INSTANCE-BASED LEARNING: Nearest Neighbour with Generalisation

A thesis

submitted to the

Department of Computer Science

in partial fulfilment

of the requirements for the

degree of

Master of Science

by

Brent Martin

Department of Computer Science
University of Waikato
Hamilton, New Zealand

March, 1995

Abstract

Instance-based learning is a machine learning method that classifies new examples by comparing them to those already seen and in memory. There are two types of instance-based learning; nearest neighbour and case-based reasoning. Of these two methods, nearest neighbour fell into disfavour during the 1980s, but regained popularity recently due to its simplicity and ease of implementation.

Nearest neighbour learning is not without problems. It is difficult to define a distance function that works well for both discrete and continuous attributes. Noise and irrelevant attributes also pose problems. Finally, the specificity bias adopted by instance-based learning, while often an advantage, can over-represent small rules at the expense of more general concepts, leading to a marked decrease in classification performance for some domains.

Generalised exemplars offer a solution. Examples that share the same class are grouped together, and so represent large rules more fully. This reduces the role of the distance function to determining the class when no rule covers the new example, which reduces the number of classification errors that result from inaccuracies of the distance function, and increases the influence of large rules while still representing small ones.

This thesis investigates non-nested generalised exemplars as a way of improving the performance of nearest neighbour. The method is tested using benchmark domains and the results compared with documented results for ungeneralised exemplars, nested generalised exemplars, rule induction methods and a composite rule induction and nearest neighbour learner. The benefits of generalisation are isolated and the performance improvement measured. The results show that non-nested generalisation of exemplars improves the classification performance of nearest neighbour systems and reduces classification time.

Acknowledgments

This research was motivated in part by the desire to further the good work of several individuals, Steven Salzberg, David Aha, Rob Holte and Dietrich Wettschereck, all of who have provided help over the past year. Grateful thanks must go to my supervisor, Ian Witten, for giving me the freedom to pursue my studies independently while always being there when needed, and to my partner, Suky, for her unending tolerance and support. Thank you also to the other machine learning students at the University of Waikato, for your valuable assistance and friendship.

Table of Contents

List of Tables

List of Figures

1 Introduction

The computer provides the means for storing huge amounts of data in a form that allows fast random retrieval. In addition to providing a convenient method for recording past events, this technology opens a new possibility; using historic data to aid in future decisions. Where a doctor once had to rely on his memory of previous cases, he may now search the online history for previously seen examples of a particular disease, and use the information recorded to suggest treatment for a new patient. A credit card company, suspicious of a new applicant's ability to pay their bills, can recall previous customers with common features in their background to help determine whether or not this customer is likely to default on their payments. Meteorological service providers can look up past instances of similar weather patterns to help forecast the next day's weather.

This method of prediction by recall is only feasible if the number of examples in the database that match the new case is fairly small. A doctor recalling leukemia patient information will have great difficulty assimilating the relevant cases if there are more than a few of them that match the current patient, particularly if they contradict each other. Also, this form of prediction only works if features in the current example exactly match those in the historic cases. Recall is difficult for problem domains where the information stored is numeric, because the number of possible values for each feature is very large, and so exact matches are less likely to occur.

Machine learning addresses this problem by assimilating historic data into a knowledge base that can be used to perform some task in the future. The user presents the system with a dataset of past cases, together with a feature that it must learn how to predict. The system then uses this dataset to learn how to classify future examples. A new patient is diagnosed by presenting his particulars to the system, which uses its knowledge base to predict the diagnosis.

Another use for machine learning is summarising data. Given a large quantity of raw data, machine learning methods such as conceptual clustering group the data into clumps of examples described by the feature values that they have in common. A dataset generalised in this manner is represented in a much more compact form. The generalised data may be both understandable by people and useable by the computer for answering high-level questions (Kolodner, 1984).

There are many different types of machine learning. Neural networks were an early approach. Hand-crafted to fit a particular problem, they use numeric functions to weight each connection of the network. The user presents new examples to the sysem, causing the weights to alter. The final state of the output nodes determines the result. Although their usefulness is limited due to the difficulty of determining the best topology for a given problem, much research is still carried out in this area as it is thought that they learn in a similar way to neurons in the human brain (Moorhead *et al*, 1989).

Induction of decision trees is another commonly used technique. C4.5 (Quinlan, 1993) is a popular example that is often used as the benchmark for testing the classification performance of new methods. The decision trees it induces, while not often intelligible to people, prove to be efficient classifiers. Researchers have tested C4.5 using a wide variety of real data with much success, demonstrating a high degree of generality. There are many rule-inducing systems that are similar, but generate production rules instead of decision trees.

Instance-based learning is a more recent development. Psychologists studying the way that people use memory to perform tasks conclude that we often recall past experiences to guide us to the solution to new problems (Kolodner, 1984). Instance-based learners do this by determining which case in memory is the most similar to the new situation. Some are very complex, using many indexes to guide the search for the best case (Bareiss and Porter, 1988), while others use a metric to measure similarity (Kibler and Aha, 1987). Instance-based learners have the advantage over rule and decision tree inducing systems that they work well when the target concepts are poorly represented by the training data.

Both induction systems and instance-based methods have strengths and weaknesses. Neither offers excellent performance across the complete spectrum of problem domains, although there are problems that each excels at. This thesis examines the possibility of combining instance-based learning with generalisation. The approach investigated—non-nested generalisation of exemplars—retains the strengths of instance-based learning, but adds the benefits of partial generalisation of the data. The result is a practical method that offers excellent classification performance over a wide variety of domains.

The remainder of this chapter describes instance-based learning and generalisation in more detail, and summarises how they can be combined.

1.1 Instance-based learning

Instance-based learners are “lazy” in the sense that they perform little work when learning from the dataset, but expend more effort classifying new examples. The simplest method, nearest neighbour, performs no work at all when learning; it stores all examples in memory verbatim. Effort is transferred to classification time, when the system decides which example in memory to use to classify the new one. Case-based reasoning systems perform a small amount of work indexing new cases, resulting in a reduction in classification effort. The examples stored in memory are called *exemplars*, and are retained in an *exemplar database*.

Instance-based learners receive new examples incrementally, giving them the freedom to learn over time, and so the set of instances in memory continues to grow. If allowed to continue indefinitely, the exemplar database eventually becomes too large to use, either because it exceeds memory capacity, or because the time taken to classify new examples becomes prohibitively long. It is therefore desirable to prune the exemplar database.

The advantage of instance-based learners is that they are able to learn quickly from a very small dataset. Whereas rule induction methods require a reasonable representation of each rule before they can be induced, an instance-based learner can begin to make useful predictions from as little as one example per class. Classification performance often exceeds 75% of the maximum possible after accepting only 25% of a complete data set. The approach therefore works well when limited data is available.

Another advantage of instance-based learners is that they work well for numeric data. Not only can they use continuous valued features, they can also predict numeric-valued classes. Induction systems can do this only if they partition the class into a small number of discrete values, because they try to represent the problem using a small number of concepts. Instance-based learners retain every example as a separate concept, and so can represent a larger number of classes.

1.2 Generalisation and rule induction

In contrast, rule induction systems are “eager.” They expend much effort during learning, by generalising the input dataset into a small set of decision rules. They then determine the class of a new example by testing each rule to see which one the new case satisfies. Two problems may arise when trying to do this: the new example may either satisfy more than one rule, or it may satisfy none. A heuristic is adopted in each of these situations.

Rule induction methods offer two advantages over instance-based learners. First, the evaluation of rules against a new example is computationally less expensive than the calculation of a real-valued distance function, and so can be performed more rapidly. Second, the resulting rule set is much smaller than the original dataset. Rule induction systems can therefore compress large datasets into a more useable form.

Induction systems generalise the input dataset by looking for the features or feature values that most strongly classify the examples, and producing rules using these features. Examples that satisfy each newly induced rule are discarded and the process is repeated until no more examples remain. A consequence of this is that the system may overlook a rule that covers a small portion of the input dataset because its examples also satisfy a larger rule, and so are discarded before the small one is induced. Section 2.5 examines this problem more fully.

1.3 Generalised instances

A major problem of instance-based learners is that classification time increases as more examples are added to memory. However, assimilating new examples is the only way to continue the learning process. Methods of pruning the exemplar database have been explored, but generally lead to a loss of classification performance (Aha, 1992).

Generalised exemplars provide an alternative solution to this problem. Rather than storing all examples verbatim, examples are merged, reducing the number in memory. As new examples are added, they may either be incorporated into existing generalised exemplars, or discarded completely if already covered by a generalised exemplar. Growth of the exemplar database is rapid at first, but decreases over time until the size becomes fairly static, with new examples resulting in minor changes to, rather than growth of, the exemplar database. By preventing overgeneralisation, this reduction in database size should not come at the expense of classification performance.

Instance-based learners perform well in certain domains, particularly those where the training data poorly represents the concepts to be learned. Induction systems, on the other hand, excel when concepts are strongly represented. Hybrid learners attempt to overcome this problem by applying more than one method. However, it is difficult to know when to apply each one. Generalised exemplars are an alternative way of combining induction with instance-based learning in a form that does not diminish the strengths of each approach. Because exemplar generalisation means that for every new classification both instance-based learning and rule induction are being applied, the merge of these two methods becomes seamless; in some cases a rule determines the class, while in others it comes from the nearest exemplar. Between these two extremes a partially satisfied rule is the closest exemplar, and so determines the class. Generalising exemplars should therefore improve the performance of instance-based learning systems.

