustering is illustrated using a simple example. The existing semi-supervised attribute clustering methods and their weaknesses are reviewed and finally our semi-supervised attribute clustering framework, to address their problems, is proposed. The result of attribute clustering could be used in different ways to improve the classifier. The proposed framework is a very general one. Different usages of this framework will solve different existing problems in machine learning. Next two chapters show different problems that are addressed using the proposed framework and the customization that needs to be done in the framework. The results include experiments using these customized frameworks and are compared with other approaches to show there are clear advantages.

Chapter 4

Attribute Clustering for Automatic View Definition in Multi-View Learning

Multi-view learning is based on the assumption that each hypothesis could be represented by different sets of attributes, called views, which look at the same problem from different angles. These attribute sets can be identified by people based on their knowledge of a domain or derived using different feature extraction algorithms. These multiple representations of a pattern are used to train different complementary learners which help each other in the learning process. Alternatively, an aggregation function is used to combine them to find the final hypothesis. Recent work has shown that utilizing the agreement between learners, based on different views, improves performance (Abney, 2002; Ando and Zhang, 2007; Blum and Mitchell, 1998; Chen et al., 2011). Yet, an effective methodology for creating effective views automatically from data sets has not been proposed. This is even more problematic in real world applications for which we do not have large labeled data sets (Du et al., 2011).

In this chapter we address this problem by combining attribute clustering with multi-view learning in an iterative semi-supervised process. The power of this approach comes from creating sufficient and diverse views during learning. This approach clusters the attributes based on their correlation in each iteration. This will help to create views which do not include highly correlated attributes and it will lead to a good spread of such attributes in different views. The result will be a good set of views which can be used to train effective multi-view learners. These learners will then be trained in each iteration and used to augment the training set used to refine the views in the next iteration.

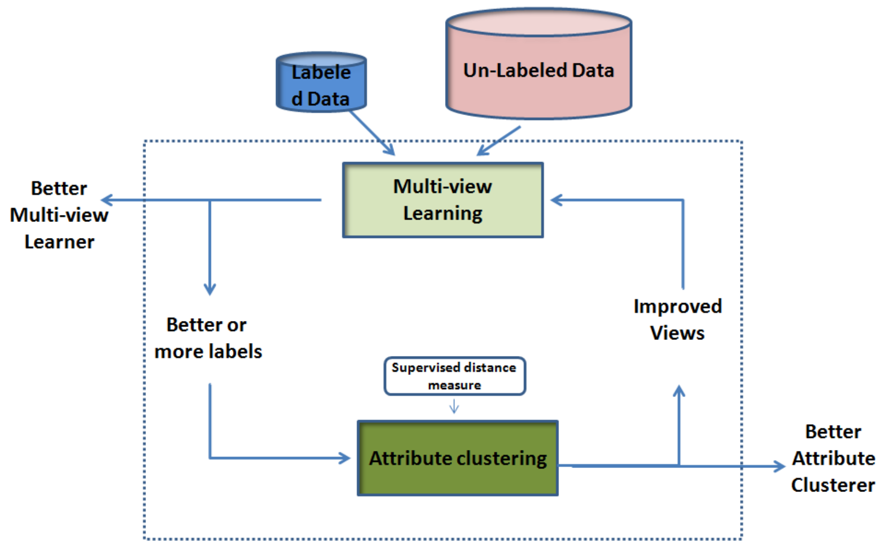
In many real world learning tasks there exist only very few labeled instances and a large number of unlabeled ones. It is fairly easy to find unlabeled instances while the labeling process is expensive, difficult or time consuming. For example the labeling process may need human insight or performance of expensive tests or experiments (Seeger, 2001). Most of the supervised learning algorithms do not work well when the labeled training set is small. In such cases we can use a small labeled data set together with a large unlabeled data set in a semi-supervised learning algorithm to improve the results.

Blum and Mitchell (1998) formalized multi-view learning for the first time by proposing co-training as a semi-supervised approach to multi-view learning. This approach is guaranteed to work well on data sets with two a priori disjoint sets of attributes, i.e. two sufficient and independent views. In many real world problems we do not have a priori views which meet these assumptions. This motivated many researchers to propose other multi-view learning algorithms and variations of standard co-training (Abney, 2002; Ando and Zhang, 2007; Balcan et al., 2004; Chen et al., 2011; Goldman and Zhou, 2000; Raskutti et al., 2002a; Zhou and Li, 2005) which weaken, change or remove these assumptions. A problem which has not been addressed is how to find such views when labeled data are rare, the automatic view definition problem. A simple way to generate multiple views is to randomly split the attributes (Bickel and Scheffer, 2004; Brefeld and Scheffer, 2004; Brefeld et al., 2005), which may lead to insufficient views which do not satisfy the assumptions required for learning (Xu et al., 2013). Du et al. (2011) designed several methods to split feature space into two views. However, their empirical results showed that, given small labeled training sets the view splitting methods are unreliable. Therefore, their proposed splitting methods could not be applied to the problem of automatic view definition for semi-supervised multi-view learning. Di and Crawford (2012) proposed active learning strategies to generate views for Hyperspectral Image Data when training set is limited. However their proposed active learning methods requires the true labels for the samples that the learner is the least confident about.

The view definition for multi-view learning can be cast as an attribute clustering problem. Attribute clustering is used to cluster similar or related attributes based on a similarity or distance measure. When the labeled training set is small it is impossible to measure the similarity between attributes so the supervised attribute clustering methods will not work well. In such cases a semi-supervised attribute clustering method is needed.

In this chapter we address the problem of automatic view definition for semi-supervised

Figure 4.1: Iterative semi-supervised multi-view learning combined with attribute clustering.



multi-view learning by combining attribute clustering with multi-view learning. In this approach, an initial split for the attributes, obtained by using attribute clustering on the original labeled data, will be used in the first iteration of the algorithm. In each subsequent iteration, each predictor will use one view to train and then weighted majority voting will be used to label unlabeled instances. Then the training set will be augmented by the most confident new labeled instances. At the end of each iteration, labeled data and new labeled instances will be used to refine views using attribute clustering.

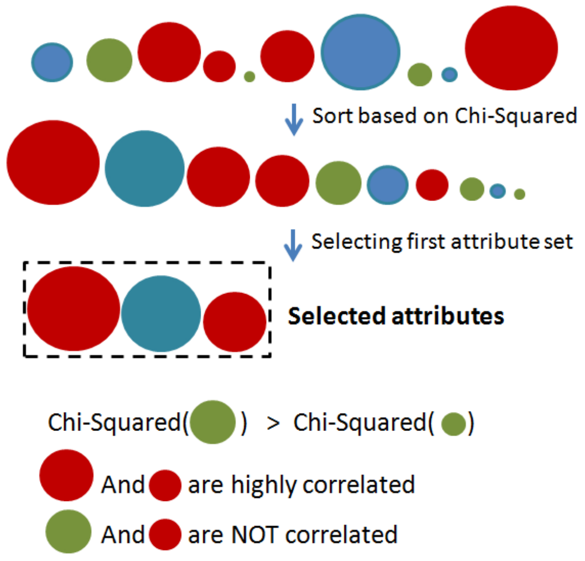
We evaluate our approach experimenting on real world data sets from two different domains, text data sets and microarray gene expression data.

4.1 Combining Multi-View Learning and Attribute Clustering

In this study we combine attribute clustering with multi-view learning to create an improved semi-supervised multi-view learning approach which creates the views automatically. This algorithm is illustrated in figure 4.1.

In each iteration, the results of attribute clustering will be used to improve the views. Then, the improved classifier will be used to improve attribute clustering by augmenting its labeled training set, with the most confident new labeled instances.

Figure 4.2: Illustration of feature selection methods which rank the attributes and select a few first ranked attributes. Red, green and blue show 3 different groups of correlated attributes and diameter of circles show the amount of information they provide for classification.

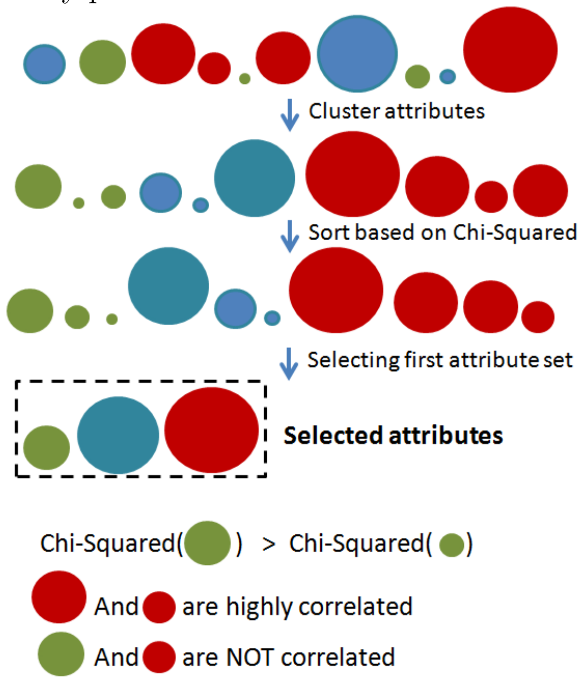

Sort based on Chi-Squared
Selecting first attribute set
Selected attributes
Chi-Squared( ) > Chi-Squared( )
And are highly correlated
And are NOT correlated

As we see in this figure, this approach has two main components: Supervised multi-view classification and supervised attribute clustering. The next two subsections discuss these two components in more detail and the last subsection shows how they interact with each other in the iterative process.

4.1.1 The Attribute Clustering Component

The lower half of figure 4.1 shows the usage of the attribute clustering component in the iterative framework. It uses the augmented training set to provide clusters of attributes. The results are then used by the multi-view learning component to refine the views. Pairwise partial correlation between two attributes given a third one, in this case the class, is used as the similarity measure for attribute clustering. It is used to generate a matrix which contains the partial correlation between every two attributes. The inverse of this similarity matrix is used as distance matrix, for attribute clustering. Agglomerative Nesting (AGNES) (Kaufman and Rousseeuw, 2009), which is a hierarchical clustering method, is used for attribute clustering in our experiments. The use of this clustering algorithm is not part of the proposed framework and any clustering algorithm which uses the similarity measure of interest could be used for this purpose. Agnes takes as input the

Figure 4.3: Illustration of feature selection using attribute clustering. Red, green and blue show 3 different groups of correlated attributes and diameter of circles show the amount of information they provide for classification.


Cluster attributes
Sort based on Chi-Squared
Selecting first attribute set
Selected attributes
Chi-Squared( ) > Chi-Squared( )
And are highly correlated
And are NOT correlated

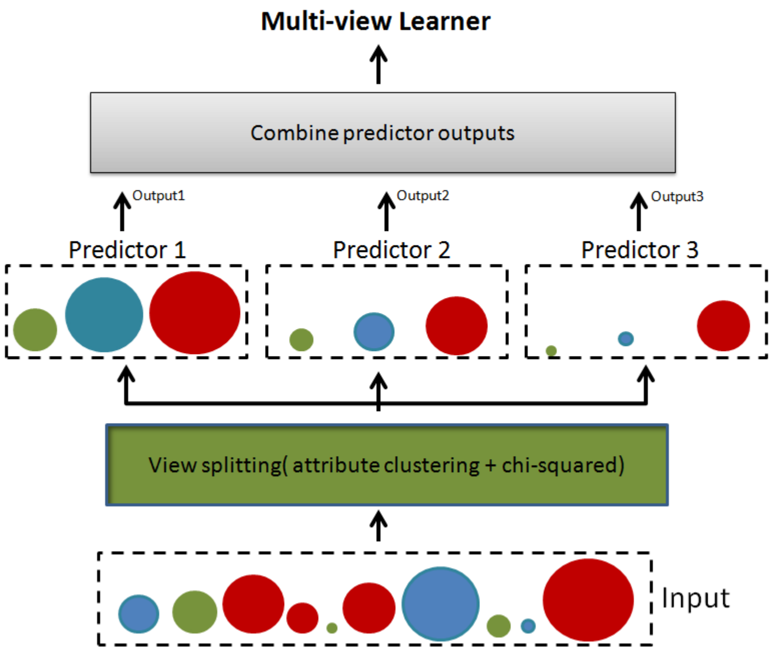
distance matrix, which in this case is calculated using correlation between attributes. It then returns hierarchical clustering results in the form of a tree. Therefore, any number of clusters, if needed, could be obtained by cutting the tree at different locations. In the first few iterations, when labeled data is limited, attribute clustering will be more like a random view splitting algorithm. Later, when the labeled training set is enriched, the attribute clustering results will improve.

4.1.2 The multi-View Learning Component

The result of attribute clustering in each iteration is used for refining the views used by the multi-view learner. The power of our approach comes from utilizing the largest number of discriminating views in multi-view learning. This is obtained by combining attribute ranking methods with attribute clustering to perform attribute selection for views.

Many feature selection methods rank the attributes based on the amount of information they provide for classification. They use a measure like chi-squared or information

Figure 4.4: View splitting process to create multi-view learner.



gain to sort the attributes and then select those which provide the most information. However, if two selected attributes with good ranks are highly correlated, then there is no point in selecting both. Figure 4.2 shows this problem. The circles represent attributes. The diameter of the circles represent the amount of information they provide and red, green and blue show three different groups of correlated attributes.

In our approach the result of attribute clustering is used to prevent this problem in feature selection and improve the multi-view classifier. As illustrated in figure 4.3, after attribute clustering, the attributes in each cluster are sorted based on the chi-squared measure in decreasing order. Then, the first attribute from each cluster is selected to create a feature set for the first view. So the first individual view classifier picks up the most useful attribute from each cluster and it will be the best individual view classifier, but we need to use more classifiers for voting. Generally, when creating the jth view, we will pick the ranked j attribute from each cluster. If a cluster k has less than j attributes, the attribute with highest Chi-Squared value from the same cluster will represent the cluster k in the jth view. This enables our approach to create overlapping views if needed.

As we go towards the last attribute in each cluster the individual view classifiers will get weaker and weaker because the attributes which remain in each cluster are the least useful ones for classification. Therefore, at some point the rest will not be good enough to be added to the multi-view classifiers. In order to find out if a created view is useful we use a function f to calculate the usefulness of the given view:

$$f(att_i) = \frac{\sum_{i=1}^{n} ChiSquared(att_i, class)}{n} \tag{4.1}$$

Where att_i represents the attribute with index i, n is the number of attributes in the created view, *class* is the class label of the instances and ChiSquared calculates the Chi-Squared measure between att_i and class. For each individual view the value of this function is compared with a threshold, 0.2, which is obtained empirically, to decide whether the view is useful or not. The minimum number of views used to train the multi-view classifier is two. Therefore, when the number of views which pass the threshold is less than two, the two views with highest chi-squared measure will be used to train the multi-view classifier. Consequently the number of views, used by the multi-view classifier, could be different in each of the iterations. Figure 4.4 shows the view splitting procedure to create our multi-view learning approach.

4.1.3 Proposed Algorithm

The algorithm for this semi-supervised attribute clustering approach is shown in algorithm 2. It receives, as input, a labeled training set L with m instances and a attributes and an unlabeled training set U with n instances and a attributes. This algorithm has two main steps, the classification and attribute clustering.

In the first iteration of the algorithm, the initialization in line 4 obtains an initial view split by using attribute clustering with original labeled data. Then the iterative process starts by training the multi-view learner on the labeled set in line 8. The multi-view classifier is used to label unlabeled data in line 9. The output of this step is a set of new labeled instances paired with a measure of confidence in the new label. This confidence is used in line 10 to add the most confident new labeled instances to the labeled training set. Then, in line 11, the new labeled training set is used with the supervised attribute clustering algorithm to find more accurate clusters of attributes. The output of the attribute clustering algorithm is then used in the next iteration, in line 6, to refine the views and improve the multi-view classifier.

The algorithm iterates until all the unlabeled data is used to augment the training set or there is no more confident new labeled instances to be added to the labeled training set. At the end, the algorithm outputs the views and the classifier which is trained on

Algorithm 2 Proposed semi-supervised attribute clustering approach

```
1: Input Labeled training set $L$ with $M$ instances and $A$ attributes, Unlabeled training set $U$ with $N$ instances and $A$ attributes, Number of Clusters of attributes $P$ (optional)
2: repeat
3:     if First Iteration then
4:         Define the initial views of attributes
5:     else
6:         Use new clusters of attributes $F$ to refine views
7:     end if
8:     Train a multi-view classifier $C$ on the Labeled training set $L$ using views
9:     Classify unlabeled instances $U$
10:    Add $K$ most confident new labeled instances to the labeled set ($M = M + K, L = L \cup \{K$ most confident new labeled instance$\}$)
11:    Use new labeled set with an attribute clustering method to provide clusters of attributes $F$
12: until No new instances added
13: Output: The set of clusters and the classifier trained in the final iteration
```

the final iteration.

4.2 Evaluation

In this section the proposed algorithm is evaluated using real text data sets and microarray gene expression data. AUC is used as evaluation measure to compare our approach with others. Our results show that the performance of our approach tends to be close to that of the ideal supervised classifier, which has access to all the data labeled, while it is statistically significantly better than that of the other approaches.

4.2.1 Data Preparation

The text data sets that we have used come from two different sources. The first group introduced by Cohen et al. (2006) in their research on automated classification of document citations for systematic review of drug classes[1]. The original data sets are in form of sets of text documents, papers. We use Term Frequency-Inverse Document Frequency (TF-IDF) to prepare text data for our experiments. TF-IDF is a numerical statistic which demonstrates the importance of a word in an individual document, considering all the documents in the collection. We extracted TF-IDF from each document, considering 1-grams and 2-grams of the title, abstract of the paper, main header and sub header keywords. The stop words are removed from 1-grams. These data sets are binary and strongly imbalanced.

The second group is obtained from the Weka data set website (Forman and Packard)[2]. The problems come from LA Times, TREC, OHSUMED, etc. These data sets are originally multi-class (1-of-n) text data sets, whose word count feature vectors have already been extracted by Han and Karypis (2000). We extracted two-class data sets from them which are fairly imbalanced.

