Dimensionality Reduction and Representation for Nearest Neighbour Learning

A thesis presented for the degree of
Doctor of Philosophy
at the University of Aberdeen

Terry R. Payne

1999

Declaration

This thesis has been composed by myself, it has not been accepted in any previous application for a degree, the work of which it is a record has been done by myself and all quotations have been distinguished by quotation marks and the sources of information have specifically acknowledged.

Terry Payne
August 30, 1999

Department of Computing Science
University of Aberdeen
Kings College
Aberdeen

Abstract

An increasing number of intelligent information agents employ Nearest Neighbour learning algorithms to provide personalised assistance to the user. This assistance may be in the form of recognising or locating documents that the user might find relevant or interesting. To achieve this, documents must be mapped into a representation that can be presented to the learning algorithm. Simple heuristic techniques are generally used to identify relevant terms from the documents. These terms are then used to construct large, sparse training vectors. The work presented here investigates an alternative representation based on sets of terms, called *set-valued attributes*, and proposes a new family of Nearest Neighbour learning algorithms that utilise this set-based representation. The importance of discarding irrelevant terms from the documents is then addressed, and this is generalised to examine the behaviour of the Nearest Neighbour learning algorithm with high dimensional data sets containing such values. A variety of selection techniques used by other machine learning and information retrieval systems are presented, and empirically evaluated within the context of a Nearest Neighbour framework. The thesis concludes with a discussion of ways in which attribute selection and dimensionality reduction techniques may be used to improve the selection of relevant attributes, and thus increase the reliability and predictive accuracy of the Nearest Neighbour learning algorithm.

Acknowledgements

How can one catalogue the support and assistance I've received during my PhD years? A multitude of folk have made my time here in Aberdeen special, and I'm indebted to them all. However, there are a notable few to whom I owe special thanks.

First of all, there is my supervisor and friend, Pete Edwards, who has been responsible for my coming to, and remaining in Aberdeen. It is to him I owe thanks for introducing me to the concept of Machine Learning, and to Agent technology. Whilst his ink strewn pages of corrections drove me to utter distraction at times, he supported me in my ideas and projects; gave reasoned arguments to my embryonic theories, and became a good drinking companion at the end of many a difficult week.

Thanks should also go to many of my colleagues in the department: to Suzanne, who introduced me to Aberdeen several years ago; Faye, who helped me through those mid-term blues by reminding me that it could only get better (pah!); Mark (aka *Ging Dude*), and Tony (the *Dude*), for the many nights out and Mexican meals; to Matt (Grouse - cool!); Claire for chumming me down the Gym and sharing too many Cappuccinos after; Nick for keeping the Departmental network going (despite the exploding Beer Barrel incident - sorry folks); Stef, for being even worse than me; Dr Hunter and Prof Sleeman for authorising funds for all the conferences I've had the privilege to attend; Alex, for a storming time down in Manchester and some great times since; and of course to all the other MSc & Postgrads in the department, who are too numerous to mention - thanks people!

A special mention should also go to our secretarial staff; to Irene, Katie, Elizabeth, Fiona and Diane, and the countless other temps that passed through the main office.

Other folk here in Aberdeen also deserve a mention: the archery crowd, who may not have won the Scottish championships, but could drink the other teams under the table; to Sue, the best flatmate ever; Sarah and Corrie, who both share my reverence for good coffee; Christopher, Crash and Phil for those endless nights of philosophical debate; and of course Richard, for simply being Richard.

Finally, come two very important people in my life. My thoughts go to Heather, who, despite now being over 6000 miles away has been an absolute star and a true friend. And then there is my 'sister', Kate, whom quite simply I love to bits.

However, I can't finish without dedicating this thesis to the two most important people in my life, my mother and father. Nothing could ever eclipse the love, dedication, and support they've shown me over the last thirty or so years. I could never have achieved my dreams and made it this far without you both ...

For Roy...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter Outline

This chapter presents an introduction to intelligent information agents, and explains how they can assist users in retrieving and processing on-line information. Different information agents employ a variety of machine learning algorithms, so that the assistance provided can be tailored to a user's individual requirements. The work presented in this thesis will focus on the utility of the nearest neighbour learning algorithm within information agents, and will illustrate some of the problems encountered when attempting to learn from natural language texts such as electronic mail or USENET news articles.

The objectives of the work are presented and motivated within this chapter. This will then be followed by an overview of the remaining chapters.

1.1 Introduction

In recent years, the volume of information available electronically has increased at an unprecedented rate. Information that was originally published or broadcast by traditional media (such as paper or video tape) is now available in computer accessible form. Much of this is due to an increase in the number people that use the Internet and the development of the World Wide Web. Although accessing and utilising these data repositories is becoming crucial within modern business practices, it is becoming increasingly difficult to identify what documents or articles are interesting or relevant to the user. For this reason, *information agents* (Mitchell, Caruana, Freitag, McDermott, & Zabowski 1994; Maes 1994) have been proposed as one possible solution to the problem of assisting users in accessing relevant information and performing simple tasks.

Many information agents have been described as personal assistants, as they can be personalised for each individual user (Maes 1994). Although many early agent systems relied on user-defined scripts to determine their behaviour, recently systems have emerged which employ machine learning techniques to induce user preferences automatically. Information agents may sort incoming information, such as prioritising unread electronic mail messages, or locate new documents (such as World Wide Web pages) that a user would find interesting. Machine learning algorithms are employed to identify regularities that occur within existing messages or documents. For example, they can identify certain terms or keywords may only appear within interesting articles, or recognise authors that periodically send important messages. These regularities can then be utilised by information agents to assist the user.

A variety of learning mechanisms have been employed within intelligent information agents, including genetic algorithms (Sheth, 1994), symbolic rule induction algorithms (Dent et al., 1992; Payne, 1994; Bayer, 1995; Cohen, 1996a; Payne et al., 1997), neural networks (McElligott & Sorensen, 1994; Mitchell et al., 1994; Pannu & Sycara, 1996; Boone, 1998), naive Bayesian techniques (Pazzani et al., 1996), Minimum Description Length techniques (Lang, 1995), clustering tech-

niques (Green & Edwards, 1996), relational learning algorithms (Cohen, 1995) and nearest neighbour (or instance based) techniques (Metral, 1993; Kozierok & Maes, 1993; Green & Edwards, 1996; Pazzani et al., 1996; Payne et al., 1997; Boone, 1998).

There are a number of differences between the task of learning from documents and other learning problems (Lewis, 1992). The organisation of text within most documents is generally either unstructured (i.e. single paragraphs of free text) or semi-structured, such as electronic mail messages (Malone et al., 1987). Documents may be organised into sections, subsections, paragraphs, etc., but any given document may contain an arbitrary number of each. In addition, there may be other information available, such as the length of the document or the type of audience for which it was intended, which may assist in the learning task. The size of vocabulary is also a significant factor in determining the representation of the document. Many of these issues also arise within *Information Retrieval* (IR) systems. Such systems attempt to locate and retrieve documents from a corpus, based on a small number of query terms presented by the user.

There are many similarities between basic information retrieval (IR) systems and simple nearest neighbour learning algorithms. The queries presented to an IR system consist of one or more terms. Each query is compared to the documents within the corpus, and those documents which are most similar to the query are presented to the user. In contrast, a nearest neighbour algorithm classifies a query instance by comparing it with all the stored training instances, and returning the class label associated with the most similar (i.e. nearest) instance. As a consequence, there may be tools used by IR systems that can be employed by nearest neighbour learning algorithms to improve their accuracy when learning to categorise documents. For example, *tfidf*[1] weighting strategies have already been used to construct prototype instances (Zhang 1992; Biberman 1995; Datta & Kibler 1997), which in turn are used to train nearest neighbour learning algorithms within some information agent systems (Lang 1995; Cohen 1996b).

[1]Term Frequency/Inverse Document Frequency (Rocchio Jr 1971; Salton & McGill 1983).

This thesis focuses on the role of nearest neighbour learning within intelligent information agents. Two issues are explored: the representation of documents within a nearest neighbour learning system, and the problems encountered by such systems when the number of attributes used to describe the documents is large.

1.2 Learning within Information Agents

Intelligent information agents such as email filters (Metral, 1993; Payne, 1994; Cohen, 1996a; Boone, 1998) and Web agents (Armstrong et al., 1995; Balabanović et al., 1995; Green & Edwards, 1996; Edwards et al., 1996; Pazzani et al., 1996) utilise a learning algorithm to determine what action should be taken on new or incoming documents. An email agent may attempt to categorise an incoming email and save it in the appropriate folder, or notify the user when a high priority message arrives. In contrast, a Web agent may simply determine whether or not a new Web page is of interest to the user. However, documents such as email messages or Web pages can be complex, and it can be difficult to map the data within the document into a representation suitable for presentation to a learning algorithm (Payne 1994; Payne & Edwards 1997).

This complexity can be illustrated by an example. Consider an application that maintains a database of bibliographic entries. Each entry refers to a published paper, and includes the abstract and title and additional details relating to the paper, such as the technical scope of the paper and author biographies, etc. A simple information agent observes the different entries examined by the user, and based on these observations, presents new entries that the user might also find interesting.

To achieve this, the agent induces and utilises a *user profile* for each user. This profile represents the user's interests, and is induced from the observations made by the application. The observations capture the entries examined, and any related actions, e.g. the query used to find the entry, or the name of the file in

which the entry was subsequently stored. New bibliographic entries are compared to the user profile, and any that match are presented to the user.

The observations have to be mapped into a representation that is suitable for presentation to a learning algorithm. Domain knowledge can be used to identify the individual fields in the database record. Fields containing text can be parsed to identify terms, remove punctuation, stem the terms (i.e. remove suffixes such as 'ing' or 'ed' (Porter, 1980)), and sort the terms in dictionary order. Figure 1.1 shows an illustrative entry that has been retrieved from a database and processed in this way. This data now has to be mapped to a training example for presentation to the learning algorithm.

Title Terms	*a agent an in interface investigation issues learn learning mail of that*
Authors	*"Peter Edwards", "Terry R. Payne"*
Abstract Terms	*a able adapt agent allow an and are assist autonomous change component construct context data develop different employ filter from has have here identify in increase interest interface internet ...*
Publication Type	*journal*
Publication Name	*"Applied Artificial Intelligence"*
Editor	*"Robert Trappl"*
Technical Scope	*advanced scientific*
Journal Scope	*a about act address administration advances ai also and application applied article artificial as comparative concerns cultural economic education emphasize engineering evaluation exchange existing ...*
Publisher	*"Taylor and Francis"*
Author Biographies	*Peter Edwards: 1988 1991 a aberdeen and arrive as been before blackboard derek fellow first for from have here in lecturer leeds live liverpool my north obtain october of on originally phd postdoctoral ...*
Year	*1997*
Pages	*1-32*
Volume	*11(1)*

Figure 1.1: A pre-processed bibliographic entry. Note that many fields contain lists of terms found in the original database entry.

Many of the fields can be easily mapped to attributes within training instances.

For example, the field *Publication Type* can be mapped to an attribute whose domain may include the values {*journal, proceedings, book, techreport*}, or the *Year* field can be mapped to a numeric attribute. However, mapping fields such as *Abstract Terms* or *Journal Scope* to attributes present a number of problems. This is due to the size of the domain of the attributes (for text, the domain may contain in the order of 20,000 - 100,000 elements (Lang, 1995)), and the number of values in each field (many supervised learning algorithms assume that an instance contains a single symbolic or numeric value for each attribute).

A number of information retrieval systems utilise a vector space representation (Salton & McGill, 1983). Each document is represented by a vector of length N, where N is the size of the vocabulary of the corpus. Each element of the vector corresponds to one of the terms in the corpus. Documents are mapped into the vector space representation by setting the elements within the vectors to 1 if the corresponding terms appear in the document[2], and setting the remainder to 0.

The vector space model can also be used to represent each field within the bibliographic entry. Figure 1.2 illustrates how multiple vectors can be used to represent three of the fields for the bibliographic entry in Figure 1.1.

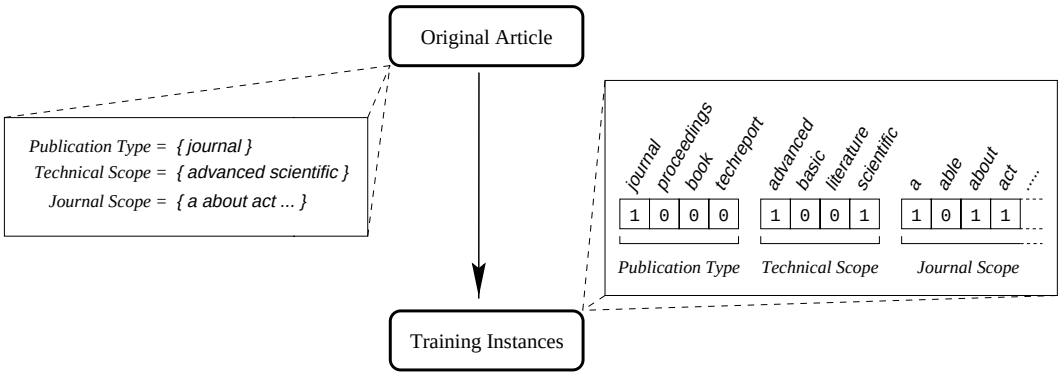


Figure 1.2: Mapping selected values to field vectors.

Although many learning algorithms can learn from data represented by the vector space model, it can introduce problems associated with *high dimensionality*. If the vocabulary of each of these field vectors is large (i.e. 20,000 - 100,000 elements),

[2]Many systems use some statistic to indicate the existence of a term, such as term frequency.

then the size of the document vectors can become unwieldy. This can result in an increase in the time taken to classify a new document, and a drop in the classification accuracy of the learning algorithm.

Various solutions have been proposed that attempt to resolve the representational problem, and reduce the size of the field vectors. One proposed solution would be to ignore the structure of the document, and map all the terms into a single field vector (Balabanović et al., 1995; Lang, 1995). However, this approach assumes that all components of the document (such as titles, header fields, etc.) refer to the same information, and so contain similar terms. This assumption is inappropriate for applications that utilise data such as that presented in Figure 1.1, where there are a number of different, possibly independent textual fields. Other systems limit the size of the separate field vectors, and thus reduce the size of the resulting document vector (Armstrong et al., 1995). A number of different statistical measures that previously were used by IR systems have been employed by different learning algorithms to weight and select terms. These measures are often based on the frequency of the term both in the document, and across the whole training set (Salton & Buckley, 1987; Edwards et al., 1996; Yang & Pedersen, 1997).

1.3 Objectives

In the previous section, the difficulties associated with the identification of a suitable representation for complex documents were introduced. The standard approach is to represent the different components of the documents (i.e. different fields) by size limited term vectors. However, this introduces the problem of selecting the most appropriate terms from the original vocabulary. The time taken for many approaches to induce a concept hypothesis (used to classify query instances) is generally a function of the number of attributes (i.e. size of the vector) used to describe the data. Hence, the need to reduce the number of elements within the vector is critical for several supervised learning algorithms. The nearest neighbour learning paradigm is well suited to learning from domains

represented by large numbers of attributes, as it does not involve any explicit training step, but rather it classifies query instances by searching through the space of training instances for the closest match. However, as each element of the query vector (i.e. attribute) is significant in determining the most similar training instance, this learning approach is susceptible to the presence of irrelevant and erroneous attributes (Langley & Iba 1993).

The objectives of the work presented here are two-fold:

- to suggest an alternative representation suitable for complex documents, and to propose alternative nearest neighbour distance metrics that can utilise such a representation;
- to investigate various approaches to the problem of identifying and eliminating irrelevant attributes, and to determine if the number of attributes can be reduced by the elimination of redundant attributes.

The representation proposed is based on the use of *sets*; each field within a complex document is mapped to a set containing a selection of terms extracted from the field. A novel nearest neighbour learning algorithm which has been modified to learn from this representation will be described. An existing information agent will then be used to evaluate the new representation.

The thesis then explores the general problem of attribute selection. The following issues will be addressed:

- The implications of high dimensionality (i.e. large numbers of attributes) on nearest neighbour learning will be discussed, with an illustration of how the presence of irrelevant or redundant attributes can have a detrimental impact on the classification accuracy of the learning algorithm.
- A number of different approaches to attribute selection have previously been proposed and tested with various learning algorithms on a variety of domains. These approaches will be combined with a nearest neighbour algo-

rithm, and systematically evaluated over a number of standard UCI data sets (Merz & Murphy 1996).

- A dimensionality reduction technique known as *Latent Semantic Indexing* has been successfully used to reduce the number of attributes used to represent documents. This technique, which is similar to a subspace mapping technique known as *Correspondence Analysis* (Greenacre 1984), will be used to reduce the dimensionality of a number of standard UCI data sets, which will then be presented to a nearest neighbour learning algorithm. The resulting behaviour will be compared and contrasted to that observed when the attribute selection techniques were evaluated.

1.4 Thesis Overview

The thesis consists of eight chapters (including this introduction) which explore the behaviour of nearest neighbour learning algorithms when presented with data sets containing irrelevant or redundant data. The second chapter introduces the nearest neighbour learning algorithm, and describes the effects of high dimensionality and feature relevance on the learning algorithm. This includes an examination of the *curse of dimensionality*, as well as an explanation of *irrelevant* and *redundant* attributes.

Chapter 3 reviews a variety of different dimensionality reduction and attribute selection techniques. The differences between *filter* and *wrapper* based selection mechanisms are discussed, and selection through feature weighting is presented. Finally the dimensionality reduction technique that underlies *Latent Semantic Indexing* is explained.

Chapter 4 describes how a novel set-based approach can be used to represent data for agent systems, and proposes a nearest neighbour learning algorithm designed to learn from this representation.

Chapter 5 presents an evaluation of the set-based data representation described in Chapter 4, and compares it with a rule-based approach within the context of

an existing, agent-based email filtering domain.

Chapter 6 discusses a number of different approaches to dimensionality reduction for nearest neighbour learning. A filter and four wrapper algorithms are described. This is then followed by a description of various weighting techniques, and concludes with a description of the sub-space mapping algorithm.

The empirical evaluation of the various dimensionality reduction techniques is then presented in Chapter 7. The results for each method are discussed, and the chapter concludes with a comparison of the different approaches.

The thesis concludes with a summary of the contributions, and a discussion focussing on the different attribute selection and dimensionality techniques of the empirical evaluations explored in the previous chapters. Possible directions for future research are then outlined.

Chapter 2

Nearest Neighbour Learning

Chapter Outline

The work presented in this thesis investigates various representational issues that are encountered when learning to categorise texts, and contrasts a number of methods designed to reduce the dimensionality of this data. The nearest neighbour learning algorithm (Section 2.2) has been used to explore various representation and dimensionality issues. This section begins by introducing the terminology used to describe data sets, and presents the nearest neighbour learning paradigm. The presence of irrelevant or redundant attributes can affect the performance of such learning algorithms. This is discussed in detail, and illustrated by means of an example.

2.1 Terminology

A *machine learning algorithm* is a program that can learn to perform a given task, such as driving autonomous vehicles (Pomerleau 1995), detecting fraud (Piatetsky-Shapiro, Brachman, Khabaza, Kloesgen, & Simoudis 1996), playing games (Mitchell 1997), or classifying unclassified data. The algorithm is generally trained by presenting it with a data set containing a number of examples of that task. If the task is a supervised classification task, the examples are tagged with a classification label, which represents the concept to be learned. The learning algorithm attempts to reproduce this concept, so it can be used to classify new, *query* examples. This section describes the terminology used when discussing machine learning algorithms in the context of a classification task.

The examples contained within the data set are known as *instances*. An instance is made up of a *classification label* and a *vector*. Each element of the vector corresponds to a characteristic of the data set, such as 'diastolic pressure' or 'shape of tumor' in medical data (Figure 2.1). These characteristics are described as *attributes*. A *feature* is a tuple that relates a value of an element to one of the attributes. For example, the feature *oval* might refer to the shape of a tumor. It is important to note that two distinct features can share the same value; for example the feature *oval* relating to the tumor shape is distinct from the feature *oval* relating to the shape of a lymph node. An attribute can be represented by a set of ordered (e.g. numerical), or un-ordered (e.g. symbolic) values. Ordered values may be continuous (i.e. real numbers) or discrete (e.g. rankings or numeric ranges). The values that appear within a given set may be restricted, for example a symbolic attribute may be represented by a small number of values. Figure 2.1 illustrates the relationship between the different terms described above.

A classification label is attached to each instance. Instances normally belong to one of a number of categories, or *classes*, and the value of the classification label indicates to which class the instance belongs. The machine learning algorithm tries to learn one or more *classification hypotheses* or *target concepts*. A target concept can be considered as a function that maps an instance (represented by the

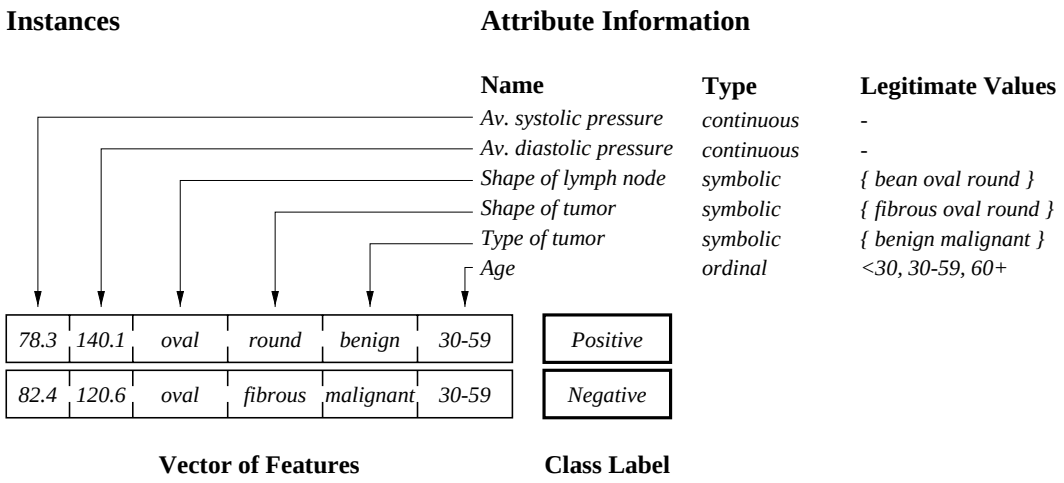


Figure 2.1: This fictitious data set illustrates the components of an instance, represented by the vector model.

feature vector) to the value of its classification label (i.e. its class). If an instance x_i belongs to the class $x_i.c$ and the target concept that maps the instance to its class is $f(x_i)$, then the class of an instance has been correctly predicted if and only if $f(x_i) = x_i.c$. Hence, a target concept can be used to predict the class of an instance from its features.

The performance of a learning algorithm is generally determined by counting the number of instances that can be correctly classified, and presenting this as a percentage of the total number of instances classified. This percentage is referred to as the *classification accuracy*. A related approach counts the number of instances that fail to be classified correctly (i.e. are classified *incorrectly*); this is known as the *error rate*, and is presented as a percentage. Both approaches rely on the data set being partitioned into mutually exclusive *training* and *test* sets. The training set is used by the learning algorithm to generate the target concept, and the test set is used to evaluate its accuracy or error rate. Various partitioning strategies exist (Kohavi 1995), and are discussed in detail in Sections 5.1.3 and 6.2.

2.2 The Nearest Neighbour Learning Paradigm

The nearest neighbour learning paradigm has been the subject of many studies for nearly half a century. It is based on the principal that the members of a population (e.g. the instances within a data set) will generally exist in close proximity with other members sharing similar properties. Hence, additional information about an instance can be obtained by observing other instances that are close to it, i.e. by observing its *nearest neighbours*. It naturally follows that this method can be used as a classification technique: if the members of the population are tagged with a classification label, then the value of the label of an unclassified instance (i.e. its *class*) can be determined by observing the class of its nearest neighbours.

The basic concepts that underlie the *nearest neighbour classifier* were presented by Fix & Hodges Jr. (1951) in their study of the nonparametric discrimination problem, in which they described a number of different procedures and demonstrated that these had asymptotically optimum properties for large sample sets (Dasarathy 1991). However, it was not until 1967 that Cover & Hart formally defined the nearest neighbour rule and applied it to the problem of pattern classification. Many aspects of this classification algorithm have since been studied (Dasarathy 1991): such as examining more than one neighbour to determine the classification of an instance; estimating the probability density functions implicit within the space to determine error rates; and editing the space to reduce the number of instances required by the algorithm.

Nearest neighbour learning algorithms are also known as *Instance-Based* (Aha 1990) or *Exemplar-Based* (Bareiss & Porter 1987) learning algorithms, and fall with the category of *lazy-learning* algorithms (Mitchell 1997), as they defer the induction or generalisation process until classifications are performed. The nearest neighbour algorithm requires less computation time during the training phase than most eager learning algorithms (such as the rule induction algorithm, C4.5 (Quinlan 1993)). However, the consequence of using this lazy learning approach is that the computational cost of classifying a new query instance can be high.

The power of the nearest neighbour approach has been demonstrated in a number

of real-world domains, such as in the pronunciation of English words (Stanfill & Waltz 1986), recognition of DNA & RNA sequences (Cost & Salzberg 1993), thyroid disease diagnosis (Kibler & Aha 1987), speech recognition (Bradshaw 1987), clinical audiology diagnosis (Bareiss & Porter 1987), meeting predictions (Kozierok & Maes 1993) and Internet information filtering (Payne, Edwards, & Green 1997). However, this learning paradigm has also been the subject of strong criticism (Aha 1992b). Breiman, Freidman, Olshen, & Stone (1984, p.17) highlight five objections to such approaches for concept learning:

1. They are expensive due to their large storage requirements;
2. They are sensitive to the choice of the similarity function used to compare instances;
3. They cannot easily work with missing attribute values;
4. They cannot easily work with symbolic attributes;
5. They do not yield concise summaries of concepts.

There has been some work to extend nearest neighbour algorithms to solve these problems. Aha developed the IBL family of instance-based learning algorithms, some of which reduce the number of instances needed to represent a concept (Aha, Kibler, & Albert 1991), and attempt to handle instances with noisy or irrelevant attributes (Aha & Kibler 1989; Aha 1992b). MBRtalk (Stanfill & Waltz 1986) defines similarity over symbolic values, and the EACH algorithm (Salzberg 1991a) uses the *Nested Generalised Exemplar* (NGE) theory to create compact representations of concepts.

There are also advantages to the nearest neighbour approach. Such methods can learn graded concepts (Barsalou 1985; Aha 1989), relational concepts (Emde & Wettschereck 1996), and provide a basis for exploring prototypical learning (Zhang 1992; Biberman 1995; Datta & Kibler 1995; Datta & Kibler 1997). There have also been a number of theoretical studies on the nearest neighbour algorithm, including PAC Analysis (Albert & Aha 1991), Average Case Analysis (Langley &

Iba 1993), and various studies on the effects of noisy domains (Dasarathy 1991; Okamoto & Yugami 1996).

2.2.1 The Basic Learning Paradigm

The basic nearest neighbour learning algorithm is trained by simply storing the training instances until classification time. When a *query* (i.e. unclassified) instance is presented for classification, its nearest neighbour is determined and used to generate a classification label. Instances can be considered as points within an n-dimensional instance space. Each of the n dimensions corresponds to one of the n attributes that are used to describe an instance. The topology of the instance space is dependent on the characteristics of the attributes, such as the type of each attribute (i.e. numeric, ordinal, symbolic etc.) and the cardinality or range of each attribute. The absolute position of the instances within this space is not as significant as the relative distance between instances. This relative distance is determined using a *distance metric* (Section 2.2.3). The choice of metric is generally dependent on the topology of the space, and a variety of different metrics have been proposed (Wilson & Martinez 1997).

The nearest neighbour paradigm can be viewed as a mapping function between the instance space and an *output* space. An instance and its classification label can be represented by the tuple (x_i, c_i), where x_i represents the location of instance i within the instance space, and c_i is the location of its classification label within the output space. Hence, given the tuple (x_i, c_i) we can say that "x_i belongs to the class c_i".

$$p(c_q|x_q) \approx p(c_i|x_i) \ , \quad \text{if } D(x_q, x_i) \approx 0 \tag{2.1}$$

The nearest neighbour learning paradigm is based on an assumption that relates the distribution of values in the instance space to the distribution of values in the output space (Cover & Hart 1967; Michie, Spiegelhalter, & Taylor 1994). It states

that if two instances, x_i and x_q, are located close to each other within the instance space (according to some distance metric D), then the posterior class probabilities of both instances will be approximately equal (Equation 2.1). In other words, if there exists a training instance x_i which is located close to the query instance x_q in the instance space, and the classification label c_i for x_i is known, then it is possible to assume that the value for the unknown class c_q will also be c_i.

2.2.2 The k-nearest Neighbour Learning Algorithm

The k-nearest neighbour learning algorithm (k-NN) locates the k nearest instances to a query instance, and determines its class by identifying the single most frequent class label associated with the nearest neighbours. An example of a 3-nearest neighbour algorithm is presented in Figure 2.2. Each instance is represented by a two-dimensional point within a continuous-valued Euclidean space. The points appear as either a + or − symbol, depending on the class label of the instances (i.e. positive or negative instances respectively). The sphere surrounding the query instance x_q contains the three nearest neighbours, n_1, n_2 and p_1. The instance x_q is classified as −, as two of the nearest neighbours, n_1 and n_2 are both negative, whereas only one instance (p_1) within the sphere is a positive instance.

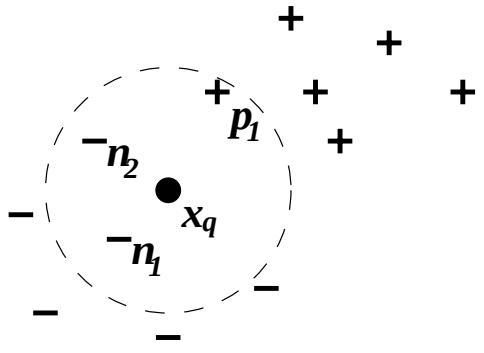


Figure 2.2: A set of positive (+) and negative (−) instances within an instance space. The class of the query instance, x_q, is determined by identifying the three nearest neighbours, n_1, n_2 and p_1.

The voting strategy used to classify x_q in Figure 2.2 is known as the *Majority Rule*. This rule groups together the nearest neighbours according to the value of their

class label, and returns the class of the largest group. Other voting strategies have been proposed, including the *Qualified k-NN Rule* (Devijer & Kittler 1982), the *Variable Threshold Rule* (Tomek 1976), and the *Distance Weighted Rule* (Dudani 1976).

The value of k can significantly affect the performance of the k-nearest neighbour algorithm. If the *Majority Rule* is used when $k \rightarrow n$ (where n is the number of instances in the instance space), and the distribution of instances across different classes is non-uniform, then the behaviour of the learning algorithm will be similar to that of a global approximator, i.e. the resulting classification will be due to the distribution of class labels, and not the location of the query instance.

Wettschereck (1994) investigated the behaviour of the k-nearest neighbour algorithm in the presence of data containing mislabelled (i.e. *noisy*) instances. He found that for small values of k, the k-NN algorithm was more robust than the single nearest neighbour algorithm (1-NN) for the majority of large data sets tested. However, the performance of the k-NN algorithm was inferior to that achieved by the 1-NN algorithm on small data sets (< 100 instances) or where the instances were sparsely distributed and located close to the decision boundary. This can be explained by considering the decision boundaries (represented by the two solid lines) in Figure 2.3. These boundaries denote a decision region containing negatively classified instances. The space either side of this region contains positively classified instances. The query instance lies within the negative space, but close to the upper boundary. If a 3-NN approach is used, the instance is correctly classified, although the sphere containing the three nearest neighbours (represented by the smaller of the dashed circles) extends beyond the upper decision boundary. However, as k is increased, the sphere grows, and encompasses some of the positive instances. Once the value of k exceeds 5, the classification of x_q will become erroneous, as more positive instances fall within the sphere than negative ones.

The sphere represents a form of localised approximation of the instance space. Hence, larger values of k result in greater approximations, which may overwhelm small areas of the instance space belonging to a certain class (such as the decision region illustrated in Figure 2.3) and result in erroneous classifications. Ideally, the

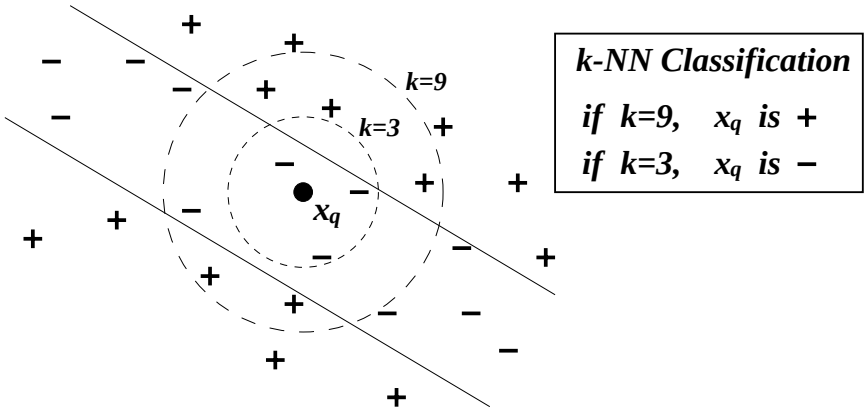


Figure 2.3: The k-NN algorithm fails to correctly classify x_q if $k = 9$, as the sphere extends beyond the decision boundaries. However, a smaller sphere ($k = 3$) produces a correct classification.

value of k should be significantly smaller than p, the size of the smallest decision region within the instance space (Dasarathy 1991); i.e. k lies within the range $1 \leq k \ll p$. If k approaches or exceeds this value, then the sphere surrounding the query instance within this region may include instances that lie outside the decision boundary, and result in incorrect classifications.

Localised approximation is especially desirable when noisy instances are present; i.e. when a misclassified instance is surrounded by similarly classified instances. If a query instance lies close to the noisy instance, then a 1-NN neighbour algorithm will generate a false classification, as the class will be determined by the noisy instance. However, a k-NN algorithm will consider the surrounding instances and is more likely to generate the correct classification. Although the k-NN algorithm is generally superior to the 1-NN algorithm, many studies have favoured the single nearest neighbour algorithm when comparing this paradigm with other machine learning algorithms, as this eliminates the need to determine the optimal value of k for each individual data set (Shepherd, 1983; Weiss & Kapouleas, 1989; Aha, 1992a; Michie et al., 1994; Rachlin et al., 1994; Domingos, 1996; Kubat et al., 1997).

2.2.3 Distance Metrics

The location of instances within the instance space is defined by the representation of the instances and the topology of the space. However, a *distance metric* is necessary to determine the relative distance between two instances. A distance metric should satisfy the following criteria for all points within a given space (Duda & Hart 1973):

i)	$D(i,j) \geq 0$ and $D(i,j) = 0$ if and only if $i = j$	Positivity
ii)	$D(i,j) = D(j,i)$	Symmetry
iii)	$D(i,j) + D(j,k) \geq D(i,k)$	Triangle Inequality

Several distance metrics have been proposed (Wilson & Martinez 1997), and include the *Chi-square* metric (Anderberg 1973), the *Mahalanobis* metric (Everitt 1974), the *Nonlinear* metric (Devijer & Kittler 1982), the *Cosine Similarity* metric (Salton & McGill 1983), the *Quadratic* metric (Fukunaga & Flick 1984), the *Minkowskian* metric (Salzberg 1991b), the *Modified Value Difference* metric (Cost & Salzberg 1993), and the *Context Similarity* metric (Biberman 1994). The choice of distance metric is significant, as it can greatly affect the learning bias of the algorithm. Ideally, the distance metric should minimise the distance between two similarly classified instances, whilst maximising the distance between instances of different classes.

Numeric Distance Metrics

The most commonly used metrics are derivatives of the *Minkowskian* metric. These include the *Manhattan* (*City-Block*), *Euclidean* and the *Chebychev* metrics, and are derived by using the values $r = 1$, $r = 2$, and $r = \infty$ respectively. These metrics calculate the distance between two instances by determining the difference between the values for each attribute (2.3), and combining these differences to generate an overall distance value (2.2).

$$D(i,j) = \left[\sum_{a=0}^{A} \delta(i_a, j_a)\right]^{\frac{1}{r}} \quad (2.2)$$

$$\delta(i_a, j_a) = |i_a - j_a|^r \quad (2.3)$$

Here, i and j refer to the two instances, and a refers to one of A attributes. The distance metrics described above differ in the approach used to compare the two values i_a and j_a in (2.3). The *Chebychev* metric is equivalent to measuring only the greatest single distance between attributes (i.e. measuring the greatest distance along a single dimension), and is often defined as follows:

$$D(i,j) = \max_{a=0}^{A} |i_a - j_a| \quad (2.4)$$

Various psychological studies have argued that the *Manhattan* metric is appropriate for domains that have separable (i.e. orthogonal) dimensions (Nosofsky 1984). Salzberg (1991b) investigated the performance of both distance metrics to determine whether or not this hypothesis could be tested, but failed to find any significant difference between either distance metric.

Normalisation of Attributes

The *Minkowskian* family of distance metrics determine $\delta(i_a, j_a)$ for each attribute a, and combine these differences to determine the overall distance (Equation 2.3). However, this overall distance may be misleading if the range of values differs greatly for each attribute. For example, an instance corresponding to a human subject may be represented by two attributes, height (in metres) and weight (in kilograms). The range of the height attribute may vary between 1.5m-2.0m, whereas the range of the weigh may vary between 50kg-150kg. If the raw data is used, then any variation in height between two instance will have little effect on the overall distance, whereas a variation in weight will have an overwhelming

effect. For this reason, values are *normalised* so that they all lie within a given range.

Two *normalisation* methods are most frequently used to standardise attributes. The first divides each value of an attribute by the range of that attribute, so that all values lie within the range [0..1]. This approach can suffer from the effects of outliers which result in the majority of values falling within a very small band within the range. The alternative, which overcomes this problem, is to make use of the standard deviation of the attribute when dividing each value (Everitt 1974).

Symbolic Distance Metrics

The *Overlap* metric is a symbolic variant of the *Manhattan* metric. It determines the distance between values for each attribute by simply comparing the values; if they are the same then it returns a value of zero, otherwise a value of one is returned (2.5).

$$\delta(i_a, j_a) = \begin{cases} 0 & \text{if } i_a = j_a \\ 1 & \text{if } i_a \neq j_a \end{cases} \tag{2.5}$$

The *Value Difference Metric* (VDM) was proposed as an alternative to the *Overlap* metric for determining the distance between two symbolic values (Stanfill & Waltz 1986). Whilst it is not strictly a distance metric (it is asymmetric and hence violates the symmetry requirement presented earlier) it has received a great deal of attention (Daelemans, Gillis, & Durieux 1994; Rachlin, Kasif, Salzberg, & Aha 1994; Ting 1994; Wettschereck, Aha, & Mohri 1997; Domingos 1996; Payne & Edwards 1997; Wilson & Martinez 1997). The VDM differs from many other distance metrics in that the distance $\delta(i_a, j_a)$ between two attribute values is not determined by comparing the values themselves, but by comparing the class conditional probability distributions for the values for each attribute a (Equation 2.7).

$$vdm(i,j) = \sum_{a=0}^{A} \delta(i_a, j_a) \cdot \omega(i_a) \quad (2.6)$$

$$\delta(i_a, j_a) = \sum_{c \in C} \left| \mathrm{P}(c|i_a) - \mathrm{P}(c|j_a) \right|^2 \quad (2.7)$$

$$\omega(i_a) = \left[\sum_{c \in C} \mathrm{P}(c|i_a)^2 \right]^{0.5} \quad (2.8)$$

Again, i and j refer to the two instances, and a refers to one of A attributes. C refers to the set of all class labels present in the data set, and $\mathrm{P}(c|i_a)$ is the class conditional probability of i_a, i.e. the probability of the value i_a occurring in the data set for attribute a in instances of class c. This probability is determined directly from the training data by counting the number of instances containing the value i_a for attribute a, and determining the proportion that also have the class label c, i.e.:

$$\mathrm{P}(c|i_a) = \frac{|\text{instances containing } i_a \wedge \text{class} = c|}{|\text{instances containing } i_a|}$$

The distance for each attribute is calculated by summing the squared difference between the class conditional probabilities of the two values for each class.

This process can be illustrated by means of an example. The top three charts in Figure 2.4 represent the discrete class distributions of three different symbolic values, 'X', 'Y' and 'Z'. Each distribution consists of three discrete probabilities, represented by the vertical bars. The lower charts illustrate how pairs of symbolic values are compared. For each class, the difference (Equation 2.7) in class condi-

	a_i = 'X'	a_i = 'Y'	a_i = 'Z'
$\mathrm{P}(c_1\|a_i)$	0.7	0.4	0.6
$\mathrm{P}(c_2\|a_i)$	0.0	0.5	0.1
$\mathrm{P}(c_3\|a_i)$	0.3	0.1	0.3

Table 2.1: Class conditional probability values for the symbols in Figure 2.4.