1.4 Thesis contributions and outline

Generalised exemplars offer an alternative to hybrid learning systems in the search for a universal learner. Salzberg (1991) introduced the method of "nested generalised exemplars," or NGE, and demonstrated its practicality, but investigations into its performance suggest that there are problems that need to be addressed (Wettschereck and Dietterich, 1994). This thesis investigates those problems. Specifically, we need to find the optimum balance between generality and specificity so that useful generalisation is performed but not at the expense of the benefits of instance-based learning.

This thesis explores non-nested generalised exemplars, comparing it to other machine learning methods and highlighting its strengths and weaknesses. It introduces NNGE, a new method for exemplar generalisation, as a promising universal learner. It proposes and experimentally evaluates the following four hypotheses.

- **Hypothesis 1:** Generalised exemplars increase the performance of nearest neighbour systems by improving the representation of large disjuncts;
- **Hypothesis 2:** Producing exclusive generalised exemplars results in a useful set of rules that may be compared with those produced by other rule induction methods;
- **Hypothesis 3:** Generalised exemplars reduce classification time without sacrificing accuracy;
- **Hypothesis 4:** A learning system using non-nested generalised exemplars shows better classification performance than one using nested generalised exemplars.

NNGE does not attempt to out-perform all other machine learning classifiers. Rather, it examines generalised exemplars as a method of improving the classification performance of instance-based learners. There are many approaches to classification, and it is likely that a combination of several will produce the best results. Furthermore, there are problems with exemplar generalisation that this thesis does not try to solve. Section 6.4 describes some of these.

Chapter Two reports some current methods of machine learning, concentrating on instance-based learning, which provides the basis for this work. Chapter Three describes generalisation of exemplars, detailing work already performed in this area and discussing problems with the approach. Chapter Four introduces NNGE, a new implementation of generalised exemplars. Chapter Five reports the experiments performed to test the validity of NNGE and to compare its performance with other systems. Finally, Chapter Six summarises the results of the research and suggests future work.

2 Background

A major goal of machine learning is the classification of previously unseen examples. Beginning with a set of examples, the system learns how to predict the class of each based on its features. The user presents new examples to the system one at a time, and it attempts to classify them. The user then reveals the correct answer. If the prediction of the class of new examples is more accurate than random guessing, the system has learned how to perform the classification task to some extent.

The methods used vary widely. Researchers have investigated neural networks and rule induction techniques for several decades, and there have been many different methods devised for performing each. More recently, instance-based learning methods have emerged as a viable third alternative in the search for a general-purpose classifier.

Instance-based systems learn new concepts by storing past cases in such a way that new examples can be directly compared with them. On the basis of this comparison the system decides the class of the new example. There are two quite different approaches. The first, nearest neighbour, uses a distance function to measure the difference between the new example and those in memory. The case of the most similar example is then used to classify the new one. The second, case-based reasoning, is a knowledge-rich approach that uses expert knowledge to link the examples in memory so that the system can quickly locate, and then search the relevant cases to find the most similar one.

2.1 Nearest neighbour

Nearest neighbour is a method that originated in statistics. It was first considered for rule production by Fix and Hodges (1951), who performed an initial analysis of the properties of k-nearest neighbour systems, and established the consistency of the method as k varies from one to infinity. They also numerically evaluated the performance of k-nearest neighbour for small samples, under the assumptions of normal distribution statistics (Fix and Hodges, 1952). It was subsequently adopted as a Bayesian approach to non-parametric classification for two-class problems (Johns, 1961), and has been widely used in the field of pattern recognition since 1963 (Kanal, 1963).

A nearest neighbour learner uses a metric that measures the distance between a new example and a set of exemplars in memory. The new example is then classified according to the class of its nearest neighbour. A pure nearest neighbour system stores all examples in memory verbatim. It then classifies new examples by finding the most similar case in memory and adopting its class. A distance function is used to determine similarity. For numeric attributes this is usually based on Euclidean distance, where each example is treated as a point in an n-dimensional feature space. It assumes that for a given point in the feature space the surrounding area will share the same class. The Euclidean function further assumes that all features are equally important, and so share the same scale in feature space, and that this scale is linear along each axis.

For the Euclidean distance function to work well the examples must be clustered into relatively few dense regions in feature space that share a common class. Conversely, if the examples are randomly distributed throughout the feature space, this violates the assumption that nearby regions in feature space classify the same. The Euclidean distance metric then fails, resulting in the same classification as random guessing. We would therefore expect a nearest neighbour system using the Euclidean distance function to work well for the same domains as conceptual clustering (Fisher and Schlimmer, 1988). Conceptual clustering is described in Section 2.4.

Symbolic features are more problematic as they do not fit the Euclidean feature-space model. To overcome this, similarity between symbolic features is determined by counting the matching features. This is a much weaker function as there may be several concepts based on entirely different features, all of which match the current example to the same degree. For domains containing a mixture of numeric and symbolic features the Euclidean distance function is adopted, with the distance between two symbolic values trivialised to zero if the features are the same, and one if they are not. This mismatch between Euclidean feature space and symbolic features means that pure nearest neighbour systems usually perform better in numeric domains than in symbolic ones.

The following sections describe some nearest neighbour learning systems.

2.1.1 The IB family

Nearest neighbour methods regained popularity after Kibler and Aha (1987) showed that the simplest of nearest neighbour models could produce excellent results for a variety of domains. They tested three simple algorithms, named PROXIMITY, GROWTH, and SHRINK. All three used a normalised Euclidean distance function to classify each new example, with the class being decided according to that of the single nearest neighbour.

PROXIMITY is a pure nearest neighbour algorithm, retaining all examples and using an unweighted Euclidean distance function to perform classification. This system gave the best performance of the three. GROWTH accepts examples incrementally, and only stores those that the current exemplar database misclassifies. This reduces the number of examples stored by up to 80% with only a small reduction in classification accuracy. SHRINK accepts all exemplars at first, and then weeds out those that the rest of the database classifies correctly. This algorithm produces impressive compression of the exemplar database (up to 80%), but at the expense of classification accuracy.

Kibler and Aha (1987) tested the above algorithms on several benchmark domains, and reported very good results for all of them. In particular, GROWTH produced excellent results for a relatively small final database. However, the results were misleading. A later study shows that the choice of domains for the initial study was fortuitous, and that in general performance of the three algorithms is much poorer than other classification methods (Aha, 1992). In comparisons with C4.5 (Quinlan, 1993), PROXIMITY performs quite well for four domains but very badly for another two, showing that the simple nearest neighbour approach has problems to overcome.

A series of improvements was introduced in the algorithms IB1 to IB5, showing how the standard Euclidean distance metric is inadequate in many domains. The aim of the study was to overcome five objections to nearest neighbour systems (Brieman, Friedman, Olshen and Stone 1984), namely:

- they are expensive due to their large storage requirements;
- they are sensitive to the choice of similarity function;
- they cannot easily work with missing attribute values;
- they cannot easily work with nominal attributes;
- they do not yield concise summaries of concepts.

There follows a brief description of each of these experimental systems.

IB1: nearest neighbour. IB1 uses a Euclidean distance function that classifies according to the nearest neighbouring example, saving all examples as they are introduced to it. The only variations from a pure nearest neighbour system such as PROXIMITY are that attribute values are linearly normalised before examples are processed, and that missing values are handled by assuming that a missing feature is maximally different to that feature in all other examples. Like PROXIMITY, IB1 performs well in four out of six of the domains tested, and very poorly on the other two. These two are characterised by noisy values, missing values, and irrelevant features.

IB2: save only misclassified instances. IB2 differs from IB1 in that it saves only instances that it misclassifies. This reduces the number of exemplars required by storing only a single exemplar for each important region of feature space, and proves to be an effective way to prune the exemplar database. Accuracy decreases because early in the learning process important examples may be discarded because there were not enough examples of conflicting classes to accurately portray the differences between the new example and the nearest neighbour. As the number of stored exemplars increases, the accuracy of the model improves, and so the system makes fewer mistakes. There are problems when the input data is noisy. Because the classification of noisy examples is poor, IB2 is more likely to store them, leading to an exemplar database where a disproportionate number of the examples contain noise. Aha (1992) observed that the performance of IB2 degraded more sharply with increased noise than IB1, and that the amount of noise in the exemplar database containing noise was higher than the percentage of noise in the input data. This confirms that IB2's method of choosing which examples to store leads to a bias towards noisy examples.

IB3: retain only good classifiers. Noisy exemplars will impact the performance of any system that does not detect them, because they will repeatedly misclassify new examples. IB3 overcomes this by pruning bad classifiers. It monitors the classification performance of each exemplar to determine whether or not they should be used. A record is kept of the number of correct and incorrect classifications each exemplar makes. If the closest exemplar does not have an acceptable performance record, its statistics are updated but it is ignored in favour of the closest acceptable neighbour. IB3 bases this decision on the exemplar's performance relative to the observed frequency of its class. If an exemplar correctly classifies new examples to a significantly higher degree of accuracy than the observed frequency of its class, it is accepted for classification. If it classifies to a significantly lower degree, it is deleted from the database.

This modification dramatically improves the performance of IB3, resulting in comparable performance to IB1 in most domains, and improvements in two. Both storage requirements and the amount of noise in the database are substantially reduced compared to IB1 and IB2.

IB4: weight attribute values. The Euclidean distance function works well for numeric domains where all attributes have similar relevance. In most domains this is not the case. The relevance of each attribute may be learned incrementally by dynamically updating feature weights. Aha (1992) proposes that these weights should be concept-specific, in that an attribute may be important to one class but not to the others. IB4 weights attributes dynamically, and performs much better than IB1, IB2 and IB3. In particular, the introduction of irrelevant attributes has very little effect on IB4 while for IB3 the exemplar database grows exponentially as the number of irrelevant attributes increases.