We have also evaluated our approach on microarray data using GEMLeR data sets [3] (Stiglic and Kokol, 2010) which include a collection of gene expression data sets. This repository consists of gene expression data from 1545 tumor samples which are distributed into nine tumor tissue types, the class labels. GEMLeR data sets are divided in two sections, 'one-versus-all' (OVA) which includes a data set for each tumor tissue type

[1]This data is available at http://skynet.ohsu.edu/cohenaa/systematic-drug-class-review-data.html

[2]This data is available at http://weka.wikispaces.com/Datasets

[3]This data is available at http://gemler.fzv.uni-mb.si/index.php

versus all other types, and ‘all-paired’ (AP) which includes a data set for every pair of tumor tissue types. We have used all of these data sets in our experiments.

4.2.2 Competitor Algorithms

The proposed algorithm (ours) is compared with seven different competitors:

- **The Simple Learner (Simple)** is a supervised classifier which only has access to the labeled data. In each experiment the base classifier is trained on the labeled training set and then tested using the test set. The number of the labeled instances, in our experiments, is very limited, similar to real world problems, so this classifier would represent the lower boundary of learners.

- **Random Split (Random)** is an unsupervised splitting method, introduced by Du et al. (2011), which randomly splits the attributes in two views. The resulting views are then used to train co-training in our experiments.

- **Entropy Split (Entropy)** is a supervised splitting method, proposed by Du et al. (2011), which uses entropy, a numeric measure that shows how predictive of the class the feature would be. This method sorts the attributes based on their entropy and then assigns features with odd and even indexes to the first and second views respectively.

- **Co-training Style (Co-Style)** is a semi-supervised classifier, proposed by Goldman and Zhou (2000), which is similar to co-training but removes the need for the views. It uses two different learners that are both trained on the original training set, instead of two separate views.

- **Self-Training (Self)** (Zhu, 2006) is a commonly used semi-supervised classifier which iteratively trains a base classifier with the labeled training set, and then classify the unlabeled data and augment the labeled training set with most confident new labeled instances. This process continues until all the unlabeled data is used or there is no more confident labeled instances.

- **Unsupervised Split (UnSup)** uses an unsupervised similarity measure (Pairwise correlation between two attributes) to create the views. The views are then used to train co-training.

- **Our approach (Ours)** is proposed in section 4.1. We used algorithm 2 together with the view definition approach that we presented in section 4.1.
- **Ideal Learner (Ideal)** is a supervised classifier which has access to the correct label of all the data. The classification results of this classifier are the best that a semi-supervised classifier can get because this classifier has all the data, including the training set and the unlabeled set correctly labeled. So this classifier represents the upper boundary of possible classifiers. We propose our approach to get performance as close as possible to this ideal, yet unrealistic, classifier.

4.2.3 Configuration

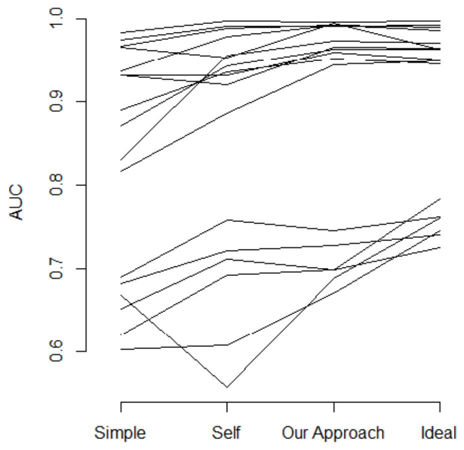
In semi-supervised learning, the smaller the labeled training set is, the more challenging the classification problem. Some of the proposed semi-supervised approaches do not work well, when the labeled training set is very small. That is why many researchers used very small labeled training sets, 2, 3, 4, 5 instances in references (Sindhwani and Rosenberg, 2008), (Lv et al., 2012), (Brefeld and Scheffer, 2006) and (Jiang et al., 2008) respectively, to show the reliability of their semi-supervised approaches. In our experiments we used 3, 5 and 10 labeled instances as training sets to explore the effect of the size of the labeled training set on our approach. The training instances are randomly selected, considering the distribution of the class labels in the original data sets, with the constraint that at least one instance from each class should be included in the training set.

Text and microarray data sets contain thousands of attributes, including many which do not have any helpful information for classification. Therefore, the classifiers applied on the whole feature set are not able to learn the pattern in the data and consequently are close to the random classifier. This problem is usually solved by selecting a subset of attributes using a feature selection algorithm (Jiang et al., 2004; Raskutti et al., 2002b). In our experiments chi-squared measure is used to reduce the size of the problem. We used 50 and 100 attributes in different experiments to compare the effect of the size of the selected feature set.

To explore the effect of the base classifier on the results, Naïve Bayes, as a simple Bayes classifier, and J48, as a representative of decision trees, are used in two different set of experiment.

We consider the experiment with 5 labeled training instances, 50 attributes and Naïve Bayes as the base classifier, to be our benchmark experiment. Then we compare benchmark experiment with new experiments with the same setting but different numbers of

Figure 4.5: Parallel coordinates plot for benchmark experiment.



training instances, different numbers of attributes or different base classifiers, to see the effect of these changes.

The average AUC has been calculated in each experiment, using five-fold cross validation. Five-fold cross validation is used because the number of instances is limited in these experiments. As Japkowicz and Shah (2011) suggested in their book, Friedman's test with 95% confidence interval, is applied to see if the proposed algorithms are significantly different in terms of AUC and the Nemenyi test is used as post hoc test to find out what these differences correspond to.

4.2.4 Results and Discussion

Table 4.2 shows the result of our benchmark experiment. Table 4.1 and 4.3 show the results of experiments with the same setting but different numbers of training instances (3 and 10). The first part of these tables includes average AUC of each approach on the text data sets over the five-fold cross-validation. In all these three experiments the average AUC of our approach is better than that of the other approaches, while it is close to that of the classifier produced by the ideal learner discussed earlier. These tables indicate that the average AUC of each learner increases when the size of the labeled training set increased. This is quite normal because more labeled instances tend to generate better results. As we see in these tables, when increasing the size of the training set, the changes in the average AUC of our approach is less than that of the other learners while the average AUC of our approach is always higher. It means that our approach is not as sensitive as that of the simple learner to the size of the training set, in terms of AUC.

Figure 4.5 illustrates the average AUC of each approach using a line for each data set, for benchmark experiment, on a parallel coordinates plot. This shows that the AUC of our approach is almost always higher than the simple approach and self-training while it is close to the ideal, yet unrealistic, supervised classifier. As mentioned in section 4.2.1 the data sets used in these experiments are all text data sets but they come from two different sources. The first group introduced by Cohen et al. (2006) and the second group is obtained from the Weka data set website (Forman and Packard). The former data sets are more imbalanced, therefore the classification task on them is more challenging. That is the reason of the significant difference between the AUC of these two groups of data sets shown in figure 4.5.

In order to find out if these results are statistically significantly different, we used Friedman's test with the null hypothesis being that the classifiers being compared are alike. The null hypothesis rejected by p-value=1.469e-05, 6.955e-05 and 6.104e-05 on the results of our experiments using 3, 5 and 10 labeled instances respectively. So we used Nemenyi test as a post hoc test to find out where the difference come from.

The second part of tables 4.1, 4.2 and 4.3 show the p-values from the Nemenyi test. We use 95% as confidence interval so each p-value less than 0.05 shows that the two compared methods are statistically significantly different.

The result of the post hoc tests for all three experiments, with different numbers of labeled instances, are the same. Our approach and the ideal one, compared with every other existing method individually, are statistically significantly better in terms of AUC. Comparing our approach to the ideal classifier, which has access to the labels for both the labeled training set and the unlabeled set, the average AUC is lower but this difference is not statistically significant. This means that although our new approach is not as good as the ideal classifier, this difference is not significant. In reality, this ideal classifier is not available. We use semi-supervised approaches as an alternatives to get similar results.

All these comparisons show that the performance of our approach tends to be close to the ideal classifier, while it is statistically significantly better than all other approaches. These tables also show that the Random Split, Entropy Split and Co-training Style methods are not statistically significantly different from each other as well as the Self-training and Unsupervised Split.

The same setting as the benchmark experiment, except that the J48 is the base classifier, is used in an experiment to explore the effect of the classifier on the results. The result of significant tests on this experiment, shown in table 4.4, is the same as that of

Table 4.1: Average AUC and P-values calculated by Friedman's and Nemenyi test on Text data sets (Setting: **3 labeled instances**, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
fbis(111,142)	0.87	0.929	0.93	0.937	0.991	0.995	**0.997**	0.995
fbis(111,189)	0.964	0.985	0.983	0.939	1	1	**0.999**	1
fbis(142,189)	0.941	0.93	0.922	0.928	0.974	0.963	**0.998**	0.973
new3s(111,142)	0.898	0.907	0.926	0.944	0.98	0.996	**0.996**	0.998
new3s(189,142)	0.872	0.919	0.923	0.906	0.932	0.976	**0.992**	0.987
new3s(189,301)	0.929	0.787	0.801	0.873	0.938	0.928	**0.961**	0.957
ohscal(An,Ca)	0.719	0.899	0.889	0.751	0.941	0.961	**0.974**	0.975
ohscal(An,Dn)	0.772	0.781	0.736	0.768	0.843	0.846	**0.943**	0.946
ohscal(Ca,Dn)	0.781	0.901	0.879	0.748	0.94	0.936	**0.963**	0.961
ohscal(Ca,To)	0.718	0.776	0.747	0.699	0.924	0.929	**0.958**	0.944
ACEInhibitors	0.702	0.694	0.704	0.692	0.7	0.741	**0.763**	0.755
ADHD	0.886	0.876	0.921	0.856	0.956	0.939	**0.959**	0.96
AtypicalA	0.633	0.627	0.636	0.622	0.658	0.689	**0.663**	0.709
ProtonP	0.654	0.672	0.693	0.636	0.701	0.779	**0.783**	0.786
Statins	0.663	0.602	0.643	0.73	0.654	0.672	**0.728**	0.802
Average	**0.8**	**0.819**	**0.822**	**0.802**	**0.875**	**0.89**	**0.912**	**0.917**

Methods	**P-value**
*Ours - Co-Style	5.821481e-06
*Ours - Entropy	4.805973e-03
*Ours - Random	1.381133e-03
*Ours - Simple	7.303006e-05
Entropy - Co-Style	5.617145e-01
Random - Co-Style	7.771378e-01

Methods	**P-value**
Ours - Ideal	0.9916245971
*Ours - Self	0.0249834276
*Ours - UnSup	0.0008691466
Self - *Ideal	0.0103920166
UnSup - *Ideal	0.0002028328
UnSup - Self	0.7433663827

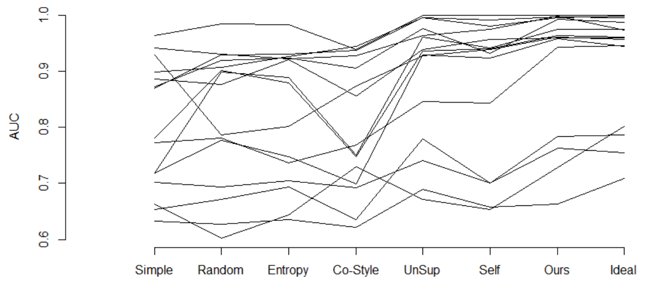


Table 4.2: Average AUC and P-values calculated by Friedman's and Nemenyi test on Text(**Benchmark Setting**: 5 labeled instances, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
fbis(111,142)	0.962	0.921	0.931	0.918	0.991	0.993	**0.993**	0.995
fbis(111,189)	0.991	0.994	0.992	0.992	0.997	1	**1**	1
fbis(142,189)	0.965	0.932	0.91	0.933	0.979	0.966	**0.997**	0.973
new3sc(111,142)	0.961	0.93	0.928	0.94	0.993	0.995	**0.994**	0.998
new3s(189,142)	0.907	0.935	0.923	0.893	0.979	0.975	**0.994**	0.987
new3s(189,301)	0.931	0.8	0.828	0.894	0.941	0.851	**0.969**	0.957
ohscal(An,Ca)	0.864	0.924	0.925	0.809	0.962	0.962	**0.974**	0.975
ohscal(An,Dn)	0.862	0.812	0.831	0.773	0.92	0.917	**0.946**	0.946
ohscal(Ca,Dn)	0.896	0.881	0.903	0.849	0.965	0.944	**0.966**	0.961
ohscal(Ca,To)	0.859	0.824	0.791	0.837	0.935	0.925	**0.955**	0.944
ACEInhibitors	0.706	0.696	0.69	0.69	0.723	0.741	**0.754**	0.755
ADHD	0.953	0.92	0.92	0.921	0.958	0.935	**0.96**	0.96
AtypicalA	0.598	0.633	0.625	0.603	0.631	0.657	**0.68**	0.709
ProtonP	0.704	0.713	0.72	0.629	0.647	0.784	**0.776**	0.786
Statins	0.65	0.656	0.645	0.622	0.63	0.671	**0.621**	0.802
Average	**0.854**	**0.838**	**0.837**	**0.82**	**0.883**	**0.888**	**0.912**	**0.917**

Methods	**P-value**
*Ours - Co-Style	8.691290e-06
*Ours - Entropy	9.753817e-04
*Ours - Random	3.643670e-03
*Ours - Simple	4.267348e-02
Entropy - Co-Style	8.615484e-01
Random - Co-Style	6.695270e-01

Methods	**P-value**
Ours - Ideal	0.914155300
*Ours - Self	0.023989818
*Ours - UnSup	0.012114262
Self - *Ideal	0.002920296
UnSup - *Ideal	0.001217282
UnSup - Self	0.996343168

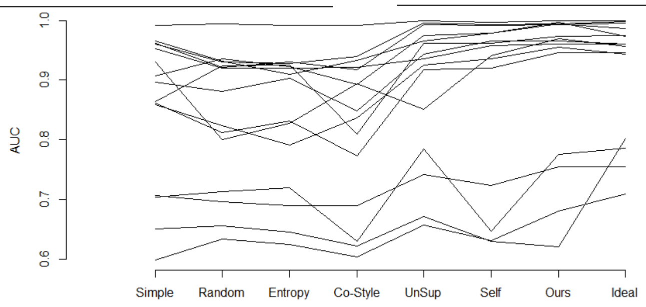


Table 4.3: Average AUC and P-values calculated by Friedman's and Nemenyi test on Text data sets (Setting: **10 labeled instances**, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
fbis(111,142)	0.988	0.918	0.927	0.93	0.996	0.993	**0.998**	0.995
fbis(111,189)	0.998	0.99	0.993	0.992	1	1	**1**	1
fbis(142,189)	0.975	0.924	0.933	0.931	0.989	0.966	**0.995**	0.972
new3s(111,142)	0.991	0.929	0.921	0.931	0.997	0.994	**0.994**	0.998
new3s(189,142)	0.968	0.923	0.93	0.911	0.991	0.977	**0.993**	0.987
new3s(189,301)	0.939	0.84	0.882	0.885	0.953	0.923	**0.964**	0.957
ohscal(An,Ca)	0.885	0.927	0.929	0.847	0.946	0.959	**0.974**	0.975
ohscal(An,Dn)	0.904	0.866	0.858	0.857	0.918	0.937	**0.946**	0.946
ohscal(Ca,Dn)	0.906	0.892	0.898	0.858	0.956	0.946	**0.972**	0.961
ohscal(Ca,To)	0.905	0.84	0.824	0.866	0.925	0.924	**0.955**	0.943
ACEInhibitors	0.707	0.682	0.681	0.685	0.715	0.709	**0.755**	0.755
ADHD	0.951	0.92	0.92	0.923	0.956	0.935	**0.963**	0.96
AtypicalA	0.611	0.629	0.62	0.595	0.663	0.656	**0.672**	0.709
ProtonP	0.617	0.71	0.692	0.634	0.689	0.788	**0.789**	0.786
Statins	0.654	0.658	0.662	0.652	0.71	0.736	**0.728**	0.802
Average	**0.867**	**0.843**	**0.845**	**0.833**	**0.894**	**0.896**	**0.913**	**0.917**

Methods	**P-value**
*Ours - Co-Style	2.117069e-06
*Ours - Entropy	8.262105e-05
*Ours - Random	6.659463e-06
*Ours - Simple	4.388736e-02
Entropy - Co-Style	9.440947e-01
Random - Co-Style	9.984764e-01

Methods	**P-value**
Ours - Ideal	0.847950316
*Ours - Self	0.000171719
*Ours - UnSup	0.013007246
Self - *Ideal	0.004517434
UnSup - Ideal	0.117659114
UnSup - Self	0.683464011

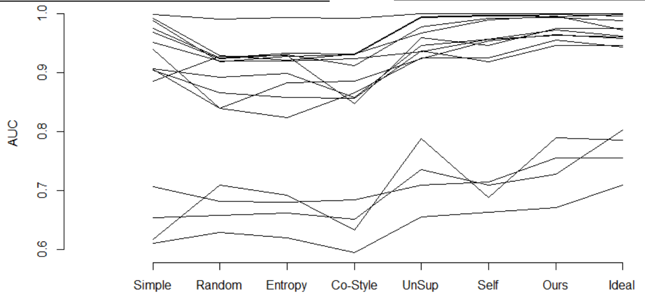


Table 4.4: Average AUC and P-values calculated by Friedman's and Nemenyi test on Text data sets (Setting: 5 labeled instances, 50 attributes, **J48**)