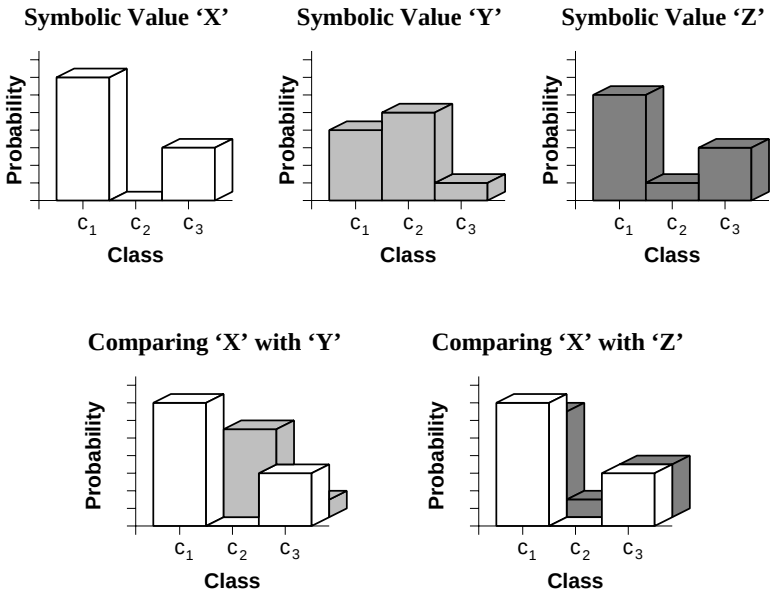


Figure 2.4: Comparing symbolic values with the value difference metric.

tional probability is determined. These differences are then combined (Equation 2.6) and result in a distance value between the two symbolic values for attribute a. Hence, to compute the distance between the two symbols 'X' and 'Y', the difference in class conditional probabilities is found for each class. For this example, the values are 0.3, 0.5 and 0.2 for classes c_1, c_2 and c_3 respectively (the class conditional values for these symbols are listed in Table 2.1). Hence the final distance between the two symbols is the sum of the squares of these values, i.e.

$$\delta(\text{'X'}, \text{'Y'}) = 0.3^2 + 0.5^2 + 0.2^2 = 0.38$$

The weight component of the VDM (Equation 2.8) provides some indication of how well an attribute value discriminates between different class labels. The weight can vary between a minimum, which is dependent on the number of classes present in the data set, and 1 which represents an ideal discriminator, i.e. an attribute value which only appears in one class. The minimum represents a uniform class distribution where an attribute value appears with equal probability in instances of all classes, and can be calculated as follows (Equation 2.9):

$$\omega(u) = |C|^{-0.5} \tag{2.9}$$

where $\omega(u)$ is this minimum value (i.e. the weight of an attribute value with a uniform class distribution), and C is the set of all class labels that appear in the data set.

The weight is used to control the influence of the attribute distance for each training instance when determining the final nearest neighbour. As the range of values of $\delta(i_a, j_a)$ will vary between zero and one, the weight can be used to restrict this range, i.e. the range of $\delta(i_a, j_a) \cdot \omega(i_a)$ will vary between zero and $\omega(i_a)$. A large attribute distance will have a greater effect on the value of $vdm(i, j)$ than a smaller one. Thus if a small weight is used (i.e. the value present in the query instance is irrelevant), then the resulting attribute distance will also be small and have little impact on the choice of nearest neighbour.

2.3 Dimensionality Issues and the Nearest Neighbour Paradigm

One of the main criticisms of the nearest neighbour paradigm is that the computational cost of classifying a query instance can be high. The query instance is compared with every stored instance to determine the nearest neighbour(s). Thus, the time taken to classify the query instance is a function of the number of stored instances, and the number of attributes (i.e. dimensions) used to describe each instance. However, some attributes may be considered as *irrelevant*, as they contribute nothing to the classification task, and may even degrade the accuracy of the resulting classifications.

Determining which of the attributes are relevant to the learning task (i.e. identifying attributes which predict the class value) is a central problem in machine learning. In the past, domain experts selected the attributes believed to be relevant to the learning task. However, in the absence of such knowledge, automatic techniques are required to identify such attributes. Rule induction algorithms have been developed which use a variety of metrics as part of their learning bias to select relevant attributes when building decision trees. Such metrics include the information gain metric (Quinlan, 1986) or the distance-based gain ratio (De Mántaras, 1991). Studies have shown that the biases used by rule induction algorithms to favour smaller numbers of attributes and smaller decision trees fail to find the minimal subset of attributes necessary to identify the concept (Almuallim & Dietterich 1991).

John et al. (1994) have attempted to define the notion of attribute 'relevance'. They claim that there are two degrees of relevance:

Strong Relevance

An attribute is indispensable in learning a concept, i.e. its omission leads to a loss in predictive accuracy;

Weak Relevance

One or more attributes make an equal contribution towards learning a con-

cept. When one of these attribute subsets is used, the others are no longer required, and hence are known as *redundant* attributes.

Nearest neighbour algorithms are especially susceptible to the inclusion of irrelevant or redundant attributes, as the distance metrics combine measurements for all of the attributes (Aha 1992b). The problems encountered can be summarised as follows:

1. Weakly relevant (i.e. redundant) attributes can bias the similarity measure away from strongly relevant attributes, which may affect the accuracy of the predicted classification;

2. Attributes which are irrelevant can influence the similarity measure, and hence have a detrimental effect on the accuracy of the predicted classification;

3. The quality of the classification mapping can be considered as a function of the number of attributes (dimensions) used to describe each instance and the number of instances in the training set. This is generally referred to as the *curse of dimensionality* (Bellman 1961).

The following subsections discuss the issues listed above with respect to the nearest neighbour learning paradigm, and discuss ways in which their impact can be minimised.

2.3.1 The Problem of Redundant Attributes

An attribute is relevant if it can make a contribution towards the construction of a concept description. However, there may be a number of attributes that are similar and make an equal contribution to that concept description. These attributes are generally referred to as *weakly relevant* (John, Kohavi, & Pfleger 1994), as the inclusion of more than one of these attributes adds nothing to the coverage of the concept. Those remaining attributes are said to be *redundant*.

Learning paradigms that sum the contributions of individual attributes are susceptible to the presence of redundant attributes. The contribution that the common dimensions (shared by the redundant attributes) make to the concept description is effectively weighted more heavily than other attributes. For example, if three independent, strongly relevant attributes are used to describe a domain, then a nearest neighbour distance function may determine the overall distance D as the sum of the individual attribute distances $d_i, i \in (1, 2, 3)$, i.e.

$$D = d_1 + d_2 + d_3$$

A fourth attribute, which is redundant, is then used to describe the instances. This attribute is perfectly correlated with the third attribute (and hence attributes three and four are weakly relevant). As expected, the overall distance D becomes a function of the four individual distances. However, as attributes three and four are perfectly correlated, the normalised individual distances for these two attributes will always be equal. Hence the overall distance function can be written as

$$D = d_1 + d_2 + 2d_3$$

This can be re-written in a more general form to include a weighted term for each attribute:

$$D = \sum_{i=0}^{A} w_i \times d_i$$

where the weights for the first two attributes are both equal to 1, i.e. $w_1 = w_2 = 1$, whereas the weight of the third attribute, $w_3 = 2$. In other words, attribute three is weighted twice as heavily as attributes one or two. This may lead to a biased, and possibly incorrect prediction. A related problem was reported by Langley & Sage (1994a), where the inclusion of additional weakly relevant attributes resulted in a polynomial change in the contribution of these attributes with respect to the other attributes when a Naive Bayesian classifier was used.

Consider the data set presented in Table 2.2 which represents ideal conditions

	C	**wind**	**rainfall**	**daylight**	d_1	d_2	d_3	D
x_q	?	*calm*	*shower*	*light*				
x_1	P	breezy	none	light	1	1	0	2
x_2	P	breezy	none	twilight	1	1	1	3
x_3	P	gusty	none	twilight	1	1	1	3
x_4	N	gusty	shower	dark	1	0	1	2
x_5	N	calm	none	dark	0	1	1	2
x_6	N	calm	shower	dark	0	0	1	1

Table 2.2: A small data set characterising the ideal conditions for flying a kite. The columns to the right of the table correspond to the distance between each training instance and the query instance x_q.

for flying a kite. The data set consists of just three relevant attributes; **wind**, **rainfall** and **daylight**. The domain has a binary class label (column C), with the two values P and N corresponding to positive and negative instances respectively. The target concept can be described by the following rules:

$$\begin{aligned}
\textbf{class} = P \quad &\longleftarrow \quad (\textbf{wind} = \textit{breezy} \vee \textbf{wind} = \textit{gusty}) \\
&\wedge \quad \textbf{rainfall} = \textit{none} \\
&\wedge \quad (\textbf{daylight} = \textit{light} \vee \textbf{daylight} = \textit{twilight}) \\
\textbf{class} = N \quad &\longleftarrow \quad \text{otherwise.}
\end{aligned}$$

A second data set (Table 2.3) consists of attributes in the first data set, and two additional redundant attributes: **visibility** and **foggy**. These new attributes are highly correlated with the **daylight** attribute, and consequently they contribute nothing new to the concept description.

A query instance is presented for classification. The values for the three relevant attributes are {*calm, shower, light*}. The correct classification label for this instance is *N*. A simple nearest neighbour learning algorithm that utilises the *Overlap* metric is used to determine the class label, given this training data. The overall distance, D, and the individual attribute distances $d_1, \ldots, d_3$ for each of the instances in the training set are listed in Table 2.2. The nearest neighbour of the query instance is x_6 which has the class label *N*. Hence the classification is correct.

	C	wind	rainfall	daylight	visibility	foggy	d_1	d_2	d_3	d_4	d_5	D
x_q	?	*calm*	*shower*	*light*	*good*	*none*						
x_1	P	breezy	none	light	good	none	1	1	0	0	0	2
x_2	P	breezy	none	twilight	poor	misty	1	1	1	1	1	5
x_3	P	gusty	none	twilight	poor	misty	1	1	1	1	1	5
x_4	N	gusty	shower	dark	bad	heavy	1	0	1	1	1	4
x_5	N	calm	none	dark	bad	heavy	0	1	1	1	1	4
x_6	N	calm	shower	dark	bad	heavy	0	0	1	1	1	3

Table 2.3: The data set in Table 2.2, with two additional redundant attributes.

This result arises due to the fact that the first two attribute values of the unclassified instance are significant in determining the classification of the instance. The value for the third attribute, *light*, is misleading in this case, as it previously only appears within the positive instance x_1. However, if two additional redundant attributes are present, then x_1 becomes the nearest neighbour, which results in an incorrect classification (Table 2.3). As the two additional attributes (**visibility** & **fog**) are highly correlated with the third attribute (**daylight**), this is equivalent to weighting the third attribute more heavily ($\times 3$ in this case) than the first two attributes, i.e. $w_1 = 1$, $w_2 = 1$, $w_3 = 3$. Hence, the *Overlap* metric will favour any instance which has the same value for the third attribute (i.e. $d_1 = 1$) over an instance that can match the values for just the first two attributes.

2.3.2 The Problem Of Irrelevant Attributes

An attribute may be considered *irrelevant* if it contributes nothing to the classification task. The inclusion of such an attribute within a concept hypothesis would be misleading, and any value contained within this attribute would be meaningless in the context of the classification task. For nearest neighbour learning algorithms, irrelevant attributes can often be responsible for generating incorrect predictions.

The assumption underlying the nearest neighbour learning paradigm is that the distribution of instances in the instance space is equal to the distribution of the corresponding classification labels in the output space. In other words, there is a correspondence between the location of an instance x_i in the instance space and its classification label c_i in the output space. Since the location of x_i is defined by

its attributes, this assumption relies on the attributes used to represent x_i being relevant to the classification label c_i in the output space. Hence the inclusion of irrelevant attributes within the representation will violate this assumption.

This can also be illustrated by an example. The kite flying data set has again been used and appears in Table 2.4. A new query instance x_q {*gusty, none, light*} is then presented for classification. In this scenario, two of the instances (x_1 & x_3) are found to be the nearest neighbours of x_q. Although the choice of nearest neighbour may be arbitrary, both will generate a correct classification, i.e. P.

	C	**wind**	**rainfall**	**daylight**	d_1	d_2	d_3	D
x_q	?	*gusty*	*none*	*light*				
x_1	P	breezy	none	light	1	0	0	1
x_2	P	breezy	none	twilight	1	0	1	2
x_3	P	gusty	none	twilight	0	0	1	1
x_4	N	gusty	shower	dark	0	1	1	2
x_5	N	calm	none	dark	1	0	1	2
x_6	N	calm	shower	dark	1	1	1	3

Table 2.4: The query instance x_q shares two nearest neighbours, x_1 and x_3, and is correctly classified as belonging to class P.

However, if two additional, irrelevant attributes, **season** and **time**, are added to the data set, the resulting classification may be incorrect. A second query instance x_q is presented for classification. It consists of three relevant values: {*gusty, none, light*} and two irrelevant values: {*winter, afternoon*}. Table 2.5 summarises the distances computed when this query instance is presented to the nearest neighbour algorithm with the two additional irrelevant attributes. This time, x_4 is found to be the nearest neighbour. Although the value for only one of the relevant attributes matched the new instance, both the values for the irrelevant attributes matched, and hence $D = 2$.

2.3.3 The Curse of Dimensionality

The nearest neighbour learning paradigm, as illustrated above, can be viewed as a mapping from an input space to an output space. If each dimension (i.e. attribute) a can be represented by up to v_a values, and the dimensionality of the

	C	**wind**	**rainfall**	**daylight**	**season**	**time**	d_1	d_2	d_3	d_4	d_5	D
x_q	?	*gusty*	*none*	*light*	*winter*	*afternoon*						
x_1	P	breezy	none	light	spring	morning	1	0	0	1	1	3
x_2	P	breezy	none	twilight	autumn	afternoon	1	0	1	1	0	3
x_3	P	gusty	none	twilight	summer	evening	0	0	1	1	1	3
x_4	N	gusty	shower	dark	winter	afternoon	0	1	1	0	0	2
x_5	N	calm	none	dark	autumn	evening	1	0	1	1	1	4
x_6	N	calm	shower	dark	summer	morning	1	1	1	1	1	5

Table 2.5: The data set in Table 2.4, with two additional irrelevant attributes.

input space (i.e. the number of attributes used to describe an instance) is A, then the total number of unique points that can occupy the input space (i.e. the *cardinality* of the space) is:

$$\prod_{a=1}^{A} v_a$$

This cardinality increases exponentially as the dimensionality of the space increases linearly. This relationship is known as the *curse of dimensionality* (Bellman 1961).

This 'curse' can have a direct effect on the performance of a nearest neighbour algorithm. The number of training instances presented to the algorithm is normally fixed. However, the number of dimensions used to describe the space can vary depending on the number of irrelevant or redundant attributes present in the data set (Almuallim & Dietterich 1991; John, Kohavi, & Pfleger 1994; Payne & Edwards 1996; Payne & Edwards 1998a), and whether any of the attributes can be combined to generate higher-order dimensions (Michie, Spiegelhalter, & Taylor 1994; Bishop 1995; Wu, Berry, Shivakumar, & McLarty 1995; Payne & Edwards 1998a). If the dimensionality of the input space is large, but the number of instances is small, the distribution of instances within this input space will be sparse. Hence, it may no longer be possible to assume that the posterior class probabilities of an instance and its nearest neighbour will approximate to each other (Section 2.2.1), and hence the quality of the mapping between the input space and the output space will be reduced.

The theoretical relationship between the number of training instances required to learn a concept and the number of dimensions used to describe the data was investigated by Langley & Iba (1993). An average case analysis was performed on a simple nearest neighbour learning algorithm when applied to a two class problem. The analysis assumed that each instance consisted of r relevant and i irrelevant Boolean attributes. The Boolean value of each attribute was determined with a probability of 0.5. The *Overlap* metric (Section 2.2.3) was used to determine the distance between two instances. The target function was conjunctive, hence the number of positive instances was dependent on the number of relevant attributes. However, the number of negative instances was a function of the total number of attributes (i.e. both relevant and irrelevant attributes).

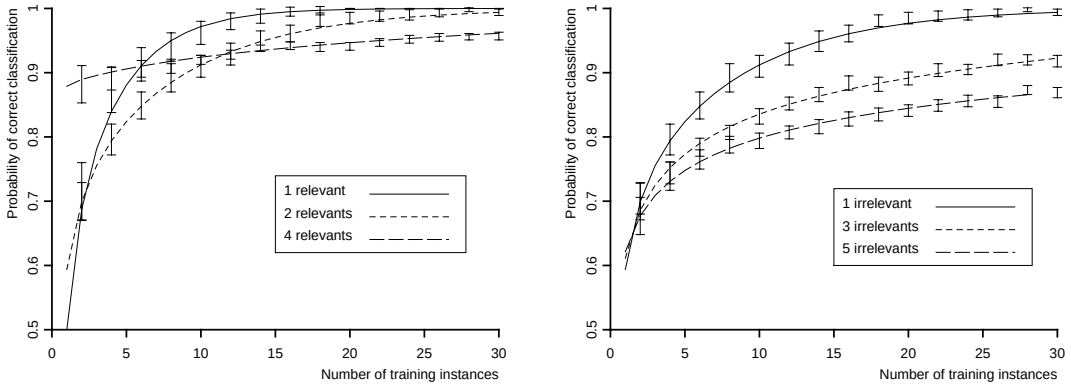


Figure 2.5: The performance of the NN algorithm as the number of relevant (left) and irrelevant (right) dimensions increases. (Langley & Iba, 1993, reproduced with permission)

This theoretical analysis was verified empirically. Various learning curves were generated, which plotted the accuracy of the learning algorithm against the number of training instances. As expected, the accuracy was low when the number of training instances was small, and increased monotonically as the number of training instances increased. The accuracy of the learning algorithm was observed as the number of relevant attributes was increased whilst the number of irrelevant attributes remained constant (Figure 2.5, left). As the number of relevant attributes increased, there was an increase in classification accuracy when small training sets were used. However, additional instances were required before the learning curve

reached its asymptote, due to an increase in the cardinality of the instance space.

The implications of an increase in the number of irrelevant attributes, whilst the number of relevant attributes remained constant, was also studied (Figure 2.5, right). No variation in the initial accuracy of the algorithm was observed, due to the fact that the target concept was a function of the relevant attributes only. Thus, there was no change in the proportion of negative to positive instances when the number of irrelevant attributes increased. However, there was a fall in the learning rate as the number of irrelevant attributes increased, partially due to an increase in the cardinality of the instance space.

Chapter 3

Dimensionality Reduction and Attribute Selection

Chapter Outline

In the previous chapter, the problems associated with nearest neighbour learning algorithms and redundant or irrelevant attributes were discussed. This chapter reviews a number of different approaches to attribute selection and dimensionality reduction. A more comprehensive review of contemporary approaches employed by various machine learning algorithms and information retrieval/text categorisation systems can be found in (Payne & Edwards 1996).

3.1 Introduction

The attribute selection techniques used by machine learning algorithms can be grouped into two broad categories: the *filter* model, where the selection technique is independent of the learning algorithm; and the *wrapper* model, where the learning algorithm is integral to the selection mechanism (John, Kohavi, & Pfleger 1994). Both models perform a search within a space of attribute subsets to determine the optimal (or sub-optimal) subset for the classification task. An alternative approach, employed by some nearest neighbour algorithms, is the use of weights to identify irrelevant attributes. We refer to this third category as the *weighted* model (Payne & Edwards, 1996).

Attribute selection techniques are also required within large information retrieval (IR) systems (Salton & McGill 1983; Deerwester, Dumais, Furnas, Landauer, & Harshman 1990), and text categorisation systems (Edwards, Bayer, Green, & Payne 1996; Yang & Pedersen 1997). These systems use large indexes to search and retrieve text documents stored in a database or *corpus*. Dimensionality reduction techniques are often used to reduce the number of words (terms) used to index the documents, and hence improve the rate at which documents are retrieved.

The attribute selection approaches used for IR/text categorisation tasks differ slightly from those used by machine learning algorithms in that they are often applied to domains with huge numbers of attributes (often $> 5\,000$). Many of the techniques used employ simple statistical metrics to determine which terms should be discarded. However, *Latent Semantic Indexing* (Deerwester, Dumais, Furnas, Landauer, & Harshman 1990) differs in that it uses an *orthogonal decomposition* technique similar to correspondence analysis (introduced in Section 3.5.1) to reduce the dimensionality of the documents.

The next four sections briefly describe the different attribute selection models used by various machine learning algorithms, and the dimensionality reduction techniques used by IR/text categorisation systems. The chapter concludes with a discussion of the relative merits of each approach.

3.2 Filter Model

The *filter* model (Figure 3.1) utilises an independent search criterion and evaluation function to find the appropriate attribute subset. This subset is then used to generate a reduced data set which in turn is presented to a learning algorithm. The evaluation function is used to determine whether or not the inclusion of an attribute will affect the classification performance of the learner. For example, the consistency measure used by Almuallim & Dietterich (1991) and Liu & Setiono (1996a) determines whether or not the removal of an attribute will result in the creation of instances that have identical attribute values but different class values. The filter model does not take into account the learning biases used by the final learning algorithm, and thus may not select the subset most suitable for that algorithm.

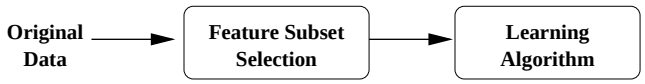


Figure 3.1: The Filter Model.

Table 3.1 lists some of the attribute selection systems that use the filter model. The *Search* column describes the type of search used. The *Forward Selection* and *Backward Elimination* searches start from either no attributes, or a full complement of attributes, and then search for solutions by greedily selecting and adding/eliminating attributes to/from the attribute subset. Cardie (1993) and Kubat et al. (1993) perform searches by presenting data which includes all the attributes to a decision tree algorithm, and selecting the attributes which appear in the resulting decision tree. The *Evaluation* column lists the evaluation function used, and the final column, *Testing Alg.* refers to the learning algorithm that utilised the attribute subset within each study.

Authors (System)	Search	Evaluation	Testing Alg.
Aha & Bankert, 1995 (BEAM)	Beam variants of forward & backward selection/elimination	Calinski-Harabasz separability index	IB1
Almuallim & Dietterich, 1991 (FOCUS)	Breadth-first	Consistency	ID3
Cardie, 1993	Forward Selection	Information Gain[a]	kNN
Kubat et al., 1993	Forward Selection	Information Gain[b]	Naive Bayes
Liu & Setiono, 1996a (LVF)	Las Vegas (i.e. Monte Carlo random sampling)	Consistency	ID3
Singh & Provan, 1996 (Info-AS)	Forward selection	Maximise 1 of 3 information metrics	Bayesian Network

[a]The C4.5 learning algorithm is used to induce a decision tree to identify the attribute subset.
[b]The ID3 learning algorithm is used to induce a decision tree to identify the attribute subset.

Table 3.1: Comparison of different attribute selection studies (filter model).

3.3 Wrapper Model

In the *wrapper* model (Figure 3.2), the attribute selection algorithm utilises the learning algorithm as part of its evaluation function. The training data is normally divided into two partitions: a training partition, and an evaluation partition. An attribute subset defines which of the attributes appear within the two partitions. The learning algorithm is trained using the instances in the training partition. The instances in the evaluation partition are then classified, and an overall predictive accuracy is computed for the current attribute subset. This accuracy measure is then used to guide the search.

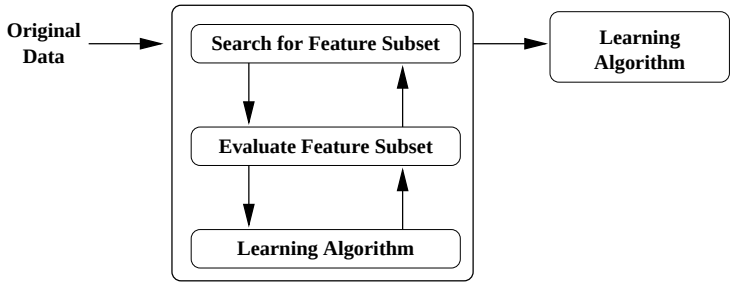


Figure 3.2: The Wrapper Model.

The study by Aha & Bankert (1995) on attribute selection for the classification of cloud patterns compared the filter and wrapper models and found that the wrapper model performed best. This supports the hypothesis proposed by (John, Kohavi, & Pfleger 1994) that attribute selection should take into account biases used by the final learning algorithm. Whilst this model may yield better results than the filter model, the time taken to evaluate each attribute subset visited during the search makes this approach infeasible for problems with very large numbers of attributes.

Table 3.2 lists some of the attribute selection systems that use the wrapper model. The *Search* column again describes the type of search used. The *Control* column refers to the control mechanism used when evaluating the attribute subsets. The last column lists the learning algorithm used to evaluate each attribute subset.

Authors (System)	Search	Control	Learning Alg.
Aha & Bankert, 1995 (BEAM)	Beam variants of forward & backward selection/elimination	leave-one-out cross validation	IB1
Bala et al., 1995 (GA-ID3)	Genetic algorithm	2 cross validated fixed partition tests	C4.5
Caruana & Freitag, 1994	Forward, backward & stepwise selection/elimination variants	fixed size train/ evaluation partitions	ID3/C4.5
Cherkauer & Shavlik, 1996 (SET-GEN)	Genetic algorithm	k-fold cross validation	C4.5
Kohavi, 1994 (BFS)	Best first search	k-fold cross validation	C4.5
Langley & Sage, 1994a (Selective Bayes)	Forward selection	accuracy measure across training set	Naive Bayes
Moore & Lee, 1994 (RACE)	Forward & backward selection, and schemata search	leave-one-out cross validation	1-NN
Richeldi & Lanzi, 1996 (ADHOC)	Genetic algorithm	k-fold cross validation	C4.5
Salzberg, 1992 (CSS)	Combined stepwise selection	fixed size train/ evaluation partitions	1-NN and EACH
Singh & Provan, 1995 (K2-AS)	Forward selection	fixed size train/ evaluation partitions	Bayesian networks
Skalak, 1994 (RMHC-PF1)	Random mutation search	k-fold cross validation	1-NN
Vafaie & De Jong, 1994	Genetic algorithm	fixed size train/ evaluation partitions	AQ15

Table 3.2: Comparison of different attribute selection studies (wrapper model).

3.4 Weighted Model

A number of learning algorithms make use of real-valued attribute vectors to weight attributes based on their past performance. A weighted attribute vector is generated, which initially gives each attribute an equal weight. The training set is then evaluated using a leave-one-out cross validation. After each instance has been evaluated, the weights are adjusted according to whether or not the classification was correct. We refer to this model as the *weighted* model (Figure 3.3).

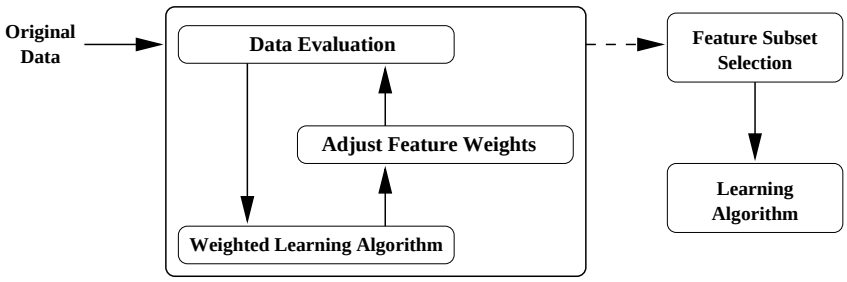


Figure 3.3: The Weighted Model.

The intuition behind this model is that irrelevant attributes will contribute very little overall to the classification task. The weighting strategies used normally reward attributes if they are responsible for correct predictions, and penalise them if they are responsible for incorrect ones. Thus, the contribution of irrelevant attributes to the classification task will decrease as the contribution of other attributes increases. Those attributes that make a small contribution can then be eliminated.

Table 3.3 lists some of the systems that employ the weighted model. The *Selection* column indicates whether the attribute weights are used as part of the final learning algorithm or used to select an attribute subset. The *Evaluation* column indicates the method used to update weights during the evaluation phase. The last column refers either to the type of algorithm used during attribute selection, or in the case of RELIEF (Kira & Rendell, 1992a) and the extensions to RELIEF (Kononenko, 1994), to the learning algorithm used once the attributes have been

Authors (System)	Selection	Evaluation	Testing Alg.
Aha, 1992b (IB4)	Weighted attributes	Adjust weights wrt accurate or inaccurate predictions	Weighted nearest neighbour
Kira & Rendell, 1992a (RELIEF)	Threshold selection	Adjust weights wrt closest +ve/-ve neighbours	ID3
Kononenko, 1994 (RELIEF-extensions)	Threshold selection	Adjust weights wrt closest neighbours from each class	ID3
Littlestone, 1988 (WINNOW)	Weighted attributes	Adjust weights wrt inaccurate predictions	Linear threshold classifier
Salzberg, 1991a (EACH)	Weighted attributes	Reduce weights wrt inaccurate predictions	Weighted nearest hyperrectangle

Table 3.3: Comparison of different attribute selection studies (weighted model).

selected.

3.5 IR/Text Categorisation Approaches

Dimensionality reduction techniques have been used by IR systems to reduce the number of terms used to index documents, resulting in an improvement in the rate at which documents are retrieved. These techniques have also been applied to the problem of reducing the number of terms presented to learning algorithms for text categorisation problems (Edwards, Bayer, Green, & Payne 1996; Yang & Pedersen 1997). Moulinier (1996) presents a framework for text categorisation, which includes a dimensionality reduction or attribute selection stage between the initial representation, that of textual data, and the final representation, presented to the learning algorithm. Whilst some studies have omitted this stage (Creecy, Masand, Smith, & Waltz 1992), the number of unique terms (typically in the region of tens or hundreds of thousands) is prohibitively high for most machine learning algorithms. For this reason, several techniques have been developed specifically to reduce the dimensionality of the final data representation.

The techniques used by many text categorisation systems are similar to those categorised by the filter model. Table 3.4 lists a sample of the different evaluation methods used by various systems. *Latent Semantic Indexing* (LSI) (Deerwester et al., 1990; Schütze et al., 1995; Weiner et al., 1995) uses an orthogonal decomposition technique to determine a lower dimensional representation for each document. A corpus can be expressed as a *term by document* matrix, where each row corresponds to a document, and each column refers to one of the terms appearing within the corpus. An orthogonal decomposition technique is then applied to this matrix, resulting in a set of decomposed matrices that describe this space and the points within it. The space can then be approximated (by approximating the decomposed matrices) resulting in a lower dimensional representation of the points in the approximated space.

Singular Value Decomposition (SVD) (Press, 1992; Greenacre, 1984, Appx A.)

Technique	Study	Learning Algorithms
Information Gain	Lewis & Ringuette, 1994	PropBayes and DT-min10
	Armstrong et al., 1995	Winnow, Wordstat and a Rocchio-based NN
	Moulinier, 1996	ID3, Charade, NN and Naive Bayes
	Moulinier, 1997	Ripper and Scar
	Yang & Pedersen, 1997	k-NN and a linear least squares fit mapping
Mutual Information	Weiner et al., 1995	Neural Network classifier
	Yang & Pedersen, 1997	k-NN and a linear least squares fit mapping
χ^2 Statistic	Schütze et al., 1995	logistic regression, linear discriminant analysis and a Neural Network classifier
	Yang & Pedersen, 1997	k-NN and a linear least squares fit mapping
Frequency Measure	Lang, 1995	Rocchio-based NN and MDL
	Edwards et al., 1996	*IBPL1* and C4.5
	Payne & Edwards, 1997	*IBPL1* and CN2
	Yang & Pedersen, 1997	k-NN and a linear least squares fit mapping

Table 3.4: Sample of dimensionality reduction methods for text categorisation.

is normally used to perform the decomposition, although a recent study has shown that other orthogonal decomposition approaches, such as ULV decomposition (Berry & Fierro 1996), can be used to replace SVD for this task. Studies have demonstrated that a significant reduction in dimensionality can be achieved when used within IR systems; for example from 5000-7000 dimensions to about 100 dimensions (Deerwester et al., 1990). SVD has also been utilised to generate subspace approximations of protein sequence data for presentation to neural networks (Wu, Berry, Shivakumar, & McLarty 1995). The size of the input vectors presented to a back-propagation neural network was reduced from 9696 to 100. In addition to this, the predictive accuracy of the neural network improved when SVD was used. Section 3.5.1 introduces some of the principals behind both LSI and a related technique, know as *correspondence analysis*, and illustrates how these techniques reduce dimensionality of a data set is reduced.

3.5.1 Correspondence Analysis

Correspondence Analysis (Greenacre 1984) is a mathematical tool that is used to graphically present multi-dimensional data within low (e.g. two or three) dimensional data plots. It is similar to *Latent Semantic Indexing* in that it can be used to reduce the number of dimensions required to describe instances of a given domain. It achieves this by representing the instances as a collection of points within a Euclidean space, and identifying a lower dimensional sub-space that approximates the original space. This section summarises the theory behind correspondence analysis, and describes how SVD can be used to identify the lower dimensional approximation of the space containing the instances.

Approximating Sub-Spaces

The way in which the space can be approximated and the vectors projected may be explained by means of an example. Consider the values given in Table 3.5. Each value represents the annual Profit or Deficit of a fictitious company over the span of three consecutive years. These values can be represented as three

vectors corresponding to Profit and Deficit tuples for each year. They can then be plotted within a two-dimensional space. These three vectors, **y1**, **y2** and **y3** are illustrated in Figure 3.4.

	Profit	Deficit
Year 1	140	1580
Year 2	290	1310
Year 3	470	410

Table 3.5: Artificial data representing fictitious annual profit and deficit figures over three years.

Any vector in a J dimensional space can be expressed as the linear combination of a *basis* and J scalar coefficients. A *basis* is a set of J linearly independent vectors that characterises a space. A particular basis is the canonical basis; it is this that characterises Euclidean space. For example, the canonical basis for the two dimensional Euclidean space $E^{(2)}$ consists of two basis vectors, $\mathbf{e_1}$ and $\mathbf{e_2}$, and can be expressed as:

$$E^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} \end{bmatrix}, \ \mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Hence, the vector **y2** can be expressed as the linear combination of the canonical basis for a two-dimensional space, and the two scalar coefficients 290 and 1310, i.e.

$$\mathbf{y2} = \begin{bmatrix} 290 \\ 1310 \end{bmatrix} = 290 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 1310 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 290\mathbf{e_1} + 1310\mathbf{e_2}$$

In Figure 3.4 we can see that the three vectors **y1**, ..., **y3** exist close to the straight line **r**. It is possible to express each of the vectors **y1**, ..., **y3** as the combination of three new vectors: a vector from the origin to a fixed point (the *centroid*) on

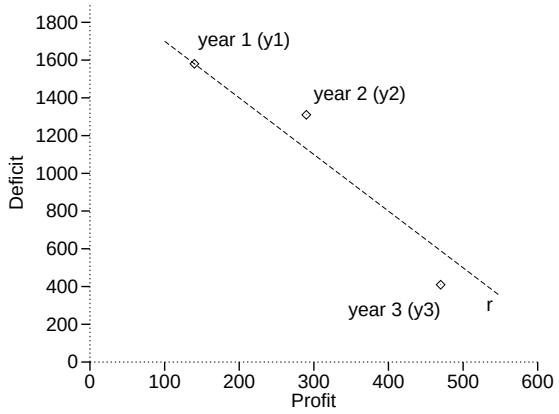


Figure 3.4: The three vectors, **y1**, **y2** and **y3**, plotted as points within a two-dimensional space. Note that each vector lies close to the straight line **r**.

the line **r**; a vector along this line from the centroid; and a vector orthogonal to the line **r**. For example, it is possible to express the vector **y2** as a combination of the three vectors $\bar{\mathbf{y}}$, **b** and **c** (Figure 3.5), where $\bar{\mathbf{y}} = [\ 300 \quad 1100\ ]^{\mathsf{T}}$ is the vector from the origin to the centroid, $\mathbf{b} = [\ 40 \quad -120\ ]^{\mathsf{T}}$ is the vector[1] running along the line **r** (from top left to bottom right in Figure 3.5), and $\mathbf{c} = [\ 30 \quad 90\ ]^{\mathsf{T}}$ is a vector which is orthogonal to the line **r**.

$$\begin{aligned}
\mathbf{y2} &= \bar{\mathbf{y}} + (-\mathbf{b}) + \mathbf{c} \\
&= [\ 300 \quad 1100\ ]^{\mathsf{T}} + \left(-[\ 40 \quad -120\ ]^{\mathsf{T}}\right) + [\ 30 \quad 90\ ]^{\mathsf{T}} \\
&= (300\mathbf{e_1} + 1100\mathbf{e_2}) + (-40\mathbf{e_1} + 120\mathbf{e_2}) + (30\mathbf{e_1} + 90\mathbf{e_2}) \\
&= 290\mathbf{e_1} + 1310\mathbf{e_2}
\end{aligned}$$

The coefficients of the vectors **b** and **c** (in conjunction with the centroid vector $\bar{\mathbf{y}}$) used to express the vector **y2** are -1 and +1 respectively. By varying these coefficients, it is possible to express every instance in the data set using the centroid vector and these two vectors. The coefficient of the centroid vector $\bar{\mathbf{y}}$ is constant

[1]The $^{\mathsf{T}}$ symbol is used in this context to refer to the transpose of each vector.

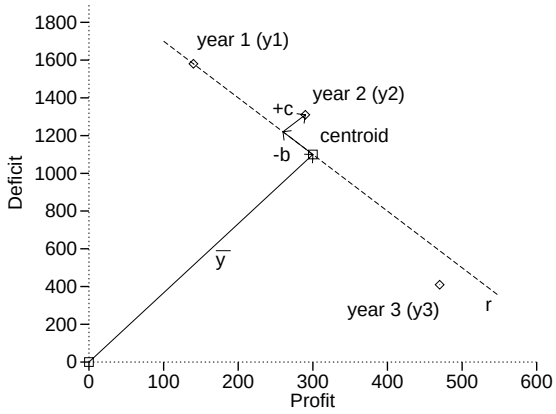


Figure 3.5: Each of the three vectors can be expressed as a combination of three new vectors: $\bar{\mathbf{y}}$, a vector running along the line $\mathbf{r}$, and a vector orthogonal to the line $\mathbf{r}$.

for all the instances in the data set. Hence, the two vectors $\mathbf{b}$ and $\mathbf{c}$ can be used to construct the *basis* of a new space that has been translated from the origin by the vector $\bar{\mathbf{y}}$. Each of the instances in Table 3.5 can be expressed as a combination of the vector $\bar{\mathbf{y}}$ and a vector whose components are the coefficients of the basis vectors $\mathbf{b}$ and $\mathbf{c}$. Figure 3.6 illustrates the three instances mapped onto this new two dimensional space.

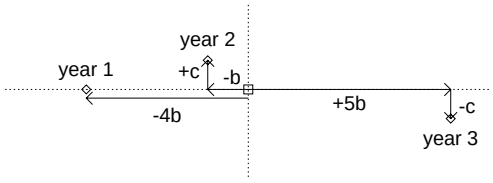


Figure 3.6: The three vectors $\mathbf{y1}$, ..., $\mathbf{y3}$, mapped onto a new, 2-dimensional space. The new space is characterised by the two basis vectors $\mathbf{b} = [40 \quad -120]^{\mathsf{T}}$ and $\mathbf{c} = [30 \quad 90]^{\mathsf{T}}$

Singular value decomposition (SVD) can be used to identify the basis $\{\mathbf{b}, \mathbf{c}\}$. The advantage of using this technique is that it is then possible to approximate the space that this new basis characterises using fewer dimensions. For example, the space illustrated in Figure 3.6 can be approximated by a space of only one dimension, characterised by the single basis vector $\mathbf{b}$. The following subsection

describes how SVD can be used to find a *best-fit* subspace within an n-dimensional space, and illustrates how the best *lower-rank* approximation of the subspace is found.

Determining the Centroid and New Basis

If singular value decomposition is used to identify a space that best fits the instances in the data set, the matrices it generates can be used to calculate a good approximation to this space. This subsection shows how the centroid vector can be determined, and describes how SVD can be applied to the task of determining a new basis and finding an approximation of the space the basis characterises.

The space characterised by the basis $\{\mathbf{b}, \mathbf{c}\}$ (see above), was translated from the origin by the centroid vector $\bar{\mathbf{y}}$. The centroid vector is used for two reasons: it is relatively simple to compute, and it is guaranteed to exist within a space containing the instances (Deerwester et al., 1990). Whilst many vectors can be used to perform this translation, it is difficult to identify such vectors if the basis of the new space is unknown.