IB5: handle novel attributes. IB1 handles missing values by assuming maximal distance for an attribute if it is missing in either the new example or the exemplar being tested. However, sometimes an attribute is missing because it is not relevant to the current example. Also, the value of an attribute may not be available when the first examples are being collected, but become so later. IB1 will incorrectly penalise those examples for which the attribute value was not available or not relevant.

IB5 overcomes this problem by assuming that the distance between a missing and present attribute is zero. Sometimes-present attributes therefore affect the distance function only when the value of that attribute is known for both examples. IB5 was tested using a domain that contained six boolean attributes, of which only one is relevant to classification. For the first 100 examples, the relevant attribute is missing, and so classification accuracy is approximately 50%. When the relevant attribute is added, IB5 quickly adapts to the new situation and classification accuracy improves. In contrast, IB4 reacts very slowly to the introduction of the new attribute.

2.1.2 KNN

Another successful variation of nearest neighbour is *k*-nearest neighbour (Kibler and Aha, 1987). This is an alternative method for dealing with noise, where the most popular class of the *k* nearest examples is used for prediction. This prevents a single noisy example from incorrectly classifying the new one, and has met with much success. The problem, however, is determining the value of *k*; the more noise in the input set, the larger *k* should be. As the system does not have this information *a priori*, a popular method is to use cross validation. The user trains and then tests the system using a variety of *k* values, and the one that produces the best result is subsequently adopted.

2.2 Case-based reasoning

Nearest neighbour methods try to define similarity and differences between historic cases by imposing a metric upon them, the Euclidean distance function. This metric is very rigid and does not represent subtleties such as irrelevant or missing features. The solution, as we have seen, is to add terms to the function such as feature and exemplar weights. The system infers these weights from the input data, usually in a somewhat *ad hoc* manner. They are not formally related to the original distance function.

Case-based reasoning methods do not try to measure the similarity between cases numerically. Instead, they form a model in memory of the relationships between examples. These relationships may either be induced (Kolodner, 1984; Lebowitz,

1987) or supplied by an expert user (Bareiss and Porter, 1988). New examples are compared to the cases in memory by determining how closely they match these relationships. Some methods use the relationships to generalise the cases into a hierarchy that, once established, is difficult to alter. Other, more flexible, methods retain all cases, the relationships instead forming threads that link similar cases.

Case-based reasoning is of particular interest to cognitive scientists. While nearest neighbour is a successful method for classifying new examples, people do not have the ability to calculate the distance between new and previously experienced instances in such a manner, and so the method provides little insight into the mechanism of human learning. It is quite widely thought, however, that people often recall past experiences when solving new problems. Case-based reasoning is considered to be a plausible model of this process (Bareiss and Porter, 1988).

Three case-based methods will be discussed. The first two, CYRUS and UNIMEM, infer generalisation hierarchies from the examples. The third, PROTOS, maintains a complex set of user-supplied relationships that are continually refined as new examples are added.

2.2.1 CYRUS

An early example of case-based reasoning by a psychologist was CYRUS (Kolodner, 1984). CYRUS contains information about events in the life of two US secretaries of state: Cyrus Vance and Edmund Muskie. A user can ask CYRUS questions about either individual's movements in natural English. Questions can either be about specific events that have occurred, or they can be more general, such as "When did you last go to the Middle East?" The facts stored in CYRUS are entirely episodic. It makes no attempt to store the knowledge known by the two men, nor does it record an entire history of events in the two men's lives.

The aim of the CYRUS project was to develop a plausible model of human remembering sufficiently detailed that it could be implemented on a computer. Work to date in this area had produced only vague models that failed to address many of the key problems in memory recall. The crux of Kolodner's work was that the memory model should not only support human memory processes but should be required by them. Also, the memory model should exhibit the shortcomings of human memory recall such as forgetting and erroneous construction of facts (Kolodner, 1984).

The memory model for CYRUS began as a semantic network allowing enumeration of branches from any particular node, such as that in Figure 2-1(a). Any object stored in such a structure may be retrieved by traversing the entire network. This model was then made more restricted by disallowing traversal. Kolodner took this step for two reasons. First, if human memory is enumerated in such a fashion, recall would be expected to slow down as more facts are learned. In practice, the reverse is usually observed; the more knowledgeable a person is about a subject, the more quickly they recall facts about it. Second, people are not very good at returning lists of facts from memory such as “name all of the museums that you have visited,” and usually go through a process of reconstructing scenarios related to the desired facts in order to retrieve them. The resulting model was therefore a semantic network where a particular node can only be accessed if the key to one of its paths is provided. In other words, CYRUS may only retrieve objects from memory if it can provide exact matches of features pertaining to them. In Figure 2-1(b) information about *Rome* can only be retrieved by supplying the index keys *area:Europe* and *city:Rome*. Sometimes the required keys are not part of the original question, in which case CYRUS constructs plausible scenarios to generate other candidate keys.

CYRUS generalises the stored information by retaining at each node a list of features shared by its subnodes. It further indexes each subnode by its differences. Figure 2-2 gives an example of the node “meetings” and its subnodes, in this case two events. Note that CYRUS indexes the two events in more than one way because there is more than one dissimilar feature, *participants* and *topic*.

One of the difficulties with indexing by differences is deciding which features to use. Indexing them all is not practical because it leads to a combinatorial explosion of indexes. CYRUS avoids this by using only those features that are likely to match a significant number of future examples. This implies that the more general a feature, the better an index it is. When deciding to index a new event by a particular feature, CYRUS uses its background knowledge to find the most general representation of the feature that still uniquely identifies the event. The steps that CYRUS takes when deciding which features to index are:

- select those features that have previously been predictive;

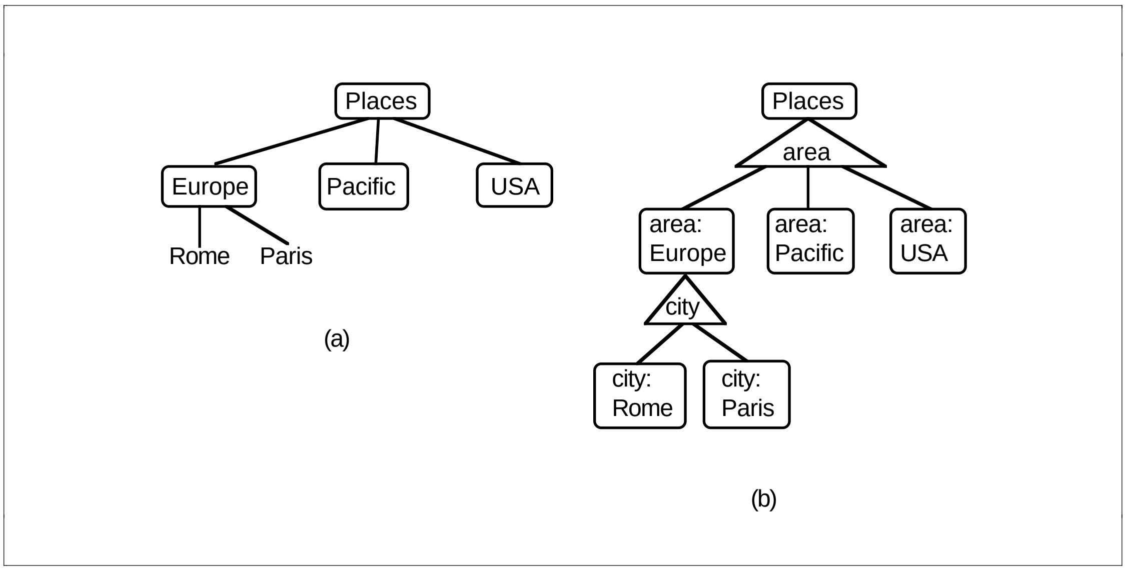

Figure 2-1. Enumerable vs non-enumerable memory

- remove those features that are present in the generalisation node that this event is to be indexed below;
- remove those features where this particular feature value is known to be non-predictive;
- choose the most general representation of each feature value that still uniquely describes the event.

Generalisation occurs whenever a new event is introduced that shares features with an existing one. When this happens, CYRUS creates a generalisation node containing only the common features, and the events are indexed below it by their differences. When others are encountered that also share the same features, the size of the generalisation node grows. Events indexed below a generalisation node may later match to new events, and so be further generalised. CYRUS therefore builds a hierarchy over time.