	Simple	**Random**	**Entropy**	**Co-Style**	**UnSup**	**Self**	**Ours**	**Ideal**
fbis(111,142)	0.962	0.916	0.924	0.921	0.908	0.841	**0.946**	0.987
fbis(111,189)	0.834	0.921	0.92	0.895	0.869	0.844	**0.936**	0.989
fbis(142,189)	0.982	0.922	0.925	0.937	0.925	0.919	**0.994**	0.989
new3s(111,142)	0.891	0.928	0.928	0.904	0.848	0.854	**0.965**	0.995
new3s(189,142)	0.854	0.855	0.87	0.878	0.875	0.823	**0.875**	0.847
new3s(189,301)	0.789	0.911	0.916	0.875	0.789	0.789	**0.943**	0.966
ohscal(An,Ca)	0.793	0.849	0.853	0.801	0.793	0.793	**0.893**	0.95
ohscal(An,Dn)	0.693	0.923	0.923	0.867	0.693	0.693	**0.89**	0.963
ohscal(Ca,Dn)	0.65	0.825	0.82	0.79	0.659	0.659	**0.848**	0.912
ohscal(Ca,To)	0.572	0.652	0.661	0.649	0.633	0.598	**0.662**	0.607
ACEInhibitors	0.891	0.92	0.919	0.922	0.891	0.891	**0.934**	0.776
ADHD	0.542	0.613	0.617	0.62	0.591	0.572	**0.606**	0.595
AtypicalA	0.583	0.62	0.62	0.642	0.6	0.6	**0.67**	0.699
ProtonP	0.573	0.6	0.621	0.617	0.605	0.599	**0.621**	0.627
Statins	0.56	0.574	0.578	0.59	0.617	0.612	**0.648**	0.6
Average	**0.745**	**0.802**	**0.806**	**0.794**	**0.753**	**0.739**	**0.829**	**0.833**

Methods	**P-value**
*Ours - Co-Style	4.433658e-02
Ours - Entropy	2.294131e-01
*Ours - Random	1.580284e-02
*Ours - Simple	9.933368e-07
Entropy - Co-Style	9.581112e-01
Random - Co-Style	9.969144e-01

Methods	**P-value**
Ours - Ideal	0.9916246481
*Ours - Self	0.0001307217
*Ours - UnSup	0.0309404987
Self - *Ideal	0.0005864788
UnSup - Ideal	0.0675156171
UnSup - Self	0.4721436347

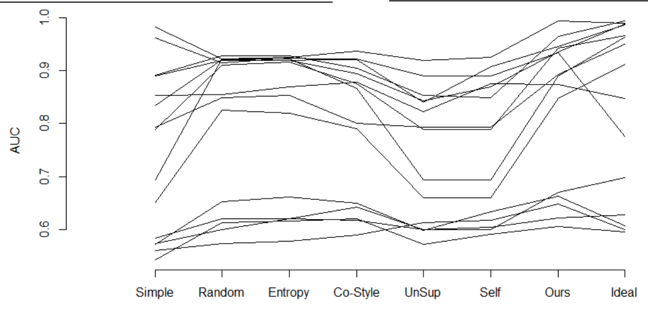
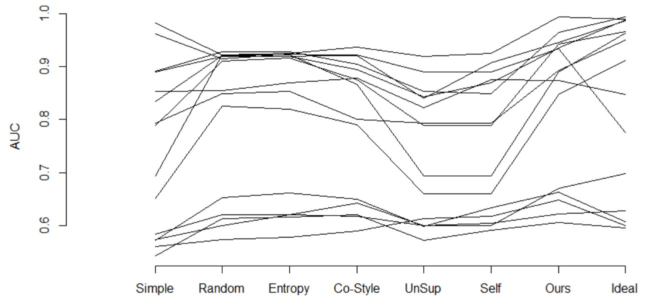


Table 4.5: Average AUC and P-values calculated by Friedman's and Nemenyi test on Text data sets(Setting: 5 labeled instances, **100 attributes**, Naïve Bayes)

	Simple	**Random**	**Entropy**	**Co-Style**	**UnSup**	**Self**	**Ours**	**Ideal**
fbis(111,142)	0.97	0.93	0.917	0.909	0.989	0.986	**0.996**	0.986
fbis(111,189)	0.983	0.984	0.984	0.956	0.993	0.998	**1**	1
fbis(142,189)	0.952	0.902	0.934	0.884	0.987	0.918	**0.996**	0.959
new3s(111,142)	0.958	0.936	0.935	0.939	0.994	0.984	**0.998**	0.992
new3s(189,142)	0.933	0.909	0.884	0.91	0.98	0.971	**0.995**	0.987
new3s(189,301)	0.866	0.836	0.816	0.889	0.884	0.902	**0.978**	0.958
ohscal(An,Ca)	0.866	0.887	0.902	0.822	0.952	0.947	**0.974**	0.962
ohscal(An,Dn)	0.723	0.769	0.808	0.668	0.92	0.877	**0.965**	0.941
ohscal(Ca,Dn)	0.87	0.919	0.923	0.834	0.94	0.951	**0.971**	0.975
ohscal(Ca,To)	0.807	0.841	0.865	0.823	0.947	0.94	**0.958**	0.959
ACEInhibitors	0.719	0.691	0.692	0.715	0.75	0.765	**0.791**	0.805
ADHD	0.949	0.917	0.915	0.919	0.954	0.934	**0.946**	0.955
AtypicalA	0.653	0.677	0.676	0.651	0.642	0.714	**0.688**	0.75
ProtonP	0.694	0.699	0.694	0.686	0.778	0.715	**0.724**	0.797
Statins	0.692	0.672	0.652	0.641	0.733	0.752	**0.748**	0.788
Average	**0.842**	**0.838**	**0.84**	**0.817**	**0.896**	**0.89**	**0.915**	**0.921**

Methods	**P-value**
*Ours - Co-Style	3.729910e-07
*Ours - Entropy	3.145261e-04
*Ours - Random	8.409118e-04
*Ours - Simple	1.080850e-02
Entropy - Co-Style	7.096187e-01
Random - Co-Style	5.617369e-01

Methods	**P-value**
Ours - Ideal	0.998991680
*Ours - Self	0.003915183
*Ours - UnSup	0.036169506
Self - *Ideal	0.002242829
UnSup - *Ideal	0.023956682
UnSup - Self	0.894325730



the benchmark experiment which helps to conclude that regardless of the base classifier our approach is statistically significantly better than other approaches while it is not significantly different from ideal classifier. The only approach which is not statistically significantly different from our approach in this experiment is the entropy split, however the average AUC of our approach is still better.

Comparing the average AUC with that of the benchmark experiment shows that the average AUC of the ideal classifier in this experiment is lower while the other three approaches have higher values. This confirms the expectations. According to Kohavi (1996), when the labeled training set is small Naïve Bayes tends to have better performance, while, when there is enough labeled instances J48 performs much better.

The same settings as the benchmark experiment, except that 100 attributes are used instead of 50, are used in an experiment to explore the effect of the size of the feature set on the results. Again the results of the significant tests, which are shown in table 4.5, are the same. The only difference is that the accuracy of our approach and ideal classifier are increased because more attributes provide more information.

We have also evaluated our approach on microarray gene expression data using the GEMLeR data sets (Stiglic and Kokol, 2010) to show that our approach works on other domains. In one experiment we used the same settings as our benchmark experiment to perform experiments on the GEMLeR data sets. In two another complementary experiments we used benchmark setting except that 3 and 10 training instances are used. Table 4.6, 4.7 and 4.8 show the average AUC and the result of Friedman's and Nemenyi test on all 42 data sets in the GEMLeR repository for these three experiments. More detailed results including the average AUC of each approach on each dataset is provided in the Appendix. The final conclusion obtained from the statistical tests are the same for these experiments, the performance of our approach, applied on microarray gene expression data, tends to be close to the ideal classifier, while it is statistically significantly better than other approaches. The difference in our approach and other existing ones is more clear because the p-values are much smaller. The reason is that the number of data sets used in experiments on Microarray data is larger than the number of text datasets which are used in our experiments.

4.3 Advantages, Limitations and Future Work

This new approach provides a solution to use multi-view learning on single view data sets, specifically when labeled data is limited. One of the main advantages of this approach is

Table 4.6: Average AUC and P-values calculated by Friedman's and Nemenyi test On **Microarray data sets** (**3 labeled instances**, 50 attributes, Naïve Bayes)

	Simple	**Random**	**Entropy**	**Co-Style**	**UnSup**	**Self**	**Ours**	**Ideal**
Average	**0.86**	**0.916**	**0.918**	**0.911**	**0.938**	**0.937**	**0.958**	**0.963**

*Ours - Co-Style	3.358526e-09	Ours - Ideal	7.948115e-01
*Ours - Entropy	1.520047e-08	*Ours - Self	1.318843e-09
*Ours - Random	3.601730e-09	*Ours - UnSup	5.127753e-09
*Ours - Simple	0.000000e+00	Self - *Ideal	1.799215e-07
Entropy - Co-Style	9.996098e-01	UnSup - *Ideal	3.036622e-07
Random - Co-Style	9.999753e-01	UnSup - Self	9.997912e-01

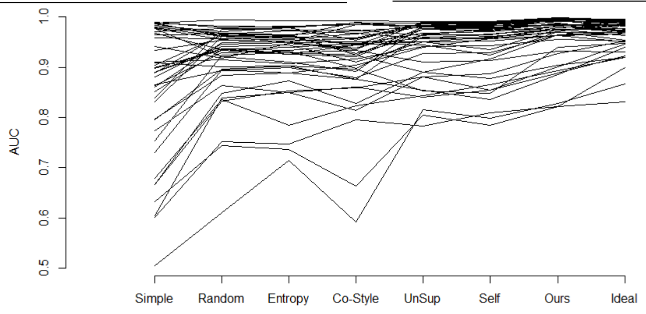


Table 4.7: Average AUC and P-values calculated by Friedman's and Nemenyi test On **Microarray data sets** (5 labeled instances, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
Average	**0.933**	**0.919**	**0.919**	**0.924**	**0.939**	**0.931**	**0.958**	**0.963**

*Ours - Co-Style	7.336999e-10	Ours - Ideal	3.383465e-01
*Ours - Entropy	3.996803e-15	*Ours - Self	3.430589e-12
*Ours - Random	1.110223e-16	*Ours - UnSup	2.231946e-08
*Ours - Simple	6.521918e-06	Self - *Ideal	3.068284e-07
Entropy - Co-Style	4.736980e-01	UnSup - *Ideal	1.627455e-04
Random - Co-Style	1.863368e-01	UnSup - Self	5.921200e-01

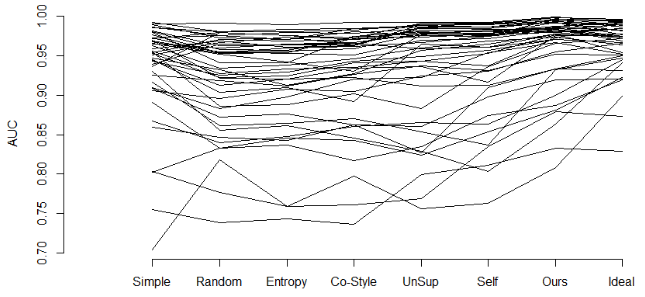
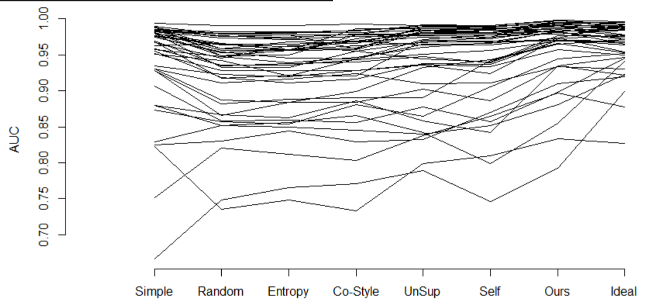


Table 4.8: Average AUC and P-values calculated by Friedman's and Nemenyi test On **Microarray data sets** (**10 labeled instances**, 50 attributes, Naïve Bayes)

	Simple	**Random**	**Entropy**	**Co-Style**	**UnSup**	**Self**	**Ours**	**Ideal**
Average	**0.939**	**0.922**	**0.923**	**0.927**	**0.939**	**0.938**	**0.959**	**0.963**

*Ours - Co-Style	6.158845e-10	Ours - Ideal	5.921217e-01
*Ours - Entropy	1.110223e-16	*Ours - Self	1.997162e-10
*Ours - Random	1.110223e-16	*Ours - UnSup	5.789313e-11
*Ours - Simple	1.169602e-02	Self - *Ideal	1.294623e-06
Entropy - Co-Style	3.484508e-01	UnSup - *Ideal	1.428770e-07
Random - Co-Style	3.484143e-01	UnSup - Self	9.871074e-01



that it improves the views during semi-supervised learning. This is done by integrating attribute clustering with multi-view learning. This advantage addressed the problem recently identified by Du et al. (2011): given a small labeled training sets the view splitting methods are unreliable.

Another advantage of our approach is that, unlike some of the proposed view splitting methods, it is not limited only to two separate views. This approach creates multiple overlapping views and uses those which represent enough information about the class, in terms of Chi-Squared measure. Di and Crawford (2012) showed that increasing the number of views increases the diversity and consequently improves the performance of multi-view learning.

When labeled data is more common, this approach will not be so beneficial. With sufficient labeled instances supervised attribute clustering could be used to create views and there is little point in using a semi-supervised approach. We will investigate, in our future work, the effect of the proposed view splitting method for creating the views in a supervised manner.

In our experiments, we used two class text and microarray gene expression data sets. In future work, we will extend the experiments to multi-class data sets. We will also investigate the effect of applying attribute clustering in additional ways to improve the classifier in the iterative process, i.e. the use of attribute clustering to improve missing attribute value handling.

4.4 Summary

In this chapter, we have proposed a methodology for creating effective views, for single view data sets with a very small number of labeled instances, to improve the learning performance of multi-view learners. In this new approach we integrate view splitting with the learning process. We combine attribute clustering with multi-view learning, in a semi-supervised manner, to solve the problem of automatic view definition for multi-view learning, when the labeled data is limited. The proposed approach is evaluated using experiments on real world text and microarray gene expression data sets. Our experimental results showed that the proposed approach is significantly better than all the other existing methods, while it is close to and not significantly different from the ideal, yet unrealistic, supervised classifier which has access to the labels for all the instances during the learning. These experiments show that when the labeled set is very small, unlike other approaches, our approach is applicable and the result is not significantly inferior to using a completely labeled data set.

Chapter 5

Improving Missing Attribute-value Handling

This chapter proposes the use of attribute clustering to improve missing attribute value handling. When talking about missing attribute-value handling, we should distinguish between missing attribute-values at induction time in the training data (Grzymała-Busse and Grzymala-Busse, 2005), and at prediction time in test data or the upcoming new instances (Saar-Tsechansky and Provost, 2007). The first section shows how attribute clustering can be used to improve missing value handling at induction time by limiting the attributes used to find the missing value. The second section shows how attribute clustering can be used to make reduced-feature models, a missing attribute-value handling method for prediction time, applicable on data sets with large number of attributes. We show, how these two usages of attribute clustering for improving missing attribute-value handling, can be used in our proposed framework to improve attribute clustering, missing attribute-value handling and classification, when labeled data is limited.

5.1 Improving Missing Attribute-value Handling at Induction Time

Various approaches have been proposed for handling missing attribute-values at induction time. A group of well-known missing attribute-value handling methods are designed based on the idea of closest fit. This idea originated in Grzymała-Busse et al. (1999) and it has two different variations: The Global and Concept Closest Fit.

The **Global Closest Fit** method looks into the pool of instances with known values and replaces a missing value by the value of another instance which is the closest fit to the one with missing value, based on Manhattan distance measure. This method considers the class label as one of the attributes. It leads to find the closest fit from other classes in some cases. Another variation of this method, the **Concept Closest Fit** method is proposed to solve this problem. The only difference, comparing to the global closest fit method, is that the search space is limited to the subset of instances with the same class label. This decreases the search time and also improves the accuracy of the value found for missing attribute-value.

Acording to Suguna and Thanushkodi (2011), although several approaches are proposed for missing attribute value handling at induction time, many based on the idea of closest fit (Gaur and Dulawat, 2011; Gaur, 2012, 2014), the one used more frequently is deleting instances containing at least one missing value. If many instances have missing attribute-values, different variations of common value methods, such as assigning an average or the most common value, are used alternatively. Grzymala-Busse et al. (2002) and Grzymala-Busse et al. (2005) concluded that even though the closet fit methods, unlike common value methods, consider the attribute-values of each instance when looking for its missing values, they are not better than common value methods. The computational time of closest fit methods are high because in order to find the missing value of an instance, they need to calculate its distance to all other instances. Additionally, the feature space can include attributes which are completely unrelated to the attribute with missing value and such attributes mislead the process of finding the closest fit. Common Value methods, have actually removed all the attributes except the one with missing attribute value from consideration. This removes the misleading attributes but it removes useful ones as well.

When looking for the closest fit in such cases, using a subset of the feature space, reduces the computational time. If the subset includes all the related attributes, it will improve the accuracy of the value found for replacing the missing attribute-value. Aha (1992); Aha et al. (1991) improved instance based learning in IBK4 and IBK5 by using the idea of removing (limiting) irrelevant attributes from feature set, when calculating the distance measure. This is done by weighting attribute values in the computation of distance measure, based on the relevance of attributes to the concept. When weights are 0 and 1 it will remove the irrelevant attributes by assigning weight 0 to them. The same idea could be used in missing attribute value handling, when calculating the distance measure to find the closest fit. In this case, when handling a missing attribute-value, the irrelevant attributes to that attribute needs to be removed from the feature set.

Li and Cercone (2006) used rough set theory to limit the attributes used to find the closest fit, to the core or one of the reducts. Each reduct is a subset of the attributes which is sufficient enough to represent the whole data. The core is the intersection of all the reducts. This approach reduces the computational time but it is not effective in terms of removing irrelevant attributes from the search. The reducts are generated considering that each one should be able to represent the whole data, so it is not necessarily gathering related attributes in each reduct. There also may be more than one reduct which contain the attribute with missing value so it is not clear which one to use. According to Du et al. (2011), when labeled data is limited it is difficult to find the underlying relationships between attributes, therefore the reducts and the core would not be reliable.

We address these problems in this research, proposing a new approach, which uses attribute clustering to limit the feature space to the related attributes. In other words, we are changing the definition of closest fit to be the closest instance from the pool, considering the subset of related attributes to the one with missing attribute-value. In first step, the attribute clustering is used to create clusters of related attributes. Then, in order to find a missing attribute-value, the cluster which the attribute belongs to will be used as the feature set to find the closest fit. This will remove the irrelevant attributes while it is drastically decreasing the computational time. When there is not enough labeled instances available to perform supervised attribute clustering, the proposed semi-supervised framework in chapter 3 would be applicable. The result of attribute clustering would be used in each iteration to improve the classifier by improving the missing attribute value handling and the classifier will be used in each iteration to improve the attribute clustering by augmenting its training set. Therefore the result would be a better missing attribute value handling and also an improved semi-supervised classifier.