The centroid vector is calculated by determining the mean vector for all the vectors that represent the instances in the data set. For example, the centroid vector $\bar{\mathbf{y}}$ for the three vectors $\mathbf{y1}, \ldots, \mathbf{y3}$ can be calculated as follows:

$$\begin{aligned}
\bar{\mathbf{y}} &= \tfrac{1}{3}(\mathbf{y1} + \mathbf{y2} + \mathbf{y3}) \\
&= \tfrac{1}{3}(140\mathbf{e_1} + 1580\mathbf{e_2} + 290\mathbf{e_1} + 1310\mathbf{e_2} + 470\mathbf{e_1} + 410\mathbf{e_2}) \\
&= \tfrac{1}{3}(900\mathbf{e_1} + 3300\mathbf{e_2}) \\
&= 300\mathbf{e_1} + 1100\mathbf{e_2}
\end{aligned}$$

To simplify the process of identifying the new basis, the vectors representing the data can be translated by the centroid vector. This has the advantage of creating a new set of vectors whose centroid is located at the origin. Hence, these vectors

can be expressed as coefficients of the new basis, without the need for an initial translation. For example, a new vector, **z2**, can be found by translating the vector **y2** with the centroid vector $\bar{y}$. It can also be expressed with respect to the basis $\{\mathbf{b}, \mathbf{c}\}$, i.e.

$$\mathbf{z2} \;=\; \mathbf{y2} - \bar{\mathbf{y}} \;=\; -1\mathbf{b} + \mathbf{c}$$

SVD (Press 1992; Greenacre 1984) is often used for solving linear least squares problems, and for performing eigenvalue/eigenvector decomposition. However, it can also be used to construct an orthonormal basis of a best-fit space. The detailed theory behind SVD is not discussed here; for a more thorough discussion see Greenacre (1984, Appendix A). The SVD of a matrix $\mathbf{X}$ of I rows and J columns, and of rank N (see below) can be expressed as:

$$\begin{array}{ccccccc} \mathbf{X} & = & \mathbf{L} & & \mathbf{D} & & \mathbf{R}^{\mathsf{T}} \\ I \times J & & I \times N & & N \times N & & N \times J \end{array}$$

where $\mathbf{L}^{\mathsf{T}}\mathbf{L} = \mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$ (the identity matrix).

The N orthonormal vectors of $\mathbf{L}$, called the left singular vectors, form an orthonormal basis for the columns of $\mathbf{X}$. Similarly, the N orthonormal vectors of $\mathbf{R}$, called the right singular vectors, form an orthonormal basis for the rows of $\mathbf{X}$. The diagonal matrix $\mathbf{D}$ contains the N singular values of $\mathbf{X}$, where the elements of $\mathbf{D}$: $d_1 \geq d_2 \geq \cdots \geq d_N > 0$. Figure 3.7 illustrates the singular value decomposition from an $I \times J$ matrix.

The matrix $\mathbf{X}$ can be constructed such that each row corresponds to each of the translated vectors representing each instance in the data set, and each column corresponds to one of the attributes of the data set. For example, a matrix constructed from the example above (Table 3.5) would consist of three rows and two columns. Each row would correspond to the translated vectors **z1**, ..., **z3** (where $\mathbf{z1} \;=\; \mathbf{y1} - \bar{\mathbf{y}}$, etc.).

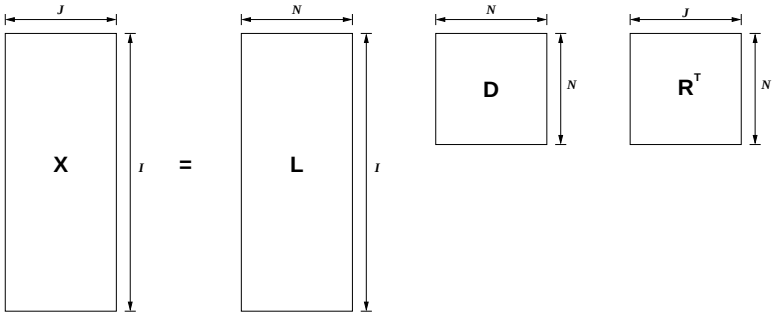


Figure 3.7: A singular value decomposition of an $I \times J$ matrix.

As stated above, the matrix $\mathbf{R}$ forms an orthonormal basis for the rows of $\mathbf{X}$. It is this matrix which characterises the best-fit space for the I instances in matrix $\mathbf{X}$. The rows of matrix $\mathbf{X}$ (corresponding to the instances in the data set) can be projected into the new space by multiplying this matrix with the basis $\mathbf{R}$. The number of dimensions of the space characterised by the basis $\mathbf{R}$ will be equal to the *rank* (Fraleigh & Beauregard 1995) of the original matrix $\mathbf{X}$. The rank N of this matrix will be equal to or less than I or J, whichever is the smaller, i.e. $N \leq \min(I, J)$.

An advantage of using SVD is that the singular values in the diagonal matrix $\mathbf{D}$ can be used to determine which of the N columns of $\mathbf{R}$ can be omitted, and hence result in a lower rank approximation of the space characterised by $\mathbf{R}$. A basis that contains the columns of $\mathbf{R}$ corresponding[2] to the largest singular values in $\mathbf{D}$ will better approximate the space (characterised by $\mathbf{R}$) than one which contains columns corresponding to the smallest singular values. Therefore, if the basis $\mathbf{R}_{(K)}$ for a K dimensional approximation of an N dimensional space is required (where $0 < K \leq N$), then the K columns of $\mathbf{R}$ corresponding to the K largest singular values of $\mathbf{D}$ should be included in the basis for this approximation. This process is called *low rank approximation* (Greenacre, 1984).

To conclude, given a matrix $\mathbf{Y}$ representing I instances and J attributes (i.e. the rows of $\mathbf{Y}$ are the vectors $\mathbf{y}_1^\mathsf{T}, \mathbf{y}_2^\mathsf{T}, \ldots, \mathbf{y}_I^\mathsf{T}$), a centroid vector $\bar{\mathbf{y}}$ can be calculated

[2]By corresponding, we mean that the nth singular value of $\mathbf{D}$, d_n, corresponds to the nth column in the matrix $\mathbf{R}$.

from $\mathbf{Y}$. If a translated matrix $\mathbf{X}$ is defined as the matrix $\mathbf{Y} - 1\bar{\mathbf{y}}^{\mathsf{T}}$, then $\mathbf{X}$ can be decomposed into the three matrices $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{R}^{\mathsf{T}}$. A $\mathbf{K}$ dimensional low rank approximation of $\mathbf{R}$ can be constructed, such that the row vectors[3] $r_1, r_2, \ldots, r_K$ of ${\mathbf{R}^{\mathsf{T}}}_{(K)}$ correspond to the $d_1, d_2, \ldots, d_K$ largest singular values of $\mathbf{D}$. Thus, the basis $\mathbf{R}_{(K)}$ characterises an approximated space of rank K. Once this basis has been determined, it can be used to project instances (represented as vectors) into the new space. For example, if there is a new vector $\mathbf{y}_i$ consisting of J components (i.e. dimensions), then we can find its projection, $\mathbf{f}_i$ which consists of K components in the space characterised by the basis $\mathbf{R}_{(K)}$ (with respect to the centroid vector $\bar{\mathbf{y}}$) as follows:

First, translate the vector $\mathbf{y}_i$ by the centroid vector:

$$\mathbf{x}_i^{\mathsf{T}} \;=\; \mathbf{y}_i^{\mathsf{T}} - 1\bar{\mathbf{y}}^{\mathsf{T}}$$

Then project the translated vector $\mathbf{x}_i$ into the new space, by finding the product of $\mathbf{x}_i$ and the basis $\mathbf{R}_{(K)}$:

$$\begin{array}{ccccc} \mathbf{f}_i & = & \mathbf{x}_i & \mathbf{R}_{(K)} \\ 1 \times K & & 1 \times J & J \times K \end{array}$$

It is possible to determine how good the approximated space is, by calculating the *variation* (as a percentage) of the space. The *total variation* of a space characterised by the basis $\mathbf{R}$ is given by $\sum_{k=1}^{N} d_k^2$, i.e. the sum of the squared singular values $d_1, d_2, \ldots, d_K$. Thus, the variation of the K-dimensional approximated space can be found by calculating the sum of the squares of the largest K singular values, $\sum_{k=1}^{K} d_k^2$, and expressing this value as a percentage of the total variation.

It is also possible to determine the importance, or *inertia* of each dimension $1 \leq k \leq K$ in the approximated space (of rank K) by squaring the corresponding singular value d_k, and expressing this value as a percentage of the total variation.

Thus, to find a K-rank approximation of a matrix $\mathbf{Y}$ containing I point vectors

[3] The row vectors of ${\mathbf{R}^{\mathsf{T}}}_{(K)}$ are equivalent to the column vectors of $\mathbf{R}_{(K)}$.

of dimension J:

1. Find the centroid vector $\bar{\mathbf{y}}$.
2. Find the translated matrix $\mathbf{X} = \mathbf{Y} - 1\bar{\mathbf{y}}^{\mathsf{T}}$.
3. Determine the basis $\mathbf{R}$ and the diagonal singular matrix $\mathbf{D}$ using singular value decomposition.
4. Select the K columns of $\mathbf{R}$ (or K rows of $\mathbf{R}^{\mathsf{T}}$) that correspond with the largest K singular values in the diagonal matrix $\mathbf{D}$.
5. Project the instances represented by the matrix $\mathbf{X}$ into the space characterised by $\mathbf{R}_{(K)}$, by multiplying $\mathbf{X}$ with $\mathbf{R}_{(K)}$.

The points plotted in Figure 3.8 illustrate how a 13 dimensional space can be approximated to a 2-dimensional subspace. The instances are represented as rows in the matrix $\mathbf{Y}$, and their attributes are represented as columns. The two dimensions correspond to the dimensions with the largest inertia values, 40.75% and 18.97%.

3.6 Discussion

The various attribute selection models described above differ in the way they identify subsets of attributes. The filter and wrapper models perform a search through a space of possible attribute subsets. The number of states within this space is exponential; if there are n attributes in the original data set, then there are a total of 2^n possible states in the search space. This exponential rise means that exhaustive, optimal searches are infeasible for all but simple problems involving few attributes. The weighted model generates weights that attempt to represent the relevance of each attribute. These weights are generally employed as a coefficient within the learning algorithm, and hence are rarely used to explicitly select attributes. Correspondence analysis differs from the above models in that it does not select or reject any of the attributes, but attempts to identify a lower dimensional representation that approximates the domain.

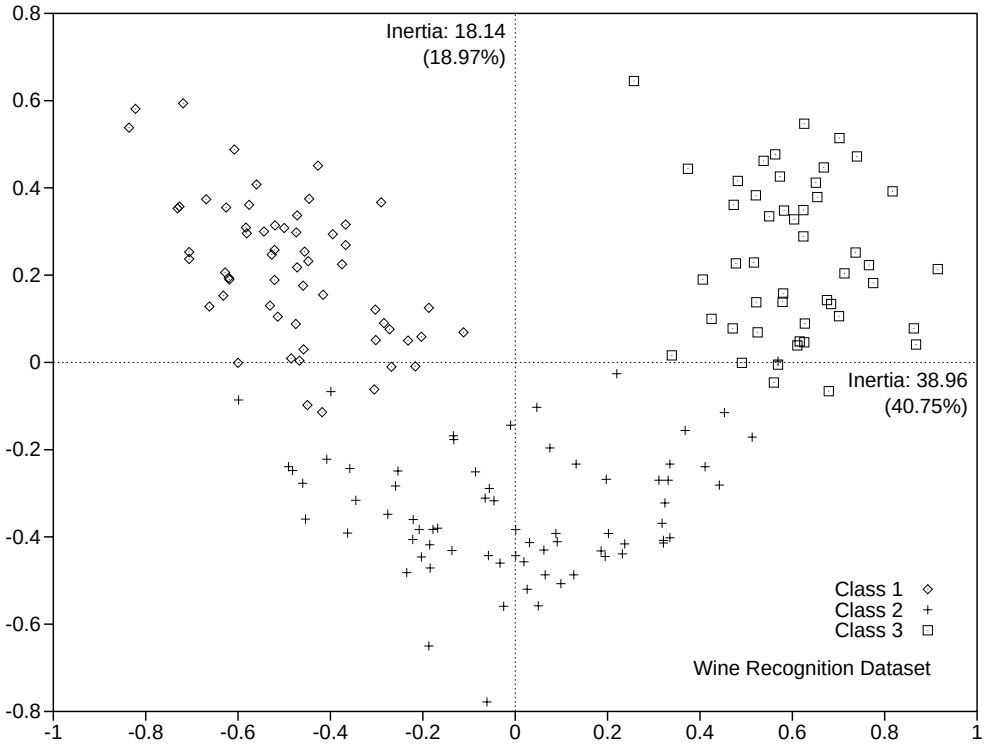


Figure 3.8: The 13-dimension UCI Wine data set approximated in a 2-dimensional subspace.

Several studies have shown that the wrapper model can identify better attribute sets that the filter model (Aha & Bankert 1995; John, Kohavi, & Pfleger 1994). However, induction is performed at every search state visited. This can result in an exponential rise in the time taken for an exhaustive search to locate the optimal subset of attributes. The number of instances, i, in the training set and the control mechanism used to evaluate each state also influences the length of time taken to determine the final attribute subset. Many systems utilise a *k-fold cross validation* approach (Kohavi, 1995) when testing each attribute state to reduce the number of times induction is performed (from i to k). Whilst this may reduce the time taken to generate the final attribute subset, this subset will be dependent on the order in which the training data is presented to the wrapper.

The weighted model differs from the other models in that no explicit search is performed. Instead, a weight vector is modified as each of the training instances is classified. As this model generally evaluates the training data using a single

leave-one-out cross validation approach (Section 3.4), the time taken to generate the final attribute subset is linearly dependent on the number of instances in the training set and the total number of attributes used to describe the training set. For this reason, this model is more suited to performing attribute selection when the number of attributes is large.

The suitability of the weighted model for numerical data is questionable. This model rewards attributes that are responsible for correct classifications, and penalises attributes that are responsible for incorrect classifications. For symbolic data, it is relatively simple to determine whether or not an attribute is responsible for a classification (if the overlap metric is used). In this case, the value (for each attribute) of the nearest neighbour is either equal to or different from the corresponding value of the instance that is being classified. However, if a numeric distance metric is used, then the distance between these two attribute values can occur somewhere within a continuous range, and therefore it is very difficult to determine whether or not the distances calculated for this attribute had an effect on the classification.

Although the filter and wrapper models involve a search through a large state space, the filter model generally takes less time to find a sub-optimal attribute subset than the wrapper model. This is due to the length of time taken to evaluate each attribute subset. However, if the number of attributes is very large, as in the case of IR and text categorisation problems, then performing any form of search becomes impractical. For this reason, a number of attribute selection approaches used by IR and text categorisation systems make an assumption that the relevance of each term is independent of the others. Although this assumption is counter-intuitive, it allows the relevance of each term (i.e. each attribute) to be assessed independently of the other terms. This reduces the number of possible evaluations that may be performed from 2^n to n, where n is the number of terms extracted from the corpus of documents.

Latent Semantic Indexing (LSI) has been demonstrated to both improve performance of IR and text categorisation systems, and reduce the number of dimensions (i.e. attributes) required. However, such studies have demonstrated that LSI and

the principals behind this method work for specific problems, but have not investigated the applicability of LSI to a broader range of classification tasks.

Chapter 4

A Set-based Approach to Data Representation

Chapter Outline

One of the central problems encountered when representing complex documents for presentation to a learning algorithm, is the way in which the dimensionality of the document can be reduced without sacrificing information about the document structure. This chapter describes a set-based approach for representing complex documents, known as the set-valued attribute representation. The representation is described, and three nearest neighbour learning algorithms that utilise set-based distance metrics are presented.

4.1 Introduction

Intelligent information agents employ machine learning algorithms to induce *user profiles* which represent the interests of the user (Section 1.1). These profiles are often induced from a data set containing pre-classified documents, such as electronic mail messages that have been labelled according to their subject matter. However, before a user profile can be induced, the document must first be mapped into a representation suitable for presentation to the learning algorithm.

To date, many intelligent information agents have used the vector space representation (described in Section 1.2) to represent these documents. The vectors consist of N elements (i.e. attributes), where N is the total number of unique terms that appear with the pre-classified documents. Each element corresponds to one of these terms, and the value of the element indicates whether or not the term appeared in a specific document. Although this representation indicates the presence of terms within the document, it fails to convey any information as to location[1] of the term within the document. In addition, the size of these vectors can often be very high (e.g. 20,000 - 100,000 elements). Although pre-processing techniques are frequently used to reduce the number of unique terms used to represent the documents, the resulting vectors can still be large, and may contain irrelevant or redundant attributes.

This chapter introduces an alternative representation which eliminates the need for large unwieldy vectors. It preserves the structure of the document by mapping each field of the document to a single attribute. An attribute is represented by a *set*, and can contain an arbitrary number of unique terms. Three nearest neighbour learning algorithms that utilise this *set-valued attribute* representation are presented below. Two of these algorithms, *IBPL1* and *IBPL2* differ in the way they compare the sets when classifying a query instance. The third, *PIBPL*, investigates a method for pruning irrelevant terms from the sets, thus reducing the size of the sets and increasing the classification accuracy of the learning algorithm.

[1] By location, we refer to the different fields of the document in which the term occurs, and not to the relative position of the term within a sentence or paragraph.

4.2 Set-Valued Attributes

Basic nearest neighbour learning algorithms (introduced in Section 2.2) generally make the assumption that an instance contains a single value for each attribute. The distance metrics determine the distance between two instances by comparing the corresponding values for each attribute (Section 2.2.3). An alternative approach is to consider the values as belonging to a *set*. Single values can be compared with these sets by performing a *set membership* test, i.e. $s \in a_i$ where s is a symbolic value, and a_i is a set of symbolic values for the ith attribute. Such an attribute is known as a *set-valued attribute*. Two set-valued attributes can be compared by performing an *intersection* test[2], i.e. $a'_i \cap a_i$. For example, an electronic mail agent might induce the following rule:

$$\begin{aligned} agents \leftarrow\ & Subject \cap \{agent\ interface\} \neq \emptyset \\ & \wedge\ Body \cap \{assistant\ ai\ engineering\ machine\} \neq \emptyset \end{aligned}$$

i.e. a message will be stored in the *agents* mailbox if it contains at least one of the terms *agent* and *interface* within the *Subject* field, and at least one of the following terms appears within the message body: *assistant*, *ai*, *engineering* or *machine*.

The three nearest neighbour learning algorithms presented below compare sets by determining an intersection distance between two sets. This is achieved by determining the distance between all combinations of pairs of symbols, and combining these distances in a variety of ways.

4.2.1 The IBPL1 Algorithm

IBPL1 is a nearest neighbour algorithm that compares two sets of symbolic values by finding the average similarity between the elements (i.e. symbolic values) of the sets. This average similarity between the a'th set-valued attribute in the query instance, i_a, and the corresponding attribute in the stored, training instance, j_a, is calculated as follows:

[2]Alternatively, a *subset* test could be performed, i.e. $a'_i \subset a_i$.

1. The distance between each element x in i_a and each of the elements y in j_a is determined, and the total distance calculated, i.e. $\sum_{y \in j_a} \delta(x, y)$;

2. The distances calculated for each element x in i_a are summed to generate an overall total distance, i.e. $\sum_{x \in i_a} \sum_{y \in j_a} \delta(x, y)$;

3. This overall total distance is then divided by the total number of distances calculated, i.e. the product of the size of both sets (Equation 4.1)[3].

The overall distance between two instances is then calculated by summing the average similarity between each of the A set-valued attributes (Equation 4.2).

$$ibpl1(i_a, j_a) = \frac{\sum_{x \in i_a} \sum_{y \in j_a} \delta(x, y)}{sizeof(i_a) \times sizeof(j_a)} \quad (4.1)$$

$$D(i, j) = \sum_{a=0}^{A} ibpl1\ (i_a, j_a) \quad (4.2)$$

The *Value Difference Metric* (described in Section 2.2.3) is used to compute the distance between values x and y (i.e. $\delta(x, y)$, reproduced in Equation 4.3). This distance metric was selected for several reasons:

- it's performance in terms of classification accuracy is superior to that of the *Overlap Metric* in several domains (Rachlin, Kasif, Salzberg, & Aha 1994; Wilson & Martinez 1997; Payne & Edwards 1998b);

- relevance weights are generated for each of the symbols present in the sets (Equation 2.8);

- the distance between two symbols is continuous, i.e. the distance falls within the continuous range $[0..\omega(x)]$ (Equation 2.8);

[3]Note that the VDM weight coefficient, $\omega(x)$, (Equation 2.8) is not shown in Equation 4.1, but is used to weight distances within *IBPL1*.

- the distance between symbols sharing similar class distributions (e.g. related terms or synonyms) will be small.

$$\delta(x,y) \quad = \quad \sum_{c \in C} \left| \mathrm{P}(c|x) - \mathrm{P}(c|y) \right|^2 \tag{4.3}$$

It is difficult to determine a distance between a known and an unknown symbol when using the VDM. Each symbol is mapped to a class conditional probability vector, generated from the data in the training set. However, as an unknown symbol is one that does not appear within the training set, there will be no information available regarding it's class conditional probabilities. Although there are various solutions to this problem, such as utilising a uniform distribution vector, *IBPL1* assumes a maximum distance, who's value is equal to 1; i.e. $\forall y$: $\delta(x_q, y) = 1$, where x_q is an unknown symbol.

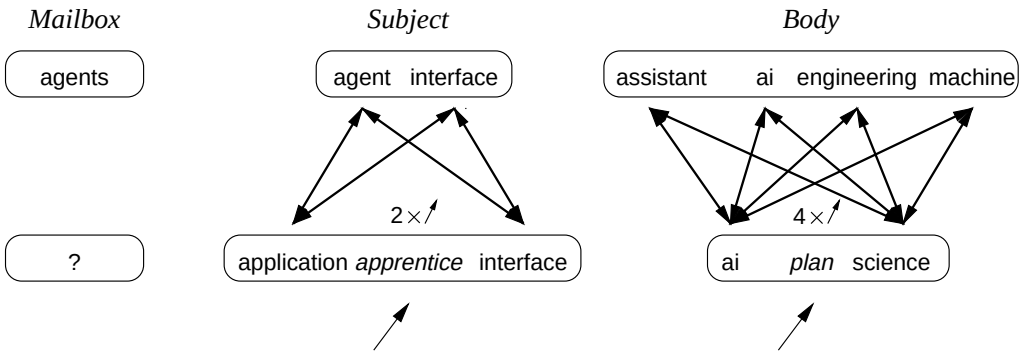


Figure 4.1: Comparing set-valued attributes using IBPL1.

Figure 4.1 illustrates how two instances can be compared. An instance consists of two attributes that correspond to the *Subject* field and the *Body* (i.e. the main text) of an email message. Each attribute contains a set of terms that appear within the corresponding field. The upper instance refers to a training instance, and the lower represents an unclassified query instance. Two of the three terms in the *Subject* attribute set (of the query instance) are compared with all the

terms within the attribute set of the training instance. The term '*apprentice*' is an unknown term and therefore no comparison can be made; instead a maximum value is assigned for each of the possible distances. These distances are then summed, and divided by the total number of distance measurements, which in this case is six (four calculations, and two maximum distances). A similar measure is then generated for the *Body* attribute, only in this case, four maximum distance values are assigned to the unknown term '*plan*'.

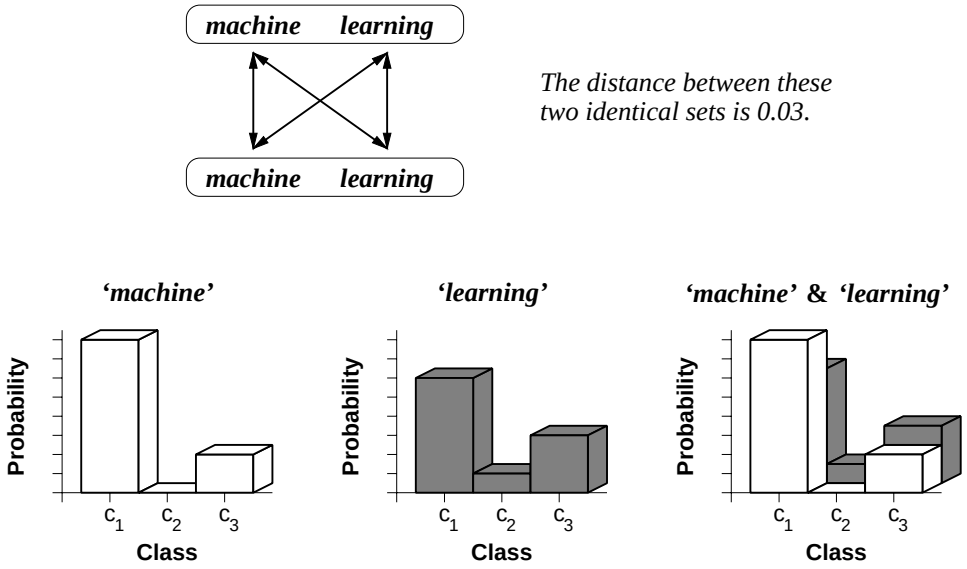
4.2.2 The IBPL2 Algorithm

Although the *IBPL1* algorithm allows distance measurements to be made between set-valued attributes, the distance metric determines the *average* distance between two sets. This results in the distance between identical sets being non-zero, unless all the terms in the set have identical class conditional distributions. For example, consider the set { *machine learning* }, where the class conditional distributions differ slightly (as illustrated in Figure 4.2). Four distance calculations will be made:

$$\begin{aligned} \textit{machine} &\longleftrightarrow \textit{machine} & \delta &= 0.00 \\ \textit{learning} &\longleftrightarrow \textit{learning} & \delta &= 0.00 \\ \textit{machine} &\longleftrightarrow \textit{learning} & \delta &= 0.06 \\ \textit{learning} &\longleftrightarrow \textit{machine} & \delta &= 0.06 \end{aligned}$$

The first two distances (between identical terms) will both be equal to zero. However, the other two distances will each be equal to $(0.8-0.6)^2 + (0.0-0.1)^2 + (0.2-0.3)^2 = 0.06$. Hence, the distance between these two identical sets will equal the average of these four distances, i.e. 0.03.

The *IBPL2* algorithm overcomes this limitation by considering only the distance between each element in the query set and its *closest element* in the corresponding training set. The distance between two sets is thus defined as:



$$\begin{array}{l|l} \mathrm{P}(c_1|\,\textit{'machine'}) = 0.8 & \mathrm{P}(c_1|\,\textit{'learning'}) = 0.6 \\ \mathrm{P}(c_2|\,\textit{'machine'}) = 0.0 & \mathrm{P}(c_2|\,\textit{'learning'}) = 0.1 \\ \mathrm{P}(c_3|\,\textit{'machine'}) = 0.2 & \mathrm{P}(c_3|\,\textit{'learning'}) = 0.3 \end{array}$$

Figure 4.2: Comparing the symbols 'machine' and 'learning'.

$$ibpl2(i_a, j_a) = \frac{\sum\limits_{x \in i_a} \underset{y \in j_a}{\mathrm{Min}}\ \delta(x, y)}{sizeof(i_a)} \tag{4.4}$$

As only the closest distances are summed for each element within the query set, the final distance between the two sets is determined by dividing the total individual distances by the number of elements within the query set. Again, the maximum distance value is assigned to unknown terms; however, only a single distance value is assigned to each unknown term. If *IBPL2* is used to determine the distance between the set { *machine learning* } (Figure 4.2) and itself, then the closest distance will be summed for each term in the query instance, i.e. *machine* $\leftrightarrow$ *machine* and *learning* $\leftrightarrow$ *learning*. Hence the distance between two identical sets will always be zero.

Figure 4.3 illustrates how *IBPL2* is applied to instances generated from the email

message illustrated in Figure 5.1. The solid lines indicate the closest distances for each element in the query set.

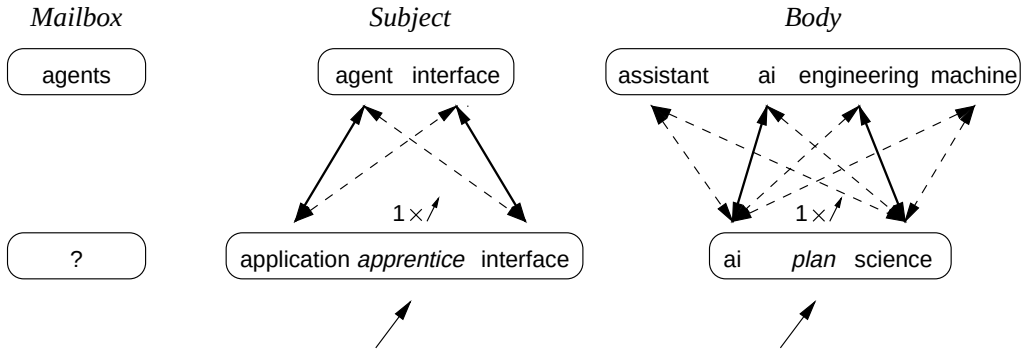


Figure 4.3: Comparing set-valued attributes using IBPL2. The solid lines denote similar matches between two symbols. The dashed lines denote distant matches between two symbols.

4.2.3 The PIBPL Algorithm

The algorithms described so far have focused on the task of learning concepts from sets of values. The *PIBPL* algorithm explores the utility of the VDM weight (Equation 2.8) as a means of selecting or rejecting terms from within the sets. The weight reflects the *typicality* of a term (Payne & Edwards 1995), i.e. how likely the term is to occur in one class but not in others. Terms which appear with equal frequency in all classes have a low weight, whereas those that mostly appear in a single class have a higher weight (page 40). The poorly weighted terms can be identified and removed from each set of extracted terms in the stored examples prior to classification. This will result in a smaller number of values in the sets during classification, and hence fewer distance calculations will be required to identify the nearest neighbour.

The *PIBPL* algorithm extends the distance metric used within *IBPL2*, to ignore the low weighted terms when determining the distance between two sets. A normalised threshold is used to determine whether or not terms should participate in the distance calculation . The weight ω for each term is calculated and compared

to the threshold. If the weight is lower than the threshold, then the corresponding term is rejected. Hence, if a zero threshold is used, then all the terms are retained in the sets; a threshold set to the value 1 will cause all the terms to be rejected. Unknown terms are always rejected.

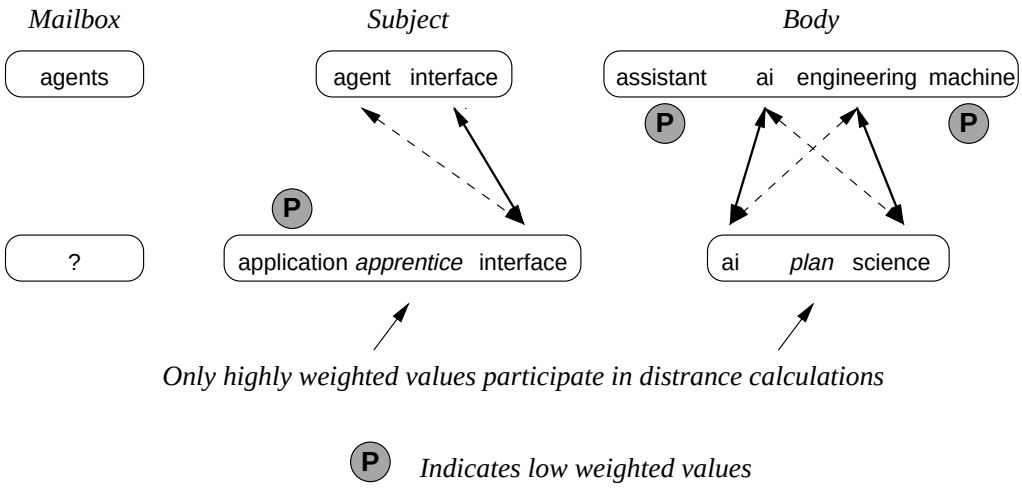


Figure 4.4: Comparing set-valued attributes using PIBPL. The solid lines denote similar matches between two symbols. The dashed lines denote distant matches between two symbols.

Figure 4.4 illustrates how *PIBPL* calculates the distance between sets of terms. Distance calculations are also performed for the highly weighted terms; unknown terms, or poorly weighted terms are ignored. The term '*interface*' in the *Subject* set is compared with the two terms in the training instance. Two of the three terms in the query instance *message body* set are compared with the two highly weighted terms in the training instance set. The unknown term '*plan*' in the query instance is ignored.

4.3 Discussion

The representation proposed above allows complex, structured documents to be represented in a compact form, whilst retaining the structure of the document. However, the size of the instances will be dependent on the number of unique terms

appearing in each of the documents, and hence the number of terms in each set can still be considerable. Irrelevant or redundant terms may also affect the distance calculated between different instances. For these reasons, *PIBPL* attempts to identify the relevant terms within each set. Chapter 5 presents results of a number of different experiments in which these learning algorithms are evaluated within the framework of an information agent.

The techniques described above have so far been discussed with respect to learning from complex documents. However, they could equally be applied to other domains, where more than one value may be required for a single attribute.

Chapter 5

An Empirical Evaluation of the Set-based Data Representation

Chapter Outline

The different set-based nearest neighbour learning algorithms, introduced in Section 4.2 are evaluated across two different domains. IBPL1 is compared to the rule induction algorithm CN2 and IBPL2 using an email filtering domain. A USENET news domain is used to evaluate PIBPL, and its use of weights to eliminate terms from the set-valued attributes is explored.

5.1 Comparison of IBPL1 and CN2

The previous chapter introduced the set valued attribute representation, and presented a selection of nearest neighbour learning algorithm that utilised this representation. The first of the new learning algorithms, *IBPL1* (Section 4.2.1), is evaluated within this Section, and its performance is compared with that achieved by an existing rule-based approach within the context of an email filtering task (Payne 1994).

5.1.1 The Email Data Set

The data set consists of electronic mail messages organised into twelve different mailboxes. Each mailbox contains varying numbers of messages with different properties. The mailboxes are listed in Table 5.1, with the number of messages in each mailbox and a description of the mail topic. Three of the mailboxes, *dai*, *kdd* and *mead* contain *digests* originating from a mailing list. These digests consist of individual messages which have been grouped together by a moderator (either manually or automatically). Thus, all the messages in the digest mailboxes will contain the same entries for the *From* and *Subject* fields. In addition, the terms selected from the *message body* represent the subject matter of the digest, as opposed to any individual topic.

Terms were selected from the *From* field, the *Subject* field and the message body of each mail message. All the terms identified in the *From* field were selected. Common words which are held in a *stop-list*, such as *'and', 'the'*, etc. were removed from the *Subject* field and the *message body*. The remaining features in the *Subject* field were then selected. However, only the top N (e.g. 10) most frequently occurring words from the *message body* were selected. Figure 5.2 illustrates the sets of terms extracted from the mail message in Figure 5.1 ($N = 4$).

Mailbox	Size	Notes
agents	61	*Messages about the specific topic of 'agent' research.*
cfp	11	*Messages containing 'calls for papers', with the majority of messages originating from the same source.*
cure	53	*Messages from a music related mailing list.*
dai	9	*Monthly periodic mailing list digest.*
jobs	25	*Messages relating to posts in academia from various sources.*
kdd	27	*Monthly periodic mailing list digest.*
mbox	20	*Messages covering a wide range of topics.*
mead	90	*Regular mailing list digest (two to three per week).*
personal	32	*Personal communications.*
phd_pos	64	*Messages regarding a specific topic from a small number of users.*
phd_stuff	13	*Various messages relating to current research.*
tea	3	*Messages on a specific topic.*

Table 5.1: Characteristics of the different mailboxes.

```
From:  terry@csd.abdn.ac.uk (Payne)
Subject:  An Agent Interface

Here is an example of a machine learning (AI based)
assistant that applies AI techniques to machine
engineering problems.
Enjoy,
         Terry.
```

Figure 5.1: An example mail message.

5.1.2 Classifying Email with CN2

CN2 (Clark & Niblett, 1989) is a supervised rule induction algorithm that uses a divide and conquer approach to induce an ordered rule set. It was previously embedded within the *Magi* email filtering agent (Payne 1994) to induce a set of email filtering rules. The learning algorithm was modified to map single email messages into multiple training instances, and from these instances, to induce the rules. The following two subsections summarise the CN2 learning algorithm, and describe the modification which enabled CN2 to induce email filtering rules.

$$\begin{array}{ll} From = \{\ terry@csd.abdn.ac.uk\ payne\ \} & m = 2 \\ Subject = \{\ agent\ interface\ \} & n = 2 \\ Body = \{\ assistant\ ai\ engineering\ machine\ \} & o = 4 \end{array}$$

Figure 5.2: Sets of terms extracted from an email message (the size of each set is shown to the right).

The CN2 Rule Induction Algorithm

The CN2 rule induction algorithm works in an iterative fashion, by applying a "best-set-so-far" beam search to a size limited set of *complexes*, where a complex is a conjunction of attribute tests. The complexes considered are then specialised to reduce the coverage of examples of other classes whilst maximising the coverage of a given class. Each new complex is evaluated and ranked using two evaluation functions, to determine its quality and significance. The first is a *Laplacian* error estimate (Equation 5.1) which assesses the quality of the complex in making a classification:

$$RelativeError(n, n_c, k) = \frac{n - n_c + k - 1}{n + k} \tag{5.1}$$

where n is the total number of examples covered by the rule, n_c is the number of positive examples covered by the rule, and there are k classes.

The second evaluation determines whether the complex is *significant*. A complex is significant if it contains a regularity unlikely to occur by chance, and thus reflects a genuine correlation between attribute values and classes. CN2 measures this by comparing the *observed* distribution among classes of examples satisfying the complex with the *expected* distribution resulting from the complex selecting examples randomly.

This significance value is calculated by using a likelihood ratio statistic (Equation 5.2) where the distribution $F = (f_1, ..., f_n)$ is the observed frequency distribution, and $E = (e_1, ..., e_n)$ is the expected frequency distribution.

$$2\sum_{i=1}^{n} f_i \log\left(\frac{f_i}{e_i}\right) \tag{5.2}$$

The resulting complexes are then combined to produce a set of ordered production rules which can be written to a file (see Figure 5.3).

```
. . .
IF feature = assistant
THEN mailbox = "agents" [0 0 0 0 11 0 0 0 0 0 0 0 0]
ELSE
IF subjattr = agent
THEN mailbox = "agents" [0 0.00 10 0 20 0 0 0 0 0 0 0 0.00]
ELSE
. . .
```

Figure 5.3: Sample rules generated by CN2.

Mapping Email Messages to Training Instances

Magi (Payne 1994) represented terms (previously extracted from email messages) in the form of sets, where each set corresponded to one of the email fields. These had to be mapped into a representation that could be used by CN2. Instead of using the vector representation for each message, multiple training instances were generated. The instances were represented by three symbolic attributes, where each attribute corresponded to one of the email fields. The number of instances generated for each email message was dependent on the number of terms extracted from the different email fields. For example, in Figure 5.4, there are 2 features in the *From* field, a single feature in the *Subject* field and 3 features in the *message body*. This will result in $2 \times 1 \times 3 = 6$ examples. Figure 5.5 shows some of the training examples generated from the term shown in Figure 5.2. The last field in the examples shown contains the action performed on that mail message, i.e. saving the message in the *agents* mailbox.

There is a risk that the generation of a large number of instances may result in a

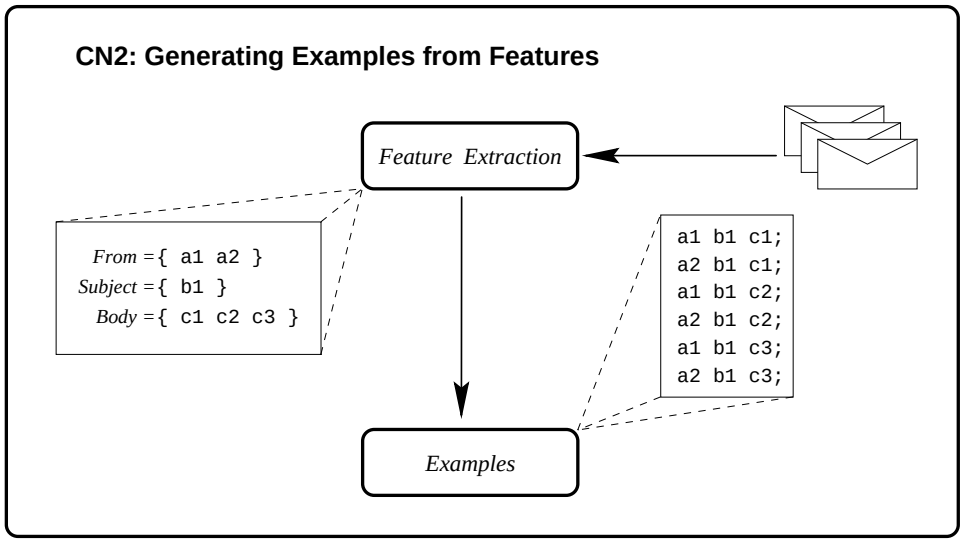


Figure 5.4: Generating multiple instances from feature sets.

bias towards features occurring within smaller sets over those within larger sets. In the example in Figure 5.4, there are two instances with the feature `c3` but all six instances contain the feature `b1`. This will affect the *Laplacian* error estimate (Equation 5.1) in exploring new complexes.

terry@csd.abdn.ac.uk	*agent*	*assistant*	`agents;`
terry@csd.abdn.ac.uk	*interface*	*assistant*	`agents;`
terry@csd.abdn.ac.uk	*interface*	*ai*	`agents;`
payne	*agent*	*learning*	`agents;`

Figure 5.5: Training instances for CN2.