A problem with this form of generalisation is that the features chosen may only be the same by coincidence, and so the generalisation node will contain the wrong ones. CYRUS avoids this by allowing an event to match to a generalisation if most of its features match. A feature that mismatches too many times is removed from the generalisation and placed in a subnode. Furthermore, generalising the wrong features may lead to a generalisation node that directly indexes few events, but which indirectly indexes a larger number of events via a subnode. When this occurs CYRUS collapses the subnode into the parent. Figures 2-3 and 2-4 illustrate these two examples of incorrect generalisation. In Figure 2-3(a) the feature *includes state dinner* has been generalised because the first two events shared this feature, but subsequent ones do not. Events are later encountered that match the other generalised features but not *includes state dinner*. When sufficient of these are observed, *include state dinner* is dropped from the node, as in Figure 2-3(b). In Figure 2-4(a), the first two events

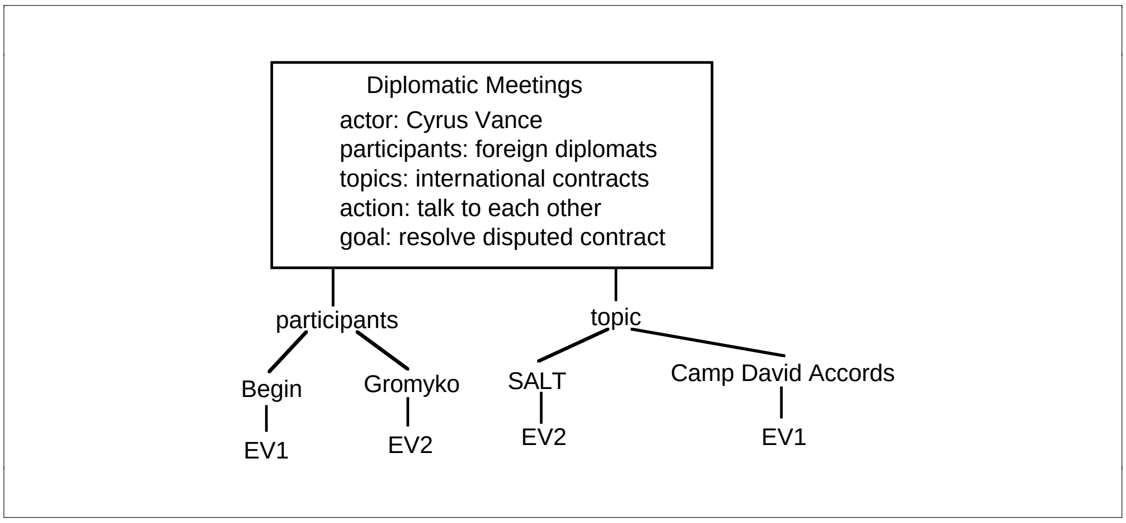
Diplomatic Meetings
actor: Cyrus Vance
participants: foreign diplomats
topics: international contracts
action: talk to each other
goal: resolve disputed contract
participants
topic
Begin
Gromyko
SALT
Camp David Accords
EV1
EV2
EV2
EV1

Figure 2-2. Example of a node and subnodes in CYRUS

EV1 and *EV2* included a different special event. Later events, however, all included a state dinner. The subnode *includes state dinner* therefore contains most of the events indexed by the parent node, so CYRUS collapses the subnode into the parent, giving the structure shown in Figure 2-4(b). *EV2* is now indexed by an exception to the generalisation *includes state dinner*.

A feature of human memory is that we may forget facts if they did not seem important at the time. For example, if a person goes to a restaurant where a band plays, and it is the first time that they have seen live music at a restaurant, they may not remember the name of the band because it was not as significant as the fact that there was live music there at all. As they come across more bands at restaurants, however, they may start to remember the names of them, because this is what

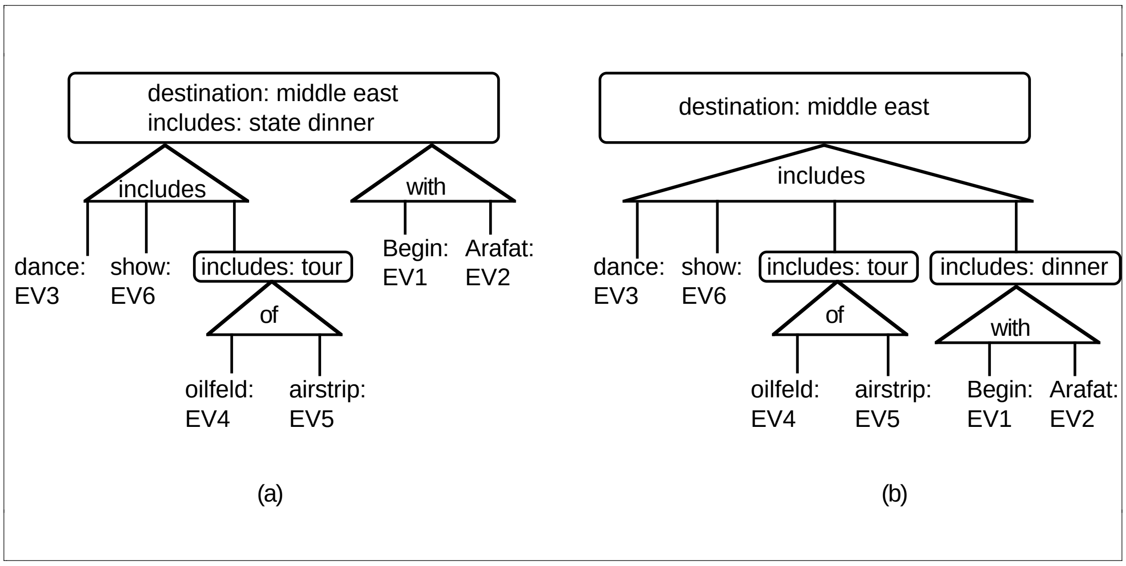

Figure 2-3. Generalisation on the wrong feature

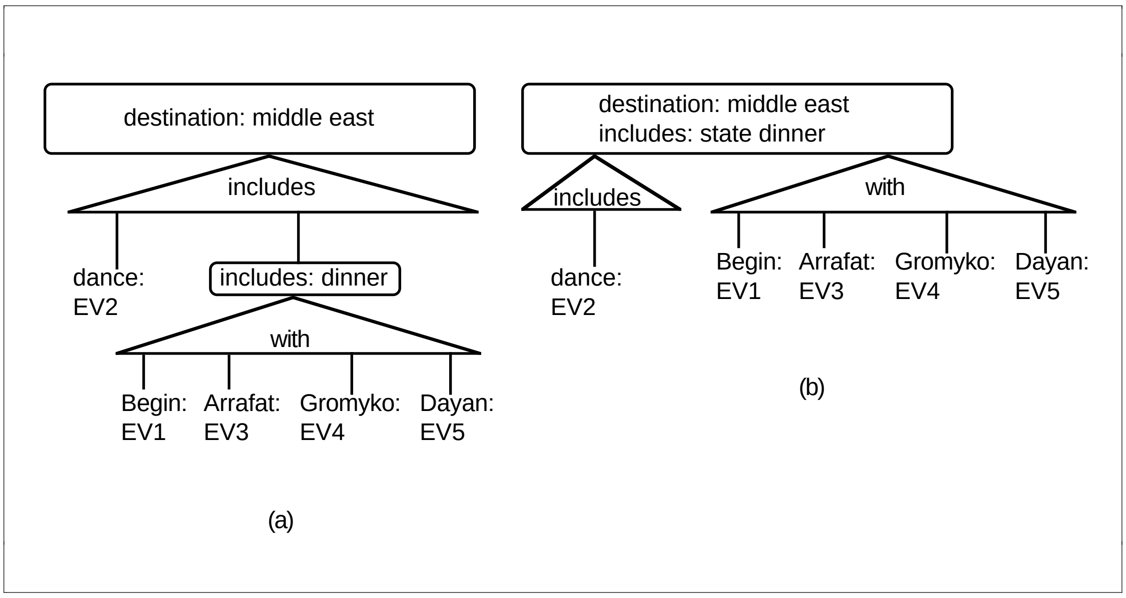

Figure 2-4. Overspecialisation

distinguishes them from each other. The name of the first band, however, has been forgotten forever. This is also true of CYRUS; whenever a new event is stored, it is indexed by feature values that are as general as possible, while still being sufficient to uniquely identify the event. In the above scenario, therefore, CYRUS would index the first visit by “includes band.” Once more restaurants are visited, this is no longer sufficient to uniquely identify each visit, and so some other feature, such as the name of the band, is required. However, CYRUS never recorded the name of the first band, so has forgotten it.

While storing exceptions helps to produce a compact and very general memory model, it also has disadvantages. When answering an inferral question of the type “What meetings did Cyrus conduct in the Middle East,” CYRUS descends the most favoured index path, preferably from a node containing “Middle East.” Two things may go wrong with this type of search. First, some of the required events may be stored in exceptions under less favourable paths, in which case they will not be retrieved by this search. More drastically, there may be no nodes at a high level containing “Middle East,” in which case CYRUS will not return any events. The network contains a node for each category of object stored in memory. Searches for an object must start at a node, so it is necessary to provide the category, which is usually the main subject of the question. For example, when asking “Where did you last go on holiday?” the initial category might be “holidays.” Because the desired object may not be indexed by this category, CYRUS requires background knowledge to suggest others. This knowledge is stored in the same memory structure.

Using background knowledge to find other plausible categories for search is called *reconstruction*. This often involves remembering other events that typically occur in a certain situation, so that what actually occurred can be pieced together. For example, when going on holiday the typical events performed on the first day might be:

- get off the plane;
- phone for a taxi;
- ride to motel;
- check into motel;
- sleep to shake off jet lag.

When asked “Did you visit the Louvre museum in Paris?” CYRUS might use the above list to search for the events that did occur, and return “No, I was only there for one day and I slept all day.” Background knowledge is therefore a necessary part of CYRUS. This information can also be used to produce an alternative context for the search if the obvious one failed to return the required information. Figure 2-5 shows the overall retrieval strategy.

CYRUS represents a significant advance in the understanding of human memory. The structure proposed exhibits many of the traits of human memory, including the need for reconstruction of events surrounding those being recalled, and the ability to forget. From a computing point of view, CYRUS shows that search without enumeration is both possible and practical. Furthermore, CYRUS shows that generalised examples can be stored and retrieved in the same way as the examples themselves, allowing a unified case-based learning approach.