5.1.1 Attribute Clustering for Missing Attribute-value Handling at Induction Time

The closest fit methods for missing attribute value handling, use a distance measure between the instance with missing attribute-value and all other instances, to find the closest fit. Then the missing attribute-value will be replaced with the value of the corresponding attribute in the closest fit. The vectors of the attribute-values, which are used to calculate the distances, in conventional closest fit methods, include all the known attributes. However, the feature space can include attributes which are completely irrelevant to the attribute with missing attribute-value. Such attributes not only mislead the process of finding the closest fit but they also increase the computational time of calculating the distance.

The following section shows a simple example which illustrates how irrelevant attributes can mislead the closest fit algorithms in calculating the distance and consequently selecting the correct closest fit. This example also shows how attribute clustering could be applied in such cases to filter the irrelevant attributes and improve the accuracy and computational time of missing attribute value handling.

An Example of Missing Attribute-value Handling in the Presence of Irrelevant Attributes

Table 5.1 shows a simple example with three cases. The attributes include symptoms of a disease and the class shows whether the patient has the disease or not. The feature space, symptoms, include three groups of correlated attributes. The three cases in this example are positive. In first case the first and third group of symptoms are strong. In second case only the third group of symptoms are obvious. The last case only shows strong appearance of the first group of symptoms. In order to find the distance between two cases we compare the vectors of attribute values. For any difference between attribute values of a specific index we add 1 to the distance.

Considering the whole feature space to find the closest fit to the first case, the distance of first case to the second and third one is 2 and 4, respectively. Therefore we will consider the second case as the closest fit to the first case. However, the first group of symptoms are strongly appeared in first case while the second case does not show any of these symptoms. Consequently this case is not a good choice to find the missing attribute-value in the first group of symptoms of the first case. Additionally, the first group of symptoms are strongly appeared in first and third case so the third case is a better candidate for this purpose. As we see in this example the irrelevant attributes in second and third group mislead the calculation of the distance and consequently the closest fit selection.

Now let's consider that we use the correlated attributes, the attributes in the same group, to calculate the distance measure and find the closest fit. Considering the first group of correlated symptoms, the distance of first case to the second and third one is 2 and 0, respectively. It means that the third case is the closest fit to the first case. As we see in this example, using correlated attributes, to the one with missing attribute-value, removes the effect of misleading attributes and leads to a better calculation of missing attribute-value.

Table 5.1: An example data set including three groups of disease symptoms for three patients.

Case	Symptom group 1			Symptom group 2			Symptom group 3			Decision
1	Yes	Yes	?	Yes	No	No	Yes	Yes	Yes	Yes
2	No	No	No	Yes	No	No	Yes	Yes	Yes	Yes
3	Yes	Yes	Yes	No	No	No	No	No	No	Yes

In order to find the distance between two instances,in first approach, all the attributes will be used, so 9 comparisons need to be done. When the cluster of relevant attributes are used this number reduced to 3 comparisons.

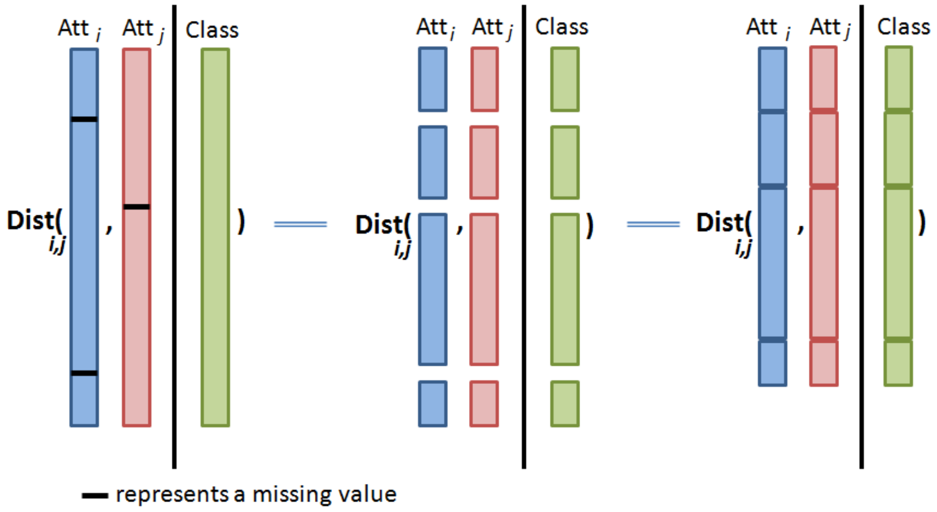
This example shows how we can improve the accuracy and computational time of the closest fit-based missing attribute-value handling methods by applying attribute clustering in the first step and then calculating the distance based on the relevant attributes.

Attribute Clustering in the Presence of Missing Attribute-values

Different methods have been proposed for attribute clustering. All of these approaches use a distance matrix for the attributes to perform attribute clustering. The entry at index $(i\ ,\ j)$ of this matrix represents the distance between attribute i and j. Once the distance matrix is calculated any clustering algorithm could be applied to cluster the attributes. We use Aglomerative Nesting (AGNES) (Kaufman and Rousseeuw, 2009), which is a hierarchical clustering method. It takes as input the distance matrix and returns hierarchical clustering results in the form of a tree. For supervised attribute clustering, a supervised similarity measure needs to be used. Pairwise partial correlation between two attributes given a third one, in this case the class, is used as the supervised similarity measure to calculate the distance matrix. It is used to generate a matrix which contains the partial correlation between every two attributes. The inverse of this similarity matrix is used as distance matrix, for attribute clustering.

When performing attribute clustering in the presence of missing attribute-values, the missing data will be handled in the calculation of the distance matrix. So the second part which uses the distance matrix as an input will not be changed. When filling the entry $(i,\ j)$ of the distance matrix, we find the distance between two attributes i and j given the class. So we use the vector of attribute values for each of the two attributes

Figure 5.1: Calculating the distance between 2 attributes in presence of missing values.



and we calculate the similarity. In case of missing attribute values we will ignore the instances with missing values when creating the two vectors for the two attributes and the class. This process is illustrated in figure 5.1. So the two vectors which will be used to calculate the distance between attribute i and j will not include any missing values. Then we use the distance matrix to perform the attribute clustering.

Filtering out the Irrelevant Attributes Using the Result of Attribute Clustering

According to Grzymała-Busse and Grzymala-Busse (2005), conventional distance between attributes x and y is computed as follows

$$Distance(x, y) = \sum_{i=1}^{n} distance(x_i, y_i), \tag{5.1}$$

where

$$distance(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i; \\ 1 & \text{if } x \text{ and } y \text{ are symbolic and } x_i \neq y_i, \\ & \text{or } x_i =? \text{ or } y_i =?; \\ \frac{|x_i - y_i|}{r} & \text{if } x_i \text{ and } y_i \text{ are numbers and } x_i \neq y_i, \text{ where } r \text{ is the} \\ & \text{difference between the maximum and minimum of} \\ & \text{the values of the numerical attribute } i. \end{cases} \tag{5.2}$$

Equation 5.1 calculates the Manhattan distance between two attributes, x and y, as the sum of the distances between their attribute-values. Equation 5.2 shows that the distance between two attribute-values, x_i and y_i, is 0 if they are equal, 1 if one of the values is missing or they are not equal but symbolic, and in other cases it is equal to the normalized distance between the two attribute values.

We propose the use of attribute clustering to limit the attributes used in calculating the distance measure. Our new approach calculates the sum of the distances between values of the attributes, from the cluster of attributes to which belongs the attribute with missing attribute-value. In order to do so the Equation 5.1 is changed in our approach to Equation 5.3 which shows the new sum of distances as follows

$$Distance(x, y|Cluster_j) = \sum_{\forall i \in Cluster_j} distance(x_i, y_i), \tag{5.3}$$

where $Cluster_j$ is the cluster of attributes to which the attribute with missing attribute-value belongs to.

Therefore different distances between two cases x and y will be calculated based on the cluster to which belongs the attribute with missing attribute value. In other words, for missing values in different attributes of the same instance but different clusters, we may find different closest fits. This will improve the value found for missing attribute-value by filtering the misleading attributes when looking for the closest fit.

Removing the irrelevant attributes, when looking for the closest fit, speeds up the search because less comparisons need to be made to calculate the distance and find the closest fit. It also dramatically decreases the range of possible values for distance. Let's consider we have n attributes. If we use the whole feature set to calculate the distance could vary from 0 to n but if we use a subset of size m relevant attributes, it will vary from 0 to m. Decreasing the range of possible values for distance will increase the chance of having more closest fit instance for a particular instance. In case of multiple closest fits we will calculate the average in case of numeric attributes and most frequently seen value in case of symbolic attributes. Beside the removal of irrelevant attributes, this averaging or voting will decrease the variance of the calculated value and consequently will improve the results.

We call these new versions of Global Closest Fit and Concept Closest Fit methods, with limited feature sets using attribute clustering, as Limited Global Closest Fit (LGCF) and Limited Concept Closest Fit (LCCF), respectively.

Combining Missing Attribute-value Handling and Attribute Clustering Using the Semi-supervised Framework

We proposed the use of attribute clustering for an improved missing attribute-value handling method. The supervised attribute clustering can provide more accurate results than unsupervised one because it considers more information, the class labels of the instances, while clustering attributes. Therefore, it is able to extract patterns, in the data, which the unsupervised attribute clustering is not able to catch. However, empirical experiments (e.g. (Du et al., 2011; Palanikkumar and Scholar, 2013; Maji, 2012, 2011)) in machine learning literature proved that the result of supervised attribute clustering is not reliable when we only use a small labeled dataset for training. The small number of samples is not enough to distinguish the real relationships between attributes. This is where we need semi-supervised attribute clustering to get benefit from both the labeled and unlabeled instances.

In many real world learning tasks very few labeled instances are available. When there is not enough labeled instances available to perform supervised attribute clustering, we can use the proposed semi-supervised framework in Semi-supervised 'Attribute Clustering'/Classification Framework chapter. This framework combines attribute clustering with classification to create a semi-supervised attribute clustering approach. The result of attribute clustering will be used, in each iteration, to improve the classifier. In this case, this improvement will be done by using attribute clustering for improving missing attribute value handling. Additionally, the classifier will be used, in each iteration, to improve the attribute clustering by augmenting its training set. This process is illustrated in figure 5.2.

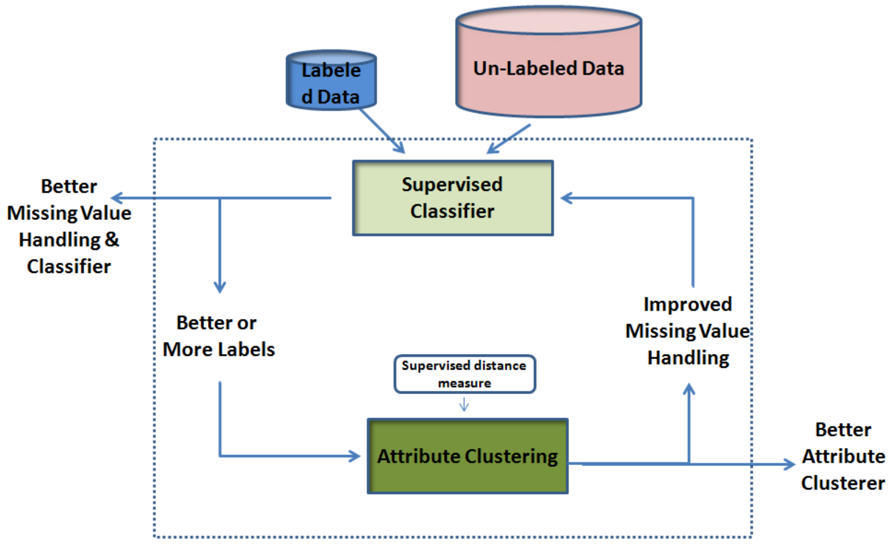
The general algorithm for this semi-supervised approach is shown in algorithm 3.

It receives, as input, a labeled training set L with M instances and A attributes and an unlabeled training set U with N instances and A attributes. This algorithm has two

Algorithm 3 Proposed semi-supervised attribute clustering approach used for missing attribute-value handling

```
1: Input Labeled training set $L$ with $M$ instances and $A$ attributes, Unlabeled training set $U$ with $N$ instances and $A$ attributes, Number of Clusters of attributes $P$ (optional)
2: repeat
3:     if First Iteration then
4:         Perform initial missing attribute-value handling
5:     else
6:         Use new clusters of attributes $F$ to perform missing attribute-value handling
7:     end if
8:     Train a classifier $C$ on the Labeled training set $L$
9:     Classify unlabeled instances $U$
10:    Add $K$ most confident new labeled instances to the labeled set ($M = M + K, L = L \cup \{K$ most confident new labeled instance$\}$)
11:    Use new labeled set with an attribute clustering method to provide clusters of attributes $F$
12: until No new instances added
13: Output: The set of clusters, missing attribute-value handling and the classifier trained in the final iteration
```

Figure 5.2: Iterative semi-supervised multi-view learning combined with attribute clustering.


Labele d Data
Un-Labeled Data
Supervised Classifier
Better Missing Value Handling & Classifier
Better or More Labels
Improved Missing Value Handling
Supervised distance measure
Attribute Clustering
Better Attribute Clusterer

main steps, the classification and attribute clustering.

In the first iteration of the algorithm, the initialization in line 4 obtains an initial value for missing attribute-values. Then the iterative process starts by training the classifier on the labeled set in line 8. The classifier is then used to label unlabeled data in line 9. The output of this step is a set of new labeled instances paired with a measure of confidence in the new label. This confidence is used in line 10 to add most confident new labeled instances to the labeled training set.

Then, in line 11, the new labeled training set is used with the supervised attribute clustering algorithm to find more accurate clusters of attributes. The output of the attribute clustering algorithm is then used in the next iteration, in line 6, to perform an improved missing attribute-value handling and improve the classifier.

The algorithm iterates until all the unlabeled data is used to augment the training set or there is no more confident new labeled instances to be added to the labeled training set. At the end, the algorithm outputs the missing attribute-value handling results, attribute clustering results and the classifier trained on the final iteration.

5.1.2 Evaluation

In this section the proposed method for missing attribute-value handling is evaluated using real UCI data sets (Bache and Lichman, 2013). Our approach is compared, from

Table 5.2: The 10 UCI data sets used in our experiments.

Data set	**Number of Instances**	**Number of Attributes**	**Class Distribution**
Spambase (Spamb)	4601	57	1813/2788
First-order theorem proving (heuristic 0) (FOTP)	3059	51	1286/1773
QSAR biodegradation (QSAR)	1055	41	356/699
Breast Cancer Wisconsin (Diagnostic) (BCWD)	569	32	212/357
Climate Model Simulation Crashes (All) (CMSC)	540	18	46/494
Ionosphere (IoPh)	351	34	225/126
Climate Model Simulation Crashes (Study 1) (CMSC1)	180	18	20/160
Climate Model Simulation Crashes (Study 2) (CMSC2)	180	18	12/168
Climate Model Simulation Crashes (Study 3) (CMSC3)	180	18	14/166
Connectionist Bench (Sonar, Mines vs. Rocks) (CB)	208	60	97/111

different aspects, with other methods. Our experimental results show that not only the performance of our new approach is better than others, in terms of AUC, but also the extracted missing values are statistically significantly closer to the real values as well as the computational time is statistically significantly lower than original closest fit method.

Configuration and Data Preparation

In order to show better the effectiveness of our proposed missing attribute-value handling method at induction time, we used 10 different UCI data sets which has numerical attributes with high variance in these experiments experiments. The higher variance of the attribute-values increases the negative effect of the missing attribute-values on the performance of the classifier trained on the data set. This helps to better show the difference between different missing value handling methods. These data sets and their

Table 5.3: Average AUC on UCI data sets (Setting: **3 missing values per instance in one third of training instances**)

Data set	**NMV**	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**	**Real**
CMSC	0.5	0.583	0.583	0.566	0.546	0.551	0.546	0.596
CMSC1	0.631	0.605	0.605	0.614	0.628	0.616	0.66	0.68
CMSC2	0.5	0.509	0.509	0.538	0.551	0.524	0.531	0.549
CMSC3	0.5	0.49	0.49	0.513	0.503	0.531	0.504	0.575
CB	0.605	0.607	0.612	0.639	0.643	0.651	0.682	0.642
BCWD	0.848	0.888	0.888	0.896	0.871	0.864	0.866	0.92
FOTP	0.549	0.549	0.542	0.542	0.554	0.554	0.539	0.592
IoPh	0.68	0.812	0.812	0.806	0.814	0.799	0.815	0.841
QSAR	0.601	0.711	0.711	0.7	0.69	0.732	0.719	0.713
Spamb	0.723	0.76	0.758	0.753	0.759	0.759	0.751	0.749
Average	**0.614**	**0.651**	**0.651**	**0.657**	**0.656**	**0.658**	**0.661**	**0.686**

basic properties are listed in table 5.2. 30 instances are used as training set in each of our experiments. The missing attribute-values are added to one third of the training instances. We use different number of missing attribute-values per instance, 3, 5 and 10, in different experiments, to see the effect of the ratio of missing attribute values on the results. The missing attribute-values are added to the attributes which has more information about the class, to make the problem more challenging. In order to do this, the attributes are sorted based on their chi-squared measure and the 3, 5 and 10 missing values are distributed between 5, 10 and 15 of the most important attributes, respectively.

AUC is used as one of our evaluation measures to compare the performance of the new variations of closest fit method with others. The Manhattan distance between the calculated vector of missing values by our approach and the real ones is compared with that of the other methods to show that more accurate values are calculated using our approach. The time elapsed to find the missing values is also compared with closest fit methods to show how these new variations are faster.