This approach was also used to generate multiple query instances from new email messages, prior to classification. As each instance was presented to the induced rule set, it would cause one of the rules to fire. However, not every fired rule necessarily had the same antecedent, and hence a conflict resolution strategy was required to determine an overall classification for the new message. A majority/threshold strategy was used (Payne, 1994), whereby the total number of times each antecedent fired was determined, and the highest total was compared with a threshold. If this highest total was greater than the threshold, then the

corresponding antecedent was returned, otherwise an *unknown* classification was returned.

5.1.3 Experimental Method

The two learning algorithms were evaluated using the *holdout* evaluation approach (Kohavi, 1995). This approach is particularly suited to identifying the shape of the learning curve, i.e. determining how the performance of the learning algorithm varies as the number of messages in the training set increases. The original data set was randomly partitioned into a training data set and a test data set. The messages within the two data sets were processed to select the sub-sets of terms from the different fields. These sub-sets were mapped into training or testing instances; either as single instances for *IBPL1*, or multiple instances for CN2 (Section 5.1.2). The two learning algorithms were then trained on the instances in the training set. The performance of each algorithm was determined by presenting it with the instances from each message in the test set, and comparing the resulting classification with the value held within the test message's class label.

The size of the training partition was varied (in 10% intervals) from 20% to 80% of all email messages. The remaining messages were used to construct the test partition. The division of messages from each mailbox was stratified[1] (Kohavi, 1995), so that the performance of both learning algorithms could be compared for individual mailboxes. The tests were repeated 30 times with randomly partitioned data sets to allow statistical analysis.

Two measurements were made for each test: *accuracy* and *coverage*. The rule-based approach used a majority/threshold strategy (Section 5.1.2) to determine an overall classification from the individual classifications of the multiple instances (generated from each mail message). If the number of instances classified as the majority class was lower than the confidence threshold (set to 30%), then no final overall classification was generated. The *coverage* denotes the number of messages

[1]For example, the 40% training data partition contains 40% of the messages from the *agents* mailbox, 40% of the messages from the *cfp* mailbox, e.t.c.

that could be classified. The *accuracy* represents the number of messages that were correctly predicted for each mail box.

IBPL1 determined whether a classification should be made by examining the 10 nearest neighbours to each query instance (Section 2.2.2). A confidence rating was generated for each class by calculating the reciprocal of each of the 10 distances, and summing the distances for instances of that class, i.e.

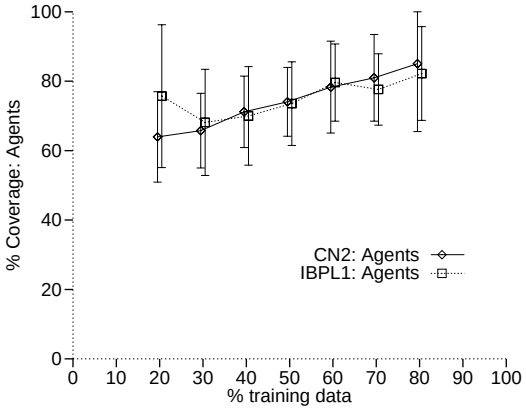
$$conf(c) = \sum_{\forall j: class(j)=c} \frac{1}{D(i,j)} \tag{5.3}$$

where i is the query instance, and j is one of its 10 nearest neighbours. The distance between i and each nearest neighbour is given by $D(i,j)$ (Equation 4.2). A final confidence rating is determined by finding the difference between the confidence rating for the majority class, and the total confidence ratings for the other classes (Payne & Edwards 1997). This final confidence rating is then compared with a threshold to determine whether or not a classification should be made.

5.1.4 Using three main Email fields

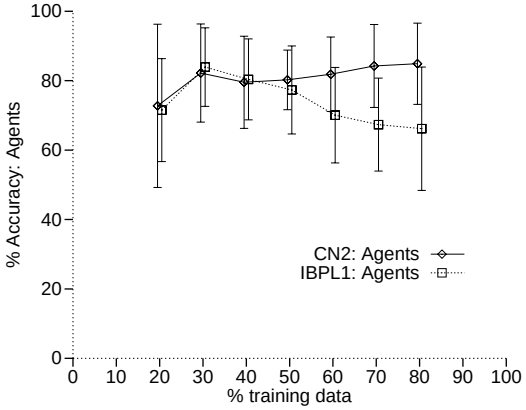
CN2 and *IBPL1* were trained with examples containing terms from all three fields (*From*, *Subject* and the 10 most frequently occurring words from the message body). Both algorithms were able to make predictions on new messages. When using CN2, predictions were made for approximately 78% of messages; this dropped to 74% when using *IBPL1*, although the exact figures varied depending on the mailbox tested. For example, when using CN2, approximately 57% of messages from the *jobs* mailbox generated predictions compared to 90% of messages from the *kdd* mailbox. The number of predictions made by CN2 steadily rises as the number of training examples increases. However, with most mailboxes, the number of predictions made by *IBPL1* initially falls, but then rises as the number of training examples increases beyond 40-50%. Figure 5.6 shows a comparison of CN2 with *IBPL1* when testing on messages from the *agents* mailbox.

%	CN2	*IBPL1*
20	63.95	75.70
30	65.75	68.14
40	71.19	70.00
50	74.06	73.55
60	78.30	79.62
70	80.97	77.63
80	85.04	82.22



Coverage - *Agents* Mailbox.

%	CN2	*IBPL1*
20	72.75	71.52
30	82.20	83.93
40	79.55	80.37
50	80.24	77.32
60	81.86	70.05
70	84.25	67.35
80	84.88	66.17



Accuracy of Predictions - *Agents* Mailbox.

Figure 5.6: Comparing CN2 with *IBPL1* on the *Agents* Mailbox.

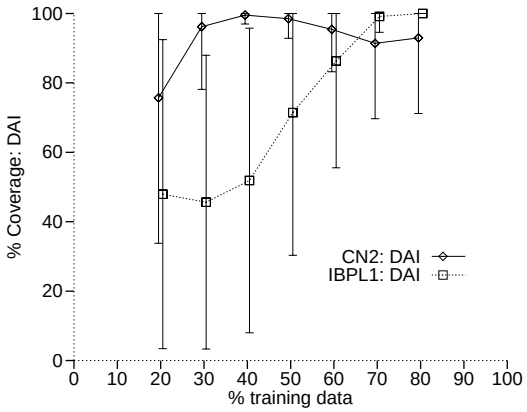
Approximately 65% of all predictions made by CN2 were correct, compared to approximately 57% for those made by *IBPL1*. The number of correct predictions made by *IBPL1* tends to increase and peak between 30-50% of the data, then falls off as the number of training examples increases beyond this point. This appears to correspond to the changes in the number of predictions made by *IBPL1* (see Figure 5.6).

Small Mailboxes

For the smaller mailboxes, such as *cfp, dai* and *tea*, CN2 gave rise to more predictions than *IBPL1*. The accuracy of the predictions made by CN2 is low for

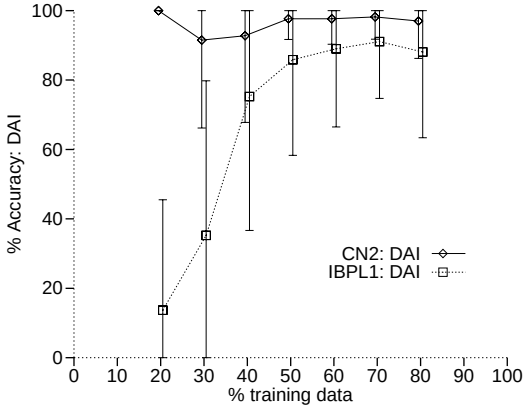
cfp (only approximately 10% of predictions were correct) but much higher for *dai* and *tea*, both of which result in prediction accuracies over 90%. The accuracy of *IBPL1* for these mailboxes varied but overall was poor, with no accurate predictions being made for *tea* and only approximately 2% to 12% of *cfp* predictions being accurate. The exception here was the *dai* digest mailbox, where the accuracy of predictions increased rapidly as the training data increased. This can be seen in Figure 5.7.

%	CN2	*IBPL1*
20	75.71	47.94
30	96.19	45.63
40	99.52	51.88
50	98.50	71.44
60	95.44	86.33
70	91.38	99.14
80	92.95	100.00



Coverage - *DAI* Mailbox.

%	CN2	*IBPL1*
20	100.00	13.68
30	91.52	35.25
40	92.78	75.24
50	97.67	85.87
60	97.67	88.99
70	98.21	91.09
80	97.00	88.10



Accuracy of Predictions - *DAI* Mailbox.

Figure 5.7: Comparing CN2 with *IBPL1* on the *DAI* Mailbox.

The poor overall performance of *IBPL1* on small data sets is a characteristic of using a k-NN approach (Section 2.2.2). The top k closest matches are considered when determining the confidence and prediction. If the number of training instances is less than k, then any similar matches, however poor, will be taken

into consideration when determining these values. Tests on smaller values of k demonstrated an increase in performance for these small mailboxes.

Digest Mailboxes

Both CN2 and *IBPL1* were able to generate predictions for a high proportion (85%+) of messages from two of the digest mailboxes, *kdd* and *mead*, with near 100% accuracy. The third, smaller digest mailbox, *dai*, resulted in a slightly lower number of predictions when CN2 was used. When using *IBPL1*, training with smaller amounts of data (e.g. 30% of the data set) resulted in a small number of predictions for the *dai* mailbox. The accuracy of these predictions was also low. However, as the number of training examples increased, the performance of *IBPL1* improved. This supports the explanation given above regarding k-NN voting strategies, as the *dai* mailbox contained only 9 messages.

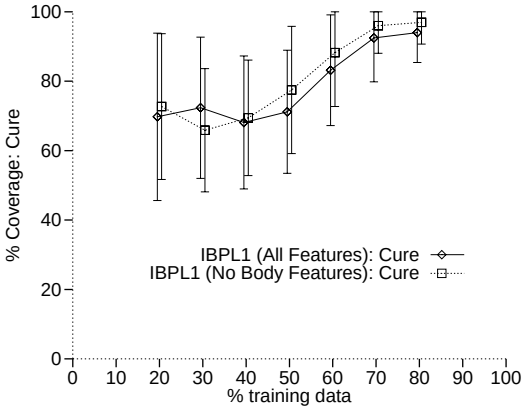
5.1.5 Changes in Body Terms

Previous work (Payne, 1994) demonstrated that whilst predictions could be made on mail messages based purely on message body terms, an improvement in performance was achieved by including the *From* and *Subject* fields. The work here expands on this previous study by exploring the effect of excluding the message body terms used whilst employing these extra fields.

With CN2 there is a slight degradation in performance overall when message body terms are reduced or omitted. A drop in the number of predictions occurs for all non-digest mailboxes with the exception of *personal* and *phd_poss*. The accuracy of the predictions is also slightly lower. The digest mailboxes were unaffected by changes in the use of message body terms, as predictions for these messages are based on the contents of the *From* and *Subject* fields only.

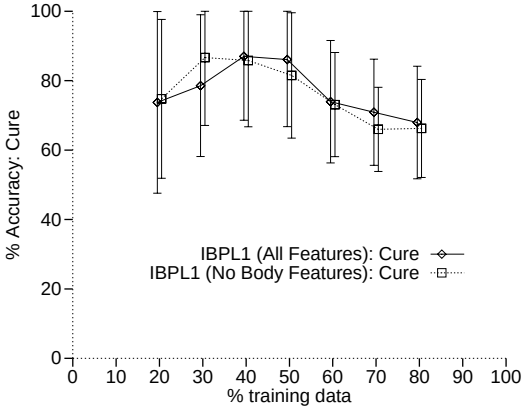
If terms from the message body are omitted when using *IBPL1*, the number of predictions increases slightly after 40% training data. However, there is a corresponding decrease in the accuracy of these predictions. Figure 5.8 shows the

%	With Body Terms	Without Body Terms
20	69.74	72.70
30	72.36	65.88
40	68.11	69.45
50	71.19	77.49
60	83.17	88.24
70	92.45	95.99
80	94.01	96.98



Coverage - *Cure* Mailbox.

%	With Body Terms	Without Body Terms
20	73.75	74.77
30	78.56	86.65
40	87.00	85.80
50	86.07	81.51
60	73.95	73.11
70	70.90	65.98
80	67.95	66.25



Accuracy of Predictions - *Cure* Mailbox.

Figure 5.8: Comparing the inclusion and omission of *message body* terms on the *Cure* Mailbox.

number of messages for which predictions were made and the accuracy of these predictions when message body terms are present and absent.

5.1.6 Omitting Other Fields (*Subject* and *From*)

If the *Subject* terms are omitted when using *IBPL1*, the number of predictions increases, while the accuracy falls. This behaviour is similar to that exhibited when omitting message body terms, and indicates the importance of these terms in making accurate predictions when using *IBPL1*.

With CN2, there was a reduction in the number of predictions made for some of

the mailboxes when *Subject* terms were omitted, namely *agents, cure* and *personal*. However, the accuracy of the predictions for messages in these mailboxes increased. No differences were observed in the number or accuracy of predictions made for messages from the other mailboxes.

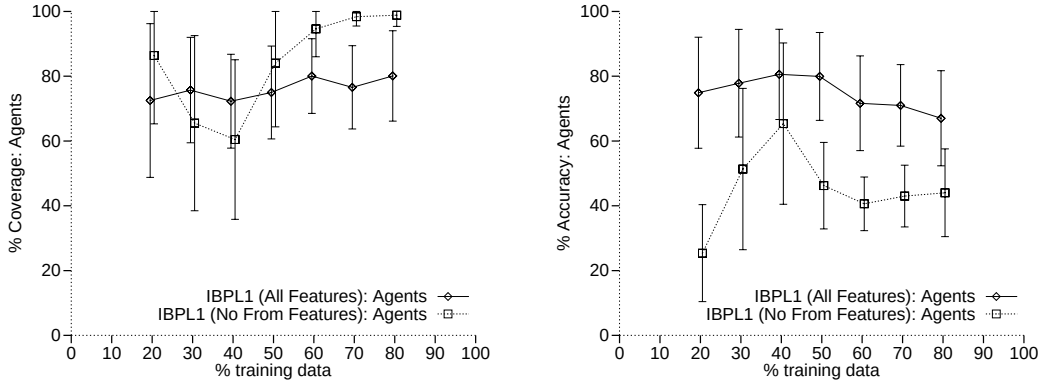


Figure 5.9: Comparing the inclusion and omission of *From* terms with *IBPL1* on the *Agents* Mailbox.

From our investigation, CN2 appears to be more dependent on the *From* terms than *IBPL1*. In most cases, omitting the *From* terms when using CN2 resulted in a reduction in the number of predictions made. This drop was more pronounced for certain mailboxes, such as *cfp, mbox, phd_stuff* and *phd_poss*. The change in accuracy of the predictions varied depending on the mailbox. For example, there was no noticeable change in the accuracy of predictions for messages from the *phd_poss* mailbox, yet the accuracy of predictions for *phd_stuff* increased by approximately 15-20%. Some mailboxes (such as *jobs*) resulted in an increase in accuracy without an overall increase in the number of predictions.

Figure 5.9 shows the behaviour of *IBPL1* for a non-digest mailbox when *From* terms are omitted. This behaviour can be explained by considering the performance of *IBPL1* before and after 40% training data:

- As the proportion of the data in the training set increases to 40%, the number of predictions made decreases. However, the accuracy of these predictions increases.

- As the training data increases beyond 40%, the number of predictions made increases again. There is an initial reduction in the accuracy which then plateaus at approximately 47%.

The indications here are that with smaller training sets, *IBPL1* depends heavily on the *From* terms to make predictions. When they are omitted, many incorrect predictions are initially made. As the training set increases in size the number of accurate predictions increases.

5.1.7 Discussion

The above evaluation has demonstrated that the performance of the two learning algorithms vary with respect to their ability to accurately predict user actions for incoming messages. The characteristics of different message types vary, so whilst one learning approach performs better with certain types of messages, another approach may perform better with other types of message. For example, the performance of IBPL1 was superior to CN2 when making predictions for messages from the *phd_stuff* mailbox, but inferior when making predictions for messages from other mailboxes.

The results indicate that the *From* and *Subject* fields help to reduce the number of incorrect predictions made. For example, let us consider the changes in performance for each algorithm when *From* terms are used. With CN2, the number of accurate predictions increases; however, with *IBPL1*, the number of accurate predictions appears to be constant, but the number of incorrect predictions is reduced.

One important difference between the two algorithms is the time taken to induce and apply user profiles to new mail messages. The instance-based approach builds a sub-symbolic representation in the form of weights and distance metrics. Unlike rule induction in CN2, these calculations do not involve searching through a large space of possible solutions. The search performed by CN2 is compounded by the large number of terms generated by the message body. It was found that

tests involving CN2 took significantly (30 to 40 times) longer than tests involving *IBPL1*. However, it should be remembered that these timings include the time taken for CN2 to induce the rules (a computationally expensive process). Hence, no conclusions can be drawn for either method regarding the time taken to predict the action for a single message.

5.2 Comparison of IBPL1 and IBPL2

Section 4.2.2 introduced an alternative distance metric to the one used by *IBPL1*. Instead of averaging all the distances between symbols within two sets, the new distance metric averaged only the closest distances to the symbols within the query set. *IBPL2* is a variant of *IBPL1* that utilises this distance metric. To evaluate its performance, *IBPL2* was compared with *IBPL1* using the Email data set described above (Section 5.1.1).

Both *IBPL1* and *IBPL2* performed equally well when applied to classifying email messages. Figures 5.10 and 5.11 illustrate the number of messages for which predictions are made (i.e. coverage), and the accuracy of those predictions for both algorithms on the *Agents* mail box (Figure 5.10) and *Phd_Possibilities* mail box (Figure 5.11).

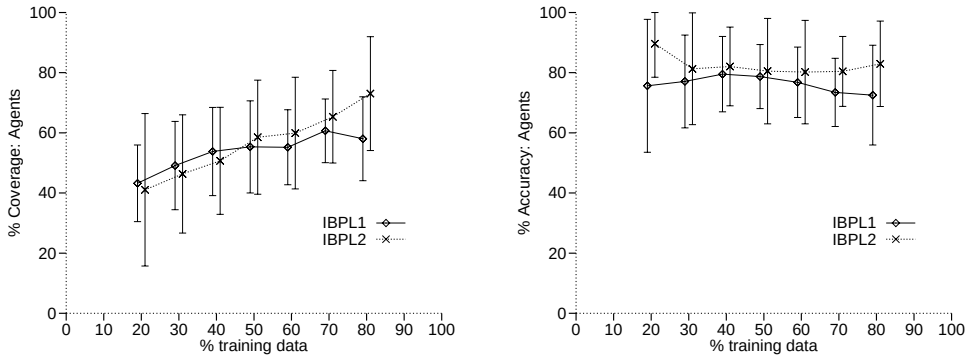


Figure 5.10: Comparing IBPL1 & IBPL2 on the *agents* data set.

These results demonstrate that the set-valued attribute representation can be successfully used by a nearest neighbour learning algorithms to filter electronic

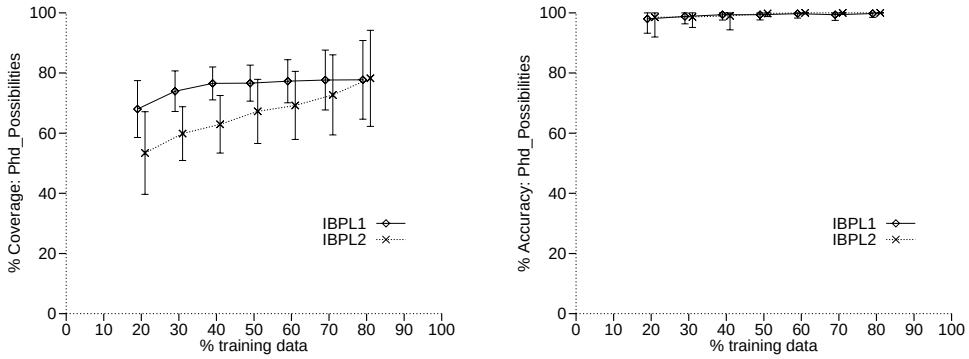


Figure 5.11: Comparing IBPL1 & IBPL2 on the *phd* data set.

mail messages. However, although the performance of *IBPL1* and *IBPL2* vary slightly for different mailboxes, there is no overall significant difference between the results of the two algorithms.

5.3 Evaluation of PIBPL

The *PIBPL* algorithm (Section 4.2.3) attempts to identify and eliminate irrelevant elements within a set-valued attribute, by utilising the VDM weights (Equation 2.8). To evaluate this algorithm, an alternative data set was used. A USENET News data set consisting of varying numbers of news articles from five different USENET news groups was obtained[2] and is summarised in Table 5.2. A similar technique to the one described in Section 5.1.1 was used to extract terms from three of the news fields: *Author*, *Subject*, and the article body. The learning task was to determine the news group from which each article had been taken.

The *holdout* evaluation approach (Kohavi, 1995) was again used to compare the learning curves of the different nearest neighbour algorithms. The size of the training partition was varied from 10% to 90% of the news articles in 10% intervals. Each test was repeated 25 times with randomly partitioned data. A number of selection thresholds were tested in the range [0..1], in intervals of 0.1.

[2]The USENET News data set was kindly supplied by Claire L. Green, Department of Computing Science, King's College, University of Aberdeen, Scotland, AB24 3UE.

Articles	USENET News Group
126	*abdn.student*
226	*alt.education.research*
112	*alt.lefthanders*
503	*rec.humor*
178	*sci.stat.math*

Table 5.2: Characteristics of the USENET News data set.

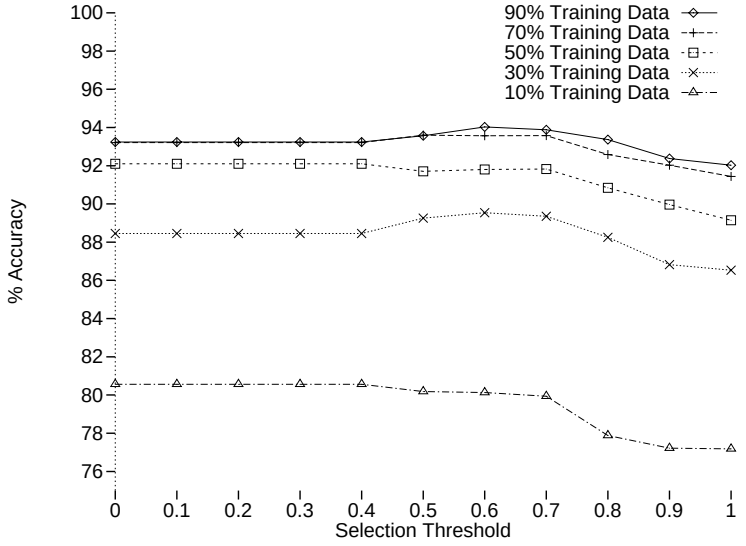


Figure 5.12: Comparing *PIBPL* with various selection thresholds on the USENET news data set.

Figure 5.12 illustrates the effects of varying the selection threshold for *PIBPL*. In general, the larger the size of the training set, the higher the classification accuracy of *PIBPL*. The accuracy of *PIBPL* remains constant with small selection thresholds, but then varies once the threshold reaches 0.5. This can be explained by considering the uniform distribution weight, $\omega(u)$ (Equation 2.9). This represents the weight of a symbolic value which appears in instances of each class with an equal (i.e. uniform) probability. It also corresponds to the minimum possible weight for the domain being tested, as its value is a function of the number of different classes. Instances within the USENET news data set belong to one of five classes, and hence $\omega(u) = |5|^{-0.5} = 0.447$.

The average classification accuracy rises slightly for selection thresholds between

the range [0.5..0.7], reaching a maximum average classification accuracy of 90.42% when the selection threshold = 0.6 (this compares to an average accuracy of 90.22% when the selection threshold ≤ 0.4, i.e. no symbols are rejected). This suggests that the elimination of low weighted symbols can result in a slight increase in the classification accuracy of *PIBPL*. However, as the selection threshold rises above 0.7, the overall accuracy falls below that achieved when no pruning was performed (i.e. when the selection threshold was ≤ 0.4). This behaviour is not surprising, as one would expect the accuracy of *PIBPL* to fall as the highly weighted terms are removed from the attribute sets.

Figure 5.13 illustrates the different learning curves for *IBPL1*, *IBPL2* and *PIBPL* (using three selection thresholds: 0.0, 0.6 and 1.0) on the USENET news data set. The learning curves are very similar for *PIBPL* with the selection thresholds 0.0 and 0.6, although there is no significant difference between these two curves at the 5% level (using a one-tailed paired t-test). However, if a selection threshold of 1.0 is used, then the curve significantly falls by an average of 2.4% from that achieved with a 0.0 selection threshold.

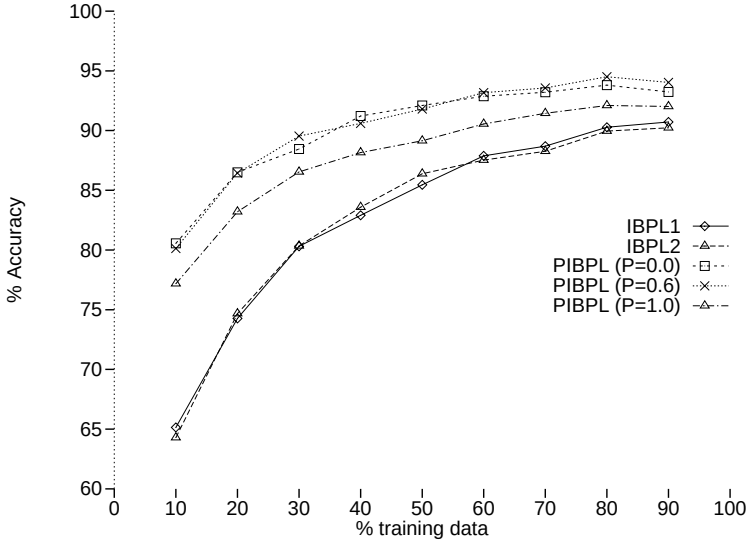


Figure 5.13: Comparing the learning curves of IBPL1, IBPL2 and PIBPL on the USENET news data set.

There is no significant difference between the performance of *IBPL1* and *IBPL2* on the USENET news data, thus supporting the results presented in Section 5.1.

However, both curves were significantly lower than those achieved by *PIBPL*. For example, the difference in accuracy between *IBPL2* and *PIBPL* (with a 0.0 threshold) ranged from an average of 16.41% (for 10% training data) up to 2.78% (for 90% training data). As the threshold is too small to reject any of the symbols, the only difference between these two algorithms is their behaviour with respect to unknown symbols. *PIBPL* discards unknown symbols, whereas *IBPL2* assigns a maximum distance to such symbols. Hence, the decision to use the maximum distance within both *IBPL1* and *IBPL2* may have been inappropriate for the domains tested so far. This may also explain the greater discrepancy in accuracy between *IBPL2* and *PIBPL* for the smaller training data sets. As the size of the training set increases, one would also expect the number of different symbols present in the training set to increase. Thus, the number of unknown symbols encountered within the test data set will decrease, and the detrimental effects of unknown symbols on the classification accuracy will diminish.

5.4 Related Work

This chapter has introduced three novel nearest neighbour algorithms designed to learn classification hypotheses from high dimensional symbolic data. So far, the techniques have been tested on domains containing English text. However, the approaches presented are suitable for any form of unordered symbolic data.

The *set-valued attribute* representation introduced in Section 4.2 was first presented as part of a nearest neighbour algorithm known as *MBR* in Payne et al. (1995). The name was then changed to *IBPL1* (Payne & Edwards 1995) to avoid confusion with the nearest neighbour algorithm employed by the MBRtalk system (Stanfill & Waltz 1986).

Set valued attributes have subsequently appeared within other learning paradigms. Cohen (1996b) proposed a method for learning rules from this representation. When the decision metric (such as ID3's Information Gain (Quinlan, 1986) or the Distance-Based Gain Ratio (De Mántaras, 1991)) is calculated, instead of per-

forming an equality test, a set membership test can be performed. The greedy rule induction algorithm, RIPPER (Cohen, 1995) was modified in this way. RIPPER constructs a set of rules by repeatedly adding rules to an initially empty set until all the positive instances of a concept are covered. The rules are constructed by greedily adding conditions (to an initially empty antecedent) until no negative instances are covered. The rules are then pruned to reduce the size of the rule set, and to avoid overfitting.

The modified RIPPER rule induction algorithm was evaluated using a variety of email filtering domains (Cohen 1996a). The learning algorithm was compared with a *tfidf* weighted classification algorithm, based on Rocchio's relevance feedback algorithm (Rocchio Jr 1971). Terms are parsed from the *From*, *To*, *Subject* and *message body* fields of electronic mail messages, and represented using the vector space representation (Figure 1.2). Each element of the vector corresponds to a term that appears within one of the four email fields. The value of each element is dependent on the number of times the corresponding term appears in the respective field of the email message; the total number of terms in the email message, and the inverse frequency of the term also in other messages. A set of prototypical class vectors are then generated from the vector representations of all the training email messages; where each vector represents a single class. New email messages are mapped into a vector space representation, and compared to the prototypical class vectors. The resulting classification is then determined by finding the most similar prototypical class vectors.

In general, the performance of RIPPER was comparable to that achieved by the *tfidf* classifier when filtering email messages, and was only occasionally statistically significant. The results demonstrated that the classification accuracy increased asymptotically as the number of training instances increased, but that the optimal classification accuracy of both algorithms varied for different email domains. These results are similar to those reported for *IBPL1* (Section 5.1).

An alternative approach to data representation was proposed by Boone (1998) as part of the *Re:Agent* email filtering system. It is based on the premiss that email messages can be used to define high-level *concept features*, which are in turn used

to represent the email messages prior to learning. The method used to construct *concept features* is similar to that used to construct *tfidf* prototypical class vectors. The email messages are first grouped together so that each group represents a concept. For example, all the email messages relating to robotics research would be grouped together to represent the *robotics* concept[3]. The messages are then mapped into a vector space representation, and the *tfidf* weights are calculated for each of the terms $(\varepsilon_1, \ldots, \varepsilon_n)$ in the message. The *centroid* vector is then calculated for each concept by averaging the values of the *tfidf* weights for each of the elements in the vectors.

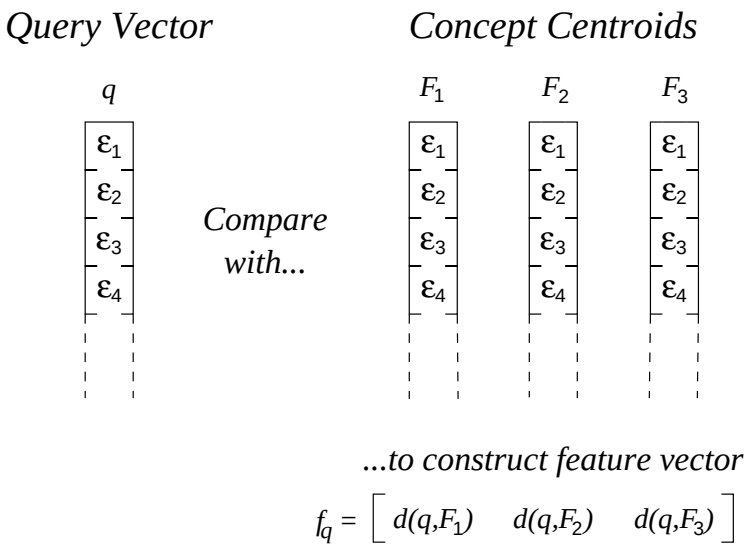


Figure 5.14: Generating features from concept centroids with *Re:Agent* (Boone, 1998).

The resulting concept centroid vectors are not directly used for classification, but are used to construct concept features. Each centroid corresponds to one of the attributes in the final representation. For example, Figure 5.14 illustrates three concept centroid vectors F_1, F_2 and F_3, which can be used to construct values for the three attributes in the final representation. An email message is converted into its vector space representation, and the *tfidf* weights calculated (e.g. q in Figure 5.14). The *cosine* distance metric (Section 2.2.3) is then used to calculate

[3]For many domains, there may be no distinction between a concept and the target classification.

the distance between the email message and each of the concept centroids. For example, the message q is mapped into the three element feature vector f_q as follows:

$$\forall a \in A : f_q[a] = d(q, F_a)$$

where a is an attribute in the set of all attributes, A, and the value $f_q[a]$ is the value of the a'th element of the vector f_q.

Once all the messages in the training set have been converted into feature vectors, they can be presented to a learning algorithm. Boone evaluated this approach using two different learning algorithms, a simple nearest neighbour algorithm, and the Backpropagation neural network algorithm (Rumelhart, Hinton, & Williams 1986). The concept feature approach was also contrasted with a *tfidf* weighted classification algorithm. The results suggested that the combination of generating concept features and presenting them to a learning algorithm could classify a significantly higher percentage of email messages than the *tfidf* approach. However, there appears to be no significant difference in the performance of the two learning algorithms tested.

5.5 Conclusions

The set-valued attribute representation provides a mechanism for encoding high dimensional data (such as large, sparse, document vectors) in a condensed, propositional form. The performance of algorithms using set-valued attributes is comparable to that of other classification approaches, such as CN2 or *tfidf* weighted algorithms. Ripper differs from *IBPL1* in that it generates *keyword-spotting rules* (Cohen 1996b) by greedily selecting elements from the training data. Thus, one would expect that irrelevant or redundant elements appearing in the set-valued attributes would not appear in the induced rule set[4]. As *IBPL1* is a nearest

[4]A number of studies have demonstrated that the decision metrics used by various rule induction techniques can be used to eliminate irrelevant and redundant attributes (Almuallim

neighbour learning algorithm, there is no implicit mechanism for eliminating irrelevant or redundant elements. Hence, it may be susceptible to the presence of such elements (Section 2.3). This may necessitate the need for an additional pre-processing stage responsible for removing such symbols. Although *PIBPL* has demonstrated that probablistic approaches can be used to eliminate irrelevant or redundant data, many other selection approaches have been proposed (Chapter 3).

& Dietterich 1991; Cardie 1993; Kubat, Flotzinger, & Pfurtscheller 1993). See Section 7.3 for details.

Chapter 6

Novel Approaches for Dimensionality Reduction

Chapter Outline

In earlier chapters, a number of existing dimensionality reduction approaches were introduced, and their applicability to the nearest neighbour paradigm discussed. Many of these techniques have been shown to improve the classification accuracy of a learning algorithm on a number of different data sets. However, some of these techniques are unsuited to domains where the dimensionality is high, such as the problem of text categorisation. This chapter presents a variety of different approaches to the task of dimensionality reduction, and discusses the rationale behind each approach. An evaluation strategy involving a nearest neighbour learning algorithm is then presented.

6.1 Introduction

Attribute selection and dimensionality reduction techniques have been proposed that attempt to reduce the detrimental effects of irrelevant or redundant attributes on the performance of a number of different learning algorithms. Some techniques utilise simple heuristics to identify whether the inclusion of certain attributes are necessary to preserve the consistency of the data set (Almuallim & Dietterich 1991), or select only certain attributes based on statistical criteria (Yang & Pedersen 1997). Other approaches determine a vector of weights and utilise these to augment or attenuate the effects of individual attributes (Wettschereck & Dietterich 1995), or to select a subset of attributes (Kira & Rendell 1992a). A number of other techniques perform a heuristic search through a space of attribute subsets, evaluating each state within this space by empirically testing the learning algorithm with a data set containing the subset of attributes.

However, these studies (summarised in Section 3) have demonstrated that there is no single approach suited to the characteristics of all data sets. Approaches that consider each attribute independently of the other attributes may reject those attributes whose values are dependent on the values of another. However, whilst approaches that evaluate the merits of attribute subsets may capture such dependencies, they may be computationally intractable when the dimensionality of the domain is large (such as in text categorisation problems).

This chapter investigates a number of alternative approaches to dimensionality reduction and attribute selection. Each approach is discussed in detail, and the results of the evaluation of each approach on a variety of data sets is presented in the next chapter. However, before the approaches are presented, the nearest neighbour algorithm and evaluation environment used must be described.

6.2 Testing Framework

The 1-Nearest Neighbour learning algorithm (Section 2.2.2) is used to evaluate the dimensionality reduction techniques in this study. The nearest neighbours are determined by utilising a specified distance metric, and by comparing the test instance with each of the instances in the training data set. Only the single nearest neighbour is used to determine the class of a test instance.

Numerical values are normalised when numerical distance metrics are used. This is performed so that attributes with large numerical ranges do not overwhelm those with relatively smaller ranges (Section 2.2.3). Normalisation is achieved by determining the maximum and minimum values for each attribute in the training set and using these values to transform all the attribute values into the range [0..1]. Any values in the test instances that fall outside this range are mapped to the nearest boundary, i.e. $i_a = 1 \quad \textit{iff} \ (i_a > 1) \quad \textit{or} \quad i_a = 0 \quad \textit{iff} \ (i_a < 0)$ where i_a is the value corresponding to the attribute a within the instance i. Different transformations are used for each attribute. The transformation used is then applied to the test data set.

An evaluation framework was designed to partition the data sets in a consistent fashion and execute various experiments on this partitioned data. Although some of the standard UCI data sets used (Merz & Murphy 1996) are available partitioned into separate training and test data sets, the majority of data sets are provided as a single file. Different evaluation strategies partition this data in different ways (Kohavi 1995). For example, the *Holdout* method partitions a data set into two sets, a training set and a test set. The training set typically consists of two-thirds of the data set, the remaining third constituting the test set. Alternatively, the *k-fold cross validation* method partitions the data set into multiple folds of approximately equal size, and then tests the learning algorithm by selecting one of the k folds as a test set, and using the remaining folds as the data set. This is repeated until all k folds have been tested. This evaluation method was chosen for the testing framework, as other studies (Kohavi 1995) have demonstrated that the variance of the results generated by this method are lower

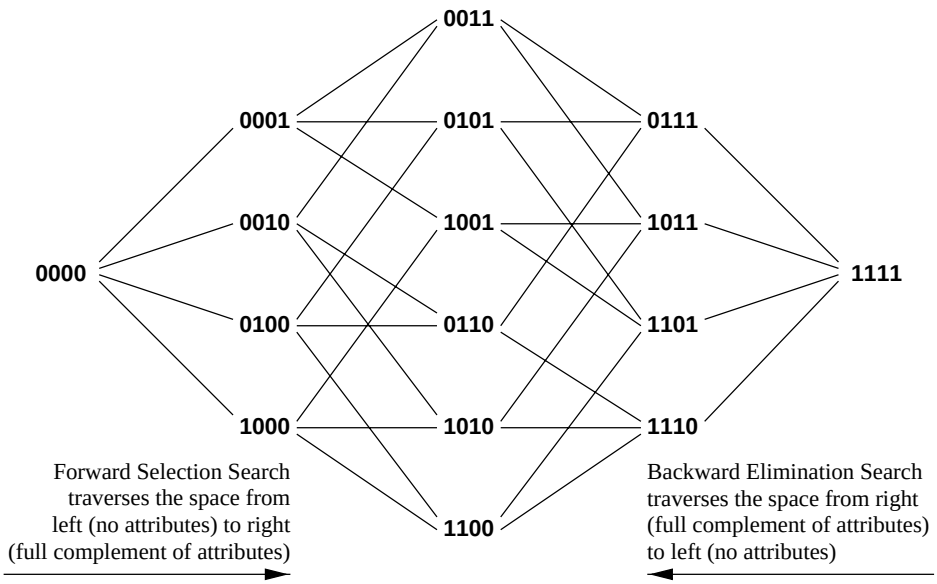


Figure 6.1: The various search states in a four dimensional ASV space (after Langley 1994).

than for other methods.

Each data set was partitioned into N folds. The value of N was dependent on the data type: 10 folds were used for the symbolic data sets, and 20 folds for the numeric data sets. The training data was then pre-processed in a variety of different ways, such as first normalising numeric training data, and then presenting the normalised data to an attribute selection algorithm. The details of each individual experiment are presented in Chapter 7.

6.3 Wrapper Method Framework

The wrapper model eliminates redundant and irrelevant attributes by evaluating the performance of a learning algorithm on different attribute subsets. A search method is used to explore the space of possible attribute subsets, and the results of each evaluation are used to guide the search. The framework used to implement the wrapper approach generates an *attribute selection vector* for a given data set. The attribute selection vector (ASV) is a binary vector whose length is equal to the

number of attributes in the original data set. Each ASV element corresponds to one of the attributes, and determines whether that attribute should be included in the new reduced data set. The search space consists of all combinations of ASVs, where each ASV corresponds to a *state* within this space (Figure 6.1). The ASVs of adjacent states differ by only one element, i.e. a move from one search state to an adjacent state is equivalent to the selection or rejection of one of the attributes in that initial state.