2.2.2 UNIMEM

UNIMEM (Lebowitz, 1987) is a machine learning system that uses a very similar memory model. Like CYRUS, it does not classify new instances, but performs learning by observation, grouping together examples into concepts according to similarities in their features. It assumes that similarities in naturally occurring examples reflect meaningful regularities. This is similar to conceptual clustering, which is described in Section 2.4.

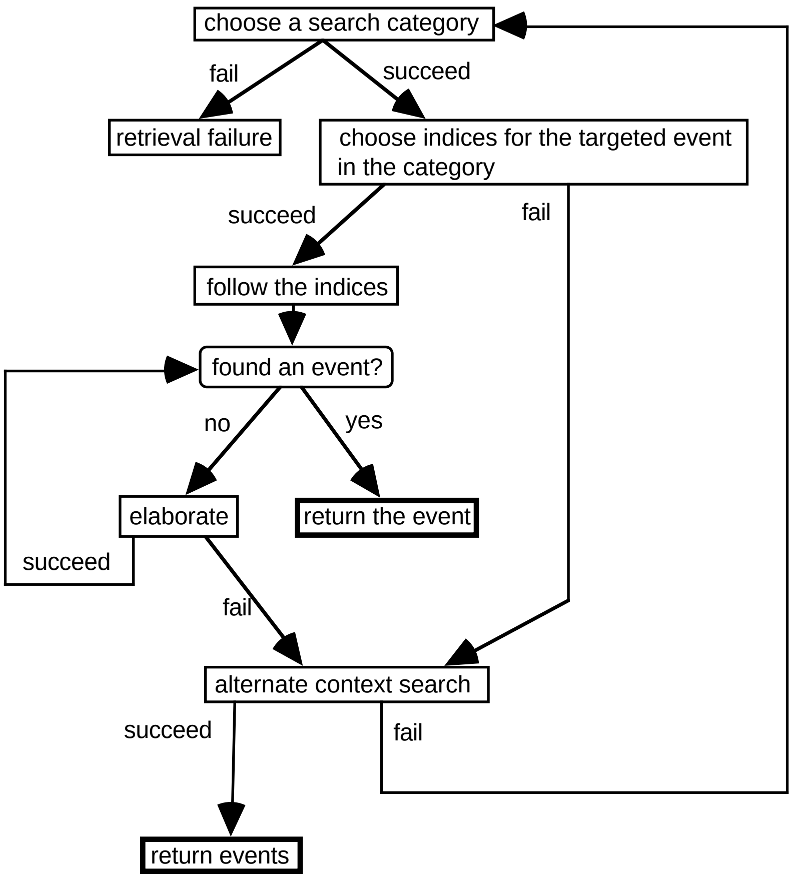

Figure 2-5. CYRUS search strategy

UNIMEM organises the examples into a generalisation based memory (Lebowitz, 1980), a hierarchy where each node represents a collection of features that the examples stored below it share in common. Figure 2-6 gives an example of such a hierarchy. UNIMEM has produced the three generalisations *G1*, *G2* and *G3* without the user specifying a class. The user can later give them meaningful names such as:

- G1 = private universities with high academic level and medium social life;
- G2 = expensive urban schools with strong applicant SAT scores;
- G3 = expensive schools with high enrolment yields.

Storing new examples in memory is a two-pass operation. UNIMEM first performs a controlled depth-first search of the generalisation hierarchy to determine the most specific node of which the new example is a member. It allows the new example to conflict with the generalisation node being considered, provided that the amount of mismatch is less than a fixed parameter. This could potentially lead to an example that is described as belonging to a particular concept but having very few features in common with it, but according to Lebowitz if the allowed discrepancy is small, this does not seem to arise. Allowing discrepancies in this manner is analogous in nearest neighbour learning to accepting that two examples are the same if their distance is sufficiently small. This practice is common for continuous-valued domains. For example, Salzberg (1991), when adjusting feature weights for real numbered domains, determines that two feature-values are the same if their difference is less than a fixed parameter.

After finding the most specific node of which the new example is a member, UNIMEM decides how to include it into the generalisation hierarchy. If it has several features in common with another in the same node, a new generalisation is formed

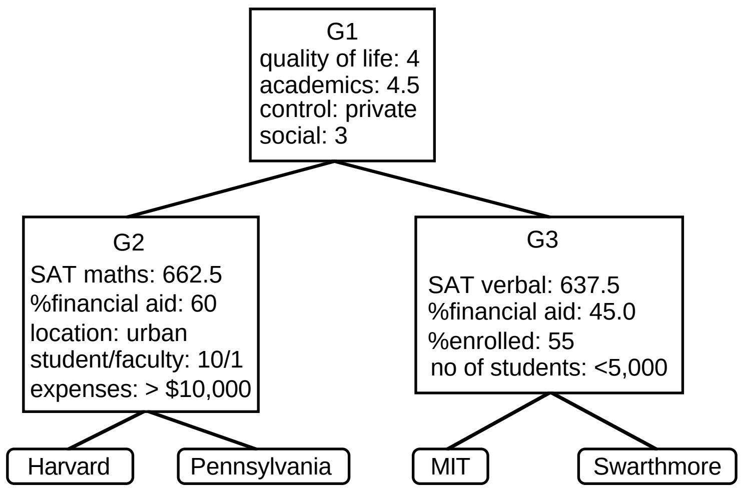

Figure 2-6. GBM hierarchy of American universities

that is a specialisation of that node, consisting of the new example and the matching one. If the new example is sufficiently similar to several others with different overlapping features, several generalisations may result. If the feature values are acceptably similar but do not match exactly, UNIMEM uses the average value.

As with CYRUS, UNIMEM risks overgeneralising, and later finding that a parent node has too many exceptions. Instead of deciding when to generalise or specialise by calculating the proportion of examples in the generalisation versus those in specialisation nodes beneath it, UNIMEM maintains weights for each feature of each generalisation. Whenever an example matches this node, the matching features have their weights increased, indicating a higher confidence that they are relevant to the generalisation. Features that contradict the new example have their confidence lowered. Each weight is altered by an amount proportional to the difference between the feature value in the generalisation and that in the new example. If a confidence level drops too low, UNIMEM removes the feature from the node and adds it to all of the subnodes. Once confidence is sufficiently high the feature becomes permanent, and its confidence level is no longer adjusted. The confidence levels do not influence the similarity decision in any way. Using this method UNIMEM prunes rather than discards generalisations that contain incorrect features. This aspect of the algorithm controls overgeneralisation by determining how closely features in the generalisation must match new examples.

When generalisations are pruned in this way they may eventually contain very few features and be too general to provide any information. UNIMEM avoids this by discarding any generalisation containing less than a fixed percentage of the total number of features. Instances contained within the discarded generalisation and all of its subnodes are themselves discarded to prevent the same faulty generalisation being recreated. Lebowitz (1987) acknowledges that discarding examples is a dubious practice that could lead to whole concepts being lost. In fact, pruning simple disjuncts is not a good idea, because it does not allow UNIMEM to induce very simple rules. It has been shown that for many classification problems a single-attribute rule covers a large fraction of the input set (Holte, 1993). UNIMEM would have great difficulty representing these data sets.

2.2.3 PROTOS

Perhaps the most complex memory model system is PROTOS (Bareiss and Porter, 1988). PROTOS learns from examples, with additional knowledge being provided by a domain expert.

PROTOS does not create a hierarchical structure. Instead, it groups exemplars by class, and adds relationships that link similarities between them. Figure 2-7 gives an example of the class *chair* which contains two exemplars, *chair1* and *chair2*.

Relationships are either trivially implied by the existence of a feature, such as "chair2 has legs(4)," or are supplied by the user in the form of background knowledge, such as "metal and wood are both examples of rigid material."

PROTOS performs classification by using three index structures to retrieve candidate classes and exemplars. The above relationships, or *remindings*, are used to retrieve the most similar class, and the most similar exemplar. Multiple remindings to a class are heuristically combined to determine how closely it matches the new example. The best matching exemplar is then chosen by selecting the most prototypical. *Prototypicality* is a function of the number of correct and incorrect classifications that each exemplar has performed. Finally, PROTOS uses *difference links* to check the best exemplar's neighbours. The user provides difference links to explain misclassification. When the class predicted for a new example is incorrect, a difference link is constructed between the incorrectly classifying exemplar and the most similar of the correct class. This link highlights important differences between the two exemplars that were overlooked during classification. PROTOS uses this link to discriminate between the two similar cases in future.

PROTOS continuously evaluates the knowledge provided by the expert. Whenever it misclassifies an example, it offers an explanation of how it arrived at its conclusion. If any step is incorrect, the user may alter the underlying domain knowledge to correct the situation. PROTOS therefore refines the knowledge structure during the learning process.