Table 5.4: Top: Manhattan Distance between the vector of calculated values by different approaches and the real values. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **3 missing values per instance**)

Data set	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	14.925	14.379	12.254	11.869	10.728	10.129
CMSC1	13.901	12.984	11.452	12.24	11.522	11.314
CMSC2	15.32	14.941	12.063	11.46	10.402	9.96
CMSC3	16.699	16.004	11.982	11.844	11.276	11.804
CB	10.498	9.529	9.987	7.903	6.824	7.006
BCWD	8.639	5.813	11.218	6.466	3.01	3.313
FOTP	8.454	9.754	9.563	8.555	3.101	4.304
IoPh	8.518	11.176	8.121	7.626	4.268	4.09
QSAR	9.444	8.98	20.59	13.427	4.835	4.911
Spamb	6.261	6.205	7.377	7.331	6.615	6.368
Average	**11.266**	**10.977**	**11.461**	**9.872**	**7.258**	**7.32**

Methods	**P-value**	**Methods**	**P-value**
CCF - CCA	0.979939284	LGCF - CCF	0.205948576
GCA - CCA	0.979939449	GCF - GCA	0.999227027
GCF - CCA	0.999227046	*LCCF - GCA	0.001847451
*LCCF - CCA	0.023172573	*LGCF - GCA	0.002896250
*LGCF - CCA	0.033507144	*LCCF - GCF	0.006987019
GCA - CCF	0.706084176	*LGCF - GCF	0.010585561
GCF - CCF	0.891213535	LGCF - LCCF	0.999996612
LCCF - CCF	0.159554971		

Table 5.5: Top: Computational time in seconds. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **3 missing values per instance**)

Data set	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	30.672	25.704	15.428	12.924
CMSC1	33.33	24.948	13.922	12.138
CMSC2	29.274	26.544	14.958	12.252
CMSC3	24.87	19.716	11.962	9.642
CB	99.156	49.224	57.918	28.666
BCWD	46.734	24.474	25.746	13.52
FOTP	71.868	35.412	41.968	20.558
IoPh	51.096	27.36	31.116	16.824
QSAR	45.114	23.316	27.134	14.55
Spamb	47.778	23.808	19.494	9.572
Average	**47.989**	**28.051**	**25.965**	**15.065**

Methods	**P-value**	**Methods**	**P-value**
GCF - CCF	4.632077e-02	*LCCF - GCF	9.765876e-07
*LCCF - CCF	4.610257e-02	*LGCF - GCF	4.613030e-02
LGCF - CCF	1.000000e+00	LGCF - *LCCF	4.625145e-02

Competitor Algorithms

According to Grzymala-Busse et al. (2002) and Grzymala-Busse et al. (2005) the closest fit methods are not preferred comparing to simple approaches like the one which removes instances with missing attribute-values from training set or the one which replaces them with common values. One reason is the poor accuracy of closest fit methods because of the existence of irrelevant attributes, to the one with missing value, in the feature set. Another reason is that it is computationally expensive comparing to the other approaches. In our approach we changed the definition of closest fit by removing the irrelevant attributes using attribute clustering. Therefore we compare our approach with original closest fit methods to show that it is faster and more accurate. It is also compared with the other methods, mentioned above, to show that it calculates better values. Therefore, the competitor algorithms in our experiments include:

- **No Missing Value (NMV)** method simply removes all the instances which include at least one missing attribute-value. This approach is fast and easy to implement and if sufficient training instances without any missing attribute-value are available, it will not affect the performance of the learning algorithms.

- **Global Closest Fit (GCF)** looks into the pool of instances with known values and replaces a missing attribute-value by the value of another instance which is the closest fit to the one with missing attribute-value. To find the closest fit, Manhattan distance measure is computed between the vector of attribute-values for the case with missing attribute-value and all other cases and the instance with the smallest distance is selected as the closest fit. This method does not consider the class of instances when looking for the closest fit. It leads to find the closest fit from other classes in some cases. This might be problematic because the two instances, which found very close to each other, can be dramatically different and this difference can come from the value of the attribute with missing attribute-value. Therefore, the corresponding attribute-value of the closest fit can be far from the real value of the missing attribute-value.

- **Concept Closest Fit (CCF)** splits the data set, in first step, into subsets based on the class label of instances. Then, for replacing each missing attribute-value, it finds the closest fit in the subset of the instances with the same class label and replaces the missing attribute-value with the known value in the closest fit. The only difference, comparing to the global closest fit method, is that the search space

Table 5.6: Average AUC on UCI data sets (Setting: **5 missing values per instance in one third of training instances**)

Data set	**NMV**	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**	**Real**
CMSC	0.5	0.594	0.594	0.59	0.59	0.59	0.59	0.596
CMSC1	0.631	0.627	0.624	0.608	0.614	0.615	0.623	0.68
CMSC2	0.5	0.534	0.519	0.535	0.539	0.55	0.52	0.549
CMSC3	0.5	0.528	0.528	0.5	0.516	0.505	0.514	0.575
CB	0.605	0.636	0.636	0.642	0.64	0.618	0.703	0.642
BCWD	0.848	0.833	0.832	0.902	0.883	0.877	0.866	0.92
FOTP	0.549	0.556	0.556	0.562	0.549	0.569	0.56	0.592
IoPh	0.68	0.728	0.787	0.726	0.759	0.813	0.806	0.841
QSAR	0.601	0.689	0.684	0.734	0.697	0.73	0.725	0.713
Spamb	0.723	0.743	0.743	0.742	0.738	0.737	0.759	0.749
Average	**0.614**	**0.647**	**0.65**	**0.654**	**0.652**	**0.661**	**0.667**	**0.686**

is limited to the subset of instances with the same class label. This decreases the search time and also improves the accuracy of the value found for missing attribute-value.

- **Global Common Attribute-Value (GCA)** is a simple approach which replaces the missing attribute-value by the most common or the average value of that attribute, in case of symbolic or numeric attributes, respectively.
- **Concept Common Attribute-Value (CCA)** is similar to Global Common Attribute-Value approach, which calculates the value within the same concept to which belongs the case with a missing attribute-value.

Results and Discussion

Our experimental results show that, limiting attributes, to those which are correlated to the attribute with missing attribute-value, improves the closet fit methods. It not only improves the AUC of the classifier trained on the corrected training set, but also makes better prediction of the missing values, as well as decreases the computational complexity.

Table 5.7: Top: Manhattan Distance between the vector of calculated values by different approaches and the real values. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **5 missing values per instance**)

Data set	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	24.9	23.954	21.343	20.774	18.579	18.319
CMSC1	23.528	22.014	20.648	20.91	19.875	18.994
CMSC2	24.828	24.153	22.054	20.409	19.454	19.537
CMSC3	27.112	26.692	22.92	22.659	18.728	19.465
CB	16.27	14.988	15.976	13.072	10.98	10.96
BCWD	15.46	10.666	21.607	13.113	6.642	6.012
FOTP	14.027	15.092	14.745	13.503	5.411	7.327
IoPh	14.447	18.092	13.1	12.882	6.46	7.066
QSAR	12.463	12.696	33.026	21.275	7.58	7.946
Spamb	8.943	8.776	10.911	10.25	10.807	9.998
Average	**18.198**	**17.712**	**19.633**	**16.885**	**12.452**	**12.563**

Methods	**P-value**	**Methods**	**P-value**
CCF - CCA	0.891190647	LGCF - CCF	0.323994389
GCA - CCA	0.991223298	GCF - GCA	0.999894113
GCF - CCA	0.999227051	*LCCF - GCA	0.001898770
*LCCF - CCA	0.015750864	*LGCF - GCA	0.002870613
*LGCF - CCA	0.023141411	*LCCF - GCF	0.004473095
GCA - CCF	0.549500178	*LGCF - GCF	0.007104076
GCF - CCF	0.706045867	LGCF - LCCF	0.999996612
LCCF - CCF	0.260984468		

Table 5.8: Top: Computational time in seconds. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **5 missing values per instance**)

Data set	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	43.78	37.82	22.634	17.438
CMSC1	44.31	36.17	19.736	14.756
CMSC2	41.27	36.96	18.468	16.252
CMSC3	37.51	29.79	17.54	14.872
CB	157.21	76.67	70.296	34.38
BCWD	75.5	37.13	28.58	14.81
FOTP	117.01	56.33	61.528	30.272
IoPh	81.68	44.24	36.134	19.368
QSAR	70.71	37.52	43.926	23.852
Spamb	78.14	38.79	34.128	16.744
Average	**74.712**	**43.142**	**35.297**	**20.274**

Methods	**P-value**	**Methods**	**P-value**
GCF - CCF	1.601002e-01	*LCCF - GCF	9.055305e-07
*LCCF - CCF	9.780764e-03	*LGCF - GCF	9.960305e-03
LGCF - CCF	7.263500e-01	LGCF - LCCF	1.600525e-01

Table 5.9: Average AUC on UCI data sets (Setting: **10 missing values per instance in one third of training instances**)

Data set	**NMV**	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**	**Real**
CMSC	0.5	0.571	0.571	0.571	0.59	0.578	0.579	0.596
CMSC1	0.631	0.563	0.563	0.589	0.643	0.561	0.633	0.68
CMSC2	0.5	0.532	0.532	0.574	0.561	0.539	0.539	0.549
CMSC3	0.5	0.553	0.529	0.514	0.514	0.54	0.534	0.575
CB	0.605	0.64	0.64	0.639	0.652	0.641	0.695	0.642
BCWD	0.848	0.885	0.885	0.879	0.868	0.888	0.879	0.92
FOTP	0.549	0.57	0.579	0.568	0.579	0.587	0.577	0.592
IoPh	0.68	0.736	0.727	0.754	0.761	0.778	0.771	0.841
QSAR	0.601	0.64	0.64	0.702	0.677	0.655	0.674	0.713
Spamb	0.723	0.751	0.75	0.744	0.752	0.777	0.756	0.749
Average	**0.614**	**0.644**	**0.642**	**0.654**	**0.659**	**0.654**	**0.664**	**0.686**

We performed different experiments with different number of missing attribute-values to investigate the effect of the missing attribute-value ratio in the data. Table 5.3, 5.6 and 5.9 show the AUC of different approaches on the UCI data sets, having 3,5 and 10 missing values in each instance of one third of the training set. The average AUC of NMV is the lowest one because this approach throws away a major part of the training set, one third of instances in this case, by removing all the instances with at least one missing attribute-value. The approach which uses the Real values has the highest average of AUC. The goal of missing attribute value handling methods is to get as close as possible to this approach. These tables show that the two proposed methods, LGCF and LCCF, have the highest average of AUC among other missing value handling methods, however, the differences are not statistically significant. AUC is not a good measure to show how these methods are different because the results are very close to each other. The missing attribute-values added to the training set does not have much impact on the AUC. Adding a lot more missing attribute-values will degrade all the missing attribute value handling methods because the number of instances are limited and there would not be enough information available to handle the missing values well. Adding more instances, on the other hand, will remove the effect of the missing attribute values.

AUC is not able to completely show the difference between the missing attribute-

Table 5.10: Top: Manhattan Distance between the vector of calculated values by different approaches and the real values. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **10 missing values per instance**)

Data set	**GCA**	**CCA**	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	48.942	46.738	46.128	44.992	35.464	35.37
CMSC1	49.856	46.896	46.536	45.353	34.18	33.121
CMSC2	50.435	49.085	44.871	41.642	36.564	36.523
CMSC3	54.824	53.949	48.451	47.641	37.263	39.096
CB	34.679	32.423	34.011	29.073	24.049	22.678
BCWD	32.304	23.987	49.549	31.498	19.158	15.588
FOTP	27.22	30.533	28.587	25.961	12.775	15.616
IoPh	29.229	35.525	27.189	25.549	14.993	14.439
QSAR	24.621	25.053	63.941	41.325	17.712	16.964
Spamb	15.197	14.94	18.801	18.652	18.007	19.295
Average	**36.731**	**35.913**	**40.806**	**35.169**	**25.016**	**24.869**

Methods	**P-value**	**Methods**	**P-value**
CCF - CCA	0.839358992	LGCF - CCF	0.470221626
GCA - CCA	0.991226281	GCF - GCA	0.999894113
GCF - CCA	0.999226991	*LCCF - GCA	0.001771404
*LCCF - CCA	0.015651952	*LGCF - GCA	0.004513406
*LGCF - CCA	0.033422593	*LCCF - GCF	0.004562116
GCA - CCF	0.470366301	*LGCF - GCF	0.010649723
GCF - CCF	0.629200724	LGCF - LCCF	0.999894115
LCCF - CCF	0.324114573		

Table 5.11: Top: Computational time in seconds. Bottom: P-values calculated by Friedman's and Nemenyi test. (Setting: **10 missing values per instance**)

Data set	**GCF**	**CCF**	**LGCF**	**LCCF**
CMSC	62.56	54.04	29.526	26.28
CMSC1	57.74	48.5	28.342	22.722
CMSC2	58.72	53.7	24.118	22.992
CMSC3	53.86	44.24	25.514	21.31
CB	284.68	138.3	133.108	66.352
BCWD	123.38	63.22	53.232	28.004
FOTP	211.12	103.84	104.242	50.836
IoPh	144.46	76.92	63.398	34.264
QSAR	126.3	67.5	75.95	41.442
Spamb	146.86	73.94	63.74	31.616
Average	**126.968**	**72.42**	**60.117**	**34.582**

Methods	**P-value**	**Methods**	**P-value**
GCF - CCF	1.602034e-01	*LCCF - GCF	7.739482e-07
*LCCF - CCF	1.004063e-02	*LGCF - GCF	9.687821e-03
LGCF - CCF	7.263464e-01	LGCF - LCCF	1.601898e-01

value handling methods because the classifier is not only trained on the calculated values but it has access to other attribute-values and consequently is not much affected by the missing attribute-values. In order to show the accuracy of calculated values we use the Manhattan distance, between the vector of calculated values and the real values, for those approaches which calculate a missing value. The result of these experiments are shown in table 5.4, 5.7 and 5.10. These tables show that, in all the three experiments, the average value of Manhattan distance for LGCF and LCCF are smaller than other approaches. In other words, the new proposed approaches calculated the missing values which are more close to the real ones comparing to the other approaches. As Japkowicz and Shah (2011) suggested in their book, Friedman's test with 95% confidence interval, is applied to see if the proposed algorithms are significantly different in terms of the average value for the Manhattan distance. The calculated p-value for Friedman's test is equal to 2.887e-05, 1.814e-05 and 2.742e-05,1 in these three experiments respectively, which means that the null hypothesis, the classifiers being compared are alike, rejected and therefore the compared algorithms are significantly different. The Nemenyi test is used as post hoc test to find out what these differences correspond to precisely. The second part of these tables show the p-values calculated by Nemenyi test. These values show that in all the experiments, both of the new proposed approaches, LGCF and LCCF, are statistically significantly better than GCA, CCA and GCF.

We have also compared computational complexity of the new proposed methods, LGCF and LCCF, with their predecessors, GCF and CCF, to show how faster the new variations are. Table 5.5, 5.8 and 5.11 show the time elapsed to calculate missing values, in seconds, using these four approaches. The average of elapsed time for LGCF is lower than GCF and for LCCF is lower than CCF. The result of statistical tests on the bottom of these tables show that these differences are statistically significant. It means that each of these new variations of closest fit is statistically significantly faster than its predecessor.

These experiments show that, removing irrelevant attributes from the feature set, using attribute clustering, will not only improve the AUC of the classifier, trained on the training set, but it will also calculate values which are statistically significantly closer to the real ones and it will perform statistically significantly faster.

5.2 Improving Missing Attribute-value Handling at Prediction Time

When talking about missing attribute value handling, we should distinguish between missing attribute-values at induction time in the training data, and at prediction time in test data or the upcoming new instances. The missing attribute value handling methods at prediction time are generally categorized as two groups, the imputation and reduced-feature models. Reduced-feature models are known to have better predictive performance comparing to imputation-based methods but they have computational difficulties when faced with high dimensionality. To make reduced-feature models practical in such cases we propose the use of multi-view learning as classifier and attribute clustering to create the views for multi-view learner automatically. The splitting of the selected attributes into views, significantly reduces the number of possible patterns and consequently the number of models which need to be precomputed and stored for reduced-feature models. Our experimental results show that not only the number of models to be trained is significantly reduced but also the classification results, when using reduced-feature models, is statistically significantly improved.

5.2.1 Missing Attribute-value Handling at Prediction Time

When facing a missing attribute-value at prediction time, one of the following strategies could be applied:

Discarding the Instance with Missing Attribute Value

The simplest missing attribute value handling method at prediction time is to discard the instances with missing attribute-values. But this means that we decline to provide prediction for some of the instances which is not acceptable most of the time.

Obtaining the Missing Value by Paying a Cost

Another approach is to obtain the missing value by paying the cost to run some complimentary tests or to get it from a third party. This method may not be applicable in some cases or the cost may not be acceptable.

Imputation

Imputation is a group of methods which handle missing attribute-values, at prediction time, by replacing it with an estimation of its value or its distribution from the data. According to Hastie et al. (2001), imputation-based methods are the most common approaches used for handling missing attribute-values.

Reduced-feature Models

Another approach is to apply a specific prediction model for each instance, based on its available attributes, or in other words, its pattern of missing values. Saar-Tsechansky and Provost (2007) called such approaches as reduced-feature models. Their empirical evaluation showed that, although imputation methods are the most common, reduced-feature models have substantial better predictive performance. The difficulty of reduced-feature models is that for each instance with a particular pattern of missing values, a different model should be applied. The models can be created on-line, which involves computation time, or precomputed and stored, which involves storage of different models. The reduced-feature models are not practical, when faced with high dimentionality, since it requires to build a large number of models which is exponential in the number of attributes, n:

$$Number\,of\,possible\,patterns = 2^n \tag{5.4}$$

Saar-Tsechansky and Provost (2007) tried to address this problem by proposing a hybrid method that uses the reduced-feature models for frequent patterns of missing values, and imputation for other cases. It allows the user to manage the tradeoff between predictive performance and storage/computation cost. Their empirical results show that when reduced-feature models used, even for a few patterns, it improves the predictive performance substantially. This method still does not offer the use of reduced-feature model for all the cases and also the maximum number of precomputed models is still exponential in the number of attributes.