Each search state is evaluated as follows:

1. An attribute selection vector (ASV) is generated for the current state;
2. The ASV is applied to the data set to generate a new data set containing only the attributes selected;
3. The nearest neighbour algorithm is then evaluated with the new, reduced data set. The leave-one-out cross validation method is used;
4. The results of the evaluation are then used to guide the search algorithm.

Some studies utilise a *k-fold cross validation* approach when evaluating each attribute state. Whilst this may reduce the time taken to generate the final attribute subset, it would render the resulting subset dependent on the order in which the training data was presented to the wrapper. This dependency is due to the way in which the data is partitioned into folds. If a k-fold cross validation method is used, then the partitioning of the data should be deterministic, so that the evaluated search states can be compared.

Four search algorithms have been investigated and are described below. None of the approaches explicitly perform an exhaustive search through the state space; instead they utilise a heuristic or stochastic search strategy to locate optimal (or sub-optimal) states in the space. The *Forward Selection* and *Backward Elimination* algorithms utilise a greedy hill climbing approach (Devijer & Kittler 1982) to search for sub-optimal solutions. These methods have been investigated using

a variety of other learning algorithms (Caruana & Freitag 1994; John, Kohavi, & Pfleger 1994; Langley & Sage 1994b; Moore & Lee 1994; Singh & Provan 1995). The two methods differ in their starting conditions; the *Forward Selection* algorithm starts with no members in its attribute subset, and incrementally adds new attributes to this set. In contrast, the *Backward Elimination* method starts with a full complement of attributes, and progresses by eliminating irrelevant attributes. However, both approaches are susceptible to the problem of finding local maxima within the state space. To overcome this, the wrapper framework was tested with a *Simulated Annealing* search algorithm. Whilst this search algorithm is essentially a hill climbing algorithm, it is less susceptible to the effects of local maxima than either of the two greedy hill climbing approaches described above. In contrast, the *Monte Carlo* algorithm uses a stochastic sampling approach to find the optimal state. Instead of performing a heuristic search though the state space, it randomly samples different states to find the state with the highest evaluation function.

6.3.1 Forward Selection

The *Forward Selection* algorithm (FSS) starts with an empty attribute selection set, represented in Figure 6.2 as *selectset*. A record of the best selection set performance, $eval_{best}$, is initially set to zero. It explores the effects of adding an additional attribute to the current selection set by evaluating the performance of the selection set with each of the remaining attributes, i.e. $selectset \cup j$. If the inclusion of any of the attributes results in an increase in performance, then the attribute responsible for the highest increase in performance, j_{best}, is added to the selection set. The process is then repeated until no further increase in performance can be achieved. Figure 6.1 illustrates the legitimate moves that the algorithm can take from each state.

```
1  var
2      AttrSet                                   Set of all attributes
3      n                                  Total number of members in AttrSet
4  end
5  proc forward_selection(AttrSet, n) ≡
6      selectset = ∅;                    Selection Set contains no attributes
7      eval_best = 0;
8      for i = 1 to n do
9          j_best = ∅;
10         foreach j in AttrSet do
11             eval = evaluate(selectset ∪ j);
12             if (eval > eval_best)                  If eval is better than ...
13                 eval_best = eval;
14                 j_best = j;
15             fi
16         done
17         if (j_best == ∅)                  Halt if no improvement is found
18             return(selectset);
19         fi
20         selectset = selectset ∪ j_best;              Add j_best to selectset
21         AttrSet : AttrSet ∩ j_best = ∅;      Remove j_best from AttrSet
22     done
23     return(selectset).
```

Figure 6.2: The *Forward Selection* Algorithm.

6.3.2 Backward Elimination

The *Backward Elimination* algorithm (BSE) is similar to the *Forward Selection* algorithm, except that it starts by including all the attributes in the attribute selection set, and greedily removes attributes from the set. Both algorithms search the state space of attribute subsets for the smallest set of attributes that can achieve the best performance. However, the termination conditions of the *Backward Elimination* search differ slightly from those used by the *Forward Selection* algorithm. If two states result in the same performance, but one of the states represents a selection set with fewer attributes, then this state is preferred over the state with more attributes. For this reason, the *Backward Elimination* algorithm (illustrated in Figure 6.3) will continue to explore attribute selection sets where the removal of an attribute has no effect on the resulting performance.

```
1  var
2      AttrSet                                          Set of all attributes
3      N                                    Total number of members in AttrSet
4  end
5  proc backward_elimination(AttrSet, N) ≡
6      selectset = AttrSet;                 Selection Set contains all attributes
7      eval_best = evaluate(selectset);
8      for i = 1 to N do
9          j_best = ∅;
10         foreach j in AttrSet do
11             eval = evaluate(selectset : selectset ∩ j = ∅);
12             if (eval ≥ eval_best)            If eval equal to or better than ...
13                 eval_best = eval;
14                 j_best = j;
15             fi
16         done
17         if (j_best == ∅)                      Halt if no improvement is found
18             return(selectset);
19         fi
20         AttrSet : AttrSet ∩ j_best = ∅;        Remove j_best from AttrSet
21         selectset : selectset ∩ j_best = ∅;    Remove j_best from selectset
23     done
24     return(selectset).
```

Figure 6.3: The *Backward Elimination* Algorithm.

6.3.3 Simulated Annealing

The *Simulated Annealing* algorithm (Kirkpatrick, Gelatt, & Vecchi 1983) is a hill climbing search algorithm that overcomes the problem of becoming trapped in a local maxima by taking downhill steps as well as uphill steps. It differs from other hill climbing algorithms in that it selects a random move to a neighbouring state instead of determining the best move from the current state. If the new state returns a higher value for the evaluation function than the current state, then the algorithm moves to this new state. Otherwise, it moves to the new state with a probability less than one. The probability is exponentially dependent on two factors; the difference between the evaluation functions of the current and new state, ΔE; and a *temperature* parameter, T.

```
1  var
2      AttrSet                                  Set of all attributes
3      InitialTemp                              Initial temperature value
4      TemperatureDrop                          Decrement factor for T
5      Moves                                    Number of states evaluated
6                                               at each temperature gradient
7  end
8  proc simulated_annealing(AttrSet) ≡
9        selectset = AttrSet;                                  Start State
10       eval_best = evaluate(selectset);
11       T = InitialTemp;
12       while (T > 0) do
13             for m = 1 to Moves do
14                 newset = Random_Step(selectset);
15                 eval = evaluate(newset);
16                 if (SA_Accept(T, eval, eval_best))
17                    selectset = newset;
18                    eval_best = eval;
19                 fi
20             done
21             T −= TemperatureDrop;                  Reduce temperature
22       done
23       return(selectset).
```

Figure 6.4: The *Simulated Annealing* Algorithm.

```
1  proc SA_accept(T, eval, eval_best) ≡
2        if (eval > eval_best)
3          return(TRUE);
4        fi
5
6        ΔE = eval_best − eval;
7        P(ΔE) = e^(−ΔE/T);
8        if (Random[0..1] < P(ΔE))
9          return(TRUE);
10       fi
11       return(FALSE).
```

Figure 6.5: Acceptance Routine for the *Simulated Annealing* Algorithm.

The parameter T is used to control the number of downward steps taken by the search algorithm. When T has a high value, the probability of moving in a downward direction is greater than when the value of T is lower. When the algorithm starts the search, the temperature parameter T is high. At this high temperature, the search is random, and can therefore *explore* the landscape without becoming trapped by local maxima. As the value of T decreases, the probability of making downward moves reduces, and hence the search starts to *exploit* the features of the landscape. Figure 6.4 illustrates the search algorithm. It searches a fixed number of states (*Moves*) for each temperature gradient. The drop in temperature ($TemperatureDrop$) is also fixed. The algorithm terminates when the value of T falls below zero.

The acceptance criteria for downward moves is a function of the difference in evaluation functions for the two states, and is given by the following equation:

$$P(\Delta E) = e^{-\frac{\Delta E}{T}} \tag{6.1}$$

Figure 6.5 illustrates how this equation is used to determine whether a downward step should be accepted. The value of $InitialTemp$ was determined to initially accept a move to a lower state (where $\Delta E = 0.5$) with a probability of 1%. It was calculated by expressing the acceptance criteria (Equation 6.1) as a function of probability and error, i.e.

$$T = \frac{-\Delta E}{\ln(P(\Delta E))}$$

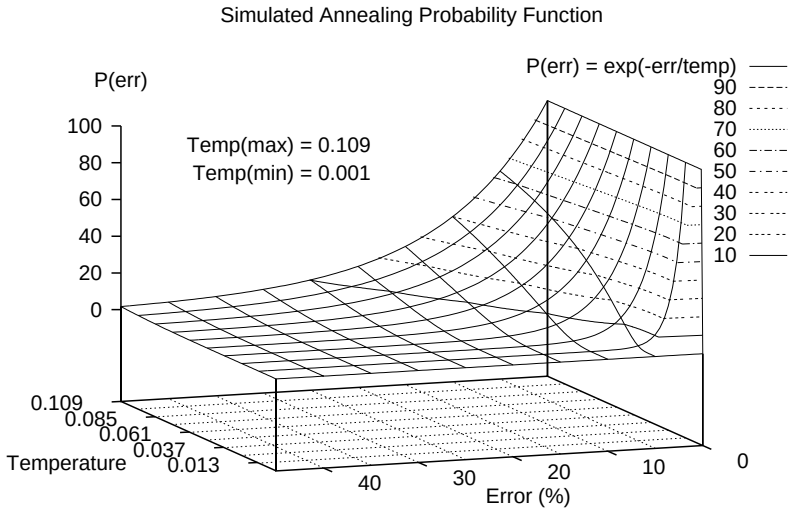


Figure 6.6: Variation in the *Simulated Annealing* probability function as the *temperature* falls.

Hence, an initial value of $T = 0.109$ can be calculated, i.e.

$$\begin{aligned} \Delta E &= 50\% = 0.5 \\ P(\Delta E) &= 1\% = 0.01 \\ T &= \frac{-0.5}{\ln(0.01)} = 0.109 \end{aligned}$$

A step size of 0.004 will reduce the temperature, T, to a lower bound (0.001) in 27 steps. At each temperature gradient, 10 iterations are performed value which results in of $10 \times 27 = 270$ steps. At the lower bound, the probability of accepting a 1% error is:

$$\begin{aligned} \Delta E &= 1\% = 0.01 \\ T &= 0.001 \\ P(\Delta E) &= e^{-\frac{0.01}{0.001}} = 0.004\% \end{aligned}$$

The variation in the probability curve is presented in Figure 6.6 as a function of the

temperature, T. The Error (%) refers to the difference between the classification of the two states (i.e. ΔE), and P(err) is the probability that the move will be taken by the search algorithm. The graph illustrates how the probability of accepting a *bad* move drops as the temperature falls.

6.3.4 Monte Carlo

```
1  var
2      AttrSet                                  Set of all attributes
3      NumIterations                 Number of states to be evaluated
4  end
5  proc monte_carlo(AttrSet) ≡
6        selectset = AttrSet;                                Start State
7        eval_best = evaluate(selectset);
8        for i = 1 to NumIterations do
9            newset = Random_State;
10           eval = evaluate(newset);
11           if (eval > eval_best)
12              selectset = newset;
13              eval_best = eval;
14           fi
15       done
16       return(selectset).
```

Figure 6.7: The *Monte Carlo* Algorithm.

The *Monte Carlo* algorithm utilises a stochastic, or random sample approach to search the state space (Skalak 1994; Liu & Setiono 1996a; Liu & Setiono 1996b). Unlike the search methods described above, this method does not traverse the search space in search of sub-optimal states, but rather evaluates a fixed number of random states. As a consequence, it is not susceptible to local maxima. It is also possible to show that as the number of states visited increases, so does the probability of finding an optimal solution (Liu & Setiono 1996a). The *Monte Carlo* search algorithm is presented in Figure 6.7. A total of 270 search states were sampled by the algorithm; this number was chosen as a similar number of states are visited by the *Simulated Annealing* algorithm described above.

6.4 Weighted Method Framework

A number of different attribute selection approaches have utilised the notion of weights to determine the significance of each of the different attributes. Many of these approaches achieve this by determining a vector of weights, where each element (i.e. weight) in the vector corresponds to one of the attributes. The weights can then be used to reduce the detrimental effects of irrelevant attributes in one of two ways. Some approaches use the weights to determine the most significant set of attributes for a data set, and then discard the remaining attributes (Kira & Rendell 1992b; Kononenko 1994). Other approaches integrate the weights within the distance metric to either augment or diminish the effect each attribute has on the distance between two instances (Aha 1992b; Salzberg 1991a). These techniques are discussed in greater detail in Chapter 3.

The relevance weights are determined by observing the evaluation of a nearest neighbour algorithm on the training data. A vector of attribute weights is generated, where each element in the vector corresponds to each of the attributes. The values of the initial weights are all equal. The leave-one out cross validation technique (Kohavi 1995) is then used to predict the class label of each of the instances in the data set. As each instance is evaluated, the weights are adjusted according to whether or not the classification is correct. This algorithm (which is presented in Figure 6.8) is similar to that used by Salzberg (1991a). The ath element of the attribute weight vector, corresponding to the ath attribute, is represented as ω_a. The update coefficient, μ, is dependent on the outcome of the classification of the instance i (where j is the nearest neighbour). If the classification is correct, then the value of the update coefficient is positive, otherwise negative.

Update Coefficient - $\Delta Coeff$

The intuition behind this algorithm is that irrelevant attributes will contribute very little overall to the classification task. The function used to update the weights is designed to reward attributes if they are responsible for making correct

```
1  var
2      i                                              New Instance
3      j                                              Nearest Instance to i
4  end
5  proc Update_Weight(i, j, classification) ≡
6                                  Determine whether a positive or negative
7                                  update coefficient should be used
8      if classification = TRUE
9        then μ = ΔCoeff;                             correct classification
10        else μ = −ΔCoeff;                           incorrect classification
11     fi
12     foreach a_i do                                 For each attribute a_i ...
13       if (δ(i_a, j_a) ≤ λ)
14         then ω_a = ω_a(1 + μ);                     Attribute values are similar
15          else ω_a = ω_a(1 − μ);                    Attribute values are different
16       fi
17     od
```

Figure 6.8: Weight Update Algorithm.

predictions, and penalise them if they are responsible for incorrect ones. Thus, the contribution made by the irrelevant attributes towards the classification task falls as the contribution made by other attributes rises. The variable μ is used to modify the different weights. Its value is either $+\Delta Coeff$ or $-\Delta Coeff$ depending on the result of the classification, where $\Delta Coeff$ is the update coefficient. The value of this coefficient determines the rate at which the weights change. If the coefficient is small, then a great many evaluations are required before there is a significant difference between low and high weights. Hence, the value of the coefficient may be predicted by determining the number of instances in the training data set[1].

Inclusion Threshold - λ

The algorithm updates weights according to whether or not the distance for each <*attribute,value*> pair is small or large. For symbolic distance metrics such as the *Overlap Metric*, this is relatively simple to determine, as the distance can return

[1]Note that the update coefficient, $\Delta Coeff$ is determined empirically in Appendix B

one of two values. However, numeric distance metrics such as the *Euclidean Metric* return a continuous valued distance measure. For this reason, the λ threshold is used to determine if the distance between two attribute values is either small or large. However, the value chosen for this threshold can strongly influence the resulting weights. If this value is too small, then all distances will exceed this threshold; likewise, if the value is too great then all the distances will lie below the threshold. In such cases, the resulting weights will tend to be equal, and thus cannot be used to differentiate between relevant and irrelevant attributes. The threshold will also be dependent on the characteristics of each data set, such as the distribution of instances within the instance space and the density of instances within clusters. For this reason, the inclusion threshold will be determined empirically for each attribute.

6.5 The Sub-space Approximation Framework

In the previous sections, various techniques were introduced that determine the most relevant attributes within a given domain. Lower dimensional instances can then be represented using this subset of attributes. The dimensionality of a domain can also be reduced by using a technique known as *Correspondence Analysis* (Greenacre 1984). This technique (described in detail in Section 3.5.1) projects the instances used to represent a domain into an alternative instance space, which can then be approximated by a lower dimensional subspace. There are several differences between this approach and the other dimensionality reduction methods discussed in this chapter:

1. Correspondence analysis identifies linear combinations of the dimensions of the original space to determine the dimensions of the new space. Low inertia dimensions are eliminated, resulting in a lower rank approximation of the original space. Thus, although the dimensionality is reduced, none of the original dimensions (i.e. attributes) are eliminated. This differs from attribute selection approaches, which select a subset of the original dimen-

sions.

2. Correspondence analysis does not make use of any class information when determining a sub-space approximation.

3. The method proposed for identifying the *basis* for a sub-space is designed to work with numerically defined spaces; i.e. the dimensions have continuous values. Therefore, this method is not suitable for symbolic data.

4. It is difficult to determine the number of dimensions that can best approximate the space. The ideal approximation is one that maximises the classification accuracy of a learning algorithm, but minimises the number of dimensions.

The sub-space approximation framework consists of two main routines: one that generates a mapping function between the original space and the transformed and approximated sub-space (Figure 6.9); and a routine that uses the mapping function to project instances from the original space into the new space (Figure 6.10). Data sets are presented to these routines as matrices, where each row of the matrix corresponds to an instance, and each column corresponds to one of the attributes of the data set. The mapping function consists of the basis $R_{(K)}$ of the approximated sub-space, and the centroid, y. Instances, represented as vectors in the matrix Y, are projected into the new space by translating them with respect

```
1  proc generate_mapping(Y, rank) ≡
2        y = get_centroid_vector(Y);
3        X = translate_data(Y, y);
4
5        [L, D, R] = SVD(X);
6
7        R_(K) = low_rank(D, R, rank);
8        map[basis] = R_(K);
9        map[centroid] = y;
10       return(map).
```

Figure 6.9: Generating a sub-space mapping.

```
1  proc apply_mapping(Y, map) ≡
2        X = translate_data(Y, map[centroid]);
3        F = matrix_multiply(X, map[basis])
4        return(F).
```

Figure 6.10: Applying the sub-space mapping to a new data set.

to the centroid vector, and then multiplied with the basis (illustrated in Figure 6.10).

```
1  proc generate_class_projected_mapping(Y, rank) ≡
2        y = get_centroid_vector(Y);
3        X = translate_data(Y, y);
4        P = get_class_prototypes(X);
5
6        [L, D, R] = SVD(P);
7
8        R_(K) = low_rank(D, R, rank);
9        map[basis] = R_(K);
10       map[centroid] = y;
11       return(map).
```

Figure 6.11: Generating a class projected sub-space mapping

The sub-space mapping approach described above is unsupervised, i.e. no class information is used when the mapping function is determined. A second approach was therefore devised, which exploits the class labels when determining the mapping function. The *generate_class_projected_mapping* routine (Figure 6.11) generates a single *prototype* point for each class, by finding the centroid of all the instances belonging to that class. Once all the prototype points have been found, they are used to generate the new basis, R.

Chapter 7

An Evaluation of the Dimensionality Reduction Approaches

Chapter Outline

In the previous chapter, a number of different dimensionality reduction approaches were discussed. Each approach reduced the number of attributes by searching through a state space of attribute combinations, and then evaluating each combination with a Nearest Neighbour algorithm. This chapter describes in detail an evaluation of each approach, when applied to different single-attribute data sets.

7.1 Introduction

This chapter reviews the different attribute selection methods discussed in chapter 6. It starts by reviewing the data sets used to compare the different methods. The performance of the basic Nearest Neighbour is determined, so that the results can be used as a benchmark with which to compare the different selection methods. Four different approaches to attribute selection and dimensionality reduction are then evaluated.

The *filter* approach proposed by Cardie (1993) is evaluated in Section 7.3. This approach employs the rule induction algorithm, C4.5 (Quinlan 1993), to select attributes. A Nearest Neighbour algorithm is then tested, either using the attributes that appear within the C4.5 decision tree, or using all the attributes available, in an attempt to determine which of the data sets contain irrelevant or redundant attributes. The classification accuracy of C4.5 is also compared with the basic Nearest Neighbour algorithm and the filtered approach.

The *wrapper* approach is tested with four different search algorithms (Section 7.4). Three of these search algorithms have previously been investigated with various other learning algorithms as part of other studies (see Chapter 3). The fourth search algorithm, *Simulated Annealing*, combines the heuristic properties of the two greedy searches, *Forward Selection* and *Backward Elimination*, with the stochastic properties of the *Monte Carlo* search. For this reason, it has been investigated here to determine whether or not it can improve on the performance on the other search methods.

Section 7.5 investigates a method for determining attribute weights, and compares a variety of different ways in which the weights can be used. The weights are generated by monitoring the performance of the learning algorithm on the training data. The algorithm used to generate the weights extends that proposed by Salzberg (1991a) to utilise distance metrics that return continuous values (such as the VDM or *Euclidean* metrics (Section 2.2.3)). The weights can be used to reject low-weighted values or attributes; they can be employed directly by the distance metric, or a combination of both methods used. These alternative strategies

are compared to determine the best approach for different domains.

The final approach investigates the whether subspace mapping techniques can be successfully used to reduce the dimensionality of the data set, and contrasts the differences between dimensionality reduction and attribute selection.

7.2 Data Sets

The approaches described in the previous chapter were evaluated on a number of different 'real-world' and 'artificial' data sets; all of which are available from the UCI Machine Learning Database Repository (Merz & Murphy 1996). The real-world data sets are summarised in Table 7.1, and are divided into two categories: *symbolic* and *numeric*. *Symbolic* data sets contain only symbolic or binary valued data, whereas *numeric* data sets contain either numeric or ordered values. Only data sets with homogeneous data types are used to simplify comparisons between different selection methods and different data types.

Several data sets contained a unique identification attribute. These artificially created attributes were removed as their values may be correlated with the class label, and hence effect the classification accuracy. For example, the *glass* data set contains an ordered numeric identifier, which is highly correlated with the class label (the correlation coefficient is 0.958 using Spearman's Rank Correlation). If used, for example, the result for a leave-one-out cross validated Euclidean test *NN* rises from 69.16% to 90.65%.

Instances containing missing values were also removed, to eliminate the requirement for an additional learning component to resolve the unknown values. Two of the attributes of the *primary-tumor* data set contained large numbers of missing values and hence were removed.

A number of artificial symbolic data sets have also been utilised. These data sets have well defined properties that allow specific characteristics (such as irrelevance or attribute dependence) to be investigated. The five symbolic and three numeric data sets used here are described below.

Data Set		Attrs	Class	Instances	Instances/Class (%) - First ten listed only
breast-cancer	Breast Cancer Data[a]	9	2	277[b]	70.8 29.2
lung-cancer	Lung Cancer Data	56	3	27[c]	29.6 37.0 33.4
lymph	Lymphography Domain[a]	18	4	148	54.7 41.2 2.7 1.4
primary-tumor	Primary Tumor Domain[a]	15[d]	22	336[d]	24.4 11.6 8.6 8.3 7.1 7.1 6.0 4.8 4.2 3.9
promoters	E.Coli Promoter Gene Sequences (DNA)	57[e]	2	106	50.0 50.0
tic	Tic-Tac-Toe Endgame Database	9	2	958	65.3 34.7
votes	1984 United States Congressional Voting Records	16	2	435	61.4 38.6
zoo	Zoo Database	16[e]	7	101	40.6 19.8 12.9 9.9 7.9 4.9 4.0
bupa	BUPA liver disorders	6	2	345	42.0 58.0
ionosphere	JHU Ionosphere Database	34	2	351	64.0 36.0
pima	Pima Indians Diabetes Database	8	2	768	34.9 65.1
sonar	Sonar, Mines vs. Rocks	60	2	208	53.4 46.6
wisconsin	Wisconsin Breast Cancer Database	9[e]	2	683[f]	65.0 35.0
wdbc	Wisconsin Diagnostic Breast Cancer	30[e]	2	569	37.3 62.7
wpbc	Wisconsin Prognostic Breast Cancer	33[e]	2	194	76.3 23.7
ecoli	Protein Localization Sites (E.Coli)	7	8	336	42.6 22.9 15.5 10.4 6.0 1.5 0.6 0.6
glass	Glass Identification Database	9[e]	6	214	32.7 35.5 7.9 6.1 4.2 13.6
iris	Iris Plants Database	4	3	150	33.3 33.3 33.3
wine	Wine Recognition Data	13	3	178	33.1 39.9 27.0
yeast	Protein Localization Sites (Yeast)	8	10	1484	31.2 28.9 16.4 11.0 3.4 3.0 2.5 2.0 1.3 0.3

[a]Thanks go to M. Zwitter and M. Soklic of the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia for providing this data.
[b]There are 286 instances in the original database, but 9 were removed as they contained missing values.
[c]There are 32 instances in the original database, but 5 were removed as they contained missing values.
[d]There are 17 attributes and 339 instances in the original database, but 2 attributes and 3 instances were removed as they contained missing values.
[e]The original data set has an additional attribute which contains a unique identifier for each instance. This attribute has been removed prior to use.
[f]There are 699 instances in the original database, but 16 had missing values and hence were removed.

Table 7.1: UCI Numeric Data Sets used in this study.

Artificial Symbolic Data Sets

The LED display problem consists of two similar data sets: the *led* and *led+17* data sets. The *led* data set contains seven binary valued attributes corresponding to the different segments within an LED seven segment numeric display. There are ten classes, each corresponding to a displayed digit. The 24-attribute *led+17* data set is a variant of the *led* data set, where the digits appear on a 24-segment display (Figure 7.1). The additional 17 segments are not used to display the digits, but can be switched on or off with a probability of 0.5 (Breiman, Freidman, Olshen, & Stone 1984; Aha, Kibler, & Albert 1991). Both data sets comprise of 200 randomly generated instances with 10% noise (i.e. each attribute value had a 10% chance of being inverted). This noise simulates the behaviour of a noisy display.

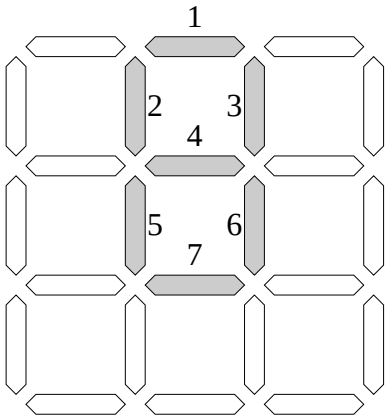


Figure 7.1: The 24 segment led display used to generate the *led+17* data set. Note that only segments 1-7 are used to display the digits.

The *Monk's* problems represent three binary classification problems (Thrun et al., 1991). The instances are described by six symbolic attributes, and their class is determined by one of three rules:

$$\textit{Monks-1} : (a_1 = a_2) \vee (a_5 = 1)$$

$$\textit{Monks-2} : (a_n = 1) \text{ for } \textbf{exactly two} \text{ choices of } n, \text{ where } 1 \leq n \leq 6$$

$$\textit{Monks-3} : (a_5 = 3 \wedge a_4 = 1) \vee (a_5 \neq 4 \wedge a_2 \neq 3)$$

The *Monks-1* and *Monks-3* data sets are in standard disjunctive normal form, and

consist of three relevant and three irrelevant attributes. The *Monks-3* data set has 5% class noise (i.e. instances may be misclassified) in the training set. The *Monks-2* problem is similar to parity problems, and thus all the attributes are weakly relevant.

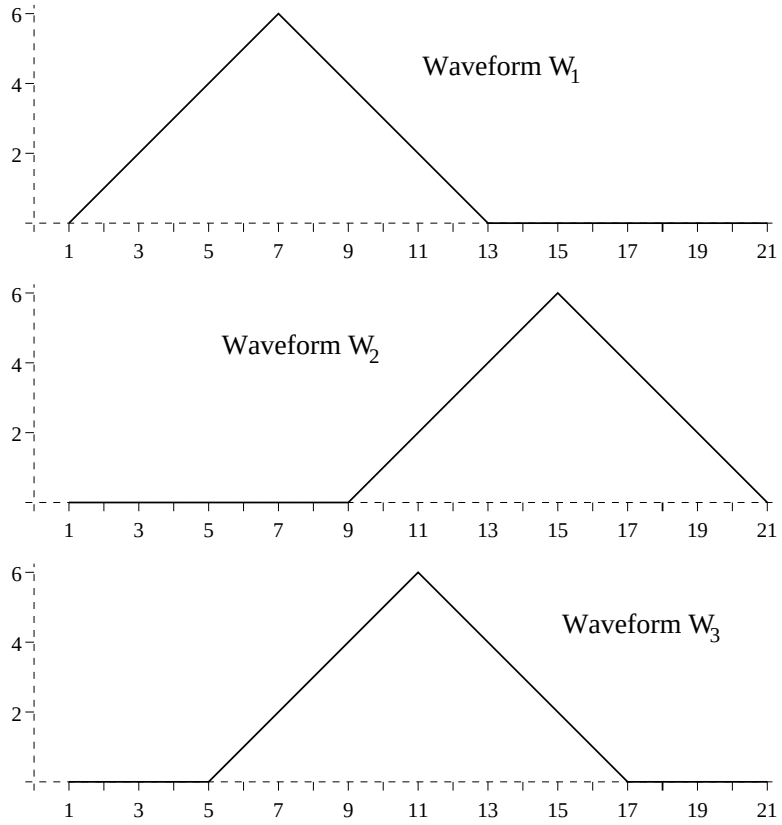
Artificial Numeric Data Sets

The *balance* scale weight problem (Seigler 1976) was generated to model one of a number of psychological experiments that were used to investigate cognitive development. It consists of four attributes: *left-weight*, *left-distance*, *right-weight* and *right-distance*. Each attribute is represented by a discrete value from the ordered set { 1 2 3 4 5 }. The class is determined by combining the values as follows:

$$c_1 \rightarrow \textit{left-weight} \times \textit{left-distance} > \textit{right-weight} \times \textit{right-distance}$$
$$c_2 \rightarrow \textit{left-weight} \times \textit{left-distance} < \textit{right-weight} \times \textit{right-distance}$$
$$c_3 \rightarrow \text{otherwise}$$

This data set contains no irrelevant or redundant attributes.

The *waveform* recognition problem (Breiman, Freidman, Olshen, & Stone 1984; Aha 1990) is based on combinations of the different waveforms presented in Figure 7.2. The resulting waveforms are grouped into three classes, depending on which pair of waveforms were combined, and are represented in the *waveform-21* data set by 21 discrete samples (corresponding to the 21 attributes). The amplitude of each sample is noisy; this noise is generated by generating a normally distributed random co-efficient ($\mu = 0$, $\sigma^2 = 1$). The *waveform-40* data set is a variant of the *waveform-21* data set with an additional 19 normally distributed random valued irrelevant attributes



$$\text{Class 1:} \quad x_a = uW_1(a) + (1-u)W_2(a) + r_a$$
$$\text{Class 2:} \quad x_a = uW_1(a) + (1-u)W_3(a) + r_a$$
$$\text{Class 3:} \quad x_a = uW_2(a) + (1-u)W_3(a) + r_a$$

Figure 7.2: The waveforms used to generate the *waveform-21* data set. Each instance belongs to one of three classes, and the value of x_a for each attribute $a \in \{1..21\}$ is defined above, where u is a uniformly distributed random number between 0..1, and r_a is a normally distributed number ($\mu = 0$, $\sigma^2 = 1$).

7.3 Evaluation of the Filter Method

The basic Nearest Neighbour algorithm (NN) was evaluated to determine a baseline accuracy for each of the data sets. The dimensionality approaches can then be compared with these results to determine whether an improvement in accuracy can be achieved. The *Overlap* and *Euclidean* distance metrics were used for the symbolic and numeric data sets (respectively)[1]. The numeric data was normalised by the NN algorithm prior to classification.

The results were obtained by performing an n-fold cross validation on each of the data sets, where the value of $n = 10$ for the symbolic data sets and $n = 20$ for the numerical data sets. Each classification is determined by identifying the single nearest neighbour to the query instance. Although the performance of the k-NN approach is generally superior to that of the 1-NN approach (Section 2.2.2), the latter avoids the necessity of determining the appropriate size of neighbourhood for each data set. This is especially important as the optimal neighbourhood size may vary depending on the number of irrelevant attributes.

The C4.5 decision tree learning algorithm (Quinlan 1993) was also tested on the data sets to provide a comparison with an alternative learning approach. This algorithm uses a *divide and conquer* approach to inducing decision trees, by recursively determining the attribute that best splits the data into homogeneously classified clusters of instances. As a consequence, many resulting decision trees utilise a subset of the available attributes, which reduces the impact of irrelevant attributes on the target concept[2]. This behaviour has been exploited as an attribute selection mechanism in its own right, with the resulting attributes being tested with other learning algorithms (Kubat, Flotzinger, & Pfurtscheller 1993; Cardie 1993).

[1]The distance metric used by the algorithm is denoted by a subscript; i.e. the NN_{OM} algorithm is the basic Nearest Neighbour algorithm using the *Overlap Metric*, whereas the NN_{EM} uses the *Euclidean Metric*.

[2]The selection metrics utilised by decision tree learning algorithms will not necessarily select the optimal set of attributes (Almuallim & Dietterich 1991).

Symbolic Data

The results of the evaluation of NN_{OM} and C4.5 with symbolic data are presented in Table 7.2. The classification accuracy achieved by C4.5 was statistically higher than that achieved by NN_{OM} for five of the eight symbolic data sets. The accuracy of C4.5 was lower than NN_{OM} for only two data sets, *lymph* and *zoo*; however, only the *zoo* result was significant at the 5% level. The numbers of attributes that appear in the pruned C4.5 decision trees were lower than the total number of attributes available in the data sets for seven of the eight symbolic data sets. Although C4.5 utilises all the attributes present in the *tic* data set, it still achieves a higher classification accuracy than NN.

	Data Sets	NN_{OM}	C4.5	FNN_{OM}
Real World	breast-cancer	70.73 (9)	↑ **75.06** (2.8)	↑ 71.84
	lung-cancer	40.00 (56)	↑ **80.01** (2.9)	↑ **71.67**
	lymph	81.24 (18)	↓ 79.81 (8.2)	↓ **73.81**
	primary-tumor	32.05 (15)	↑ **40.68** (13.1)	↓ 32.04
	promoters	77.00 (57)	↑ 79.17 (4.7)	↑ 82.82
	tic	80.90 (9)	↑ **86.21** (9.0)	—
	votes	92.44 (16)	↑ **95.20** (4.8)	↑ 94.06
	zoo	96.09 (16)	↓ **92.00** (7.2)	↓ **87.18**
Artificial	led	69.50 (7)	↑ 71.50 (7)	—
	led+17	34.50 (24)	↑ **62.00** (15.3)	↑ **46.00**
	monks-1	78.70 (6)	↓ 75.70 (5)	↑ 86.11
	monks-2	73.84 (6)	↓ 65.00 (6)	—
	monks-3	82.87 (6)	↑ 97.20 (2)	↑ 88.89

Table 7.2: A Comparison of the basic Nearest Neighbour algorithm using the *Overlap* distance metrics with the C4.5 rule induction algorithm on symbolic data. Values in bold indicate a significant difference in classification accuracy with respect to the NN_{OM} results (at the 5% confidence level). The vertical arrows indicate whether or not an increase or decrease in accuracy was achieved. The average number of attributes that appear in the pruned C4.5 decision trees are given in parentheses.

The two artificial *led* domains illustrate that the performance of both learning algorithms deteriorate in the presence of irrelevant attributes, but at different rates. Both C4.5 and NN_{OM} achieve a similar classification accuracy for the basic *led* data set, and C4.5 utilises all the seven attributes in the resulting decision tree. However, when irrelevant attributes are present (*led+17*), both algorithms

experience a drop in classification accuracy. The decision tree generated by C4.5 included all the relevant attributes and nine of the irrelevant attributes. The resulting accuracy fell by 7.5% (when compared to the corresponding result for the *led* data set).

The performance of NN_{OM} is superior to that of C4.5 for the *Monks-1* and *Monks-2* data sets[3]. The inferior performance of NN_{OM} on the *Monks-3* data set was due to the inclusion of noise within the training set. If a k-NN variant of the NN_{OM} is used, then the degradation in accuracy due to noise can be reduced (e.g. 3-NN_{OM} increases the classification accuracy to 85.88%). The accuracy of NN_{OM} rises to 86.57% if the noise is removed (i.e. the noisy class labels are corrected); however the decision tree generated by C4.5 is unaffected. These *Monks-3* results confirm that the 1-NN algorithm is susceptible to the presence of noise.

C4.5 utilises all the attributes available in the *Monks-2* data set, but selects subsets of attributes for the *Monks-1* and *Monks-3* data sets. However neither subsets are optimal: two irrelevant attributes are included in the *Monks-1* subset, whereas one of the relevant attributes is omitted in the *Monks-3* subset.

The final column of Table 7.2 lists the results obtained when the NN_{OM} algorithm is used with the attribute subsets that appear in the pruned C4.5 decision trees. This approach is similar to that used by Cardie (1993), where the C4.5 decision tree algorithm is used as an attribute filter (hence $F\mathrm{NN}_{OM}$). The results for the *tic*, *led* and *Monks-2* data sets are omitted from this list, as all the attributes in the data sets appeared in the pruned decision trees.

The filtered NN algorithm $F\mathrm{NN}_{OM}$ achieved higher classification accuracies than NN_{OM} for four of the seven symbolic data sets, although only the result for the *lung-cancer* data set was significant. For two of the data sets (*lymph* and *zoo*), the accuracy of the NN algorithm fell significantly when the corresponding attribute subsets were used. However, these were the only symbolic data sets for which the accuracy of C4.5 was less than that of NN_{OM}, and hence the attribute subsets

[3]The performance of C4.5 with the *Monks* data sets can be improved if the attribute values are grouped (Quinlan 1993, pages 63-69).

may not have been appropriate to the concepts represented by the data sets.

The filtered *led+17* data set contains eight fewer irrelevant attributes than the original, and there is a corresponding increase in accuracy for FNN, although this is significantly lower (p=0.001) than the accuracy achieved by C4.5. This suggests that NN is more susceptible (than C4.5) to the presence of irrelevant attributes in the data set, even when the irrelevant attributes appear in the in the induced decision trees. The rejection of some of the irrelevant attributes from the *Monks-1* and *Monks-3* data sets also results in an increase in classification for the filtered NN algorithm. The accuracy of $F\text{NN}_{OM}$ is still inferior to that achieved by C4.5 on the *Monks-3* data set, although this accuracy rises from 88.89% to 97.22% if the noisy class labels are corrected.

Numeric Data

C4.5 achieved superior classification results for only seven of the twelve numerical data sets (Table 7.3), of which only two results were significant (for the *ionosphere* and *yeast* data sets). Of these seven data sets, the decision trees utilised a subset of attributes for the *ionosphere*, *wdbc*, *ecoli* and *glass* data sets. In contrast, NN_{EM} achieved significantly better classification accuracies for two of the five remaining data sets.

The *iris* data set is known to contain two relevant attributes and two irrelevant attributes (Michie, Spiegelhalter, & Taylor 1994). Although the C4.5 decision trees only utilised the two relevant attributes, the classification accuracy of C4.5 was lower (though not significant) than that achieved by NN_{EM}. If the decision trees are used to eliminate the irrelevant attributes from the data set (i.e. using $F\text{NN}_{EM}$) then the accuracy increases significantly to 98.13%.

C4.5 successfully utilised all four attributes in the *balance* data set, and generally selected only the most relevant attributes in the *waveform-21* data set. The NN_{EM} classification accuracy for the *waveform-40* was lower than that for the *waveform-21*; this may possibly be due to the additional attributes. This hypothesis is supported when a histogram of the attributes selected by C4.5 is examined

	Data Sets	NN_{EM}	C4.5	FNN_{EM}
Real World	bupa	61.98 (6)	↑ 65.17 (6.0)	—
	ionosphere	87.17 (34)	↑ **90.81** (9.6)	↑ **92.60**
	pima	70.99 (8)	↑ 72.91 (8.0)	—
	sonar	85.96 (60)	↓ **75.45** (13.6)	↓ 82.32
	wisconsin	95.90 (9)	↓ 95.16 (5.7)	95.90
	wdbc	95.40 (30)	↑ 95.60 (7.8)	↑ 95.43
	wpbc	69.06 (33)	↓ 68.22 (13.4)	↑ 70.50
	ecoli	80.59 (7)	↑ 84.19 (5.3)	↓ 79.71
	glass	68.09 (9)	↑ 71.76 (8.8)	68.09
	iris	96.16 (4)	↓ 94.82 (2.0)	↑ **98.13**
	wine	94.86 (13)	↓ **90.91** (3.7)	↑ 96.04
	yeast	52.56 (8)	↑ **54.80** (8.0)	—
Artificial	balance	78.10 (4)	↓ 76.98 (4.0)	—
	waveform-21	73.33 (21)	↓ *69.99* (16.1)	↓ 73.00
	waveform-40	68.33 (40)	↑ 69.33 (17.6)	↑ **74.33**

Table 7.3: A Comparison of the basic Nearest Neighbour algorithm using the *Euclidean* distance metric with the C4.5 rule induction algorithm on numeric data. Values in bold or italic indicate a significant difference in classification accuracy with respect to the NN_{EM} results (at the 5% or 10% confidence level respectively). The vertical arrows indicate whether or not an increase or decrease in accuracy was achieved. The average number of attributes that appear in the pruned C4.5 decision trees are given in parentheses.