Generalisation within PROTOS takes two forms. First, common features amongst exemplars of the same class provide a reminding to that class. In Figure 2-7, both *chair1* and *chair2* have seats, and so *seat* becomes a feature of the class *chairs*. These

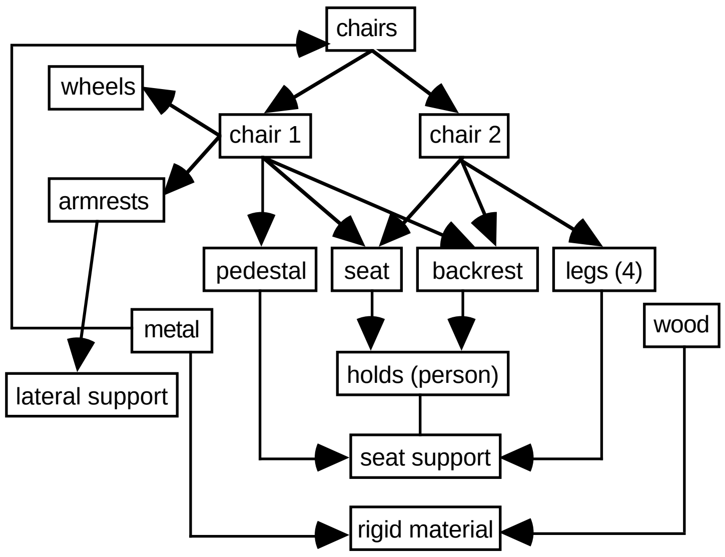

Figure 2-7. PROTOS knowledge structure for class *chairs*

remindings are not explicitly included in the description of the concept, but rather are evaluated each time a new example is classified. The importance of each of these implicit remindings is determined by observing how typical the feature is to the class; a feature that all exemplars within a class have in common is considered essential, while a feature shared by a very small number is considered unimportant. Second, new examples in which all features can be successfully matched to the features of the most similar case are merged into that exemplar by generalising any different features to the feature generalisation that allowed them to be matched. In Figure 2-7, *pedestal* and *legs(4)* are both examples of *seat support*, and so can be generalised to this value. Unlike the implicit generalisation described above, this action is explicit and irreversible.

As a result of having several different types of indexes, PROTOS' search strategy is quite complex. Rather than performing a single search for the most similar case, PROTOS begins by finding a group of promising cases, and then prunes and refines this group using the various indices until it finds the best exemplar. The overall search strategy is:

- collect remindings to each category, heuristically combining multiple ones to the same category, and retaining the *c* most promising categories;
- select the *p* most prototypical exemplars from each of the *c* categories;
- collect remindings to each exemplar, heuristically combining multiple ones to the same exemplar, and add the *e* exemplars with the strongest remindings to the list;
- order the list of promising exemplars by reminding strength;
- for the most promising exemplar, determine its strength of similarity with the new example. If not sufficiently strong, discard and try the next exemplar in the list. Repeat until either a suitable match is found or the list is exhausted;
- try to find a better match using the difference links;
- return the class of the best exemplar.

Bareiss and Porter (1988) tested PROTOS on the domain of clinical audiology. While learning to become practising audiologists, students are typically subjected to between 150 and 200 cases of patients with ear abnormalities. PROTOS was trained on 200 such cases and tested on a normal mix of 26, correctly classifying all of them, compared to a classification accuracy of 38% for ID3 (Quinlan, 1986) and 77% for PROXIMITY (Kibler and Aha, 1987). Interestingly, when Bareiss and Porter replaced PROXIMITY's matching algorithm by that used in PROTOS, they reported an accuracy of only 62%, indicating that the low-level matching algorithm employed by PROTOS could be improved (Bareiss and Porter, 1988).

The high performance achieved by PROTOS for this domain compared to other learning methods showed that learning systems that interact with the user can learn complex concepts more effectively than non-interactive systems.

2.2.4 Discussion

The three methods described all maintain a memory structure that helps them make inferences about future examples. Incremental learning is advantageous because it allows the concepts being learned to be refined indefinitely without requiring that the input dataset be completely relearned each time a new example is added. However, a disadvantage is that the memory structure initially takes shape when the system has observed only a small number of examples. This causes two problems: the memory structure may be over-general, and it may use the wrong features when defining concepts.

CYRUS and UNIMEM overcome the first problem by detecting and remedying overgeneralisation. CYRUS decides that a concept is over-general when there are too many more examples in exception nodes below it than in the node itself. When this occurs it removes the offending node so that the exceptions become simply generalisation nodes. UNIMEM checks features, rather than whole generalisations. When a feature has mismatched too many of the examples in the node being tested and those below it, UNIMEM removes it from the generalisation node. If a generalisation has too few features, it is deleted. UNIMEM cannot take CYRUS' approach because it does not store the actual examples in memory. If an example is sufficiently similar to a generalisation, it is stored as though it matched, and information about dissimilar feature values is lost.

Neither CYRUS nor UNIMEM copes well with the second problem of incorrect feature use. While CYRUS does detect and remove features in a generalisation node that fail to match events indexed by that node, it is unable to alter the features used to index an event, because it only stores the features needed to discriminate the new example at the time that the example is observed. If these features later prove to be spurious, and the important ones have been dropped, the example cannot be indexed efficiently.

If UNIMEM has discriminated using the wrong features, they will gradually be deleted from the generalisation. These features are not propagated down to the subnodes as the values of the feature at the subnode level are not retained. Similarly, when the number of features in a generalisation falls below the negative threshold UNIMEM deletes it from memory. Doing this loses all nodes and examples below, as the values of the features within the subnodes and their examples are not retained. Failure to store the lower node features is not an oversight: Lebowitz (1987) claims that if this information were stored, attempting to rebuild the portion of a hierarchy below a dropped node would result in the same erroneous generalisation being built again. He further claims that in the domains he tested this was not a serious problem as there were always sufficient examples for useful concepts to be created. However, this may not always be true.

PROTOS relies on an expert user to guide it in learning the importance of feature similarities, and so the extent that it suffers from the above two problems depends to some degree on the competence of the user. An incorrect generalisation presented by the user early on may cause PROTOS to perform badly when collecting the set of candidate examples, without the cause of the problem being obvious. The user may therefore continue to add unnecessary concepts to try to correct the problem, when in fact removing the spurious one would be a better cure.

Of the three systems, only PROTOS has the ability to correct its memory structures when it goes wrong, and does so by relying on the user to spot the mistakes that it has made and correct them. It is possible, however, for non-interactive incremental learning systems to correct their memory structures. ID5 (Utgoff, 1989) is an incremental tree builder based on ID3 (Quinlan, 1986) that reconstructs portions of its tree whenever it becomes apparent that it is discriminating on the wrong features. It does so by retaining enough information about all examples observed to redo the calculations necessary to decide which features to use. CYRUS, UNIMEM and PROTOS do not perform such calculations. Instead they build the memory structures *ad hoc*, preventing re-evaluation when they find mistakes.

A common feature of all three methods is the use of exceptions. CYRUS and UNIMEM both use exceptions to overcome explicit overgeneralisation, while PROTOS uses them to overcome implicit overgeneralisation by allowing the user to add difference links that discriminate between otherwise similar exemplars.

2.3 Rule induction

Instance-based learning systems retain the set of examples, along with some means of comparing a new example to those in memory. In contrast, Rule induction methods generalise the training set into rules that they can evaluate directly to classify new examples. These rules may be represented in many ways, including decision trees and modular rules. Rule induction systems evaluate the features of the training set and decide which ones to use to discriminate between the different classes. The following sections describe two popular rule induction systems that use a similar method of choosing the next attribute to discriminate, but differ in the final representation of the learned rules.

2.3.1 ID3

Quinlan (1986) developed the tree-inducing system ID3 while trying to devise a method to compress chess end-games. ID3 reduces a set of input examples to a decision tree, where the value of a single attribute determines the outcome at each decision node. Figure 2-8 gives an example of a typical decision tree.

ID3 produces decision trees in a top-down fashion. Beginning with the tree root, it chooses the first attribute to discriminate on, and produces a subnode for each value. If all examples with a particular attribute value have the same class, the node becomes a leaf node, otherwise ID3 chooses another attribute to further discriminate between the classes. The tree is complete when all examples are represented by a leaf node.

To determine which attribute to branch on at each level, ID3 calculates the information gained by discriminating on each, and uses the one that maximises the gain. The formula used is:

$$\text{gain}(A) = I(p,n) - E(A)$$

where

$$E(a) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \left(-\frac{p_i}{p_i + n_i} \log_2 \frac{p_i}{p_i + n_i} - \frac{n_i}{p_i + n_i} \log_2 \frac{n_i}{p_i + n_i} \right)$$

and

$$I(p,n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

ID3 performs well for a large number of domains, producing compact decision trees with a high classification accuracy. Its main drawback is sensitivity to noise, a problem that can be reduced by pruning the resulting decision tree. A commercial version of ID3 called C4.5 (Quinlan, 1993) provides facilities for decision tree pruning, as well as supporting continuous-valued domains by partitioning the range of possible values.

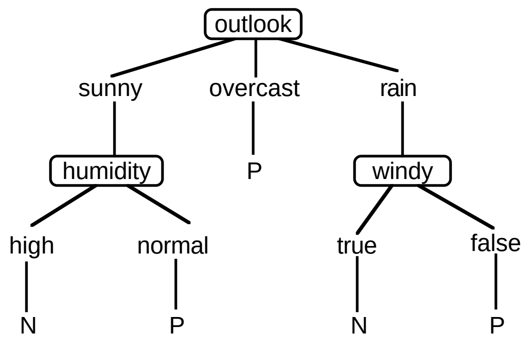

Figure 2-8. Example of a decision tree

2.3.2 PRISM

A problem with decision trees that discriminate by attribute is that they assume all values of each attribute are important, and so determine the class by finding the most discriminating attribute. For many domains, to descibe the concept being learned may require several rules that do not share common attributes. In this case the decision tree created will be large and overly specific, and so will not classify new examples very well. PRISM overcomes this problem by inducing modular rules where each attribute value is tested individually to decide whether or not to include it in the rule (Cendrowska, 1987).