To reduce the number of attributes and consequently the number of precomputed models for reduced-feature models, Hong et al. (2009b) proposed an unsupervised attribute clustering approach for feature selection. The number of possible models, in this approach, is exponential to the number of all selected attributes, m:

$$Number\,of\,possible\,patterns = 2^m \quad m <= n \tag{5.5}$$

Therefore, if the number of selected attributes m is large, it is still problematic, because of the huge number of possible models. Additionally, if the dataset has many missing values, the reduced-feature models for m selected attributes will be problematic. In some cases very few attributes of the m selected ones may contain values, making it difficult to extract the pattern in the data. To address this problem Hong et al. (2009b) suggested to replace selected features that have missing attribute values, with another from the same cluster. This replacement will help to create effective feature sets, in such cases, by replacing the attribute with missing value with one of the available attributes from the same cluster. Then a new model will be trained on the new feature set. However, the feature replacement increases the number of possible models. The number of possible models with replacement is the m-combination of attribute set:

$$Number\,of\,possible\,patterns = \binom{n}{m} \quad m <= n \tag{5.6}$$

where n is the size of attribute set and m is the number of selected attributes by feature selection. The size of each model to create, is also based on the size of all selected attributes, m. If the number of selected attributes is high, this model, as a reduced-feature model, will still have high storage/computation cost. The attribute clustering method in this approach is unsupervised and it will also reduce the effectiveness of this approach, in terms of placing attributes into related clusters.

5.2.2 Multi-view learning with reduced-feature models

To make reduced-feature models practical, when faced with high dimensionality, we propose the use of multi-view learning as classifier and attribute clustering to create the views for multi-view learner automatically. The splitting of the selected attributes into views significantly reduces the number of possible patterns and, consequently, the number of models which need to be precomputed and stored for reduced-feature models. The number of possible patterns for each view is exponential in the number of attributes in that view. So the total number of possible patterns for all possible models for reduced-feature model is the sum of possible patterns of different views:

$$Number\,of\,possible\,patterns = \sum_{i=1}^{k} 2^{x_i}, \qquad \sum_{i=1}^{k} x_i = m \tag{5.7}$$

Where k is the number of views, x_i is the number of attributes in view i and m is the total number of selected attributes which distributed into different views.

Table 5.12: Comparing the number of possible models for reduced-feature model, using each approach; where n is the total number of attributes, m is the number of selected attributes, k is the number of views and is considered to be 2, and x_i is the number of attributes in view i.

n, m and x_i	2^n	$\binom{n}{m}$	2^m	$\sum_{i=1}^{k}\binom{n}{x_i}$	$\sum_{i=1}^{k}2^{x_i}$
50, 20 and 10	1e+15	4e+13	1e+6	**2e+10**	**2,048**
20, 10 and 5	1e+6	1e+5	1024	**3e+4**	**64**
10, 6 and 2	1,024	210	64	**135**	**12**

In the proposed framework, each attribute of a view comes from a cluster of relevant attributes. If there is a missing value for such an attribute, it could be replaced with an attribute from the same cluster in that view. This replacement will only affect the learner corresponding to that view and that is the only part of the model which will be changed and we just need to create a small model for the learner which used the view. If attribute replacement is used in our model, the upper boundary of the number of the possible patterns for view i is x_i-combination of the n attributes. Therefore the upper boundary for the total number of possible patterns with replacement is

$$Number\,of\,possible\,patterns = \sum_{i=1}^{k}\binom{n}{x_i}, \qquad \sum_{i=1}^{k}x_i = m \tag{5.8}$$

Equation 5.9 illustrates the relationship between the models without replacement, in terms of the size of the possible patterns. It shows how the size of the possible patterns is reduced respectively.

$$2^n \gg 2^m \gg \sum_{i=1}^{k}2^{x_i} \tag{5.9}$$

Equation 5.10 illustrates the relationship between the different models with replacement, in terms of the size of the possible patterns

$$2^n \gg \binom{n}{m} \gg \sum_{i=1}^{k}\binom{n}{x_i} \tag{5.10}$$

For a better illustration of the differences between these approaches, table 5.12 compares the number of possible models for reduced-feature models in each approach. In this table specific values are considered for n, m, k and x_i. For convenience we assume equal size views for our model in this table, so x_i is a constant for every i. This table shows how significantly our approach reduced the number of possible models which need to be precomputed and stored.

Table 5.13 also shows the number of possible models for reduced-feature models in each approach, for some of the well-known UCI data sets which contain missing attribute values. In this table, the number of selected attributes m is considered to be half of the total number of attributes n. The number of views k is considered to be fixed to 2. For convenience we assume equal size views for our model in this table, so x_i is a constant for every i and it is equal to m/k. This table shows how significantly our approach reduced the number of possible models which need to be precomputed and stored.

5.2.3 Evaluation

We theoretically showed that, applying reduced-feature models with multi-view learning, significantly reduces the number of possible models and makes it practical. As a new variation of reduced-feature models we need to run experiments to show if it still has better predictive performance comparing to imputation-based models. To show this we perform experiments using the data sets used in our previous experiments on multi-view learning.

Data Preparation and Configuration

The data sets used in our experiments include all the text data sets that defined and used in our experiments in chapter 4. Text data sets contain thousands of attributes, including many which does not have any helpful information for classification. Therefore, the classifiers applied on the whole feature set are not able to learn the pattern in the data and consequently are close to the random classifier. This problem is usually solved by selecting a subset of attributes using a feature selection algorithm (Jiang et al., 2004; Raskutti et al., 2002b). In these experiments, chi-squared measure is used to select 50 attributes and reduce the size of the problem. One fifth of each data set is used for testing, with a minimum of 100 instances for small data sets. The rest is used for training. 5 missing attribute-values are randomly added to each instance of the test data set.

J48 is used as the base classifier in these experiments. The average AUC has been calculated in each experiment, using five-fold cross validation. Five-fold cross validation is used because the number of instances is limited in these experiments. As Japkowicz and Shah (2011) suggested in their book, Friedman's test with 95% confidence interval, is applied to see if the proposed algorithms are significantly different in terms of AUC and the Nemenyi test is used as post hoc test to find out what these differences correspond to.

Competitor Algorithms

The proposed algorithm is compared with two different imputation-based methods and also the ideal case without missing attribute-values:

- **The Central Imputation** (Batista and Monard, 2002; Torgo, 2010) is one of the most frequently used imputation methods. It replaces the missing attribute-value by the mean (quantitative attribute) or mode (qualitative attribute) of all known values of that attribute.
- **The KNN Imputation** (Batista and Monard, 2002; Torgo, 2010) uses the k-nearest neighbors to fill in the missing attribute-values in a data set. For each instance with missing attribute-values, it will search for its k most similar instances and use the values of them to fill in the missing attribute-values. For discrete attributes it uses the most frequent value and for continuous attributes it uses the mean in the k-nearest neighbors.
- **The Ideal Case With No Missing Attribute-value** is an approach which has access to all the attribute-values and is used to show how accurate each missing attribute value handling method is in calculating or predicting the missing values.

Results and Discussion

Table 5.14 shows our experimental results. The first part includes the average AUC of different approaches compared on different data sets. It shows that the average AUC of reduced-feature models is higher than the ones of the two imputation-based methods, while it is pretty close to the average AUC of the case without any missing attribute value.

To find out if these differences are statistically significant we used Friedman's test with 95% confidence interval. The resulting p-value is 0.0001023 which indicates that the results are statistically significantly different. The Nemenyi test is used as post hoc test to find out what these differences correspond to. The second part of table 5.14 contains

Table 5.13: Comparing the number of possible models for reduced-feature model, using each approach for UCI datasets; where n is the total number of attributes, m is the number of selected attributes and is considered to be half of n, k is the number of views and is considered to be 2, x_i is the number of attributes in view i, and is considered to be equal to m/k

UCI Dataset	n, m and x_i	2^n	$\binom{n}{m}$	2^m	$\sum_{i=1}^{k}\binom{n}{x_i}$	$\sum_{i=1}^{k}2^{x_i}$
Spambase	57 , 28 and 14	1e+17	1e+16	2e+8	**7e+12**	**32768**
Annealing	38 , 19 and 10	2e+12	3e+10	5e+5	**4e+8**	**2048**
Soybean	35 , 18 and 9	3e+10	4e+9	2e+5	**7e+7**	**1024**
Ionosphere	34 , 17 and 8	1e+10	2e+9	1e+5	**1e+7**	**512**
Dermatology	33 , 16 and 8	8e+9	1e+9	6e+4	**1e+7**	**512**
HIGGS	28 , 14 and 7	2e+8	4e+7	1e+4	**1e+6**	**256**
Cardiotocography	23 , 12 and 6	8e+6	1e+6	4096	**1e+5**	**128**
Parkinsons	23 , 12 and 6	8e+6	1e+6	4096	**1e+5**	**128**
Mushroom	22 , 11 and 6	4e+6	7e+5	2048	**7e+4**	**128**
Hepatitis	19 , 10 and 5	5e+5	9e+4	1024	**1e+4**	**64**
Lymphography	18 , 9 and 4	2e+5	4e+4	512	**3092**	**32**
Bank Marketing	17 , 8 and 4	1e+5	2e+4	256	**2412**	**32**
Adult	14 , 7 and 4	1e+4	3432	128	**1033**	**32**
Statlog (Heart)	13 , 6 and 3	8192	1716	64	**302**	**16**

the p-values of this post hoc test. It indicates that reduced-feature models is statistically significantly better than both imputation based methods in terms of AUC, while the two imputation-based methods are not considered different.

5.3 Summary

This chapter proposes two new usages for attribute clustering to improve missing attribute value handling at induction and prediction time. When labeled data is limited, these approaches are applicable using our semi-supervised attribute clustering framework.

In first approach, closet fit-based attribute clustering approaches are improved by changing the definition of closest fit to be the closest instance from the pool, considering the subset of related attributes to the one with missing attribute-value. Attribute clustering is used in this approach to create the clusters of related attributes. Our experimental results, on real world data sets from the UCI data repository, show that removing the irrelevant attributes from the feature set using attribute clustering in the new variation of closest fit method, will not only improve the AUC of the classifier, trained on the training set, but it will also calculate values which are statistically significantly closer to the real ones and it will perform statistically significantly faster.

In second part of this chapter, the attribute clustering and multi-view learning are used to make the reduced-feature models practical, by decreasing the number of possible patterns for missing attribute values. We theoretically showed that splitting attributes in views will significantly decrease the number of precomputed classifiers, needed in order to apply reduced-feature models as a missing attribute-value handling method. Using some of the well known UCI data sets, we showed that how this approach significantly reduces the number of possible models which need to be precomputed and stored to apply reduced-feature models. Our experimental results also showed that applying reduced-feature models with multi-view learning and attribute clustering, not only reduces the number of models, but it also increases the predictive performance of the classifer comparing to imputation-based missing attribute-value handling methods.

Next chapter summarizes the proposed framework and the usages of it to solve different problems. It mentions the limitations of proposed approach and our future work to address them.

Table 5.14: Comparing the AUC of applying reduced-feature models in multi-view learning with central and KNN imputation.

UCI Dataset	**Central Imputation**	**KNN Imputation**	**Reduced -feature Models**	**Ideal (No missing value)**
fbis(111,142)	0.931	0.989	**0.998**	1
fbis(111,189)	0.92	0.985	**0.999**	1
fbis(142,189)	0.838	0.898	**0.996**	0.997
new3s(111,142)	0.881	0.978	**0.992**	0.999
new3s(189,142)	0.898	0.987	**0.994**	0.999
new3s(189,301)	0.745	0.91	**0.986**	0.996
ohscal(An,Ca)	0.886	0.895	**0.964**	0.981
ohscal(An,Dn)	0.894	0.903	**0.94**	0.944
ohscal(Ca,Dn)	0.928	0.912	**0.968**	0.967
ohscal(Ca,To)	0.808	0.853	**0.915**	0.931
ACEInhibitors	0.54	0.569	**0.52**	0.535
ADHD	0.876	0.738	**0.937**	0.955
BB	0.601	0.589	**0.587**	0.565
CalciumC	0.646	0.678	**0.696**	0.733
Average	**0.787**	**0.82**	**0.869**	**0.872**

Methods	**P-value**
KNN Imputation - Central Imputation	2.495400e-01
Reduced-feature - Central Imputation	6.821229e-05
Reduced-feature - KNN Imputation	2.184101e-02

Chapter 6

Conclusion

This thesis proposed a novel iterative approach to attribute clustering and semi-supervised learning. This iterative approach exploits the strength of semi-supervised classification to gradually improve the quality of attribute clustering, particularly when labeled data is limited. It also applies the result of attribute clustering to improve the classification results iteratively. This improvement is done based on the usages of attribute clustering. In this study, we proposed two new usages for attribute clustering to improve classification: solving the automatic view definition problem for multi-view learning and improving missing attribute-value handling at induction and prediction time. The following two sections talk about these two specific contributions of this study. The third section discusses the overarching framework, which is used and evaluated in these two applications and is anticipated to be helpful in solving many other machine learning problems. The advantages, limitations and future work come in different sorts. Those of the overarching framework itself and those of its applications. Therefore, a separate subsection is included for this in each of the following sections.

6.1 Automatic View Definition for Multi-view Learning

Recent work has shown that utilizing the agreement between learners, based on different views, improves performance. Yet, a methodology for creating views automatically from data sets, in real world applications for which we do not have large labeled data sets, has not been proposed. In this study, we proposed the use of attribute clustering in our semi-supervised framework, in order to create effective views for single view data sets, when only a small number of labeled instances are available. This will make the multi-view learners applicable and improve their learning performance. In this new approach we integrate view splitting with the learning process by combining the attribute clustering with multi-view learning, in a semi-supervised manner. The proposed approach is evaluated using experiments on real world text and microarray gene expression data sets. Our experimental results showed that the proposed approach is significantly better than other methods. In fact, in many cases it is close to and not significantly different from the ideal, yet unrealistic, supervised classifier, which has access to labels for all instances.

6.1.1 Advantages, Limitations and Future Work

This new approach provides a solution to use multi-view learning on single view data sets, specifically when labeled data is limited. This is done by integrating attribute clustering with multi-view learning. This approach addresses the problem recently identified by Du et al. (2011): given small labeled training sets the view splitting methods are unreliable.

Another advantage is that, unlike other proposed view splitting methods, it is not limited to two separate views. This approach creates multiple overlapping views and uses those which represent enough information about the class, in terms of, Chi-Squared measure.

When labeled data is more common, this approach will not be so beneficial. With sufficient labeled instances supervised attribute clustering could be used to create views and there is little point in using a semi-supervised approach. We will investigate, in our future work, the effect of the proposed view splitting method for creating the views in a supervised manner.

6.2 Missing Attribute-value Handling

Many well-known missing attribute-value handling methods, where the value is replaced during induction, are designed based on the idea of closest fit. However, the computational cost of such methods is high because to find the missing value of an instance, they need to calculate its distance to all other instances. Additionally, the feature space can include attributes which are completely unrelated to the one with the missing value and these make the distance estimate unreliable. When looking for the closest fit in such cases, using a subset of the feature space, reduces the computational cost. If the subset includes all of the related attributes, it will improve the accuracy of the value found for replacing the missing attribute-value.

In this study we proposed a new approach, which uses attribute clustering to limit the feature space to the related attributes. This will remove the irrelevant attributes while it is drastically decreasing the computational cost. When there is not enough labeled instances available to perform supervised attribute clustering, the proposed semi-supervised framework in chapter 3 would be applicable. The result of attribute clustering is used in each iteration to improve the classifier by improving the missing attribute value handling and the classifier is used in each iteration to improve the attribute clustering by augmenting its training set. Therefore the result should be a better missing attribute value handling and also an improved semi-supervised classifier. Our experimental results show that removing irrelevant attributes from the feature set using attribute clustering will not only improve the AUC of the classifier, trained on the training set, but it will also calculate values which are statistically significantly closer to the real ones and it will calculate the values statistically significantly faster.

Empirical evaluation showed that, among different approaches which are proposed for missing attribute-value handling at prediction time, reduced-feature models have substantial better predictive performance. This approach applies a specific prediction model for each instance, based on its available attributes, or in other words, its pattern of missing values. The difficulty of reduced-feature models is that for each instance with a particular pattern of missing values, a different model should be applied. The models can be created on-line, which involves long computational cost, or precomputed and stored, which involves storage of different models which is exponential in the number of attributes. Some new variations of reduced-feature models slightly reduced the number of precomputed models needed. We addressed this problem in our proposed semi-supervised attribute clustering framework, when multi-view learner is used as classifier and attribute clustering is used to create the views for multi-view learner automatically. We showed that splitting of the selected attributes into views, in our approach, significantly reduces the number of possible patterns and consequently the number of the models which need to be precomputed and stored for reduced-feature model.

6.2.1 Advantages, Limitations and Future work

One of the main advantages of the proposed missing attribute value handling method is that it calculates values which are very close to the real ones. It is also faster than other methods. Another advantage is that, using the semi-supervised attribute clustering framework, it could be applied in many real world problems, which do not have a large labeled training set.

If the relationships between attributes is weak then the attribute clustering approach might not create meaningful clusters of attributes. In that case, removing part of the feature set will increase the speed to find the closest fit but it will not necessarily help to find a better closest fit. If the number of missing attribute-values is high in the remaining attributes in the feature set it might also be problematic. Therefore, a decision needs to be made for handling each missing value, whether to use the cluster of attributes to reduce the size of feature set or not, based on the relevance of the attributes in corresponding cluster. We will investigate this in future work. We will also investigate the effect of the new definition for closest fit on other closest fit-based missing attribute value handling methods.

The main advantage of the proposed variation of the reduced-feature models method for missing value handling at prediction time is that it significantly reduced the number of the models which need to be precomputed. This makes the reduced-feature models approach applicable in many real world problems with large number of attributes.

Although the proposed method, significantly decreased the size of the reduced-feature models, but the number of precomputed models could still be high, depending on the number of attributes in the clusters. This new approach, if used with other missing attribute value handling methods in a hybrid model, which creates the reduced-feature models only for frequent patterns of missing values, would even reduce the number of precomputed models more. We will investigate the performance of such a hybrid model in our future work.

6.3 The Overarching Semi-supervised Attribute Clustering Framework

In real world problems, usually labeled data is limited, while large number of unlabeled instances are available. The machine learning techniques, which rely on the relationships between attributes, extracted using supervised similarity or distance measures, are not reliable, because these measures are not able to find the real relationships between attributes. Unsupervised measures, which are used as an alternative in such cases, are not able to extract the relationships between attributes which are based on the class values.