(Figure 7.3). The first twenty-one bars refer to the attributes generated to represent the waveforms, and the remaining nineteen bars refer to the attributes with random values. The relevant attributes were selected in most cases, although some attributes were frequently rejected (such as attributes 3, 9 and 18). However, the frequency of the selection of irrelevant attributes much lower than that of relevant attributes.

FNN_{EM} succeeded in improving the classification accuracy with respect to both NN_{EM} and C4.5 for five of the real world data sets. The results for *ionosphere* and *iris* were significant at the 5% level. Although there was a drop in classification accuracy for two of the data sets, the results were not significant. The rejection of attributes had no effect on the results for the *wisconsin* and *glass* data sets.

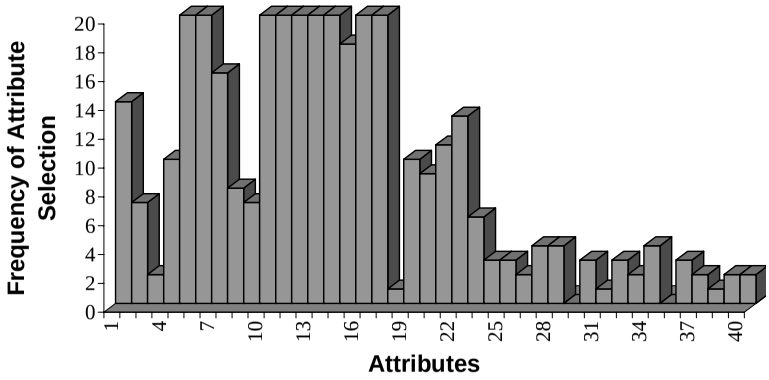


Figure 7.3: Histogram representing the frequency of attributes appearing in the C4.5 decision tree for the different n-fold evaluations (with the *waveform-40* data set).

Discussion

In general, the C4.5 classification algorithm appears to be more accurate than the Nearest Neighbour algorithm when presented with symbolic data sets. However, the accuracy of both algorithms are comparable when presented with numeric data sets. For the majority of data sets, C4.5 utilised a subset of the available attributes. This suggests that not all the attributes are required to represent the target hypothesis, and that the rejected attributes may be either irrelevant or redundant. However, the decision trees induced by C4.5 for the artificial data sets suggest that the attributes selected are not optimal and that relevant attributes may be rejected in favour of irrelevant ones. The results for the filtered Nearest Neighbour algorithm fail to demonstrate any overall increase in the accuracy when used with symbolic data sets, although there a slight overall increase when applied to numeric data sets.

The filtered approach was also used by Cardie (1993) to improve the case-based learning (CBL) component of a natural language processing system. The CBL component employed a k-NN approach (Section 2.2.2) to predicting the necessary lexical or semantic characteristics of unknown terms encountered when parsing text. The classification accuracy of the CBL algorithm was compared with that

of the C4.5 algorithm when predicting one of three different characteristics[4]: the *part of speech*, *general semantic class* and *specific semantic class* characteristics. It was believed that each of these characteristics were dependent on only a subset of the attributes used to describe the unknown terms. The decision trees induced by C4.5 utilised different attribute subsets for each characteristic, confirming this hypothesis. The classification accuracy of the resulting trees was significantly greater than that achieved by CBL when $k = 1$, but was significantly lower when $k = 10$. A hybrid approach was then investigated, which determined the attribute subsets selected by C4.5 for each of the characteristics, and discarded the remaining attributes prior to classifying new instances with the CBL algorithm. The classification accuracy of this hybrid system was significantly better than either C4.5 or the CBL approach (when $k = 10$).

This study demonstrated that the filtered approach could be used to eliminate irrelevant attributes and thus improve the classification of a k-NN algorithm for a single symbolic domain. The results presented above (Section 7.3) extend this study, and demonstrate that for some domains, C4.5 fails to identify the relevant attributes, and may retain irrelevant attributes instead. This results in a deterioration in classification accuracy.

7.4 Evaluation of the Wrapper Method

The wrapper method (Section 6.3) utilises a search algorithm to explore a space of different attribute subsets. This approach has been evaluated using the four search algorithms described above: *Forward Selection (FSS)*, *Simulated Annealing (SA)*, *Monte Carlo (MC)* and *Backward Elimination (BSE)*.

The results for the different wrapper methods were obtained by performing an n-fold cross validation on each of the data sets, with $n = 10$ for the symbolic data sets and $n = 20$ for the numerical data sets. During the training stage, the search algorithm is used to determine the best attribute subset for the current training

[4]The three characteristics represent three different learning tasks.

data. The best attribute subset is that which achieves the best leave-one-out cross validation result on the training set. The *Overlap* and *Euclidean* distance metrics were used to determine the nearest neighbours for instances from the symbolic and numeric data sets (respectively). The numeric data was normalised by the NN algorithm prior to classification.

Table 7.4 lists the delta accuracies of each of the search methods with respect to the NN_{OM} and NN_{EM} accuracies (listed in Tables 7.2 and 7.3 respectively). Positive results indicate an increase in accuracy due to the selection of attributes, whereas negative results denote a decrease in accuracy. The numbers to the right of the names of the data sets (first column) represent the total number of attributes in the data sets, whereas the numbers in parenthesis represent the *average* number attributes selected in each fold.

Symbolic Data

Each of the search methods succeeded in improving the accuracy for all but one of the symbolic data sets (*zoo*) whilst reducing the average number of attributes used. Of these seven results, four were significant, although these were achieved either by the *Forward Selection* approach (*breast-cancer* & *votes*) or the *Backward Elimination* approach (*lung-cancer* & *tic*). The *Forward Selection* approach had a significant affect on the classification accuracy for more data sets than the other search techniques at the 5% level (two results were significantly higher than the NN_{OM} result, and three others were significantly lower). The *Monte Carlo* method achieved an increase in accuracy for four of the eight symbolic data sets, but of these, only two (*tic* & *votes*) were significant at the 5% level. In contrast, the *Simulated Annealing* method failed to significantly increase the classification accuracy for any of the data sets, but significantly reduced the accuracy for the *breast-cancer* and *zoo* data sets at the 5% level.

Significant increases in accuracy were achieved for the *breast-cancer*, *lung-cancer*, *tic* and *votes* data sets only. This suggests that these data sets may contain irrelevant and possibly redundant attributes, as the elimination of some attributes

results in a higher classification accuracy. The filter approach ($F\text{NN}_{OM}$) also succeeded in reducing the number of attributes required for learning accurate concept hypotheses for three of the four data sets. In each case, the resulting classification accuracy was greater than that achieved by NN_{OM}, although only the results for the *lung-cancer* data set were significant at the 5% level.

In contrast, each of the four search methods resulted in a degradation in classification accuracy for the *zoo* data set and only the *Backward Elimination* search method succeeded in generating a slight, but not significant improvement in classification accuracy for the *lymph* data set; this search method also selected the largest of the four attribute subsets (with an average of 13.2 attributes). The filter approach ($F\text{NN}_{OM}$) also failed to increase the classification accuracy for these two data sets.

Of the four search methods investigated, the *Forward Selection* search consistently selected the smallest attribute subset, with an average of 3.6 attributes over all the real world symbolic data sets. In comparison, the *Backward Elimination* search selected the largest subsets (average 15.4 attributes) whereas the stochastic searches selected slightly smaller subsets on average (10.5 attributes for *SA* and 12.1 for *MC*). The bar chart in Figure 7.4 illustrates the average proportional size of the attribute subsets for each of the real world symbolic data sets.

The artificial symbolic data sets confirm the hypothesis that the wrapper method can be used to eliminate irrelevant attributes. Of the four search methods tested, the *Simulated Annealing* search was the only one that rejected any of the attributes in the *led* data set; this resulted in a reduction in classification accuracy. However, when applied to the *led+17* data set, all four methods attempted to reject some of the irrelevant attributes and hence increased the classification accuracy. The *Forward Selection* search found the smallest attribute subsets (there were an average of 7.3 attributes in each subset), although one of the relevant attributes was rejected in favour of an irrelevant attribute for most of the folds. However, it succeeded in raising the classification accuracy of NN_{OM} from 34.5% to 61.0%, which was only 8.5% short of the accuracy achieved when all the irrelevant attributes were rejected (i.e. the *led* data set). The *Backward Elimination* search

	Data Sets		Δ(FSS)	Δ(SA)	Δ(MC)	Δ(BSE)
Real World	breast-cancer	9	**5.42** (2.5)	**-6.11** (3.7)	2.48 (3.0)	**-4.35** (6.9)
	lung-cancer	56	*20.00* (3.6)	0.00 (16.4)	-10.00 (24.9)	**23.34** (16.5)
	lymph	18	**-8.90** (5.3)	-1.48 (9.5)	-0.86 (9.5)	0.52 (13.2)
	primary-tumor	15	*-4.14* (3.5)	0.65 (7.9)	1.23 (8.9)	-2.07 (10.3)
	promoters	57	6.82 (4.5)	-4.64 (26.0)	-2.64 (28.6)	0.09 (50.2)
	tic	9	**-15.58** (1.0)	-0.03 (6.3)	**3.24** (7.3)	**5.22** (7.0)
	votes	16	**2.07** (2.8)	*1.61* (5.6)	**1.83** (5.6)	*0.91* (10.0)
	zoo	16	**-5.91** (5.6)	**-6.82** (8.2)	*-3.00* (8.9)	-1.09 (9.1)
Artificial	led	7	0.00 (7.0)	-2.00 (6.9)	0.00 (7.0)	0.00 (7.0)
	led+17	24	**26.50** (7.3)	**19.50** (10.1)	**16.50** (12.0)	**17.50** (14.7)
	monks-1	6	-12.03 (4)	18.52 (3)	18.52 (3)	18.52 (3)
	monks-2	6	-6.71 (1)	-6.71 (2)	-13.42 (4)	0.00 (6)
	monks-3	6	10.19 (3)	6.94 (4)	10.19 (3)	6.94 (4)
Real World	bupa	6	-3.53 (1.5)	-0.08 (4.4)	-1.60 (4.5)	-0.13 (4.6)
	ionosphere	34	1.18 (5.2)	**3.66** (13.5)	*3.46* (14.7)	*2.02* (21.9)
	pima	8	**-6.14** (3.2)	**-4.20** (4.7)	**-3.04** (4.3)	**-2.35** (7.5)
	sonar	60	**-8.50** (8.3)	-1.86 (25.7)	-2.27 (28.7)	-1.86 (41.5)
	wisconsin	9	-0.58 (5.0)	-0.58 (5.3)	*-0.87* (5.5)	**-1.31** (6.5)
	wdbc	30	-0.36 (5.1)	0.54 (14.6)	0.71 (14.5)	1.06 (19.6)
	wpbc	33	1.06 (1.2)	-2.78 (14.9)	2.11 (15.2)	-2.61 (26.1)
	ecoli	7	**-3.05** (6.3)	**-3.94** (5.9)	**-4.23** (6.4)	**-4.23** (6.4)
	glass	9	2.32 (5.4)	3.36 (5.5)	2.91 (5.4)	*4.18* (5.7)
	iris	4	**1.97** (2.0)	**1.97** (2.1)	**1.97** (2.0)	-0.18 (2.3)
	wine	13	0.07 (3.9)	-0.49 (6.8)	-0.07 (7.7)	-0.56 (8.8)
	yeast	8	**-1.75** (7.5)	**-1.35** (7.4)	*-0.41* (7.7)	**-0.88** (7.7)
Artificial	balance	4	**-15.68** (1.7)	0.00 (4.0)	0.00 (4.0)	0.00 (4.0)
	waveform-21	21	**-7.33** (5.8)	0.00 (11.4)	*-3.33* (12.2)	-0.33 (18.0)
	waveform-40	40	0.33 (6.4)	1.33 (20.8)	-1.33 (20.4)	0.00 (32.8)

Table 7.4: Accuracies of different wrapper methods for the UCI data sets. These values indicate the difference in accuracy between *NN* and the different wrappers. Those values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The average number of selected attributes appear in parentheses.

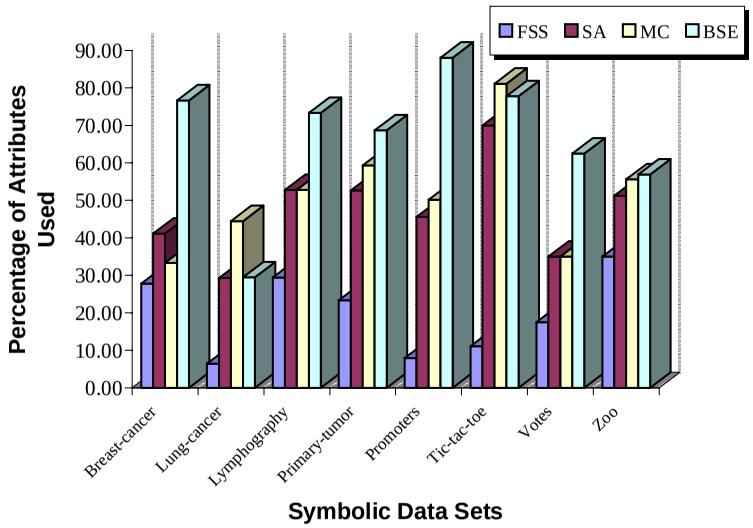


Figure 7.4: Bar Chart illustrating the average proportional size of the attribute subsets selected by the different wrapper search methods.

included all of the relevant attributes in the subsets, but the overall average size of the subsets was 14.7 attributes. The importance of selecting relevant attributes over irrelevant ones was demonstrated by the *Monte Carlo* search; although fewer attributes were selected for each subset than by the *BSE* search (an average of 12.0 attributes), only six of the seven relevant attributes were selected for most of the folds. As a consequence, the resulting accuracy of the *MC* search was 1.0% lower than that achieved when using the *BSE* approach on the *led+17* data set.

The *Forward Selection* approach was the only search method not to select the three relevant attributes for the *Monks-1* data set. The concept hypothesis included a clause that was represented a relationship between two attributes ($a_1 = a_2$). Thus, it was impossible to identify one of these relevant attributes without having selected the other, as no increase in classification accuracy would be observed until both attributes have been selected.

The *Backward Elimination* search was the only one to successfully identify the optimal subset for the *Monks-2* data set (i.e. all six attributes). The *Forward Selection* search failed to to identify a single optimal attribute, as when evaluated, all the single attribute subsets were equal. However, the inclusion of a second

attribute resulted in a drop in classification accuracy, and hence the resulting attribute subset contained only one attribute.

All four search methods succeeded in identifying at least two of the three relevant attributes for the *Monks-3* data set, but included one additional irrelevant attribute. The *FSS* & and *MC* methods both selected the same three attributes; whereas the other two search methods selected all three relevant attributes and the same additional irrelevant attribute. Surprisingly, the resulting classification accuracies (when evaluated on the test data) were higher for the smaller attribute subsets, even though one relevant attribute was missing. This may be due to the noise in the training data; if the noise is removed then all four search methods succeed in identifying the three relevant attributes.

Numeric Data

All four search methods failed to maintain or increase the classification accuracy after selecting attribute subsets for six of the twelve data sets. The results obtained for the *pima*, *ecoli* and *yeast* data sets were all significant lower than for NN_{EM}, whereas only a single search method yielded a significant drop in accuracy with the *sonar* (*FSS*) and *wisconsin* (*BSE*) data sets. None of the results obtained for the *bupa* data set were significant.

Mixed results were obtained for the *wdbc*, *wpbc* and *wine* data sets, although none were significant. The *Forward Selection* method selected an average of 5.1 attributes for the *wdbc* data set, resulting in a slight decrease in accuracy. However, the other search methods selected a greater number of attributes which all yielded improvements in the classification accuracy. In contrast, selecting larger attribute subsets for the *wine* data set has a detrimental effect on the classification accuracy.

There was an increase in classification accuracy with the *ionosphere* and *glass* data sets for all four search methods, although not all the results were significant. The filter approach also succeeded in selecting attribute subsets that significantly increased the accuracy of the Nearest Neighbour algorithm for the *ionosphere* data set, but failed to identify smaller attribute subsets for the *glass* data sets.

Figure 7.5 illustrates the frequency of which each attribute was selected by the different search methods for the *glass* data set. The most frequently selected attributes were $\{ a_1\ a_2\ a_3\ a_6\ a_7\ a_8\}$, although in some of the folds one or more of these attributes were rejected. If this attribute subset is used for all the training folds, then the classification accuracy rises significantly by 10.41% to 78.502%. As the different search methods select a subset of attributes that maximises the evaluation function for each training fold, this suggests that whilst these subsets may be optimal for the training data, they are not optimal for the test data. This behaviour, known as *overfitting*, was also observed within a study that examined the performance of a *Best-First* search within the wrapper framework (Kohavi & Sommerfield 1995).

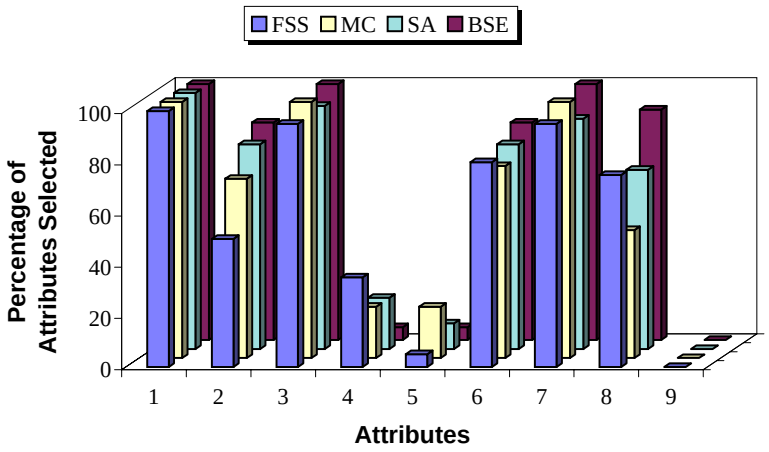


Figure 7.5: Histogram illustrating the frequency that attributes were selected by the different search techniques for the *glass* data set.

Three of the four search methods successfully identified the two relevant attributes in the *iris* data set, although the *Simulated Annealing* search also selected all four attributes for one of the folds. However, the *Backward Elimination* search selected all the attributes in some of the folds, and rejected one of the relevant attributes in others. As a consequence, the average classification accuracy fell slightly.

Discussion

The attribute subsets selected by the different search methods varied in both size and corresponding classification accuracy. The *Forward Selection* search selected the smallest attribute subset in the majority of cases, whereas the *Backward Elimination* search selected the largest subsets. This behaviour has also been reported in another study (John, Kohavi, & Pfleger 1994), and may be due to both searches becoming trapped within local maxima. For example, the *Forward Selection* failed to determine that all the attributes were relevant for the *monks-2* data set, as the selection of any single attribute appeared to be more optimal than the selection of two attributes. The average size of the attribute subsets selected by the two stochastic searches were similar (the *Simulated Annealing* search selected an average of 9.7 attributes across all the data sets, compared to 10.7 selected by the *Monte Carlo* search). A similar trend was also observed for the average classification accuracy across all the data sets; the lowest average accuracy was achieved when using the *FSS* search (73.79%), whereas the highest average accuracy was achieved by the *BSE* search (76.67%). The averages for the two stochastic search methods were 75.12% (*SA*) and 75.23% (*MC*).

Figure 7.6 illustrates how the accuracy of the Nearest Neighbour algorithm (plotted along the X axis) compares with the results obtained when the different search methods were used (plotted along the Y axis) for the individual data sets. Points plotted above the diagonal line represent cases where the wrapper method succeeded in increasing the classification accuracy of the Nearest Neighbour algorithm, whereas points plotted below this line represent a drop in accuracy.

Several other studies have investigated the utility of the wrapper method for attribute selection when learning with a rule induction algorithm (Caruana & Freitag, 1994; John et al., 1994; Kohavi, 1994) or a Nearest Neighbour algorithm (Salzberg, 1992; Aha & Bankert, 1994; Moore & Lee, 1994; Skalak, 1994) (see Table 3.2). The rule induction studies utilised variants of the *Forward Selection* and *Backward Elimination* searches. In general, the wrapper method was observed to identify smaller decision trees, but any resulting improvements in classifica-

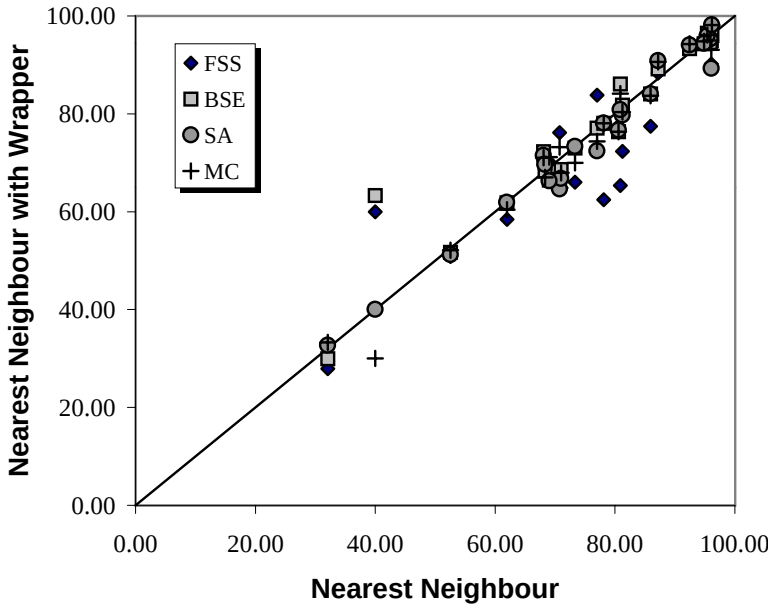


Figure 7.6: Scatter chart to illustrate the comparative performances of the different wrapper methods to the Nearest Neighbour algorithm.

tion accuracy were slight. The Nearest Neighbour studies found that the wrapper method could be used to identify small subsets of attributes when learning target concepts for certain domains. This resulted in a significant increase in classification accuracy for these domains.

The results presented in Table 7.4 suggest that the *lymph* and *zoo* data sets may not contain any irrelevant or redundant attributes, as all the subsets generated by the wrapper and filter selection techniques reduce the resulting classification accuracy of the NN algorithm. However, the *primary-tumor* and *promoters* data sets may contain redundant attributes, as although it is possible in most cases to reduce the number of attributes for each data set, this does not yield an increase in classification accuracy.

7.5 Evaluation of the Weighted Method

In Section 6.4, a method for determining a vector of attribute weights was introduced. This vector of weights could be utilised by a Nearest Neighbour algorithm to reduce the effects of irrelevant attributes present in a data set. This section describes three different ways in which this can be achieved, and evaluates each approach on both numeric and symbolic data sets. The three approaches are as follows:

Threshold Selection: The weights are first generated to rate the relevance of each attribute. A *selection threshold* is then used to determine whether attributes should be rejected, or retained within the attribute subset. If the attribute weight is greater than this threshold, the attributes are selected. This approach is similar to that used by *Relief* (Kira & Rendell 1992a; Kira & Rendell 1992b; Kononenko 1994).

Weighted Distance Metric: Weights are used adjust the effect each attribute has on the distance between two features, by utilising the weights within the distance metric. No explicit selection is performed with this method; however, if the value of the weight approaches zero, then this is equivalent to having removed the attribute from the attribute subset.

Weighted Threshold Selection: This hybrid approach combines the selection of attributes based on the selection threshold, and the inclusion of weights within the inter-instance distance calculations. This method should eliminate most of the irrelevant attributes from the attribute subset, and weight the remaining attributes accordingly.

Each of the three weighted methods described above was evaluated by performing an n-fold cross validation on the real-world and artificial data sets. A 10-fold cross validation approach was used for the symbolic data sets whereas a 20-fold approach was used for the numerical data sets. The weights were generated by

observing the distance metric during an evaluation of the training set[5]. The function *Update_Weight()* presented in Figure 6.8 was used to modify the weights during this phase. The resulting vector of weights was then used to either weight or select attributes, depending on the weighting method used. The numeric data was normalised by the NN algorithm prior to this weight generation process.

The *Update Coefficient* (page 117) determines the rate at which the weights are modified during the generation phase. A single coefficient of 0.02 was determined empirically and is used to update weights for all the data sets. The selection of a single coefficient is described in detail in Appendix B. When generating weights for numeric domains, an *inclusion threshold*, λ is also required (page 118). The value of λ varies for each data set. Table 7.5 lists the values of λ used for each data set.

Data Set	λ	Data Set	λ
bupa	*0.05*	ecoli	*0.2*
ionosphere	*0.05*	glass	*0.02*
pima	*0.5*	iris	*0.02*
sonar	*0.05*	wine	*0.05*
wisconsin	*0.3*	yeast	*0.1*
wdbc	*0.2*	balance	*0.05*
wpbc	*0.001*	waveform-21	*0.05*
		waveform-40	*0.2*

Table 7.5: The different inclusion thresholds used by the numeric data sets.

7.5.1 Threshold Selection

This approach generates a vector of attribute weights and then utilises them to determine whether or not attributes should be selected for the attribute subset. The intuition behind this approach is that the weights represent the *relevance* of each attribute; the greater the weight, the more frequently the respective attribute participated in determining the correct (or discriminating against the incorrect) nearest neighbour during the generation phase. A *selection threshold* is used to

[5]A leave-one-out cross validation approach was used to evaluate the training set when generating the weights.

select highly weighted attributes, and add these attributes to the attribute subset. The higher the selection threshold, the fewer attributes are selected.

The *Threshold Selection* algorithm, ThS, was evaluated using a variety of different selection thresholds in the range [0..1]. Tables 7.6 and 7.7 list the highest classification accuracies achieved for each data set. The number of attributes belonging to each data set is listed to the right of the name of the data set. The classification accuracy achieved for each data set is listed with the difference in accuracy between that method and the NN algorithm. The final two columns reflect the threshold used to achieve the listed result, and the number of attributes (averaged over all the folds used in the evaluation) that were selected. For some data sets, the selection of attributes resulted in a decrease in classification accuracy. In such cases, no entries for the threshold and average number of attributes are listed.

Symbolic Data

ThS_{OM} succeeded in raising the classification accuracy for five of the eight symbolic real world data sets (Table 7.6). Of these, four were significant (at either the 5% or 10% level). The rejection of attributes from the *primary-tumor*, *tic* and *zoo* data sets resulted in a decrease in accuracy: for this reason no threshold values have been listed for these data sets. However, none of the attribute selection methods tested so far have succeeded in identifying attribute subsets that significantly increased the accuracy for either the *primary-tumor* or *zoo* data sets. No significant increase in accuracy was achieved for the *lymph* data set, although the average number of attributes selected was reduced from 18 to 13.7. A similar result was also obtained by the *Backward Elimination* wrapper; an average of 13.2 attributes were selected which resulted in a slight but not significant increase in accuracy. The other selection methods identified smaller subsets, but the corresponding classification accuracies were worse than when all the attributes were used (i.e. using NN_{OM}).

The accuracy achieved for the *promoters* data set was higher than the respective accuracies achieved by any other method tested so far. However, unlike the

Forward Selection wrapper and the filtered NN algorithm which both selected an average of approximately 4.5 attributes, this method selected an average of 34 attributes. This could suggest that there are a number of redundant attributes present within this data set, although there is no evidence for this within the other wrapper results.

	Data Sets		ThS_{OM}	$\Delta(\text{ThS}_{OM})$	T'hold	Attrs.
Real World	breast-cancer	9	75.70	*4.97*	0.80	3.5
	lung-cancer	56	50.00	*10.00*	0.74	37.4
	lymph	18	83.10	1.86	0.50	13.7
	primary-tumor	15	32.05	0.00	—	—
	promoters	57	83.55	**6.55**	0.56	34.0
	tic	9	80.90	0.00	—	—
	votes	16	94.29	**1.85**	0.44	2.6
	zoo	16	96.09	0.00	—	—
Artificial	led	7	69.50	0.00	—	—
	led+17	24	53.00	**18.50**	0.56	5.4
	monks-1	6	97.22	18.52	0.50	3.0
	monks-2	6	73.84	0.00	—	—
	monks-3	6	94.44	11.57	0.80	3.0

Table 7.6: The results of applying the *Threshold Selection* algorithm to the UCI symbolic data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively.

None of the relevant attributes were rejected for the *led* artificial data set; however, only an average of 5.4 attributes were selected for the *led+17* data set. Although the elimination of the majority of irrelevant attributes reduced the detrimental impact of the additional irrelevant attributes on the overall accuracy, the rejection of two of the relevant attributes (a_1 & a_6) compromised the resulting concept hypothesis. Hence, the difference in classification accuracy between the two *led* data sets was 16.5%. The relevant attributes were selected for all three *Monks* data sets, and thus ThS_{OM} achieved the maximum classification accuracy possible for each.

Numeric Data

The classification accuracy increased significantly for four of the twelve data sets, and increased slightly for a further six when the *Threshold Selection* algorithm was used. It failed to detect the presence of any irrelevant attributes within the *pima* and *wdbc* data sets. Although none of the other selection approaches have so far identified any irrelevance in the *pima* data set, all but the *Forward Selection* wrapper approaches have succeeded in reducing the dimensionality for the *wdbc* data set (although no significant increase in accuracy was achieved). In contrast, small increases in accuracy were achieved (with a corresponding reduction in dimensionality) for the *bupa*, *ecoli*, *sonar*, *wisconsin* and *yeast* data sets, whereas the wrapper and filter methods failed to yield such improvements. This suggests that the *Threshold Selection* approach may identify different attributes as being relevant to those identified by either wrapper or filter approaches. For example, five of the six attributes predominantly selected by the wrapper approaches appeared in the subsets selected by ThS_{EM} for the *glass* data set (Figure 7.5). However, ThS_{EM} selected a_9 in preference to a_2 for the majority of folds tested, and consequently achieved a higher classification accuracy.

The result presented for the *balance* data set is somewhat misleading. Although the average number of attributes selected could be reduced slightly without any detrimental effect on the classification accuracy, this was actually due to the removal of two attributes from only one of the twenty folds tested. The removal of any further attributes from any of the other folds resulted in a drop in classification accuracy for ThS_{EM}. Significant increases in accuracy were achieved for both *waveform* data sets, although the reduction in dimensionality was less than that achieved by other methods. The majority of the twenty-one significant attributes in the *waveform-40* data set were selected in most of the folds (attributes $a_2 \ldots a_{18}$ were selected in at least 17 of the 20 folds), although three irrelevant attributes were also selected with this frequency. This differs somewhat to the subsets selected by the filter method (Figure 7.3). If a higher threshold is selected for the *waveform-40* data set (e.g. 0.42), then although the average number of

	Data Sets		ThS$_{EM}$	Δ(ThS$_{EM}$)	T'hold	Attrs.
Real World	bupa	6	64.72	2.74	0.52	4.9
	ionosphere	34	90.38	**3.21**	0.06	17.6
	pima	8	70.99	0.00	—	—
	sonar	60	87.55	1.60	0.36	42.6
	wisconsin	9	96.77	*0.87*	0.60	7.4
	wdbc	30	95.40	0.00	—	—
	wpbc	33	73.17	4.11	0.50	1.2
	ecoli	7	81.18	0.59	0.86	6.2
	glass	9	73.36	*5.27*	0.36	6.0
	iris	4	98.13	**1.97**	0.60	2.0
	wine	13	96.04	1.18	0.40	8.3
	yeast	8	53.03	0.47	0.50	7.8
Artificial	balance	4	78.10	0.00	0.32	3.9
	waveform-21	21	76.67	*3.34*	0.24	18.4
	waveform-40	40	75.67	**7.34**	0.32	24.4

Table 7.7: The results of applying the *Threshold Selection* algorithm to the UCI numeric data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively.

attributes selected by ThS$_{EM}$ is similar to that selected by FNN$_{EM}$ (i.e. 17.8 and 17.6 respectively), the accuracy achieved by ThS$_{EM}$ drops significantly (p=0.07) by 5.0%, suggesting that of the two methods, the filter method is more successful in identifying the relevant attributes.

7.5.2 Weighted Distance Metric

The Nearest Neighbour algorithm described in Section 7.5.1 uses the vector of relevance weights to select relevant attributes. This is achieved by comparing each weight with a selection threshold. The *Weighted Distance Metric* approach (ωNN) combines the weights with the distance metric to adjust the effect each attribute has on the distance between two instances.

Most distance metrics (Section 2.2.3) determine the distance between two instances by summing the individual distances between the respective features for each attribute (Equation 2.2). In the case of the *Overlap* or *Euclidean*[6] metrics,

[6]If the numeric attributes are not normalised, then this range will be determined by the

the distance between each feature pair will lie within the range [0..1]. An attribute weight can be used to restrict this range, and thus reduce the apparent distance between the two features. For example, if the distance between the two respective features of instances i and j is $\delta(i_a, j_a)$, and the two features are diametrically positioned within the (normalised) feature space, then $\delta(i_a, j_a) = 1$ (Equation 2.5). If this distance is then multiplied by the weight ω_a, then $\delta(i_a, j_a) \cdot \omega_a = \omega_a$. However, if the two features occupy the same position within the feature space, then $\delta(i_a, j_a) = 0$, and hence $0 \cdot \omega_a = 0$.

The larger the attribute distance, the greater effect it will have on the value of $D(i, j)$, the distance between the instances i and j. Thus if a small weight is used (i.e. if the attribute is irrelevant), then $\delta(i_a, j_a) \cdot \omega_a$ will also be small and thus have a reduced impact on the choice of nearest neighbour[7].

The classification accuracies obtained by the *Weighted Distance Metric* are presented in Tables 7.8 and 7.9. The results obtained by the basic nearest neighbour algorithm have been listed in the second columns of both tables for reference. The difference in accuracies between the nearest neighbour algorithm and the *Weighted Distance Metric* algorithm are presented in the final columns.

Symbolic Data

The classification accuracy increased for five of the eight symbolic data sets when the weights were employed by the *Overlap* distance metric. Of these, only the results for the *lung-cancer* and *votes* data sets were significant at the 5% significance level. However, there was a significant drop in the classification accuracy for the *breast-cancer* data set. In general, the performance of ωNN_{OM} was slightly worse than that of ThS_{OM} (Figure 7.6), although the ωNN_{OM} classification accuracy was significantly lower when tested on the *breast-cancer* data set (the accuracy of ωNN_{OM} was 65.37%, compared to 75.70% achieved by ThS_{OM}).

The *Threshold Selection* method (ThS_{OM}) failed to reject any of the attributes

maximal value of each attribute.

[7] A similar principal is utilised by the Value Difference Metric (page 37).

	Data Sets		NN_{OM}	ωNN_{OM}	$\Delta(\omega NN_{OM})$
Real World	breast-cancer	9	70.73	**65.37**	**-5.36**
	lung-cancer	56	40.00	*46.67*	*6.67*
	lymph	18	81.24	78.53	-2.71
	primary-tumor	15	32.05	32.37	0.32
	promoters	57	77.00	79.91	2.91
	tic	9	80.90	82.88	1.97
	votes	16	92.44	**94.74**	**2.29**
	zoo	16	96.09	95.09	-1.00
Artificial	led	7	69.50	**65.50**	**-4.00**
	led+17	24	34.50	**55.50**	**21.00**
	monks-1	6	78.70	98.38	19.68
	monks-2	6	73.84	70.83	-3.01
	monks-3	6	82.87	94.91	12.04

Table 7.8: Classification accuracies of the *Weighted Distance Metric* algorithm (ωNN_{OM}) for the UCI symbolic data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The results of the basic nearest neighbour algorithm have been reproduced from Table 7.2 for reference.

of the *primary-tumor* or *tic* data sets, and hence had no affect on the resulting classification accuracy. However, there was a slight (but not significant) increase in accuracy for ωNN_{OM} with these data sets. This suggests that some of the domains tested may contain attributes that are only slightly relevant to the target concept. As a consequence, the inclusion of such attributes may be essential for the accurate classification of some of the instances of a domain, but they may also adversely bias the algorithm when classifying others. Hence, the weighting of such attributes can minimise this adverse behaviour, without degrading the overall classification accuracy.

The ωNN_{OM} algorithm significantly increased the accuracy for all of the artificial data sets that contained irrelevant attributes (i.e. *led+17*, *monks-1* and *monks-3*). This confirms the hypothesis that combining weights with the distance metric can improve the classification accuracy of a nearest neighbour algorithm when irrelevant attributes are present within the data set. However, there was an unexpected drop in accuracy for the (*led* and *monks-2*) data sets. Neither of these data sets contained irrelevant or redundant attributes, and thus one would expect all the attributes to be highly weighted. One possible explanation for this behaviour may

be the way in which the weights were generated. If a data set consists of a set of attributes that were equally relevant to the target concept (such as within the *monks-2* problem; see page 126), then the resulting vector of weights should all be equal. However, if the data set contains a skewed sample of instances that represent only part of this concept (e.g. if the classification of the majority of the positive instances was determined by only a subset of the attributes), then the vector of weights generated from this training set would reflect this skew in attribute relevance.

Numeric Data

The *Weighted Distance Metric* algorithm increased the classification accuracy of the nearest neighbour algorithm for all but one of the numeric real world data sets. The increase was significant at the 5% level for the *ionosphere*, *wisconsin* and *wine* data sets, and significant at the 10% level for a further four data sets. There was a slight, but not significant reduction in classification accuracy for the *bupa* data set.

The results obtained by the *Weighted Distance Metric* algorithm were higher than those obtained by the *Threshold Selection* algorithm for seven of the twelve real world data sets. In addition, the other attribute selection methods tested so far have failed to increase the classification accuracy for four of these data sets: *pima*, *sonar*, *ecoli* and *yeast*.

There was a slight decrease in the accuracy for the *balance* data set. However, a significant increase in accuracy was observed for the two *waveform* data sets (both at the 5% confidence level). In general, the weights for the relevant and irrelevant attributes in the *waveform-40* data set differ. The median weight for the 21 relevant attributes is 1.308 ± 1.040, whereas the median weight for the remaining 19 irrelevant attributes is 0.708 ± 0.448.

	Data Sets		NN_{EM}	ωNN_{EM}	$\Delta(\omega NN_{EM})$
Real World	bupa	6	61.98	61.70	-0.28
	ionosphere	34	87.17	**91.78**	**4.61**
	pima	8	70.99	71.25	0.26
	sonar	60	85.96	*88.96*	*3.00*
	wisconsin	9	95.90	**96.64**	**0.74**
	wdbc	30	95.40	95.59	0.18
	wpbc	33	69.06	70.06	1.00
	ecoli	7	80.59	*81.49*	*0.90*
	glass	9	68.09	*72.00*	*3.91*
	iris	4	96.16	96.79	0.63
	wine	13	94.86	**97.15**	**2.29**
	yeast	8	52.56	*53.37*	*0.81*
Artificial	balance	4	78.10	77.15	-0.94
	waveform-21	21	73.33	**76.67**	**3.33**
	waveform-40	40	68.33	**76.00**	**7.67**

Table 7.9: Classification accuracies of the *Weighted Distance Metric* algorithm (ωNN_{EM}) for the UCI symbolic data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The results of the basic nearest neighbour algorithm have been reproduced from Table 7.3 for reference.

7.5.3 Weighted Threshold Selection

The *Weighted Threshold Selection* algorithm is a hybrid algorithm that combines the *Threshold Selection* approach (Section 7.5.1) for selecting highly weighted attributes, and utilising the attribute weights of the selected attributes within the distance metric (Section 7.5.2). It was evaluated using a variety of different selection thresholds in the range [0..1]. The highest classification accuracies are listed in Tables 7.10 (for symbolic data sets) and 7.11 (for numeric data sets). The third columns list the differences in accuracy when compared to the Nearest Neighbour algorithms (see Tables 7.2 and 7.3). As in previous tables, the final two columns reflect the threshold used to achieve the listed result, and the number of attributes (averaged over all the folds used in the evaluation) selected. For some data sets, the selection of attributes resulted in a decrease in classification accuracy. In such cases, no entries for the threshold and average number of attributes are listed.

Symbolic Data

The performance of the *Weighted Threshold Selection* algorithm was no better than that of of either the *Threshold Selection* algorithm (ThS_{OM}) or the *Weighted Distance Metric* algorithm (ωNN_{OM}) for five of the eight real world symbolic data sets (Table 7.10). The classification accuracy of the hybrid algorithm was slightly lower than either ThS_{OM} or ωNN_{OM} for the *lymph* data set, but higher for the *promoters* data set ($\text{ThS}_{OM} = 83.55\%$, $\omega\text{ThS}_{OM} = 83.64\%$), and for the *votes* data set ($\omega\text{NN}_{OM} = 94.74\%$, $\omega\text{ThS}_{OM} = 95.43\%$). However, none of these differences between weighted methods were statistically significant.