Each rule begins as an empty set. Each attribute value is then tested to determine which one has the highest information gain, and a clause is added to the rule for that attribute of the form

$$A_i = V_{ij}$$

where V_{ij} is the most promising attribute value and A_i is the corresponding attribute. The set of examples with this attribute value is then isolated. If all of these examples share the same class, the rule is complete and they are removed from the set. If they do not, PRISM finds the next best attribute value and adds another clause to the rule. On completing a rule those examples that satisfy it are removed and the process is repeated until no examples remain.

2.4 Clustering

Another method related to nearest neighbour learning is clustering. Clustering systems build a generalisation hierarchy by partitioning the set of examples in such a way that similarity is maximised within a partition and minimised between them. At the lowest level of the hierarchy are the individual examples.

Clustering is especially suited to unsupervised learning, where the concepts to be learned are not known in advance, but it may also be applied to learning from examples. A new example is classified by considering adding it to each cluster, and determining which one it fits best. This process is repeated down the hierarchy until a cluster is reached that contains only examples of a single class. The new example adopts the class of this cluster.

The main differences between different clustering methods are the similarity measure, and the method used to evaluate each cluster to determine the best fit for the new example. Approaches range from Euclidean distance to Bayesian statistics. Clustering is therefore the broad approach of concept formation by grouping similar examples.

2.5 The problem of small disjuncts

Studies into the cause of misclassification in induced rules show that rules that cover a very small proportion of the training set, or *small disjuncts*, cause most classification errors. While these rules may individually represent only a small proportion of the input set, they may collectively account for over half of the examples, and contribute up to 95% of the errors (Clarke and Niblett, 1987). It is thought that this is mainly due to the use by most systems of a bias that is inappropriate for inducing small disjuncts (Holte *et al*, 1989).

Decision tree and rule inducing methods implement a maximum generality bias; whenever they create a new rule, it is made as general as possible while still discriminating the concept being learned. This approach favours more general rules, or *large disjuncts* (Holte *et al*, 1989). Instance-based learners, on the other hand, retain the actual examples, and so the definition of a particular class consists of a disjunct for each training example, which is the most specific bias possible. This favours small disjuncts, but does not induce large disjuncts very effectively.

The are several ways to improve either the small disjunct performance of rule inducing systems or the large disjunct performance of instance-based learners. Rule induction systems can produce more specific rules by using the generality bias for large disjuncts, but selecting the most specific rule that covers each set of examples that represents a small disjunct. Instance-based learners can adopt a more general bias by considering the *k* nearest neighbours rather than the single closest exemplar.

The effect is to shift the underlying bias further towards the middle. The problem is that this tends to reduce the performance of the system in the area that it performs best. For example, a decision tree inducer with a reduced generality bias will perform worse for large disjuncts. Since large disjuncts usually account for the majority of the training set, this means that overall performance decreases. Holte also experimented with a selective bias version of CN2 (Clarke and Niblett, 1987) that accepts a maximally general small disjunct only if it matches less than 25% of examples of the wrong class. This reduces the error rate for small disjuncts, but decreases overall classification performance. Shifting the bias of a single learning method almost always does this (Holte *et al*, 1989).

2.5.1 Small disjuncts in decision trees

Ting (1994) investigated the problem of small disjuncts in decision trees. By comparing the performance of the decision-tree generating system C4.5 (Quinlan, 1993) and the IBL nearest neighbour systems (Aha, 1992) for a variety of disjunct sizes, Ting shows that for small disjuncts the instance-based learning system out-performs the tree inducer, while for larger disjuncts the reverse is true.

To address this problem, Ting implemented COMPOSITE LEARNER, a system that combines C4.5 (Quinlan, 1993) with IBL (Aha, 1992). The composite learner uses a selection rule that determines which system to believe depending on the size of the disjunct. For small disjuncts the instance-based learner provides the class, while for larger disjuncts the decision tree is used. This approach eliminates the problem of reduced accuracy due to bias shift, because the bias used remains invariant.

In implementing a hybrid nearest neighbour and decision tree system Ting discovered two serious problems. First, no particular instance-based system performs satisfactorily for all domains. He overcomes this by refining the selection criteria to use a particular IBL system according to several characteristics of the data set, such as the presence of missing values. Second, the definition of small disjunct is crucial to the performance of the hybrid system. The definitions Ting (1994) studied are:

- the disjunct size is less than or equal to a fixed value;
- the disjunct size is less than or equal to a fixed proportion *p* of the training set;
- the total small disjuncts coverage is less than or equal to a fixed proportion *t* of the training set;
- the error rate exceeds a fixed error rate;
- the error rate exceeds the decision tree estimated error rate.

Ting (1994) concludes that no particular measure stands out as a clear winner. This ultimately limits the performance improvements observed in the composite learner. Overall it performs quite well, offering improvement over C4.5 in ten of the twelve domains, and outperforming both C4.5 and IBL in eight of the twelve domains. The composite learner often outperforms C4.5 when the instance-based learner alone is significantly worse, supporting the theory that each of these two methods is better than the other for a subset of the complete range of disjunct sizes.

2.5.2 Effect on instance-based learning

Just as generality based inducers often overlook small disjuncts, instance-based learning systems may poorly represent large ones. Given sufficient training examples, large disjuncts should be adequately represented, the examples forming a cluster over the area covered by the large disjunct. Small disjuncts will similarly be represented by such a cluster, albeit a much smaller one. The smallest may be represented by a single example.

Unfortunately, training sets are often too small to represent large disjuncts fully. As a result, small disjuncts may be over-represented by even a single point, and will have an unduly large effect on classification performance in their local area. Figure 2-9 illustrates a two-class problem where new examples are classified as *black* or *white*. The rectangle represents the target concept for a classification of *white*. In Figure 2-9(a) there are many examples of this disjunct, and so the area of feature space that gives a classification of *black* does not significantly encroach upon the area represented by the large disjunct. In Figure 2-9(b), however, the *white* disjunct is poorly represented by the examples, and the area of feature space with class *black* now encroaches heavily upon it. New examples falling within the shaded region will therefore be misclassified.

As mentioned earlier, there are several ways to overcome this problem, by shifting the underlying bias. Nearest neighbour systems can be biased to favour large disjuncts over small by considering the *k* closest points and either selecting the most popular class, or multiplying each result by distance and computing the sum for each class, a sort of "exemplar gravity." Both of these methods, however, reduce the effect of small disjuncts, and so do not provide a completely satisfactory solution. K-nearest neighbour does work well when there is noise in the dataset. This is a consequence of the loss of specificity, because noise tends to appear in the learning system as spurious small disjuncts.

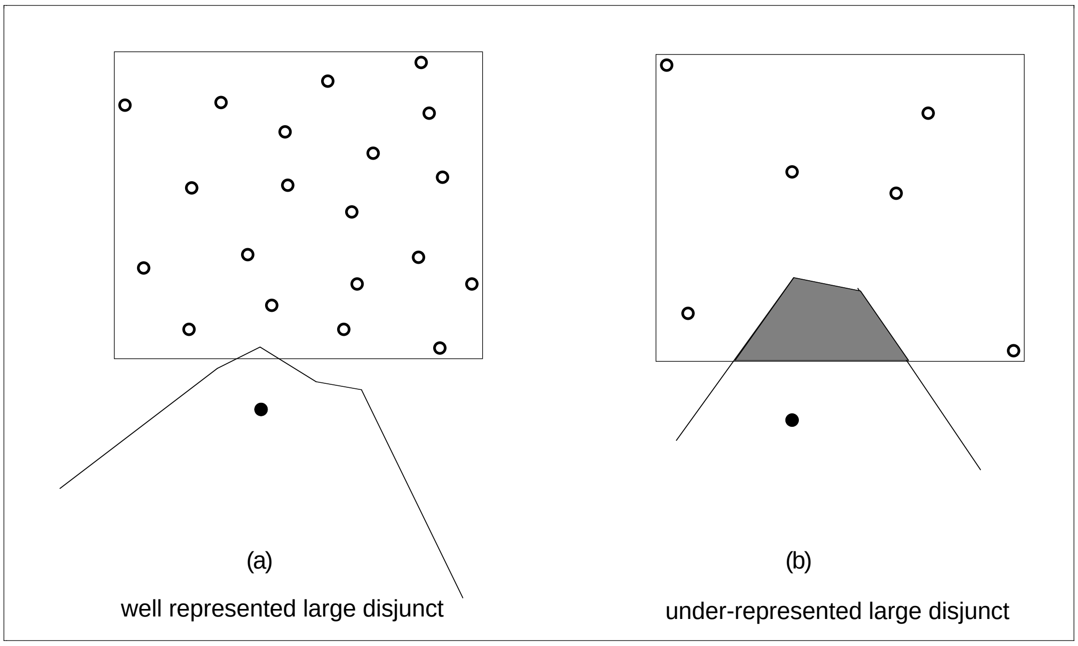
(a)
well represented large disjunct
(b)
under-represented large disjunct

Figure 2-9. Under-representation of large disjuncts

3 Generalised exemplars

Hybrid learners as described in Section 2.5 are one way of combining both a generality and specificity bias. Another approach is to generalise the exemplars in an instance-based learner (Salzberg, 1991; Smith *et al*, 1990).

Generalised exemplars represent more than one of the original examples in the training set. In a geometric sense, if an instance database is a set of points in an *n*-dimensional problem space with *n* being the number of features in each example, a generalised exemplar is an *n*-dimensional region covering a finite area of the problem space. This study explores axis-parallel *n*-dimensional rectangles, or hyperrectangles. Many rule-inducing systems use this form of generalisation, including C4.5 (Quinlan, 1993) and PRISM (Cendrowska, 1987). Hyperrectangles represent each generalisation by an exemplar in which each feature value is replaced by either a range of values for a continuous-valued domain, or a list of possible values for a discrete-valued domain. The exemplar database for a given problem may contain a mixture of both generalised and ungeneralised exemplars. An ungeneralised exemplar is a trivial hyperrectangle of size zero. The exemplar database in Figure 3-1 contains both generalised and ungeneralised exemplars, of class *a* and *b*.