Therefore, a semi-supervised approach would be beneficial in such cases. Attribute clustering is a machine learning technique, which uses the relationships between attributes to cluster similar or related attributes. Although there is a strong appeal for a more efficient and effective attribute clustering method, most of the work in this area is concentrated on supervised and unsupervised attribute clustering. This problem motivated us, in this study, to propose an iterative semi-supervised attribute clustering framework, shown in figure 1.4. In this framework, in each iteration, a classifier will be used to improve attribute clustering by augmenting its labeled training set. Then attribute clustering will be used to improve the classifier in the first step of the next iteration. How this improvement is realized will depend on the usages of attribute clustering: feature selection, extraction or replacement.

6.3.1 Advantages, Limitations and Future work

One of the main advantages of this semi-supervised attribute clustering framework is that, it could be applied on many real world problems in which we have limited training set and large number of unlabeled instances. This framework not only outputs semi-supervised attribute clustering but it also outputs an improved semi-supervised classifier.

The usage of attribute clustering in real world problems used to be limited because of the lack of a semi-supervised attribute clustering approach. One of the advantages of the proposed semi-supervised approach is that the attribute clustering could be used in different ways to improve the classifier in the framework. We have proposed two new usages for attribute clustering and investigated their performance when used in our semi-supervised attribute clustering framework. In our future work we will extend the usages of attribute clustering using our semi-supervised attribute clustering framework.

Although one of the primary goals of using attribute clustering is to reduce the computational complexity, the iterative algorithm of the proposed framework tends to be computationally expensive, especially when unlabeled data is unlimited. Therefore, the conventional stopping criterion for semi-supervised algorithms, which continues while more new confident labeled instances are available, is not of much interest here. A stricter stopping criterion would stop the iterative algorithm after a few iterations, when the learning curve becomes almost a straight line. Future work will investigate altering the stopping criterion to eliminate this problem.

Appendix A

Appendix

	Sim ple	Ran dom	Entr opy	Co-Style	Un Sup	Self	Ours	Ideal
OVA:								
Ovarian	0.941	0.918	0.948	0.935	0.99	0.991	**0.998**	0.996
Uterus	0.678	0.836	0.785	0.824	0.866	0.843	**0.894**	0.92
Prostate	0.967	0.935	0.939	0.948	0.925	0.944	**0.971**	0.992
Ovary	0.774	0.863	0.85	0.861	0.855	0.841	**0.888**	0.922
Omentum	0.6	0.752	0.746	0.795	0.81	0.783	**0.821**	0.899
Lung	0.797	0.884	0.889	0.876	0.836	0.854	**0.886**	0.94
Kidney	0.881	0.947	0.949	0.952	0.935	0.95	**0.976**	0.978
Endometrium	0.667	0.832	0.852	0.859	0.855	0.881	**0.901**	0.946
Colon	0.795	0.896	0.903	0.897	0.849	0.853	**0.94**	0.947
Breast	0.753	0.925	0.929	0.925	0.916	0.89	**0.965**	0.971
AP:								
Uterus_Kidney	0.986	0.964	0.958	0.967	0.986	0.987	**0.993**	0.99
Prostate_Uterus	0.988	0.994	0.993	0.993	0.991	0.991	**0.999**	0.994
Prostate_Ovary	0.99	0.979	0.981	0.985	0.99	0.984	**0.994**	0.982
Prostate_Lung	0.989	0.962	0.962	0.99	0.976	0.98	**0.995**	0.977
Prostate_Kidney	0.98	0.969	0.965	0.973	0.978	0.981	**0.991**	0.992
Ovary_Uterus	0.667	0.848	0.873	0.828	0.878	0.891	**0.905**	0.92
Ovary_Lung	0.864	0.929	0.933	0.898	0.958	0.96	**0.971**	0.954
Ovary_Kidney	0.977	0.922	0.91	0.891	0.984	0.984	**0.984**	0.989
Omentum_Uterus	0.73	0.895	0.888	0.904	0.958	0.94	**0.963**	0.951

Omentum_Ovary	0.633	0.744	0.737	0.663	0.785	0.804	**0.822**	0.831
Omentum_Lung	0.909	0.9	0.903	0.877	0.962	0.966	**0.97**	0.969
Omentum_Kidney	0.971	0.977	0.974	0.974	0.977	0.963	**0.974**	0.984
Lung_Uterus	0.89	0.937	0.926	0.921	0.975	0.976	**0.986**	0.967
Lung_Kidney	0.965	0.966	0.963	0.944	0.989	0.989	**0.994**	0.995
Endomet_Uterus	0.505	0.611	0.715	0.591	0.798	0.816	**0.828**	0.867
Endomet_Ovary	0.604	0.838	0.85	0.814	0.887	0.881	**0.928**	0.952
Endomet_Lung	0.86	0.949	0.953	0.957	0.978	0.981	**0.986**	0.983
Endomet_Kidney	0.839	0.975	0.979	0.988	0.978	0.977	**0.998**	0.989
Endomet_Colon	0.889	0.967	0.973	0.968	0.959	0.957	**0.979**	0.985
Endomet_Breast	0.898	0.936	0.934	0.944	0.96	0.951	**0.975**	0.974
Colon_Uterus	0.977	0.956	0.963	0.927	0.966	0.965	**0.982**	0.973
Colon_Prostate	0.987	0.982	0.98	0.988	0.988	0.989	**0.996**	0.995
Colon_Ovary	0.866	0.894	0.9	0.88	0.929	0.928	**0.966**	0.966
Colon_Omentum	0.91	0.915	0.908	0.917	0.943	0.942	**0.956**	0.952
Colon_Lung	0.933	0.954	0.956	0.947	0.971	0.968	**0.982**	0.972
Colon_Kidney	0.983	0.97	0.978	0.955	0.986	0.988	**0.994**	0.987
Breast_Uterus	0.891	0.942	0.945	0.911	0.953	0.95	**0.964**	0.967
Breast_Prostate	0.979	0.962	0.95	0.986	0.983	0.982	**0.994**	0.993
Breast_Ovary	0.832	0.962	0.949	0.938	0.972	0.975	**0.988**	0.972
Breast_Omentum	0.851	0.935	0.929	0.933	0.914	0.91	**0.932**	0.931
Breast_Lung	0.9	0.934	0.925	0.945	0.967	0.958	**0.976**	0.963
Breast_Kidney	0.905	0.963	0.965	0.924	0.98	0.98	**0.984**	0.985
Breast_Colon	0.959	0.957	0.96	0.959	0.99	0.987	**0.991**	0.989
Average	**0.86**	**0.916**	**0.918**	**0.911**	**0.938**	**0.937**	**0.958**	**0.963**

Table A.1: Average AUC calculated by Friedman's and Nemenyi test On **Microarray data sets** (**3 labeled instances**, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
OVA:								
Ovarian	0.936	0.98	0.976	0.972	0.991	0.991	**0.998**	0.996
Uterus	0.803	0.833	0.836	0.817	0.874	0.835	**0.887**	0.92

Prostate	0.956	0.928	0.933	0.935	0.937	0.943	**0.972**	0.992
Ovary	0.891	0.833	0.846	0.843	0.854	0.824	**0.881**	0.922
Omentum	0.804	0.777	0.759	0.797	0.763	0.757	**0.809**	0.898
Lung	0.93	0.855	0.861	0.846	0.803	0.829	**0.863**	0.941
Kidney	0.97	0.921	0.921	0.935	0.916	0.937	**0.976**	0.978
Endometrium	0.868	0.84	0.848	0.86	0.863	0.866	**0.9**	0.945
Colon	0.917	0.862	0.864	0.87	0.837	0.854	**0.933**	0.947
Breast	0.955	0.886	0.888	0.902	0.936	0.883	**0.966**	0.971
AP:								
Uterus_Kidney	0.973	0.976	0.98	0.979	0.984	0.984	**0.993**	0.99
Prostate_Uterus	0.991	0.991	0.99	0.993	0.993	0.993	**0.999**	0.993
Prostate_Ovary	0.99	0.973	0.97	0.984	0.987	0.987	**0.993**	0.982
Prostate_Lung	0.968	0.953	0.954	0.974	0.977	0.974	**0.992**	0.977
Prostate_Kidney	0.98	0.958	0.956	0.972	0.981	0.98	**0.987**	0.992
Ovary_Uterus	0.908	0.872	0.876	0.862	0.898	0.861	**0.92**	0.921
Ovary_Lung	0.947	0.932	0.913	0.928	0.965	0.957	**0.978**	0.954
Ovary_Kidney	0.981	0.963	0.963	0.964	0.983	0.985	**0.985**	0.989
Omentum_Uterus	0.91	0.884	0.898	0.92	0.954	0.922	**0.967**	0.952
Omentum_Ovary	0.755	0.738	0.744	0.736	0.811	0.799	**0.833**	0.829
Omentum_Lung	0.945	0.903	0.91	0.892	0.96	0.965	**0.97**	0.967
Omentum_Kidney	0.972	0.98	0.976	0.971	0.977	0.976	**0.979**	0.984
Lung_Uterus	0.965	0.952	0.942	0.93	0.982	0.977	**0.984**	0.968
Lung_Kidney	0.951	0.967	0.973	0.964	0.989	0.989	**0.994**	0.995
Endomet_Uterus	0.704	0.818	0.759	0.761	0.835	0.769	**0.879**	0.874
Endomet_Ovary	0.859	0.847	0.843	0.863	0.909	0.827	**0.933**	0.95
Endomet_Lung	0.977	0.954	0.956	0.959	0.978	0.978	**0.985**	0.983
Endomet_Kidney	0.986	0.975	0.971	0.98	0.974	0.975	**0.997**	0.989
Endomet_Colon	0.94	0.964	0.964	0.966	0.961	0.959	**0.978**	0.985
Endomet_Breast	0.953	0.923	0.925	0.941	0.953	0.943	**0.977**	0.974
Colon_Uterus	0.972	0.954	0.96	0.961	0.975	0.966	**0.983**	0.973
Colon_Prostate	0.993	0.98	0.98	0.983	0.99	0.99	**0.996**	0.995
Colon_Ovary	0.905	0.896	0.907	0.904	0.93	0.925	**0.955**	0.966
Colon_Omentum	0.926	0.919	0.912	0.927	0.931	0.937	**0.952**	0.951
Colon_Lung	0.968	0.955	0.953	0.951	0.972	0.969	**0.983**	0.972
Colon_Kidney	0.985	0.981	0.983	0.985	0.986	0.988	**0.995**	0.987

Breast_Uterus	0.943	0.926	0.93	0.943	0.957	0.949	**0.971**	0.967
Breast_Prostate	0.98	0.941	0.941	0.974	0.976	0.98	**0.986**	0.993
Breast_Ovary	0.964	0.96	0.965	0.967	0.975	0.977	**0.986**	0.975
Breast_Omentum	0.944	0.913	0.92	0.923	0.913	0.911	**0.933**	0.931
Breast_Lung	0.963	0.934	0.938	0.946	0.968	0.96	**0.974**	0.964
Breast_Kidney	0.954	0.971	0.969	0.97	0.98	0.983	**0.981**	0.985
Breast_Colon	0.959	0.969	0.961	0.963	0.991	0.987	**0.992**	0.989
Average	**0.933**	**0.919**	**0.919**	**0.924**	**0.939**	**0.931**	**0.958**	**0.963**

Table A.2: Average AUC calculated by Friedman's and Nemenyi test On **Microarray data sets** (5 labeled instances, 50 attributes, Naïve Bayes)

	Sim ple	**Ran dom**	**Entr opy**	**Co-Style**	**Un Sup**	**Self**	**Ours**	**Ideal**
OVA:								
Ovarian	0.985	0.98	0.978	0.978	0.991	0.991	**0.998**	0.996
Uterus	0.825	0.83	0.844	0.829	0.87	0.832	**0.91**	0.919
Prostate	0.969	0.935	0.938	0.94	0.939	0.948	**0.974**	0.992
Ovary	0.879	0.851	0.849	0.846	0.851	0.839	**0.881**	0.923
Omentum	0.666	0.748	0.765	0.77	0.746	0.789	**0.792**	0.899
Lung	0.829	0.852	0.856	0.866	0.799	0.842	**0.855**	0.941
Kidney	0.951	0.928	0.928	0.928	0.923	0.936	**0.973**	0.978
Endometrium	0.906	0.858	0.859	0.856	0.858	0.877	**0.898**	0.945
Colon	0.88	0.866	0.863	0.887	0.842	0.858	**0.935**	0.947
Breast	0.929	0.882	0.888	0.891	0.942	0.891	**0.965**	0.971
AP:								
Uterus_Kidney	0.982	0.98	0.981	0.981	0.986	0.984	**0.992**	0.99
Prostate_Uterus	0.994	0.991	0.991	0.993	0.991	0.991	**0.998**	0.993
Prostate_Ovary	0.98	0.966	0.956	0.986	0.987	0.987	**0.99**	0.985
Prostate_Lung	0.99	0.948	0.95	0.977	0.981	0.978	**0.991**	0.978
Prostate_Kidney	0.989	0.959	0.958	0.969	0.981	0.981	**0.987**	0.992
Ovary_Uterus	0.928	0.866	0.884	0.884	0.886	0.903	**0.934**	0.922
Ovary_Lung	0.951	0.943	0.921	0.945	0.963	0.965	**0.974**	0.954
Ovary_Kidney	0.985	0.959	0.968	0.966	0.984	0.987	**0.982**	0.989

Omentum_Uterus	0.931	0.887	0.884	0.899	0.944	0.933	**0.966**	0.953
Omentum_Ovary	0.823	0.735	0.748	0.733	0.81	0.799	**0.833**	0.827
Omentum_Lung	0.968	0.917	0.922	0.921	0.969	0.97	**0.97**	0.969
Omentum_Kidney	0.983	0.977	0.972	0.976	0.978	0.977	**0.979**	0.984
Lung_Uterus	0.974	0.954	0.946	0.944	0.978	0.971	**0.983**	0.965
Lung_Kidney	0.988	0.976	0.978	0.958	0.989	0.99	**0.994**	0.995
Endomet_Uterus	0.752	0.821	0.812	0.803	0.865	0.837	**0.897**	0.878
Endomet_Ovary	0.874	0.857	0.853	0.88	0.906	0.864	**0.944**	0.952
Endomet_Lung	0.981	0.956	0.956	0.957	0.984	0.981	**0.986**	0.984
Endomet_Kidney	0.985	0.975	0.974	0.979	0.974	0.978	**0.999**	0.987
Endomet_Colon	0.976	0.964	0.962	0.969	0.965	0.964	**0.983**	0.985
Endomet_Breast	0.98	0.932	0.932	0.955	0.94	0.944	**0.977**	0.975
Colon_Uterus	0.975	0.953	0.957	0.963	0.975	0.968	**0.981**	0.973
Colon_Prostate	0.99	0.979	0.98	0.984	0.988	0.989	**0.995**	0.995
Colon_Ovary	0.955	0.918	0.911	0.916	0.937	0.938	**0.97**	0.967
Colon_Omentum	0.934	0.923	0.92	0.928	0.933	0.937	**0.957**	0.949
Colon_Lung	0.967	0.946	0.963	0.955	0.968	0.968	**0.983**	0.972
Colon_Kidney	0.985	0.983	0.983	0.983	0.985	0.99	**0.994**	0.987
Breast_Uterus	0.959	0.936	0.936	0.943	0.957	0.951	**0.969**	0.967
Breast_Prostate	0.98	0.948	0.956	0.972	0.981	0.984	**0.994**	0.993
Breast_Ovary	0.977	0.964	0.964	0.968	0.972	0.974	**0.986**	0.977
Breast_Omentum	0.931	0.911	0.917	0.924	0.911	0.91	**0.934**	0.93
Breast_Lung	0.957	0.948	0.939	0.946	0.967	0.961	**0.97**	0.963
Breast_Kidney	0.983	0.973	0.971	0.969	0.98	0.98	**0.983**	0.985
Breast_Colon	0.963	0.966	0.957	0.959	0.987	0.983	**0.988**	0.989
Average	**0.939**	**0.922**	**0.923**	**0.927**	**0.939**	**0.938**	**0.959**	**0.963**

Table A.3: Average AUC calculated by Friedman's and Nemenyi test On **Microarray data sets** (Setting:**10 labeled instances**, 50 attributes, Naïve Bayes)

Figure A.1: Parallel coordinates plot for experiments on **Microarray data sets** (**3 labeled instances**, 50 attributes, Naïve Bayes)

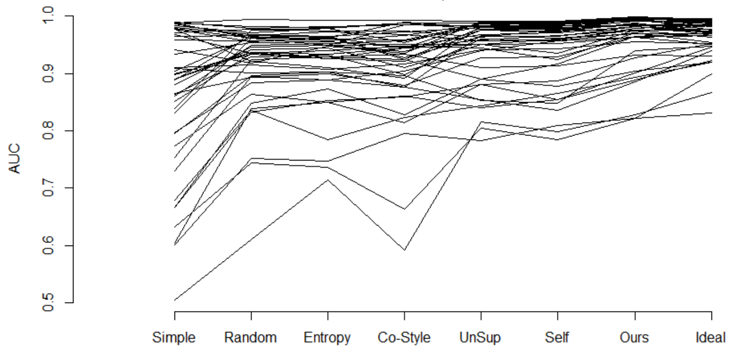


Figure A.2: Parallel coordinates plot for experiments on **Microarray data sets** (5 labeled instances, 50 attributes, Naïve Bayes)

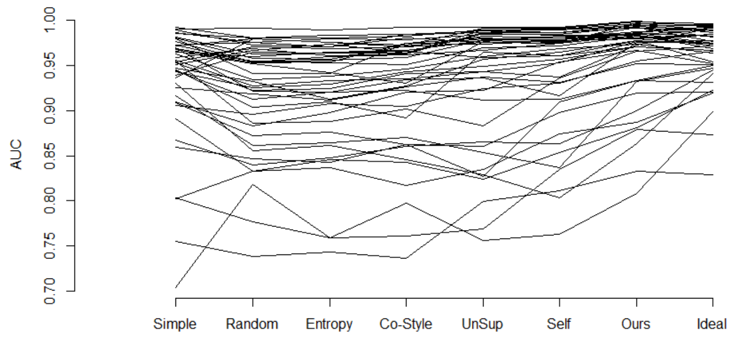
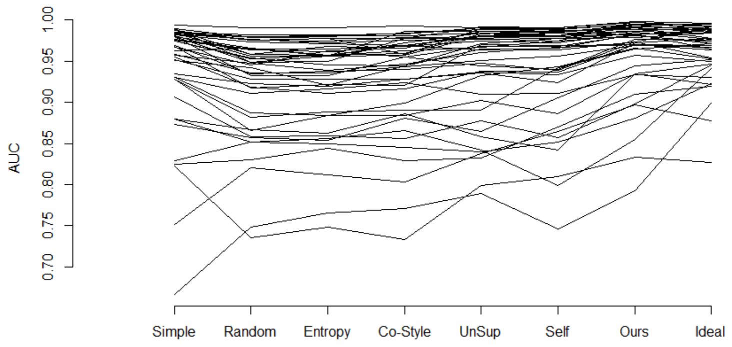


Figure A.3: Parallel coordinates plot for experiments on **Microarray data sets** (Setting:**10 labeled instances**, 50 attributes, Naïve Bayes)



Bibliography

Abney, S. Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 360–367. Association for Computational Linguistics, 2002.