	Data Sets		ωThS_{OM}	$\Delta(\omega\text{ThS}_{OM})$	T'hold	Attrs.
Real World	breast-cancer	9	*75.70*	*4.97*	0.80	3.5
	lung-cancer	56	**50.00**	**10.00**	0.74	37.4
	lymph	18	79.19	-2.05	0.34	16.2
	primary-tumor	15	32.37	0.32	—	—
	promoters	57	*83.64*	*6.64*	0.58	31.0
	tic	9	82.88	1.97	—	—
	votes	16	**95.43**	**2.99**	0.28	8.4
	zoo	16	95.09	-1.00	—	—
Artificial	led	7	**65.50**	**-4.00**	—	—
	led+17	24	**56.00**	**21.50**	0.32	19.2
	monks-1	6	98.38	19.68	—	—
	monks-2	6	70.83	-3.01	—	—
	monks-3	6	95.14	12.27	0.50	5.0

Table 7.10: The classification accuracies obtained by the *Weighted Threshold Selection* algorithm (ωThS_{OM}) for the UCI symbolic data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The threshold and average number of selected attributes are only given for data sets where attribute selection resulted in an increase in classification accuracy.

ωThS_{OM} achieved a significant rise in accuracy for the *led+17* data set, although only an average of 4.8 attributes were rejected. This result was higher than that obtained by the *Threshold Selection* algorithm, which had rejected the majority of attributes (including relevant attributes). No attributes were rejected from the *led*, *monks-1* and *monks-2* data sets. Thus, the classification accuracies achieved for these data sets were the same as those obtained by the *Weighted Distance Metric*

algorithm. The ωThS$_{OM}$ algorithm rejected one of the three irrelevant attributes for the *monks-3* data set, and yet obtained a higher accuracy than that for ThS$_{OM}$ (95.14% and 94.44% respectively). This result was surprising, as the *Threshold Selection* algorithm had rejected all the irrelevant attributes. However, the result obtained by ωNN_{OM} was also high (94.91%, see Table 7.8), demonstrating that weights employed within the distance metric can often be as effective as attribute selection when irrelevant attributes are present.

Numeric Data

An overall increase in accuracy was achieved by ωThS$_{EM}$ for all twelve numeric data sets when compared to the NN$_{EM}$ classification accuracy (see Table 7.11). The hybrid algorithm also achieved an increase in accuracy for six of the twelve data sets, when compared to either ThS$_{OM}$ or ωNN$_{OM}$. In each case, the number of attributes selected (on average) by the ωThS$_{EM}$ algorithm was the same or greater than that by ThS$_{EM}$. The ThS$_{EM}$ algorithm achieved slightly higher (but not significantly) classification results for a further four data sets, but on average selected a similar number of attributes.

There was degradation in classification accuracy for the *balance* data set. As this data set contains no irrelevant attributes, this behaviour could be due to the distance metric being weighted. In Section 7.5.2, the classification accuracy was observed to fall for data sets containing no irrelevant attributes when weights were integrated into the distance metric. The resulting classification accuracy of ωNN$_{EM}$ for the *balance* data set was 77.15%. The accuracy for ωThS$_{EM}$ was slightly higher than ωNN$_{EM}$ (77.48%), but this was due to the thresholding and subsequent rejection of some of the attributes. However, it should be noted that the number of attributes rejected was very small (i.e. 0.1 attributes rejected, which is equivalent to one attribute being omitted from the subset in only two of the 20-fold evaluations), and hence their rejection was almost certainly due to the effects of overfitting.

There was also a significant rise in accuracy for the two *waveform* data sets.

	Data Sets		ωThS$_{EM}$	$\Delta(\omega$ThS$_{EM})$	T'hold	Attrs.
Real World	bupa	6	63.55	1.57	0.54	4.8
	ionosphere	34	**92.32**	**5.15**	0.06	17.6
	pima	8	71.25	0.26	—	—
	sonar	60	*89.50*	*3.55*	0.32	47.4
	wisconsin	9	**96.92**	**1.02**	0.54	8.6
	wdbc	30	95.93	0.53	0.70	20.6
	wpbc	33	72.61	3.55	0.90	1.0
	ecoli	7	*81.49*	*0.90*	0.88	6.2
	glass	9	*72.91*	*4.82*	0.22	7.8
	iris	4	98.04	1.88	0.52	2.1
	wine	13	**97.71**	**2.85**	0.38	9.3
	yeast	8	53.91	*1.35*	0.50	7.8
Artificial	balance	4	77.48	-0.62	0.34	3.9
	waveform-21	21	**76.67**	**3.33**	—	—
	waveform-40	40	**76.33**	**8.00**	0.16	38.3

Table 7.11: A comparison of the delta classification accuracies for the *Weighted Distance Metric* algorithm ($\Delta(\omega$NN$_{EM})$) and the *Weighted Threshold Selection* algorithm ($\Delta(\omega$ThS$_{EM})$) for the UCI symbolic data sets. Values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The threshold and average number of selected attributes are only given for data sets where attribute selection resulted in an increase in classification accuracy.

However, none of the attributes were rejected for the *waveform-21* data set, and hence this result was the same as that achieved by ωNN$_{EM}$. Although an average of 1.7 attributes were eliminated from the *waveform-40* data set, the threshold selected was very low (0.16) which suggests that these attributes had little effect on the classification accuracy when the *Weighted Distance Metric* was used (their omission from the subset yielded an increase in accuracy of 0.33%).

Discussion

The behaviour of each of the three weighting strategies presented above appears to be dependent on the data type of each data set. Although the average classification accuracies achieved by each of the three methods were generally better than those achieved by other techniques described above, the difference between the *Weighted Distance Metric* algorithm and the *Threshold Selection* algorithm

was much more pronounced for the symbolic data set than the numeric data set. ThS_{OM} achieved a greater average accuracy for the symbolic data sets (75.67%) than either ωNN_{OM} (73.90%) or ωThS_{OM} (75.40%). In contrast, the weighted threshold selection algorithm ωThS_{EM} achieved a slightly greater average accuracy (81.11%) for the numeric data sets than other weighed approaches (80.74% with ThS_{OM} and 81.11% with ωNN_{OM}).

The performance of the *Weighted Distance* method was weaker than that of the other two weighted methods across the majority of data sets, and in several cases resulted in a drop in accuracy (relative to the performance of the basic nearest neighbour algorithm). This method was observed to improve the performance of the nearest neighbour algorithm only when there were a number of irrelevant attributes (such as the artificial data sets). However, the accuracy degrades when there is little irrelevance present in the data set.

The *Weighted Threshold Selection* algorithm selected fewer attributes than the *Threshold Selection* algorithm, but performed as well as the *Threshold Selection* algorithm (with symbolic data sets) or better than this algorithm (with numeric data sets). This suggests that there may be a number of attributes that may be only slightly relevant to the learning task for the majority of instances, but who's values are important in determining the class of a few instances.

A small number of studies have contrasted different weighting strategies. As part of a study that evaluated a number of different methods for generating weights, Wettschereck, Aha, & Mohri (1997) compared the performance of two weighting strategies; one that assigned weights to attributes, and another that performed selection based on weights. The results suggested that attribute selection yielded higher classification accuracies when attributes were either highly correlated with the class labels (i.e. highly relevant), or if attributes were irrelevant. However, attribute weighting is more appropriate when the relevance of attributes can vary. Kohavi, Langley, & Yun (1997) investigated the utility of generating discrete weights as an alternative to continuous weights or binary weights (i.e. selecting or rejecting attributes). They found that although the use of weights improved the classification accuracy of a nearest neighbour learning algorithm, a drop in

accuracy was observed when more than a few discrete weights were used, and typically that the best performance was observed with only one non-zero weight (i.e. binary weights).

7.6 Evaluation of Sub-space Method

The sub-space approximation technique differs from the attribute selection methods evaluated so far in that instances are mapped from the instance space into an alternative, lower dimensional approximation of the instance space. Thus, no explicit attribute selection stage is performed, although the number of dimensions used to describe the data sets should fall. The various mapping functions employed by the sub-space approximation method were introduced in Section 6.5. Two alternative methods for generating the mapping between the instance space and the approximated subspace were presented: an unsupervised mapping function *generate_mapping* (Figure 6.9), and a supervised variant *generate_class_projected_mapping* (Figure 6.11). These two functions have been embedded within two nearest neighbour learning algorithms: *CA-NN* utilised *generate_mapping* to determine the mapping between the two spaces, whereas *CACP-NN* used *generate_class_projected_mapping*.

The *quality* of the sub-space approximation, i.e. how well the approximated subspace represents the original space, is determined by the *rank* of the approximation (see Section 3.5.1). The rank is an integer in the range $[1..J]$, where J represents the number of attributes used to describe the original data set (i.e. the dimensionality of the instance space). The rank determines the dimensionality of the approximated sub-space, so that a high rank sub-space will better approximate the original instance space than a low rank sub space.

The two learning algorithms were evaluated by performing an 20-fold cross validation on each of the data sets tested. As the sub-space mapping functions were designed to work with numerical data, only the numerical data sets were tested. The *Euclidean* distance metric was used to determine the distance between in-

	Data Sets		CA	Δ(CA)	CACP	Δ(CACP)
Real World	bupa	6	61.98 (6)	0.00	61.98 (6)	0.00
	ionosphere	34	90.90 (22)	**3.72**	91.19 (11)	**4.02**
	pima	8	70.99 (8)	0.00	70.99 (8)	0.00
	sonar	60	86.96 (23)	1.00	85.50 (30)	-0.46
	wisconsin	9	97.36 (6)	**1.47**	96.19 (3)	0.30
	wdbc	30	96.65 (5)	*1.25*	96.29 (16)	0.89
	wpbc	33	71.61 (16)	**2.56**	73.06 (15)	4.00
	ecoli	7	80.59 (7)	0.00	81.82 (5)	1.23
	glass	9	68.09 (8)	0.00	70.00 (8)	1.91
	iris	4	96.16 (4)	0.00	96.70 (3)	0.54
	wine	13	97.08 (6)	*2.22*	97.64 (6)	**2.78**
	yeast	8	52.62 (7)	0.06	52.62 (7)	0.06
Artificial	balance	4	78.12 (4)	0.02	88.95 (1)	**10.86**
	waveform-21	21	81.00 (3)	**7.67**	79.33 (3)	**6.00**
	waveform-40	40	83.00 (2)	**14.67**	84.67 (3)	**16.33**

Table 7.12: Accuracies of the two sub-space approximation algorithms for numeric UCI data sets. These values indicate the relative difference in accuracy between *NN* and *CA-NN*/*CACP-NN*. Those values in bold or italic indicate a significant difference at the 5% or 10% confidence level respectively. The average number of selected attributes appear in parentheses.

stances. The rank of the approximation was varied between 1 and J for each of the data sets. The results, presented in Table 7.12, list the highest accuracies and corresponding ranks (in parentheses) for each of the data sets. For some of the data sets, the approximation of the instance space resulted in a drop in classification accuracy. In these cases, the full rank is listed.

Numeric Data

CA-NN succeeded in raising the classification accuracy whilst reducing the dimensionality for seven of the twelve real-world data sets; three of which were significant at the 5% significance level. It also succeeded in reducing the dimensionality for one further data set (*glass*); however, this reduction was slight and the classification accuracy was unaffected. No reduction in dimensionality was achieved for the *pima*, *ecoil* and *iris* data sets, although the *iris* data set is known to contain irrelevant attributes.

The performance of *CA-NN* on both *waveform* data sets was superior to any other dimensionality reduction method tested so far. The classification accuracy was significantly greater than any other method, yet only three dimensions were required to represent the *waveform-21* data set (two for the *waveform-40* data set). However, as the waveform data set consists of linearly dependent attributes (Section 7.2), this suggests that the subspace mapping function may be combining these multiple attributes into a smaller number of dimensions (an example of this is illustrated in Figure 3.6).

Although the class projected algorithm, *CACP-NN*, achieved an increase in classification accuracy for nine of the real-world data sets, only two of these results were significant (*ionosphere* & *wine*). In addition there was a slight drop in classification accuracy for the *sonar* data set. Again, a significant increase in classification accuracy (and drop in dimensionality) was observed for the *waveform* data sets.

One of the most interesting results however, was observed for the *balance* data set. Although this data set contains no irrelevant or redundant attributes, *CACP-NN* succeeded in identifying a single dimensional subspace that could accurately approximate the original instance space. This one dimension space yielded a significant increase in classification accuracy (88.95%, compared to 78.10% for *NN*). The *balance* data set consists of linearly dependant attributes. However, it differs from the *waveform* data set in that the instances are randomly distributed within the instance space, and hence this attribute dependency may only be apparent when the class centroids are calculated. This may explain why *CA-NN* was unable to identify a low dimensional approximated subspace, and yet *CACP-NN* was more successful.

Figure 7.7 illustrates how the classification accuracy of both *CA-NN* and *CACP-NN* change as the number of dimensions (i.e. rank) is varied for three data sets. The accuracy initially rises as the rank increases, and then plateaus out, indicating that the additional dimensions make little contribution (if any) to the performance of the algorithm. In the case of the *ionosphere* data set, the accuracies of both algorithms fall slightly as additional dimensions are included. This could be due to noise present in the data, which would be eliminated by performing lower rank

(i.e. cruder) approximations.

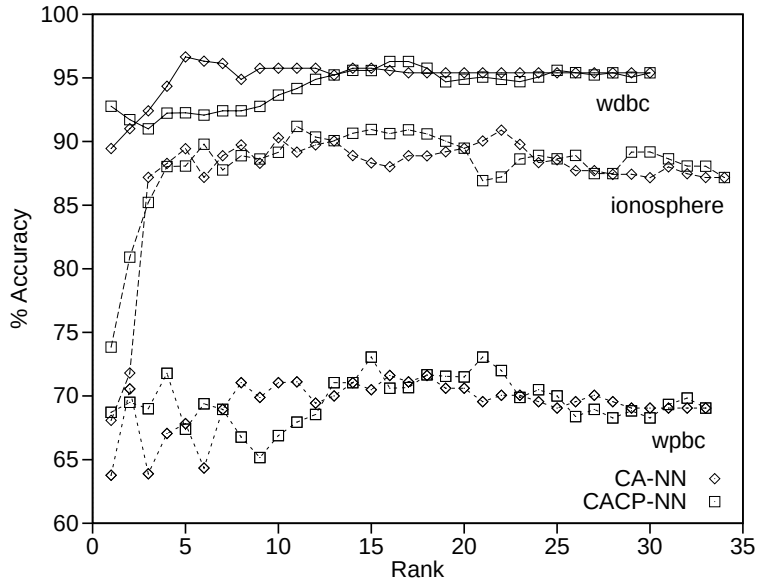


Figure 7.7: The learning curves for three data sets: *ionosphere, wdbc* and *wpbc*. Each curve is plotted as a function of the rank of the approximated sub space.

The results obtained for the *iris* data set were lower than expected, as two of the four attributes (*petal length* and *petal width*) are known to be highly relevant to the classification task (Duda & Hart 1973; Michie, Spiegelhalter, & Taylor 1994). If only these two attributes are included in the data set, then the accuracy for *NN* increases from 96.16% to 98.13%, and there is a corresponding increase in accuracy for *CA-NN* (*CACP-NN*) from 92.05% (93.93%) to 96.67% (96.67%) within a single rank subspace. This suggests that the classification accuracy of both *CA-NN* and *CACP-NN* may degrade in the presence of irrelevant attributes.

The result achieved by *CA-NN* for the *balance* data set suggests that when all the dimensions are present (i.e. no approximation is generated), the subspace mapping may still affect the classification accuracy of the learning algorithm. Figure 7.8 illustrates distribution of instances from the *iris* data set (using only the *petal length* and *petal width* attributes) in the original, canonical space, and their distribution within the new (full rank) subspace. This figure illustrates the effects of the mapping function generated by *CA-NN* on the instances. The mapping function performs a rotation and a linear translation. The rotation should not

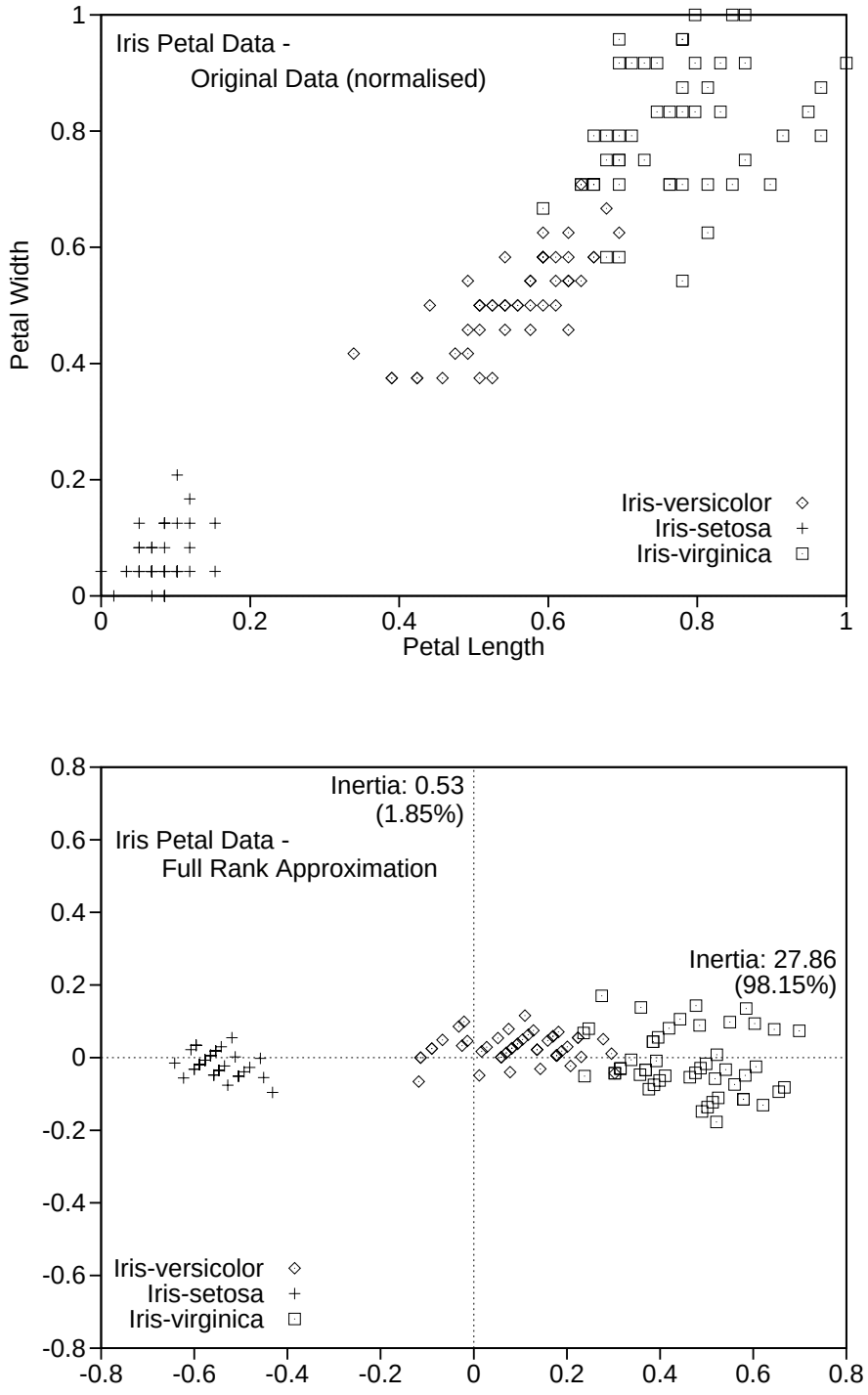


Figure 7.8: Mapping the two most relevant attributes of the *iris* data set into a full ranked subspace.

affect the performance of the nearest neighbour algorithm. However, the varying translation of each dimension has the affect of distorting the subspace with respect to the original space, and hence can affect the distance function.

Results from artificial data sets

The results for the *iris* data set suggested that the performance of *CA-NN* and *CACP-NN* may degrade in the presence of irrelevant attributes. To investigate this hypothesis, two further data sets were created, consisting of 100 instances each. These two dimension, binary class data sets are illustrated in Figure 7.9. The first data set comprises of two linearly separable partitions. As *CACP-NN* identifies and utilises class centroids, the second data set contains four linearly inseparable partitions, two per class. A set of fifty, single attribute data sets were constructed, each containing a single random value for each instance. New data sets were created for each experiment by combining one of the binary class data sets with a random sample of these irrelevant attribute data sets.

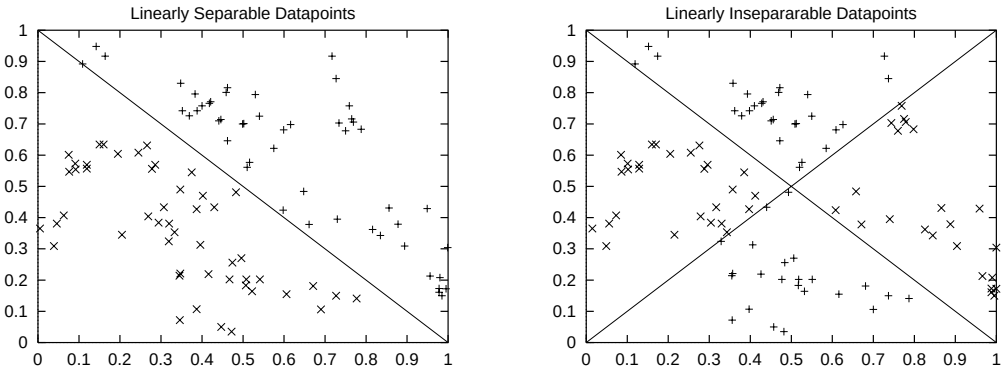


Figure 7.9: Two dimensional artificial data. Datapoints are either Positive (+) or Negative (×).

Various experiments were performed to investigate the behaviour of *CA-NN* and *CACP-NN* in the presence of irrelevant attributes. For each experiment, the two data sets containing the relevant attributes were augmented with an increasing number of irrelevant attributes. Each data set was then tested with *NN*, *CA-NN* and *CACP-NN*. This was repeated fifteen times for different combinations of

irrelevant attributes.

Figure 7.10 illustrates the results obtained from experiments on the linearly separable data set. The classification accuracy of all three algorithms falls exponentially, as the number of irrelevant attributes increase. The classification accuracies for *NN* and *CA-NN* are comparable with data sets containing small numbers of irrelevant attributes. However, after the number of irrelevant attributes exceeds fourteen, the difference in classification accuracy between the two algorithms becomes small but significant (a one-tailed t-test shows significance at the 5% level), with *CA-NN* achieving a slightly higher accuracy than *NN*. The number of dimensions used by *CA-NN* varies as the number of irrelevant attributes in the data set increases. There is no reduction in dimensionality for data sets with few irrelevant attributes. As the number of irrelevant attributes increases beyond eight to forty nine, the number of dimensions selected by *CA-NN* increases slowly from eight to twenty nine.

The error rate of *CACP-NN* is much lower than that achieved by either *CA-NN* or *NN*. *CACP-NN* achieved a mean accuracy of 74.74% with forty nine additional attributes, whereas *CA-NN* and *NN* achieved mean accuracies of 57.47% and 55.93% respectively. The number of dimensions selected to approximate the space remained relatively constant (between three to five dimensions).

The results for the three algorithms on the linearly inseparable data sets are shown in Figure 7.11. Although *CACP-NN* achieved superior results for these data sets, the overall performance was much lower than with linearly separable data. This drop in accuracy for *CACP-NN* may be due to the method used to generate the centroids for each class. These centroids occur within a close proximity to each other, due to the distribution of the instances of each class. Although the instance space is divided into multiple partitions for each class, only a single centroid is generated for each class.

The initial drop in accuracy in *NN* is not surprising, as there is an additional boundary separating the points of each class, and a small number of points lie along this new boundary. However, the results after the addition of only a few

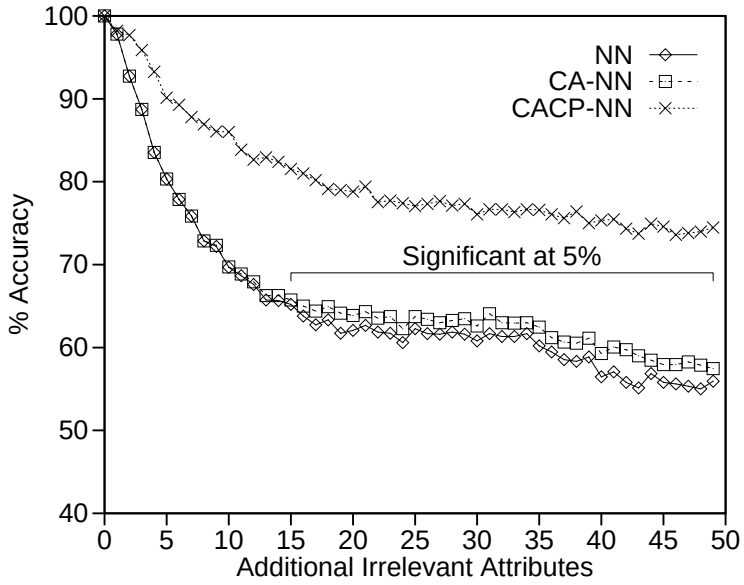


Figure 7.10: The effects of additional irrelevant attributes for a linearly separable data set on three learning algorithms.

attributes (e.g. 11 attributes) are little better than that achieved by pure chance, indicating that any contribution that the relevant attributes have to any classification hypothesis has been obscured by the effects of the irrelevant attributes. The results show an unusual increase in accuracy for *CACP-NN* for data sets containing between five and fourteen additional attributes. As yet, no explanation has been found for this behaviour.

The above experiments were repeated to investigate the behaviour of both *CA-NN* and *CACP-NN* in the presence of redundant attributes. A set of forty-eight, single attribute data sets were constructed, each containing a single value for each instance. These values were based on those in the two data sets illustrated in Figure 7.9, and were calculated in one of several ways: values were copied from one of the dimensions of the original data sets; or values were calculated by inverting one of the dimensions using the function $f(x) = 1 - x$. In addition, some of the single attribute data sets were modified using the function $f(x) = x \times (1 \pm rnd(\delta))$, where $rnd(\delta)$ generates a small random number between 0 and δ; for this study we used $\delta = 0.05$.

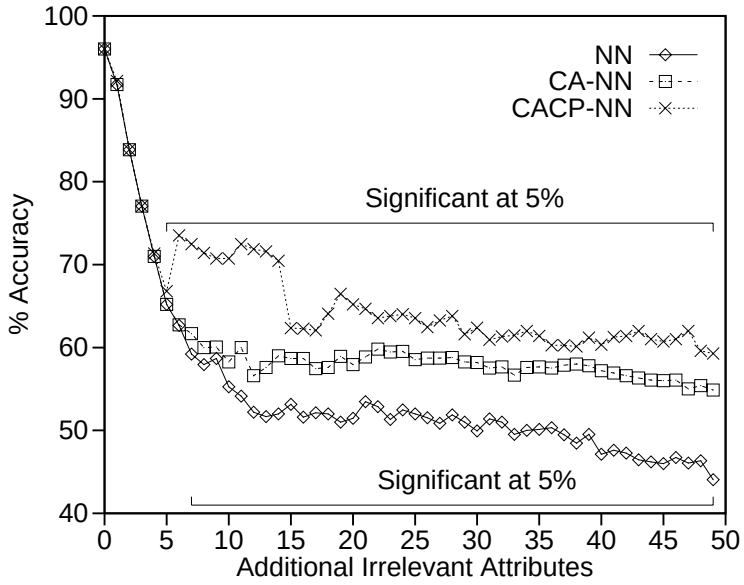


Figure 7.11: The effects of additional irrelevant attributes for a linearly inseparable data set on three learning algorithms.

All three algorithms achieved approximately 100% accuracy for the linearly separable data set and 96.00% for the linearly inseparable data set. A rank of two was always selected for *CA-NN*, whereas the mean rank varied between one and four for *CACP-NN*.

Discussion

The *Sub-space Approximation* techniques described above could successfully identify lower rank sub-spaces that maintained or increased the classification accuracy of a nearest neighbour learning algorithm for the majority of data sets tested. *CAC-NNP* achieved the highest average classification accuracy (81.80%, compared to 78.57% for the basic *NN* algorithm) of all the methods tested (Appendix A), whilst finding the second highest average reduction in dimensionality (an average of 8.33 dimensions compared to 19.07 in the original data sets).

In general, attribute selection techniques have been used to identify and eliminate both redundant and irrelevant attributes. The dimensionality reduction techniques used by *CA-NN* and *CACP-NN* appear to be very successful in removing

redundant dimensions from the data set. The data points are represented by an attribute-by-instance matrix. Once this matrix has been decomposed, the rank of the matrix can be determined by the resulting diagonal matrix. This rank represents the number of linearly independent, orthogonal dimensions within a subspace. Therefore, the addition of any duplicate attributes, or any linear combinations of attributes, will not result in an increase in rank, and so will be eliminated by the decomposition. If two or more attributes contain very similar, but not identical values, then there will be additional orthogonal dimensions to express the slight deviations between them. Because the inertia of such dimensions will be small, a lower rank subspace that excludes these dimensions will closely approximate the original subspace.

Unlike many of the existing attribute selection techniques, the dimensionality reduction techniques described here appear to have little impact in reducing the effects of irrelevant attributes. However, the performance of the class projected variant *CACP-NN* degrades at a slower rate than either *CA-NN* or a simple nearest neighbour in the presence of irrelevant attributes.

Chapter 8

Summary, Conclusions and Further Work

Chapter Outline

This final chapter summarises the results obtained from the empirical investigation presented in Chapters 5 and 7. The impact of these results are discussed with respect to nearest neighbour learning algorithms and the integration of such systems within information agents. Finally, a number of the issues that have been raised by this work are then discussed, and presented as directions for further study.

8.1 Summary

The objectives of the thesis were to propose a compact data representation that could preserve the structure of complex documents (such as electronic mail messages), and to investigate the role of attribute selection for nearest neighbour learning algorithms. A summary of the representation proposed, the evaluation of this representation, and the investigation of attribute selection techniques are presented below.

A novel set-based representation was presented in Chapter 4. This representation maps each of the fields in a complex document to a *set-valued attribute*. It differs from the traditional representation in that it relaxes the constraint that attributes can only express a single value within an instance. A set-valued attribute contains a set of unordered values that can be used to represent a single field. For example, a single set-valued attribute corresponding to the *Journal Scope* field within the bibliographic entry example in Figure 1.1 may express all the terms that appear in that field as an unordered collection of terms.

Three different nearest neighbour learning algorithms that utilise this representation were proposed:

- *IBPL1* determines the distance between two set-valued attributes by finding the average of all the individual distances between combinations of feature tuples, where each tuple contained two features, one from each of the set-valued attributes;
- *IBPL2* identifies the minimal distance for each of the features present in the set taken from the query instance, by comparing each of these features with all the features within the corresponding set of the training instance. The minimal distances are then averaged to determine the overall distance between the two sets.
- *PIBPL* utilises relevance weights to reject features from the set-valued attributes prior to calculating the distance between sets.

The results of the evaluations of each of these three nearest neighbour learning algorithms were presented within Chapter 5. *IBPL1* was evaluated within the framework of an intelligent mail filtering agent (Payne 1994). The motivation for using this framework was to contrast the performance of *IBPL1* with a variant of the rule induction algorithm, CN2. The learning algorithms were employed not only to predict a classification for each mail message within the test data set, but also to determine a measure of reliability for each prediction. For this reason, the performance of both learning algorithms was measured along two dimensions: coverage and accuracy. The coverage represented the number of reliable predictions made (i.e. a threshold was used to determine whether or not a prediction was reliable). The accuracy represented the percentage of correct classifications reliably predicted.

The results suggested that the performance of *IBPL1* was comparable to CN2 across both dimensions. However, this performance varied depending on the learning algorithm used and the characteristics of the data tested. The performance of both algorithms increased when a greater number of fields were mapped from the email messages to training and test instances.

The performance of *IBPL2* (Section 5.2) was determined across the email data sets, and compared with that of *IBPL1*. Although there was a slight variation in the results obtained for both algorithms (in terms of both coverage and accuracy), there was no significant difference between the two algorithms.

The work on *IBPL1* and *IBPL2* focussed on the use of set-valued attributes. The agent framework used to evaluate both algorithms mapped only a small number of terms from the fields in an email message to the set-valued attribute. These terms were selected by utilising simple statistical measurements about the frequency of various terms within the different email messages. *PIBPL* was designed to automatically select which terms should be used when attempting to classify new test instances. It utilised statistical weights to measure the relevance of each term, and to reject the low weighted terms. The results demonstrated that the removal of many of the terms from the set-valued attributes could significantly improve the performance of the algorithm.

The results obtained for *PIBPL* (Section 5.3) suggested that an approach for selecting terms based on the learning biases could result in superior results to methods that used naive selection heuristics. However, before further investigation on selection methods within *PIBPL* could proceed, a greater understanding of general selection methods for nearest neighbour learning was required. To achieve this, a number of different selection methods were described in Chapter 6, and systematically evaluated in Chapter 7.

The selection methods investigated were divided into four categories; the *Filter Method*, the *Wrapper Method*, the *Weighted Method* and the *Sub-space Approximation Method*. The principals behind each category were introduced in Chapter 3. Although many of the selection methods investigated have been used with other learning algorithms, some of the methods had not previously been tested with a nearest neighbour learning algorithm, and many algorithms had been evaluated within a single domain.

To better understand the differences between the different selection methods, they were evaluated using traditional single-valued attribute data. Some of the algorithms tested can be computationally complex, and as such unsuited for learning from high dimensional problems (such as in term selection when learning from documents). Thus, the motivation to use the single-valued attribute data sets, was to compare all the different selection algorithms across a selection of different, standardised domains.

A single filtering approach was evaluated as part of the *Filter Method* in Section 7.3. It utilised the information gain metric (Quinlan 1986) to greedily select attributes for each data set. The rule induction algorithm C4.5 (Quinlan 1993) was employed to perform this selection. The results confirmed that attribute selection could result in an increase in classification accuracy for the nearest neighbour learning algorithm with some data sets. However, further testing suggested that this method failed to identify all the relevant attributes in a data set, and in many cases selected irrelevant ones.

The *Wrapper Method* (Section 7.4) employed a search algorithm to locate optimal

(or sub-optimal) attribute subsets. The nearest neighbour algorithm was used as an evaluation function to score each search state. Four search algorithms were evaluated: two heuristic searches (*Forward Selection* and *Backward Elimination*), and two stochastic searches (*Simulated Annealing* and *Monte Carlo*). The *Forward Selection* search consistently found the smallest attribute subsets, but at the expense of classification accuracy. This behaviour may be due to the greedy search becoming trapped in local maxima. Although the *Backward Elimination* search resulted in the highest classification accuracies for several data sets, the size of the attribute subsets was larger than other searches. There was little difference in the classification accuracy or size of subsets with either stochastic approach, although the *Monte Carlo* search achieved a slightly higher average accuracy than the *Simulated Annealing* search.

The investigation of the *Weighted Method* (Section 7.5) focussed on three different strategies for utilising attribute weights within the nearest neighbour learning algorithm. The *Threshold Selection* approach utilised a threshold to determine whether or not attributes should be discarded. The *Weighted Distance Metric* did not explicitly select attributes; instead the distance metric was modified to utilise the weights when determining the nearest neighbour of a query instance. The third approach combined selection of attributes through thresholding with the weighted distance metric approach. This third approach was found to select the smallest subset of attributes for most data sets, but yielded the best gains in classification accuracy.

The *Sub-space Approximation Method* (Section 7.6) differed from the previous methods in that instances were mapped from one representation into an alternative, lower dimensional representation. Two algorithms were tested on numerical data, and both were found to be successful in reducing the number of dimensions required to represent the data. In addition, they achieved the highest average classification accuracy of all the algorithms tested. Further investigation suggested that this was not due to the elimination of irrelevant attributes, but to the elimination of redundant attributes.

8.2 Conclusions

This thesis makes a number of practical contributions towards an understanding of the way nearest neighbour learning algorithms can be employed within high dimensional domains, and how different dimensionality reduction techniques can be used to improve the classification accuracy of the learning algorithm.

8.2.1 Set-Valued Attributes

The set-valued attribute representation provides a means of concisely representing a number of different symbolic values associated with the same characteristic, or attribute. Previous representations utilised single-valued attributes, where an attribute was exclusively represented by a single value. However, this representation could not be used when several different values could equally be used to represent that attribute. The alternative was to utilise a binary vector to indicate the presence or absence of each value. However, this results in large vectors when the number of possible values is large. The set-valued attribute representation offers a solution to this problem; the different attributes can be used to represent different characteristics of a domain, whether or not the value of each attribute is mutually exclusive.

Although two different distance metrics have been proposed, there is little difference in the performance of each. The size of the sets can be reduced by removing the irrelevant members of each set by utilising relevance weights. This not only reduces the time taken to classify new instances, but can reduce the deterioration in classification accuracy due to the presence of irrelevant attributes.

8.2.2 Attribute Selection and Dimensionality Reduction

Many different attribute selection algorithms that improve the performance of learning algorithms across several dimensions already exist. They can be used to simplify the induced hypothesis from rule induction algorithms, or reduce the

number of dimensions that describe the instance space. The classification accuracy may improve due to the elimination of irrelevant or redundant attributes, and fewer training instances may be required to learn a concept. However, there have been few studies to date that have used the nearest neighbour algorithm to systematically compare the different approaches. This thesis discusses these issues, and presents an empirical comparison of a selection of different attribute selection approaches in the context of a nearest neighbour learning algorithm.

Many nearest neighbour learning algorithms have favoured the integration of weights within the distance metric. Although this can improve the classification accuracy of data containing several irrelevant attributes, a drop in accuracy can occur if no irrelevant attributes are present. However, weighted attributes can be used to complement other selection techniques, resulting in an improvement in classification accuracy at the expense of slightly larger attribute subsets. The sub-space approximation methods were found to be as successful at increasing the average classification accuracy as the weighted methods, but this was achieved with smaller attribute subsets.

8.3 Future Work

The work presented in this thesis represents an initial investigation of dimensionality reduction for nearest neighbour learning algorithms. It has demonstrated the suitability of set-valued attributes as a representation for learning, and a variety of different attribute selection and dimensionality reduction techniques have been explored. Although this work constitutes a significant contribution towards an understanding of these issues, there are many directions in which this work could now proceed.

The classification accuracy of the set based nearest neighbour algorithms may vary significantly depending the choice of distance metric used. The *Value Difference Metric* (Section 2.2.3) was chosen as it determines the distance between symbolic values on the basis of class distribution. Thus, the distance between sets

of symbols with similar class distributions will be small, even if the sets contain syntactically different symbols (Figure 2.4). However, if the *Overlap* metric was used, then the distance between such sets would be expected to rise. These and other symbolic distance metrics should be evaluated within a set-based nearest neighbour learning algorithm (such as *IBPL1*) to try to determine the optimal characteristics of a suitable distance metric for use with set-valued attributes

The VDM utilises weights to represent the relevance of each symbolic value (Equation 2.8). This weight was also used by *PIBPL* to eliminate irrelevant symbolic values from the set-valued attributes. However, a recent investigation has demonstrated that the weights have little impact on the classification accuracy of the VDM (Payne & Edwards 1998b), and that the performance of the VDM does not degrade when irrelevant attributes are added to a data set (see Appendix C). The VDM should be investigated further to determine why this distance metric is robust when irrelevant attributes are present in the data set.

The sub-space approximation method can be successfully applied to the task of dimensionality reduction for numeric data sets. However, this is not due to the elimination of irrelevant attributes present in the data set, as the inclusion of such attributes can lead to a degradation in classification accuracy. An attribute selection algorithm could be combined with the sub-space approximation algorithms (Section 6.5) to first eliminate irrelevant attributes, and then further reduce the dimensionality of the resulting attribute subset. These algorithms should also be investigated to determine if the lower rank sub spaces eliminate the effects of noise within the data set.

The four selection techniques presented in Section 6.3 utilised a leave-one-out cross validation strategy to identify the optimal sub-set of relevant attributes for the training data. However, when evaluated, some of the resulting classification accuracies were lower than expected. One possible explanation might be that the wrapper is *overfitting* the training data: i.e. whilst the subset of selected attributes might yield the optimal classification accuracy when cross validated with the training data, they may be too specific for the test data. An investigation is necessary to determine whether or not the attribute subset is overfitting the

training data, and if so whether or not this is due to the cross validation strategy used or the number of folds selected.

The *Threshold Selection* algorithm uses a threshold value to determine whether attribute should be discarded. Similarly, both CA-NN and CACP-NN reduce the dimensionality of the data set by reducing the rank of the approximated subspace. At present, these parameters are determined empirically. Some existing learning algorithms utilise n-fold cross validation techniques to identify suitable estimates (Kohavi & John 1995), or utilise a stopping criteria to identify partition points (Fayyad & Irani 1993). An automated method for reliably estimating values for the threshold and the rank of these algorithms would be extremely useful.