Aha, D. W. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2):267–287, 1992.

Aha, D. W., Kibler, D., and Albert, M. K. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

Ando, R. K. and Zhang, T. Two-view feature generation model for semi-supervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 25–32. ACM, 2007.

Au, W.-H., Chan, K. C., Wong, A. K., and Wang, Y. Attribute clustering for grouping, selection, and classification of gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(2):83–101, 2005.

Bache, K. and Lichman, M. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Balcan, M.-F., Blum, A., and Yang, K. Co-training and expansion: Towards bridging theory and practice. In *Advances in neural information processing systems*, pages 89–96, 2004.

Batista, G. E. and Monard, M. C. A study of k-nearest neighbour as an imputation method. *HIS*, 87:251–260, 2002.

Bickel, S. and Scheffer, T. Multi-view clustering. In *ICDM*, volume 4, pages 19–26, 2004.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Bollier, D. and Firestone, C. M. *The promise and peril of big data.* Aspen Institute, Communications and Society Program Washington, DC, USA, 2010.

Boyd, D. and Crawford, K. Six provocations for big data. In *A decade in internet time: symposium on the dynamics of the internet and society*, 2011.

Brefeld, U. and Scheffer, T. Co-em support vector learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 16. ACM, 2004.

Brefeld, U. and Scheffer, T. Semi-supervised learning for structured output variables. In *Proceedings of the 23rd international conference on Machine learning*, pages 145–152. ACM, 2006.

Brefeld, U., Büscher, C., and Scheffer, T. Multi-view discriminative sequential learning. In *Machine Learning: ECML 2005*, pages 60–71. Springer, 2005.

Cha, S.-H. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International journal of mathematical models and methods in applied sciences*, 1(4):300–307, 2007.

Chen, M., Chen, Y., and Weinberger, K. Q. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 953–960, 2011.

Chizi, B. and Maimon, O. Dimension reduction and feature selection. In *Data Mining and Knowledge Discovery Handbook*, pages 83–100. Springer, 2010.

Choi, S.-S., Cha, S.-H., and Tappert, C. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

Cohen, A. M., Hersh, W. R., Peterson, K., and Yen, P.-Y. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2):206–219, 2006.

Covões, T. F., Hruschka, E. R., de Castro, L. N., and Santos, Á. M. A cluster-based feature selection approach. pages 169–176, 2009.

Di, W. and Crawford, M. M. View generation for multiview maximum disagreement based active learning for hyperspectral image classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(5):1942–1954, 2012.

Domany, E. Cluster analysis of gene expression data. *Journal of Statistical Physics*, 110 (3-6):1117–1139, 2003.

Douglas, L. The importance of big data: A definition. *Gartner (June 2012)*, 2012.

Du, J., Ling, C. X., and Zhou, Z.-H. When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering*, 23(5):788–799, 2011.

Elder IV, J. F. and Pregibon, D. A statistical perspective on knowledge discovery in databases. *Advances in knowledge discovery and data mining*, pages 83–113, 1996.

Forman, G. and Packard, H. 19 multi-class (1-of-n) text data sets available at weka data set website. `http://weka.wikispaces.com/Datasets`.

Frigui, H. and Mahdi, R. Semi-supervised clustering and feature discrimination with instance-level constraints. In *IEEE International Fuzzy Systems Conference, FUZZ-IEEE'07.*, pages 1–6. IEEE, 2007.

Gaur, S. Closest fit approach to handle odd size missing block values. *International Journal of Mathematical Archive (IJMA) ISSN 2229-5046*, 3(7), 2012.

Gaur, S. Estimation of missing value at extremes in data mining. *International Journal of Advance Foundation and Research in Computer (IJAFRC) ISSN 2348-4853*, 1(3), 2014.

Gaur, S. and Dulawat, M. A closest fit approach to missing attribute values in data mining. *International Journal of advances in Science and Technology*, 2(4):18–24, 2011.

Goldman, S. and Zhou, Y. Enhancing supervised learning with unlabeled data. In *ICML*, pages 327–334. Citeseer, 2000.

Grzymała-Busse, J. W. and Grzymala-Busse, W. J. Handling missing attribute values. In *Data mining and knowledge discovery handbook*, pages 37–57. Springer, 2005.

Grzymała-Busse, J. W., Grzymała-Busse, W. J., and Goodwin, L. K. A closest fit approach to missing attribute values in preterm birth data. In *New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, pages 405–413. Springer, 1999.

Grzymala-Busse, J. W., Grzymala-Busse, W. J., and Goodwin, L. K. A comparison of three closest fit approaches to missing attribute values in preterm birth data. *International journal of intelligent systems*, 17(2):125–134, 2002.

Grzymala-Busse, J. W., Goodwin, L. K., Grzymala-Busse, W. J., and Zheng, X. Handling missing attribute values in preterm birth data sets. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 342–351. Springer, 2005.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1): 10–18, 2009.

Hall, M. A. *Correlation-based feature selection for machine learning.* PhD thesis, The University of Waikato, 1999.

Han, E.-H. S. and Karypis, G. *Centroid-based document classification: Analysis and experimental results.* Springer, 2000.

Hastie, T., Tibshirani, R., and Friedman, J. J. H. *The elements of statistical learning*, volume 1. Springer New York, 2001.

Heyer, L. J., Kruglyak, S., and Yooseph, S. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9(11):1106–1115, 1999.

Hoi, S. C., Wang, J., Zhao, P., and Jin, R. Online feature selection for mining big data. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 93–100. ACM, 2012.

Hong, T.-P. and Liou, Y.-L. Attribute clustering in high dimensional feature spaces. In *2007 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2286–2289. IEEE, 2007.

Hong, T.-P., Wang, P.-C., and Lee, Y.-C. Automatic attribute clustering based on genetic algorithms. In *IEEE International Conference on Granular Computing, GRC'09.*, pages 214–218. IEEE, 2009a.

Hong, T.-P., Wang, P.-C., and Lee, Y.-C. An effective attribute clustering approach for feature selection and replacement. *Cybernetics and Systems: An International Journal*, 40(8):657–669, 2009b.

Hong, T.-P., Wang, P.-C., and Ting, C.-K. An evolutionary attribute clustering and selection method based on feature similarity. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–5. IEEE, 2010.

Japkowicz, N. and Shah, M. *Evaluating learning algorithms: a classification perspective.* Cambridge University Press, 2011.

Jiang, D., Tang, C., and Zhang, A. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

Jiang, J.-Y., Yin, K.-T., and Lee, S.-J. A confidence-based hierarchical feature clustering algorithm for text classification. In *The 2007 International Conference on Intelligent Pervasive Computing, IPC'07*, pages 161–164. IEEE, 2007.

Jiang, J.-Y., Liou, R.-J., and Lee, S.-J. A fuzzy self-constructing feature clustering algorithm for text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 23(3):335–349, 2011.

Jiang, W., Chang, S.-F., Jebara, T., and Loui, A. C. Semantic concept classification by joint semi-supervised learning of feature subspaces and support vector machines. In *Computer Vision–ECCV 2008*, pages 270–283. Springer, 2008.

John, G. H., Kohavi, R., and Pfleger, K. Irrelevant features and the subset selection problem. In *ICML*, volume 94, pages 121–129, 1994.

Kaufman, L. and Rousseeuw, P. J. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

Kim, S.-H., Kim, N.-U., and Chung, T.-M. Attribute relationship evaluation methodology for big data security. In *IT Convergence and Security (ICITCS), 2013 International Conference on*, pages 1–4. IEEE, 2013.

Kira, K. and Rendell, L. A. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992.

Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207, 1996.

Kohavi, R. and Frasca, B. Useful feature subsets and rough set reducts. In *Third International Workshop on Rough Sets and Soft Computing*, RSSC, pages 310–317, 1994.

Laney, D. 3d data management: Controlling data volume, velocity and variety. *META Group Research Note 6*, 2001.

Lanquillon, C. Partially supervised text classification: Combining labeled and unlabeled documents using an em-like scheme. In *Machine Learning: ECML 2000*, pages 229–237. Springer, 2000.

Li, G., Hu, X., Shen, X., Chen, X., and Li, Z. A novel unsupervised feature selection method for bioinformatics data sets through feature clustering. In *IEEE International Conference on Granular Computing, GrC'08*, pages 41–47. IEEE, 2008.

Li, J. A source coding approach to classification by vector quantization and the principle of minimum description length. In *Data Compression Conference, 2002. Proceedings. DCC'02*, pages 382–391. IEEE, 2002.

Li, J. and Zha, H. Simultaneous classification and feature clustering using discriminant vector quantization with applications to microarray data analysis. In *IEEE Computer Society Bioinformatics Conference, 2002. Proceedings.*, pages 246–255. IEEE, 2002.

Li, J. and Cercone, N. Comparisons on different approaches to assign missing attribute values. Technical report, SCS, Univ. of Waterloo, 2006.

Ling, X., Xue, G.-R., Dai, W., Jiang, Y., Yang, Q., and Yu, Y. Can chinese web pages be classified with english data source? In *Proceedings of the 17th international conference on World Wide Web*, pages 969–978. ACM, 2008.

Liu, H. and Motoda, H. *Feature selection for knowledge discovery and data mining.* Springer, 1998a. ISBN 079238198X.

Liu, H. and Motoda, H. *Feature extraction, construction and selection: A data mining perspective*. Springer, 1998b.

Lv, J., Chen, X., Huang, J., and Bao, H. Semi-supervised mesh segmentation and labeling. In *Computer Graphics Forum*, volume 31, pages 2241–2248. Wiley Online Library, 2012.

Maji, P. Fuzzy–rough supervised attribute clustering algorithm and classification of microarray data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(1):222–233, 2011.

Maji, P. Mutual information-based supervised attribute clustering for microarray sample classification. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):127–140, 2012.

Mampaey, M. and Vreeken, J. Summarising data by clustering items. In *Machine Learning and Knowledge Discovery in Databases*, pages 321–336. Springer, 2010.

Matloff, N. Long live (big data-fied) statistics! In *Proceedings of JSM*, pages 98–108, 2013.

Mendelson, E. Introduction to mathematical logic. the university series in undergraduate mathematics, 1964.

Mining, O. D. Oracle data mining concepts, 11g release 1. Technical report, Oracle Company, 05 2008.

Minsky, M. A framework for representing knowledge. 1974.

Mitra, P., Murthy, C., and Pal, S. K. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3): 301–312, 2002.

Nagi, S., Bhattacharyya, D. K., and Kalita, J. K. Gene expression data clustering analysis: A survey. In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science (NCETACS)*, pages 1–12. IEEE, 2011.

Niu, K., Zhang, S., and Chen, J. Subspace clustering through attribute clustering. *Frontiers of Electrical and Electronic Engineering in China*, 3(1):44–48, 2008.

Niu, Z.-Y., Ji, D.-H., and Tan, C. L. A semi-supervised feature clustering algorithm with application to word sense disambiguation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 907–914. Association for Computational Linguistics, 2005.

Orłowska, E. and Pawlak, Z. Expressive power of knowledge representation systems. *International Journal of Man-Machine Studies*, 20(5):485–500, 1984.

Palanikkumar, J. J. and Scholar, P. Combining supervised attribute clustering and ga-svm classifier for microarray sample classification. *International Journal of Engineering Research & Technology*, 2(6):785–789, 2013.

Pawlak, Z. Rough sets-theoretical aspect of reasoning about data. *Proceedings of Kluwer Academic Publishers*, 1991.

Paykel, E. Classification of depressed patients: a cluster analysis derived grouping. *The British Journal of Psychiatry*, 118(544):275–288, 1971.

Quillian, R. A notation for representing conceptual information: An application to semantics and mechanical english paraphrasing, sp-1395. *System Development Corporation, Santa Monica, California*, 1963.

Quinzán, I., Sotoca, J. M., and Pla, F. Clustering-based feature selection in semi-supervised problems. In *Ninth International Conference on Intelligent Systems Design and Applications. ISDA'09.*, pages 535–540. IEEE, 2009.

Rainie, H. and Wellman, B. Networked: The new social operating system. 2012.

Raskutti, B., Ferrá, H., and Kowalczyk, A. Combining clustering and co-training to enhance text classification using unlabelled data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 620–625. ACM, 2002a.

Raskutti, B., Ferrá, H., and Kowalczyk, A. Combining clustering and co-training to enhance text classification using unlabelled data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 620–625. ACM, 2002b.

Saar-Tsechansky, M. and Provost, F. Handling missing values when applying classification models. *Journal of machine learning research*, 2007.

Sammut, C. and Webb, G. I. *Encyclopedia of machine learning.* Springer-Verlag New York Incorporated, 2011. ISBN 0387307680.

Seeger, M. Learning with labeled and unlabeled data. Technical report, technical report, University of Edinburgh, 2001.

Seifi, F., Drummond, C., Japkowicz, N., and Matwin, S. Improving classification and attribute clustering, an iterative semi-supervised aproach. *Submitted to the Journal of Machine Learning Research.*

Simmons, R. F. *Synthetic language behavior.* System Development Corporation, 1963.

Sindhwani, V. and Rosenberg, D. S. An rkhs for multi-view learning and manifold co-regularization. In *Proceedings of the 25th international conference on Machine learning*, pages 976–983. ACM, 2008.

Slonim, D. K. and Yanai, I. Getting started in gene expression microarray analysis. *PLoS computational biology*, 5(10), 2009.

Snijders, C., Matzat, U., and Reips, U.-D. Big data: Big gaps of knowledge in the field of internet science. *International Journal of Internet Science*, 7(1):1–5, 2012.

Stiglic, G. and Kokol, P. Stability of ranked gene lists in large microarray analysis studies. *Journal of Biomedicine and Biotechnology*, 2010, 2010.

Suguna, N. and Thanushkodi, K. G. Predicting missing attribute values using k-means clustering. *Journal of Computer Science*, 7(2), 2011.

Tan, M., Tsang, I. W., and Wang, L. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research*, 15:1371–1429, 2014.

Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J., and Church, G. M. Systematic determination of genetic network architecture. *Nature genetics*, 22(3):281–285, 1999.

Tjhi, W.-C. and Chen, L. Flexible fuzzy co-clustering with feature-cluster weighting. In *9th International Conference on Control, Automation, Robotics and Vision. ICARCV'06*, pages 1–6. IEEE, 2006.

Torgo, L. *Data Mining using R: learning with case studies.* CRC Press, 2010. ISBN 9781439810187.

Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Van der Pouw Kraan, T., Wijbrandts, C., van Baarsen, L., Voskuyl, A., Rustenburg, F., Baggen, J., Ibrahim, S., Fero, M., Dijkmans, B., Tak, P., et al. Rheumatoid arthritis subtypes identified by genomic profiling of peripheral blood cells: assignment of a type i interferon signature in a subpopulation of patients. *Annals of the rheumatic diseases*, 66(8):1008–1014, 2007.

Vinh, N. X. and Phuong, N. M. An information theoretic divergence for microarray data clustering. In *BioInformatics and BioEngineering, 2008. BIBE 2008. 8th IEEE International Conference on*, pages 1–7. IEEE, 2008.

Wan, C., Pan, R., and Li, J. Bi-weighting domain adaptation for cross-language text classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1535, 2011.

Wang, W. and Zhou, Z.-H. Analyzing co-training style algorithms. In *Machine Learning: ECML 2007*, pages 454–465. Springer, 2007.

Wang, W. and Zhou, Z.-H. A new analysis of co-training. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1135–1142, 2010.

Wang, W. and Zhou, Z.-H. Co-training with insufficient views. In *Asian Conference on Machine Learning*, pages 467–482, 2013.

Wehenkel, L. *Machine Learning Approaches to Power System Security Assessment.* PhD thesis, University of Liège - Faculté des Sciences appliquées, May 1994. URL `http://www.montefiore.ulg.ac.be/services/stochastic/pubs/1994/Weh94`. Thèse d'agrégation de l'enseignement supérieur.

Winograd, T. Frame representations and the declarative/procedural controversy. *Representation and understanding: Studies in cognitive science*, pages 185–210, 1975.

Wu, X. and Srihari, R. Incorporating prior knowledge with weighted margin support vector machines. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–333. ACM, 2004.

Xu, C., Tao, D., and Xu, C. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

Yao, Y., Zhao, Y., Wang, J., and Han, S. A model of machine learning based on user preference of attributes. In *Rough Sets and Current Trends in Computing*, pages 587–596. Springer, 2006.

Zeng, Y., Hernandez, J. C., and Lin, S. Spanning tree based attribute clustering. pages 681–688, 2009.

Zhou, Z.-H. and Li, M. Semi-supervised regression with co-training. In *IJCAI*, pages 908–916, 2005.

Zhu, X. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

Ziarko, W. and Shan, N. A method for computing all maximally general rules in attribute-value systems. *Computational Intelligence*, 12(2):223–234, 1996.