Appendix A

A Summary of the Attribute Selection Results

This Appendix contains a summary of the results obtained from the empirical study presented in Chapter 7. Tables A.1 and A.2 compare the classification accuracy of the nearest neighbour learning algorithm with different attribute selection algorithms. The average numbers of attributes selected by each method are presented in Tables A.3 and A.4.

	Data Sets	NN_{OM}	FSS_{OM}	SA_{OM}	MC_{OM}	BSE_{OM}	FNN_{OM}	ThS_{OM}	ωNN_{OM}	ωThS_{OM}
Real World	breast-cancer	70.73	76.15	64.62	73.21	66.38	71.84	75.70	65.37	75.70
	lung-cancer	40.00	60.00	40.00	30.00	63.33	71.67	50.00	46.67	50.00
	lymph	81.24	72.33	79.76	80.38	81.76	73.81	83.10	78.53	79.19
	primary-tumor	32.05	27.91	32.70	33.28	29.98	32.04	32.05	32.37	32.37
	promoters	77.00	83.82	72.37	74.37	77.09	82.82	83.55	79.91	83.64
	tic	80.90	65.33	80.88	84.14	86.12	80.90	80.90	82.88	82.88
	votes	92.44	94.51	94.05	94.28	93.35	94.06	94.29	94.74	95.43
	zoo	96.09	90.18	89.27	93.09	95.00	87.18	96.09	95.09	95.09
Artificial	led	69.50	69.50	67.50	69.50	69.50	69.50	69.50	65.50	65.50
	led +17	34.50	61.00	54.00	51.00	52.00	46.00	53.00	55.50	56.00
	monks-1	78.70	66.67	97.22	97.22	97.22	86.11	97.22	98.38	98.38
	monks-2	73.84	67.13	67.13	60.42	73.84	73.84	73.84	70.83	70.83
	monks-3	82.87	93.06	89.81	93.06	89.81	88.89	94.44	94.91	95.14
Average		69.99	71.35	71.49	71.84	75.03	73.74	75.67	73.90	75.40

Table A.1: Summary of classification accuracies for each of the experiments on UCI Symbolic Data Sets used in this study.

	Data Sets	NN_{EM}	FSS_{EM}	SA_{EM}	MC_{EM}	BSE_{EM}	FNN_{EM}	ThS_{EM}	ωNN_{EM}	ωThS_{EM}	CA	CACP
Real World	bupa	61.98	58.45	61.90	60.38	61.85	61.98	64.72	61.70	63.55	61.98	61.98
	ionosphere	87.17	88.35	90.83	90.64	89.20	92.60	90.38	91.78	92.32	90.90	91.19
	pima	70.99	64.85	66.79	67.96	68.65	70.99	70.99	71.25	71.25	70.99	70.99
	sonar	85.96	77.46	84.09	83.68	84.09	82.32	87.55	88.96	89.50	86.96	85.50
	wisconsin	95.90	95.32	95.31	95.03	94.59	95.90	96.77	96.64	96.92	97.36	96.19
	wdbc	95.40	95.04	95.94	96.11	96.46	95.43	95.40	95.59	95.93	96.65	96.29
	wpbc	69.06	70.11	66.28	71.17	66.44	70.50	73.17	70.06	72.61	71.61	73.06
	ecoli	80.59	77.54	76.65	76.36	76.36	79.71	81.18	81.49	81.49	80.59	81.82
	glass	68.09	70.41	71.46	71.00	72.27	68.09	73.36	72.00	72.91	68.09	70.00
	iris	96.16	98.13	98.13	98.13	95.98	98.13	98.13	96.79	98.04	96.16	96.70
	wine	94.86	94.93	94.38	94.79	94.31	96.04	96.04	97.15	97.71	97.08	97.64
	yeast	52.56	50.81	51.21	52.15	51.68	52.56	53.03	53.37	53.91	52.62	52.62
Artificial	balance	78.10	62.42	78.10	78.10	78.10	78.10	78.1	77.15	77.48	78.12	88.95
	waveform	73.33	66.00	73.33	70.00	73.00	73.00	76.67	76.67	76.67	81.00	79.33
	waveform-40	68.33	68.67	69.67	67.00	68.33	74.33	75.67	76.00	76.33	83.00	84.67
Average		78.57	75.90	78.27	78.17	78.09	79.31	80.74	80.44	81.11	80.87	81.80

Table A.2: Summary of classification accuracies for each of the experiments on UCI Numeric Data Sets used in this study.

	Data Sets	NN_{OM}	FSS_{OM}	SA_{OM}	MC_{OM}	BSE_{OM}	FNN_{OM}	ThS_{OM}	ωThS_{OM}
Real World	breast-cancer	9	2.5	3.7	3.0	6.9	2.8	3.5	3.5
	lung-cancer	56	3.6	16.4	24.9	16.5	2.9	37.4	37.4
	lymph	18	5.3	9.5	9.5	13.2	8.2	13.7	16.2
	primary-tumor	15	3.5	7.9	8.9	10.3	13.1	15.0	15.0
	promoters	57	4.5	26.0	28.6	50.2	4.7	34.0	31.0
	tic	9	1.0	6.3	7.3	7.0	9.0	9.0	9.0
	votes	16	2.8	5.6	5.6	10.0	4.8	2.6	8.4
	zoo	16	5.6	8.2	8.9	9.1	7.2	16.0	16.0
Artificial	led	7	7.0	6.9	7.0	7.0	7.0	7.0	7.0
	led+17	24	7.3	10.1	12.0	14.7	15.3	5.4	19.2
	monks-1	6	4.0	3.0	3.0	3.0	5.0	3.0	6.0
	monks-2	6	1.0	2.0	4.0	6.0	6.0	6.0	6.0
	monks-3	6	3.0	4.0	3.0	4.0	2.0	3.0	5.0
Average		18.85	3.93	8.43	9.67	12.15	6.77	11.97	13.82

Table A.3: Average size of attribute subsets selected for each of the UCI Symbolic Data Sets.

	Data Sets	NN_{EM}	FSS_{EM}	SA_{EM}	MC_{EM}	BSE_{EM}	FNN_{EM}	ThS_{EM}	ωThS_{EM}	CA	CACP
Real World	bupa	6	1.5	4.4	4.5	4.6	6.0	4.9	4.8	6	6
	ionosphere	34	5.2	13.5	14.7	21.9	9.6	17.6	17.6	22	11
	pima	8	3.2	4.7	4.3	7.5	8.0	8.0	8.0	8	8
	sonar	60	8.3	25.7	28.7	41.5	13.6	42.6	47.4	23	30
	wisconsin	9	5.0	5.3	5.5	6.5	5.7	7.4	8.6	6	3
	wdbc	30	5.1	14.6	14.5	19.6	7.8	30.0	20.6	5	16
	wpbc	33	1.2	14.9	15.2	26.1	13.4	1.2	1.0	16	15
	ecoli	7	6.3	5.9	6.4	6.4	5.3	6.2	6.2	7	5
	glass	9	5.4	5.5	5.4	5.7	8.8	6.0	7.8	8	8
	iris	4	2.0	2.1	2.0	2.3	2.0	2.0	2.1	4	3
	wine	13	3.9	6.8	7.7	8.8	3.7	8.3	9.3	6	6
	yeast	8	7.5	7.4	7.7	7.7	8.0	7.8	7.8	7	7
Artificial	balance	4	1.7	4.0	4.0	4.0	4.0	3.9	3.9	4	1
	waveform-21	21	5.8	11.4	12.2	18.0	16.1	18.4	21.0	3	3
	waveform-40	40	6.4	20.8	20.4	32.8	17.6	24.4	38.3	2	3
Average		19.07	4.57	9.80	10.21	14.23	8.64	12.58	13.63	8.47	8.33

Table A.4: Average size of attribute subsets selected for each of the UCI Numeric Data Sets.

Appendix B

Determining the Weight Update Coefficient

The weight function described in Section 6.4 is induced by evaluating the training set, and updating the weight for each attribute according to whether the attribute value pairs for the query and nearest neighbour are equal, and whether or not the classification of the query instance was correct. The way the weight is updated is dependent on the *weight update coefficient*, $\Delta Coeff$.

Various weight update coefficients were evaluated to determine their effects on the classification accuracy for the different UCI symbolic data sets. A 10-fold cross validation was performed to determine the classification accuracy of the *Overlap* metric when combined with the weights. Five coefficients were investigated: 0.1, 0.05, 0.02, 0.01 and 0.005. To provide a set of base-line comparisons, two additional distance metrics were used. The basic *Overlap* metric was used to determine the increase in classification accuracy when the induced weights were employed. In addition, a *Randomised Weighted Overlap* metric was evaluated. This metric was similar to the weighted Overlap metric, except that a random number generator was used to determine the weights.

The results of the evaluation are given in Table B.1. These results show the difference in classification accuracy with respect to the *Overlap* metric. The bottom

row of the table indicates the average differences in accuracy for each of the weight update coefficients. Although the 0.05 coefficient achieved the highest average accuracy, this was largely due to the results achieved for the *lung-cancer* data set. This result was significantly greater than that achieved with the *Overlap* metric if a one-tailed t-test was used with p=0.08. At the 5% level, the increase in accuracy was significant for only one data set (*votes*). However, a significant increase was achieved for two data sets with update coefficients of 0.02 and 0.01. The average increase in accuracy was higher for 0.02 coefficient, than 0.01 coefficient. If the results for the *lung-cancer* data set are omitted, then the 0.02 update coefficient achieves the highest average increase in accuracy.

The *Randomised Weighted Overlap* metric achieved a lower classification accuracy than the *Overlap Metric* for all the data sets tested, although the results were significant for only two of the data sets.

	0.1	0.05	0.02	0.01	0.005	random
breast-cancer	-3.20	-3.92	-4.29	-4.30	-3.93	-3.18
lung-cancer	3.33	6.67	0.00	0.00	0.00	-5.00
lymph	-6.19	-2.76	1.33	2.05	2.05	-0.14
primary-tumor	-1.76	0.03	0.02	0.02	0.02	-3.86
promoters	-0.09	2.73	5.73	3.73	3.73	-0.18
tic	1.56	1.66	1.98	2.71	2.81	-6.90
votes	1.61	2.75	2.29	1.82	1.36	0.01
zoo	-1.00	0.00	-1.00	-1.00	-1.00	-1.00
Average	-0.72	0.89	0.76	0.63	0.63	-2.53

Table B.1: Relative change in classification accuracy (with respect to the *Overlap* Metric) as the weight coefficient was varied.

Appendix C

Implicit Feature Selection with the Value Difference Metric

The Value Difference Metric differs from many other distance metrics in that the location of an instance within the instance space is not defined directly by the values of its attributes, but by the class conditional distributions of these values. The distributions vary from being skewed, where an attribute value appears in instances of only one class, to a uniform distribution, where the attribute value appears equally in instances of each class. In other words, attribute values with skewed distributions may be highly relevant to the target concept, and attribute values with a uniform distribution may be irrelevant. However, these distributions assume that each attribute value is independent of any other value for any of the attributes. The value weight component $\omega(i_a)$ provides some indication of the skew of the class distribution for an attribute value, and can be used to control the influence each attribute distance has on the final distance $vdm(i, j)$.

The inclusion of a weight within the VDM has been questioned by a number of studies. PEBLS (Cost & Salzberg 1993) is a NN learning algorithm which uses the Modified Value Difference Metric (MVDM). This distance metric is a variant of the VDM which omits $\omega(i_a)$, and it has been argued that the distance between two attribute values should be symmetrical, i.e. $vdm(i, j) = vdm(j, i)$. A recent study compared MVDM with the VDM, but concluded that there was no difference in

the classification accuracies of either metric over several data sets (Wettschereck, Aha, & Mohri 1997).

We have investigated the utility of $\omega(i_a)$, both as a component of the VDM, and when combined with another distance metric. Several symbolic data sets from the UCI Machine Learning Repository (Merz & Murphy 1996) were used to evaluate the performance of five distance metrics: three of which were based on class conditional probabilities (VDM, MVDM & VW_{OM}); and two which were used for baseline comparisons with other studies. The MVDM differs from the metric given in Section 2.6 in that the term $\omega(i_a)$ is omitted. In contrast, the VW_{OM} utilises the value weight (Equaton 2.8), but instead of using the attribute distance defined in (Equaton 2.7), the attribute distance for the Overlap metric (2.5) is used.

The *Weighted Distance Metric* (Section 6.4) and the simple Overlap metric (NN_{OM}) were included to provide a comparison of the VDM, MVDM and VW_{OM} with other distance metrics. ωNN_{OM} and VW_{OM} differ in that the weights used by the VW_{OM} are probabilistic and can be rapidly determined from the training set, whereas the weights in ωNN_{OM} are induced, and thus require a separate training stage.

Three hypotheses were investigated:

H$_1$ There is no difference between the performance of MVDM and VDM, i.e. $\omega(i_a)$ has no significant effect on the performance of the VDM.

H$_2$ The value weight component $\omega(i_a)$ of the VDM can be effectively utilised by the Overlap metric to improve performance in terms of accuracy. The performance of the VW_{OM} should be comparable to that of ωNN_{OM}, and both metrics should achieve better results (in general) than NN_{OM}.

H$_3$ The use of class conditional probabilities within the distance metric should improve the classification accuracy by reducing the effects of irrelevant attributes, i.e. the performance of the VDM, MVDM and VW_{OM} should not degrade in the presence of irrelevant attributes.

	NN_{OM}	ωNN_{OM}	VW_{OM}	MVDM	VDM
Breast-cancer	70.73	*66.44*	71.46	*67.51*	*67.16*
Lung-cancer	40.00	40.00	46.67	**68.33**	**65.00**
Lymphography	81.24	82.57	**83.24**	83.24	83.19
Primary-tumor	32.05	32.07	33.54	30.83	31.15
Promoters	77.00	**82.73**	**79.73**	**89.36**	**87.36**
Tic-tac-toe	80.90	82.88	*72.75*	**90.71**	**90.71**
Votes	92.44	**94.73**	**93.80**	94.51	**94.97**
Zoo	96.09	95.09	95.09	97.09	97.09

Table C.1: 10-fold cross validated classification accuracies.

To evaluate the performance of each of the distance metrics, a 10-fold cross validation (Kohavi 1995) was performed on a number of different UCI data sets. The results, given in Table C.1, list the classification accuracies achieved by each metric for each of the data sets. Results presented in bold were found to be significantly higher (p=*0.05*) than those achieved by the Overlap metric (NN_{OM}), whereas those in italics were significantly lower. A one-tailed paired t-test was used to determine this significance. Figure C.1 plots the difference in the results obtained by NN_{OM} and the other distance metrics.

There was no significant difference between the performance of the VDM and MVDM for any of the data sets, which appears to support H_1. The VDM succeeded in significantly improving the classification accuracy for four data sets (p=*0.05*), and succeeded in raising the accuracy (though not significantly) for two other data sets. The MVDM achieved similar success, except for the *Votes* data set, where the increase in accuracy became significant when p=*0.054*. The results for both distance metrics were significantly lower than NN_{OM} for only one data set (*Breast-Cancer*). These results demonstrate that the VDM (and MVDM) can achieve better classification accuracies than the Overlap metric. The accuracy of the MVDM was significantly higher than ωNN_{OM} for three data sets (*Lung-cancer*, *Promoters* and *Tic-tac-toe*).

The VW_{OM} also succeeded in raising the classification accuracy for six of the data sets, although the increase was only significant for three. A significant increase

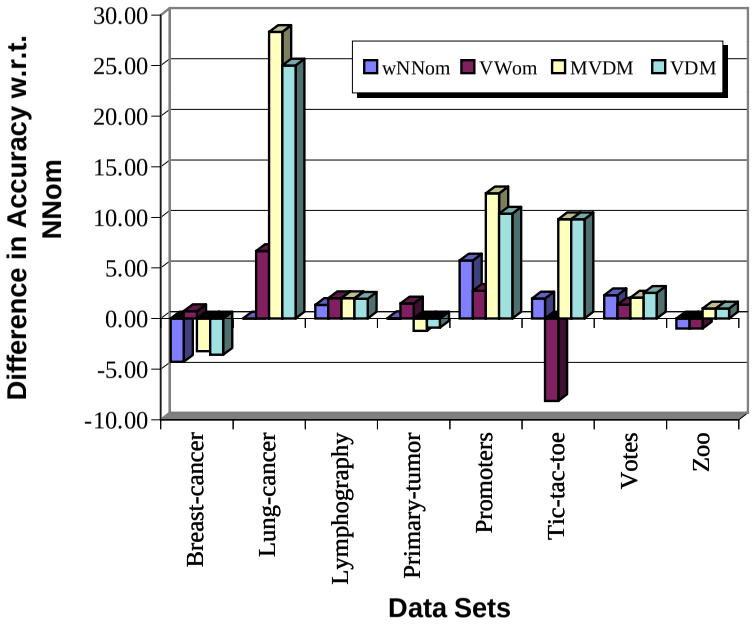


Figure C.1: Comparison of the 10 fold cross validated classification accuracies of the distance metrics relative to the Overlap metric (NN_{OM}).

in accuracy was also achieved by $\omega\mathrm{NN}_{OM}$ for two of the same three data sets. As $\omega\mathrm{NN}_{OM}$ has previously been demonstrated to be robust in the presence of irrelevant attributes, and given that a similar increase in accuracy can be observed for the VW_{OM}, this suggests that the value weight $\omega(i_a)$ can be used to limit the impact of irrelevant attributes on the classification accuracy. This appears to support H_2.

Although the distance metrics based on the VDM performed well with the data sets, a further investigation was required to determine if the performance of these metrics would degrade in the presence of irrelevant attributes. For this reason, the metrics were evaluated on the 24-attribute LED display problem. This problem contains seven binary valued attributes (corresponding to the different segments within an LED seven segment numeric display), and an additional seventeen irrelevant attributes (Aha, Kibler, & Albert 1991). If this number of additional attributes is varied, it is possible to observe the effect of irrelevant attributes on different learning algorithms. Data sets were constructed containing 200 randomly generated instances with 10% noise (i.e. each attribute value had a 10% chance

of being inverted). The number of irrelevant attributes was varied from zero to seventeen, and each test was repeated ten times. The results are plotted in Figure C.2.

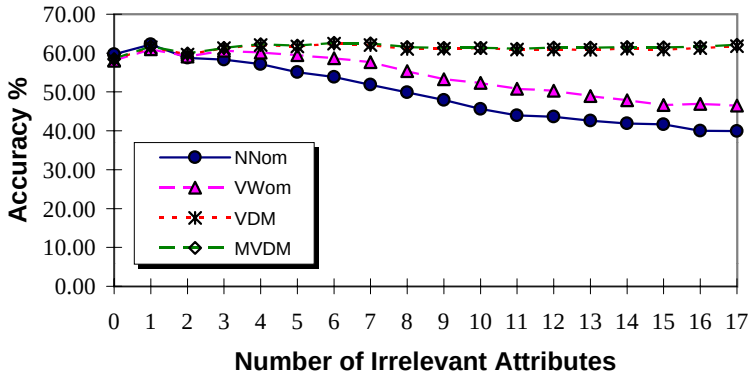


Figure C.2: LED artificial results.

As the number of irrelevant attributes increased, the performance of the Overlap metric (NN_{OM}) fell from 59.7% to 40.0%. There was a similar degradation in the performance of the VW_{OM}, although this degradation was not as acute as NN_{OM}, and the classification accuracy of the VW_{OM} was significantly higher than that of NN_{OM} when three or more irrelevant attributes were present. The VDM and MVDM showed no signs of degradation as the number of irrelevant attributes increased. These results support both H_2 and H_3, though it would appear that $\omega(i_a)$ succeeds only in reducing the impact of the irrelevant attributes, not eliminating their effects.

Conclusions

The Value Difference Metric is an alternative symbolic distance metric which can be successfully applied to classification problems containing irrelevant attributes. The distance metric utilises a set of value weights, which can be determined 'on the fly' from the training data. These value weights modify the distance between attribute values such that the distances between class discriminant values are augmented, but otherwise diminished. The exclusion of these value weights appears

to have no effect on the performance of the VDM. However, if combined with the Overlap metric, the value weights improve the performance of the distance metric (in terms of accuracy) on data containing irrelevant attributes. This increase in performance is comparable to that achieved when attribute weights are induced, and utilised by the Overlap metric. However, the value weights have the advantage that no training is required.

Bibliography

Aha, D. (1989). Incremental Instance-Based Learning of Independent and Graded Concept Descriptions. In *Proceedings of the 6th International Machine Learning Workshop*, pp. 387–392.

Aha, D. (1990). *A Study of Instance-Based Algorithms for Supervised Learning Tasks.* Ph. D. thesis, University of California, Irvine.

Aha, D. (1992a). Generalising from Case Studies: A Case Study. In *Proceedings of the 9th International Conference on Machine Learning*, pp. 1–10. San Francisco, CA:Morgan Kaufmann.

Aha, D. (1992b). Tolerating Noisy, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms. *International Journal of Man-Machine Studies 36*, 267–287.

Aha, D. and Bankert, R. (1994). Feature Selection for Case-Based Classification of Cloud Types: An Empirical Comparison. In *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning.*, pp. 106–112. Menlo Park, CA:AAAI Press.

Aha, D. and Bankert, R. (1995). A Comparative Evaluation of Sequential Feature Selection Algorithms. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pp. 1–7. Menlo Park, CA:AAAI Press.

Aha, D. and Kibler, D. (1989). Noise-Tolerant Instance-Based Learning Algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, IJCAI-89*, pp. 794–799.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-Based Learning Algorithms. *Machine Learning 6*, 37–66.

Albert, M. and Aha, D. (1991). Analyses of Instance-Based Learning Algorithms. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pp. 553–558.

Almuallim, H. and Dietterich, T. (1991). Learning With Many Irrelevant Features. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pp. 547–552. MIT Press.

Anderberg, M. (1973). *Cluster Analysis for Applications.* New York:Academic Press.

Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*. Menlo Park, CA:AAAI Press.

Bala, J., Huang, J., Vafaie, H., DeJong, K., and Wechsler, H. (1995). Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 719–724. San Mateo, CA:Morgan Kaufmann.

Balabanović, M., Shoham, Y., and Yun, Y. (1995). An Adaptive Agent for Automated Web Browsing. *Journal of Image Representation and Visual Communication 6*(4).

Bareiss, E. and Porter, B. (1987). Protos: An Exemplar-Based Learning Apprentice. In *Proceedings of the 4th International Workshop on Machine Learning*, pp. 12–23.

Barsalou, L. (1985). Ideas, Central Tendency, and Frequency of Instantiation as Determinants of Graded Structures in Categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition 11*(4), 629–654.

Bayer, D. (1995). A Learning Agent for Resource Discovery on the World Wide Web. MSc Thesis, Department of Computing Science, University of Aberdeen, Scotland.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour.* New Jersey:Princeton University Press.

Berry, M. and Fierro, R. (1996). Low-Rank Orthogonal Decompositions for Information Retrieval Applications. *Numerical Linear Algebra with Applications 1*(1), 1–27.

Biberman, Y. (1994). A Context Similarity Measure. In *Proceedings of the 7th European Conference on Machine Learning (ECML94)*, pp. 49–63. Berlin, Germany:Springer-Verlag.

Biberman, Y. (1995). The Role of Prototypicality in Exemplar-Based Learning. In *Proceedings of the 8th European Conference on Machine Learning*, pp. 77–91. Berlin, Germany:Springer-Verlag.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford, UK:Clarendon Press.

Boone, G. (1998). Concept Features in Re:Agent, an Intelligent Email Agent. In *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pp. 141–148. New York, NY:ACM Press.

Bradshaw, G. (1987). Learning About Speech Sounds: The NEXUS Project. In *Proceedings of the 4th International Workshop on Machine Learning*, pp. 1–11.

Breiman, L., Freidman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA:Wadsworth International Group.

Cardie, C. (1993). Using Decision Trees to Improve Case-Based Learning. In *Proceedings of the 10th International Conference on Machine Learning*, pp. 25–32. San Francisco, CA:Morgan Kaufmann.

Caruana, R. and Freitag, D. (1994). Greedy Attribute Selection. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 28–36. San Francisco, CA:Morgan Kaufmann.

Cherkauer, K. and Shavlik, D. (1996). Growing Simpler Decision Trees to Facilitate Knowledge Discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 315–318. Menlo Park, CA:AAAI Press.

Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning 3*, 261–283.

Cohen, W. (1995). Fast Effective Rule Induction. In *Proceedings of the 12th International Machine Learning Conference (ML95)*, pp. 115–123. San Francisco, CA:Morgan Kaufmann.

Cohen, W. (1996a). Learning Rules that Classify E-Mail. In *Machine Learning in Information Access: Papers from the 1996 AAAI Spring Symposium*, pp. 18–25. Menlo Park, CA:AAAI Press.

Cohen, W. (1996b). Learning Trees and Rules with Set-valued Features. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 709–716. Menlo Park, CA:AAAI Press.

Cost, S. and Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning 10*, 57–78.

Cover, T. and Hart, P. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory IT-13*(1), 21–27.

Creecy, R., Masand, B., Smith, S., and Waltz, D. (1992). Trading Mips and Memory for Knowledge Engineering. *Communications of the ACM 35*(8), 48–64.

Daelemans, W., Gillis, S., and Durieux, G. (1994). Skousen's Analogical Modeling Algorithm: a Comparison with Lazy Learning. In D. Jones (Ed.), *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, pp. 1–7. UMIST: Manchester.

Dasarathy, B. V. (1991). *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, California:IEEE Computer Society Press.

Datta, P. and Kibler, D. (1995). Learning Prototypical Concept Descriptions. In *Proceedings of the 12th International Conference on Machine Learning*, pp. 158–166. San Francisco, CA:Morgan Kaufmann.

Datta, P. and Kibler, D. (1997). Learning Symbolic Prototypes. In *Proceedings of the 14th International Conference on Machine Learning*, pp. 75–82. San Francisco, CA:Morgan Kaufmann.

De Mántaras, R. (1991). A Distance-Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning 6*, 81–92.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Dent, L., Boticario, J., McDermott, J., Mitchell, T., and Zabowski, D. (1992). A Personal Learning Apprentice. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 96–103.

Devijer, P. and Kittler, J. (1982). *Pattern Recognition: A statistical approach.* Prentice Hall International.

Domingos, P. (1996). Unifying Instance-Based and Rule-Based Induction. *Machine Learning 24*, 141–168.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* John Wiley and Sons.

Dudani, S. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics SMC-6*(4), 325–327.

Edwards, P., Bayer, D., Green, C., and Payne, T. (1996). Experience with Learning Agents which Manage Internet-Based Information. In *Machine Learning in Information Access: Papers from the 1996 AAAI Spring Symposium*, pp. 31–40. Menlo Park, CA:AAAI Press.

Emde, W. and Wettschereck, D. (1996). Relational Instance-Based Learning. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 122–130. San Francisco, CA:Morgan Kaufmann.

Everitt, B. (1974). *Cluster Analysis.* New York:Halsted Press.

Fayyad, U. and Irani, K. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1022–1027. San Mateo, CA:Morgan Kaufmann.

Fix, E. and Hodges Jr., J. (1951). Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties. In *Project 21-49-004 Report No. 4*, pp. 261–279. USAF School of Aviation Medicine, Randolph Field, Tex.

Fraleigh, J. and Beauregard, R. (1995). *Linear Algebra.* Menlo Park, CA:Addison-Wesley.

Fukunaga, K. and Flick, T. (1984). An Optimal Global Nearest Neighbour Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6*(3), 314–318.

Green, C. and Edwards, P. (1996). Using Machine Learning to Enhance Software Tools for Internet Information Management. In *Proceedings of the AAAI-96 Workshop on Internet-based Information Systems.*

Greenacre, M. (1984). *Theory and Applications of Correspondence Analysis.* London, UK:Academic Press.

John, G., Kohavi, R., and Pfleger, K. (1994). Irrelevant Features and the Subset Selection Problem. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 121–129. San Francisco, CA:Morgan Kaufmann.

Kibler, D. and Aha, D. (1987). Learning Representative Exemplars of Concepts: An Initial Case Study. In *Proceedings of the 4th International Workshop on Machine Learning*, pp. 24–30.

Kira, K. and Rendell, L. (1992a). A Practical Approach to Feature Selection. In *Proceedings of the 9th International Workshop on Machine Learning*, pp. 249–256. San Mateo, CA:Morgan Kaufmann.

Kira, K. and Rendell, L. (1992b). The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 129–134. MIT Press.

Kirkpatrick, S., Gelatt, C., and Vecchi, P. (1983). Optimization by Simulated Annealing. *Science 220*, 671–680.

Kohavi, R. (1994). Feature Subset Selection as Search with Probabilistic Estimates. In *AAAI Fall Symposium on Relevance*, pp. 121–126. Menlo Park, CA:AAAI Press.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1137–1145. San Mateo, CA:Morgan Kaufmann.

Kohavi, R. and John, G. (1995). Automatic Parameter Selection by Minimizing Estimated Error. In *Proceedings of the 12th International Conference on Machine Learning (ML95)*, pp. 304–312. San Francisco, CA:Morgan Kaufmann.

Kohavi, R., Langley, P., and Yun, Y. (1997). The Utility of Feature Weighting in Nearest-Neighbour Algorithms. In *Proceedings of the 9th European Conference on Machine Learning (ECML97)*. Berlin, Germany:Springer-Verlag.

Kohavi, R. and Sommerfield, D. (1995). Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pp. 192–197.

Kononenko, I. (1994). Estimating Attributes: Analysis and Extensions of RELIEF. In *Proceedings of the 7th European Conference on Machine Learning*, pp. 171–182. Berlin, Heidelberg:Springer-Verlag.

Kozierok, R. and Maes, P. (1993). A Learning Interface Agent for Scheduling Meetings. In *Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces*, pp. 81–88. New York, New York:ACM Press.

Kubat, M., Flotzinger, D., and Pfurtscheller, G. (1993). Discovering Patterns in EEG-Signals: Comparative Study of a Few Methods. In *Proceedings of the 6th European Conference on Machine Learning*, pp. 366–371. Berlin, Heidelberg:Springer-Verlag.

Kubat, M., Holte, R., and Matwin, S. (1997). Learning when Negative Examples Abound. In *Proceedings of the 9th European Conference on Machine Learning (ECML97)*, pp. 146–153. Berlin, Germany:Springer-Verlag.

Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th International Machine Learning Conference (ML95)*, pp. 331–339. San Francisco, CA:Morgan Kaufmann.

Langley, P. and Iba, W. (1993). Average-case Analysis of a Nearest Neighbor Algorithm. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 889–894. San Mateo, CA:Morgan Kaufmann.

Langley, P. and Sage, S. (1994a). Induction of Selective Bayesian Classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pp. 399–406. Seattle, WA:Morgan Kaufmann.

Langley, P. and Sage, S. (1994b). Oblivious Decision Trees and Abstract Cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pp. 113–117. Seattle, WA:AAAI Press.

Lewis, D. (1992). *Representation and Learning in Information Retrieval*. Ph. D. thesis, University of Massachusetts, MA.

Lewis, D. and Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, pp. 81–93.

Littlestone, N. (1988). Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning 2*, 285–318.

Liu, H. and Setiono, R. (1996a). A Probabilistic Approach to Feature Selection - A Filter Solution. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 319–327. San Francisco, CA:Morgan Kaufmann.

Liu, H. and Setiono, R. (1996b). Feature Selection and Classification - A Probabilistic Wrapper Approach. In *The 9th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA-AIE'96)*, pp. 419–424.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM 37*(7), 30–40.

Malone, T., Grant, K., Turbak, F., Brobst, S., and Cohen, M. (1987). Intelligent Information-Sharing Systems. *Communications of the ACM 30*(5), 390–402.

McElligott, M. and Sorensen, H. (1994). An Evolutionary Connectionist Approach to Personal Information Filtering. In *Irish Neural Networks Conference '94*. University College Dublin.

Merz, C. and Murphy, P. (1996). UCI repository of machine learning databases.

Metral, M. (1993). Design of a Generic Learning Interface Agent. BSc Thesis, Department of Electrical Engineering and Computer Science, MIT.

Michie, D., Spiegelhalter, D., and Taylor, C. (Eds.) (1994). *Machine Learning, Neural and Statistical Classification.* UK:Ellis Horwood Ltd.

Mitchell, T. (1997). *Machine Learning.* New York:The McGraw-Hill Companies, Inc.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., and Zabowski, D. (1994). Experience with a Learning Personal Assistant. *Communications of the ACM 37*(7), 81–91.

Moore, A. and Lee, M. (1994). Efficient Algorithms for Minimizing Cross Validation Error. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 190–198. San Francisco, CA:Morgan Kaufmann.

Moulinier, I. (1996). A Framework for Comparing Text Categorisation Approaches. In *Machine Learning in Information Access: Papers from the 1996 AAAI Spring Symposium*, pp. 61–68. Menlo Park, CA:AAAI Press.

Moulinier, I. (1997). Feature Selection: A Useful Preprocessing Step. In *Proceedings of the 19th Annual BCS-IRSG Colloquium on IR Research*, pp. 140–158.

Nosofsky, R. (1984). Choice, Similarity, and the Context Theory of Classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition 10*(1), 104–114.

Okamoto, S. and Yugami, N. (1996). Theoretical Analysis of the Nearest Neighbor Classifier in Noisy Domains. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 355–363. San Francisco, CA:Morgan Kaufmann.

Pannu, A. and Sycara, K. (1996). A Learning Personal Agent for Text Filtering and Notification. In *Proceedings of the International Conference on Knowledge Based Systems (KBCS96)*, pp. 255–266.

Payne, T. (1994). Learning Email Filtering Rules with Magi, A Mail Agent Interface. MSc Thesis, Department of Computing Science, University of Aberdeen, Scotland.

Payne, T. and Edwards, P. (1995). Learning Mechanisms for Information Filtering Agents. In J. Nealon and N. Taylor (Eds.), *Proceedings of the UK Intelligent Agents Workshop*, pp. 163–183. Oxford, UK:SGES Publications.

Payne, T. and Edwards, P. (1996). A Survey of Feature Selection Methods. Unpublished Draft.

Payne, T. and Edwards, P. (1997). Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence 11*(1), 1–32.

Payne, T. and Edwards, P. (1998a). Dimensionality Reduction through Correspondence Analysis. Unpublished Draft.

Payne, T. and Edwards, P. (1998b). Implicit Feature Selection with the Value Difference Metric. In *ECAI 98 Conference Proceeedings*, pp. 450–454.

Payne, T., Edwards, P., and Green, C. (1995). Experience with Rule Induction and k-Nearest Neighbour Methods for Interface Agents that Learn. In *ML95 Workshop on Agents that Learn from Other Agents*. http://www.cs.wisc.edu/~shavlik/ml95w1/procs.html.

Payne, T., Edwards, P., and Green, C. (1997). Experience with Rule Induction and k-Nearest Neighbour Methods for Interface Agents that Learn. *IEEE Transactions on Knowledge and Data Engineering 9*(2), 329–335. *Presented at the ML95 Workshop on Agents that Learn from Other Agents.*

Pazzani, M., Muramatsu, J., and Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. In *Machine Learning in Information Access: Papers from the 1996 AAAI Spring Symposium*, pp. 69–77. Menlo Park, CA:AAAI Press.

Piatetsky-Shapiro, G., Brachman, R., Khabaza, T., Kloesgen, W., and Simoudis, E. (1996). An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96).*, pp. 89–95. Menlo Park, CA:AAAI Press.

Pomerleau, D. (1995). RALPH: Rapidly Adapting Lateral Position Holder. In *IEEE Symposium on Intelligent Vehicles*. Detroit, MI.

Porter, M. (1980). An Algorithm for Suffix Stripping. *Program (Automated Library and Information Systems) 14*(3), 130–137.

Press, W. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press.

Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning 1*, 81–106.

Quinlan, J. (1993). *C4.5 Programs for Machine Learning.* San Mateo, CA:Morgan Kaufmann.

Rachlin, J., Kasif, S., Salzberg, S., and Aha, D. (1994). Towards a Better Understanding of Memory-Based Reasoning Systems. In *Proceedings of the 11th International Machine Learning Conference (ML94)*, pp. 242–250. San Francisco, CA:Morgan Kaufmann.

Richeldi, M. and Lanzi, P. (1996). Performing Effective Feature Selection by Investigating the Deep Structure of the Data. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 379–382. Menlo Park, CA:AAAI Press.

Rocchio Jr, J. (1971). Relevance Feedback in Information Retrieval. In G. Salton (Ed.), *The Smart Retrieval System*, pp. 313–323. Englewood Cliffs, NJ:Prentice Hall Inc.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning Internal Representations by Error Propagation. In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing*, Volume 1, pp. 318–362. Cambridge, MA:MIT Press.

Salton, G. and Buckley, C. (1987). Term Weighting Approaches in Automatic Text Retrieval. Technical Report 87-881, Department of Computer Science, Cornell University, Ithaca, NY.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

Salzberg, S. (1991a). A Nearest Hyperrectangle Learning Method. *Machine Learning 6*, 251–276.

Salzberg, S. (1991b). Distance Metrics for Instance-Based Learning. In *ISMIS'91 6th International Symposium, Methodologies for Intelligent Systems*, pp. 399–408.

Salzberg, S. (1992). Improving Classification Methods via Feature Selection. Technical Report TR JHU-92/12, Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218.

Schütze, H., Hull, D., and Pedersen, J. (1995). A Comparison of Classifiers and Document Representations for the Routing Problem. In *18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pp. 229–237. New York, New York:ACM Press.

Seigler, R. (1976). Three Aspects of Cognitive Development. *Cognitive Psychology 8*, 481–520.

Shepherd, B. (1983). An Appraisal of a Decision Tree approach to Image Classification. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 473–475. Los Altos, CA:William Kaufmann.

Sheth, B. (1994). A Learning Approach to Personalized Information Filtering. Master's Thesis, Department of Electrical Engineering and Computer Science, MIT.

Singh, M. and Provan, G. (1995). A Comparison of Induction Algorithms for Selective and non-Selective Bayesian Classifiers. In *Proceedings of the 12th International Conference on Machine Learning*, pp. 497–505. San Francisco, CA:Morgan Kaufmann.

Singh, M. and Provan, G. (1996). Efficient Learning of Selective Bayesian Network Classifiers. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 453–461. San Francisco, CA:Morgan Kaufmann.

Skalak, D. (1994). Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 293–301. San Francisco, CA:Morgan Kaufmann.

Stanfill, C. and Waltz, D. (1986). Toward Memory-Based Reasoning. *Communications of the ACM 29*(12), 1213–1228.

Thrun, S., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S., Fisher, D., Hamann, R., Kaufman, K., Keller,

S., Kononenko, I., Kreuziger, J., Michalski, R., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., and Zhang, J. (1991). The MONK's Problems - A Performance Comparison of Different Learning Algorithms. Technical Report CMU-CS-91-197, School of Computer Science,Carnegie Mellon University,PA.

Ting, K. (1994). Discretization of Continuous-Valued Attributes and Instance-Based Learning. Technical Report TR #941, Basser Department of Computer Science, University of Sydney, NSW 2006, Australia.

Tomek, I. (1976). A Generalization of the k-NN Rule. *IEEE Transactions on Systems, Man, and Cybernetics SMC-6*(2), 121–126.

Vafaie, H. and De Jong, K. (1994). Improving a Rule Induction System using Genetic Algorithms. In R. Michalski and G. Tecuci (Eds.), *Machine Learning: A Multistrategy Approach. Vol 4*, pp. 453–469. San Francisco, CA:Morgan Kaufmann.

Weiner, E., Pedersen, J., and Weigend, A. (1995). A Neural Network Approach to Topic Spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, pp. 317–332.

Weiss, S. and Kapouleas, I. (1989). An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 781–787. San Mateo, CA:Morgan Kaufmann.

Wettschereck, D. (1994). *A Study of Distance-Based Machine Learning Algorithms.* Ph. D. thesis, Oregon State University.

Wettschereck, D., Aha, D., and Mohri, T. (1997). A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. *Artificial Intelligence Review 11*(1-5), 273–314. Special Issue on Lazy Learning.

Wettschereck, D. and Dietterich, T. (1995). An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. *Machine Learning 19*, 5–28.

Wilson, D. and Martinez, T. (1997). Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research 6*, 1–34.

Wu, C., Berry, M., Shivakumar, S., and McLarty, J. (1995). Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with Singular Value Decomposition. *Machine Learning 21*, 177–193.

Yang, Y. and Pedersen, J. (1997). A Comparative Study on Feature Selection in Text Categorisation. In *Proceedings of the 14th International Conference on Machine Learning.*, pp. 412–420. San Francisco, CA:Morgan Kaufmann.

Zhang, J. (1992). Selecting Typical Instances in Instance-Based Learning. In *Proceedings of the 9th International Machine Learning Workshop*, pp. 470–479.