Generalised exemplars may also be thought of as rules. For a given problem, the set of hyperrectangles has the following properties:

- it may be represented as a disjunctive set of conjunctive rules;
- there are as many conjuncts as there are features in an example;
- a conjunct may be expressed as a pair of magnitude comparisons (for continuous values);

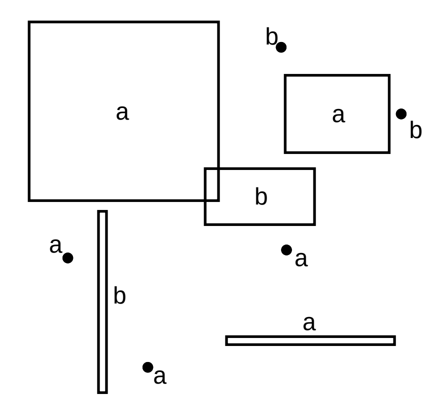

Figure 3-1. Exemplar database containing generalised exemplars

- a conjunct may be expressed as an internal disjunct (for symbolic feature values);
- no external disjuncts are permitted within a hyperrectangle.

Figure 3-2 gives an example of a hyperrectangle represented in such a manner.

3.1 Motivation

There are three reasons why it might be desirable to generalise exemplars. These are now described. We present a hypothesis for each motive, and verify them experimentally in Chapter 5.

3.1.1 Classification performance

Section 2.5 described how nearest neighbour methods implement a maximum specificity bias. Because of this, large disjuncts tend to be under-represented when the training set is small. In particular, if an oracle assembles a set of training instances that represents all known rules in a small number of examples, large disjuncts will almost certainly be under-represented compared to small ones.

Generalised exemplars provide a solution. By generalising where possible, a large disjunct that is represented by a small number of points will influence a larger area of the problem space, by filling in the otherwise empty region between the examples representing it. With such a system, large disjuncts tend to be over-represented early in the learning process, but as more conflicting examples are introduced they are pruned down to their correct size. Unlike the composite learner of Ting (1994), exemplar generalisation does not require an explicit definition of small disjuncts. This allows a gradual change in representation from large to small disjuncts. This suggests the following hypothesis.

Hypothesis 1: Generalised exemplars improve the performance of nearest neighbour systems by improving the representation of large disjuncts.

3.1.2 Rule induction

A very desirable feature of rule induction systems is that in addition to performing classification, they can be used to summarise data. The induced rules can then be applied to future cases by either a person or an expert system. The rules must be kept simple, and capture human-understandable concepts. Instance-based learning does not

```
class A if    f1 = (1 or 4 or 9 )            AND
              f2 = (23)                      AND
              (f3 >= 11 AND f3 <= 45)        AND
              f4 = (a or d)
```

Figure 3-2. Example of a hyperrectangle

do this. The knowledge learned by an instance-based system is understandable only by it. Generalised exemplars, however, can be converted into modular rules that other systems can use.

The form of the rules produced depends on the method used to generalise the exemplars. If it allows exemplars to nest within each other, the rules produced must be interpreted as defaults and exceptions. On the other hand, if the generalised exemplars are non-overlapping, the rules form a set of mutually exclusive conjunctive disjuncts. If exemplars overlap, the resulting rules will conflict with each other, and therefore be more difficult to interpret. This thesis proposes the following hypothesis.

> *Hypothesis 2: Producing exclusive generalised exemplars results in a useful set of rules that may be compared with those produced by other rule induction methods.*

3.1.3 Speed

A classification system performs two tasks: learning and classification. Rule inducing methods put considerable effort into the first task. Classification is very simple, as the system need only test a new example against each rule. The drawback is that the system cannot classify the new example if it does not fit any of the rules. In this situation a heuristic is adopted, such as

> "If the new example does not satisfy any rules, choose the class with the highest *a priori* probability."

In instance-based learning the effort required to learn from a new case is roughly equal to that required to classify it. The learning effort is very small compared to many other approaches. Furthermore, instance-based learning has the advantage that a conclusion is drawn for every new example, without needing a heuristic to handle examples that do not fit the learned rule set. The disadvantage is that the effort required to classify a new example is much higher than that of rule-based systems. This effort can become untenable if the number of cases in memory grows too large, or the number of features is very large.

There are several methods for pruning the number of examplars. One approach is to retain exemplars only when examples misclassify correctly (Aha, 1992). The assumption is that if an instance classifies correctly, it is a typical example and is already adequately represented. This is not always the case, however, because the incremental nature of instance-based learning systems means that an example may appear that would have caused the discarded example to classify incorrectly had it existed in the database at the time. Aha (1992) verified this notion experimentally; a decrease in classification performance is commonly observed when exemplars are pruned in this way.

Generalised exemplars offer an alternative method for reducing classification effort. We expect this decrease for two reasons:

- generalising exemplars decreases the number of exemplars in memory;
- comparing the new example against a generalised exemplar is simpler than computing a distance function.

Unlike exemplar pruning, exemplar generalisation does not involve an explicit loss of information, so classification performance need not suffer. However, the time taken to teach the system increases. This suggests the following hypothesis.

Hypothesis 3: Generalising exemplars reduces classification time without sacrificing accuracy.

3.2 Nested generalised exemplars

Salzberg (1991) describes a method of learning using nested generalised exemplars (NGE). “Nested” means that exemplars may be completely contained within one another, analogous to default rules with exceptions. Furthermore, generalised exemplars may overlap one another.

The learning process begins with a database containing a small number of seed exemplars. This seeding is necessary to prevent all examples of the same class being generalised into a single rectangle, and covering most of the problem space. Salzberg reports that the number of seed exemplars and whether they are closely packed or widely spaced is not important. In fact, the number and distribution of seeds does effect classification performance, as they affect the amount of overgeneralisation that occurs. Section 3.3.2 discusses this further.

NGE then classifies each new example using the current exemplar database and a Euclidean distance function that returns the class of the single nearest neighbour. The distance function handles hyperrectangles by returning a distance of zero if the new example falls within a hyperrectangle. Because NGE allows hyperrectangles to nest and overlap, an example may fall within more than one hyperrectangle. In this case a heuristic is used; NGE returns the class of the hyperrectangle covering the smallest area of feature space. For discrete features, the distance function is set to zero if two features match, and one if they do not. For hyperrectangles, the feature matches if it appears in the list of covered values.

The distance function is further modified by two dynamic weights. The first, W_H, weights exemplars according to the observed accuracy with which they make predictions, while the second, W_i, weights features according to their importance. The complete distance function is

$$D_{EH} = W_H \sqrt{\sum_{i=1}^{m} \left(W_i \frac{E_i - H_i}{\max_i - \min_i} \right)^2}$$

where E_i is the i th feature value in the example and H_i is the i th feature value in the exemplar. The feature difference $E_i - H_i$ for ungeneralised exemplars is the difference between the feature value of the example and that of the exemplar, while for hyperrectangles it is defined as

$$E_i - H_i = \begin{cases} E_i - H_{upper} & \text{when} \quad E_i > H_{upper} \\ H_{lower} - E_i & \text{when} \quad E_i < H_{lower} \\ 0 & \text{otherwise} \end{cases}$$

where H_{upper} and H_{lower} are the boundaries of the hyperrectangle for this feature. The exemplar weight W_H is

$$W_H = \frac{p+n}{p}$$

This weighting scheme penalises poorly performing exemplars, which may represent noisy cases or over-represented small disjuncts.

The feature weight W_i is initialised to one and increased or decreased according to the following rules:

- for correct predictions, increase W_i if this feature in the new exemplar did not match that in the chosen exemplar, otherwise decrease it;
- for incorrect predictions increase W_i if this feature matched, otherwise decrease it.

Both of these schemes are dynamic, the weights being updated after classifying each new example. Section 4.4 explains how they can be improved.

After classifying the new example, NGE attempts to generalise one of the existing exemplars to cover it. If the prediction was correct, the nearest neighbour is generalised to include the new example. If it is a hyperrectangle, it is grown to cover the new example. If it is a single exemplar, a new hyperrectangle is created that covers the nearest neighbour and the new example. If the prediction was incorrect, NGE employs a fallback heuristic in a further attempt to perform generalisation. This means that if the nearest neighbour classifies incorrectly, the second best neighbour is tried. If this exemplar classifies correctly, NGE generalises it to cover the new example. Furthermore, if the incorrect best match was a hyperrectangle, its boundaries are altered so that it shrinks away from the new example and is no longer the closest. If neither the best nor the second-best exemplars are of the correct class, no generalisation is performed.

The second-chance heuristic promotes nesting of hyperrectangles. If an example falls within a rectangle of the wrong class that already contains an exemplar of the same class, NGE generalises the two into a new hyperrectangle, nested within the other.

Salzberg also experimented with a greedy version of the algorithm that does not include the fallback heuristic. According to Salzberg it cannot create nested hyperrectangles because any examples that fall within a generalised exemplar of the wrong class do not get generalised into a nested rectangle, although it can still store exceptions by nesting ungeneralised exemplars within hyperrectangles. Hyperrectangles may also overlap as they are not checked after generalisation to see if they have overlapped with their neighbours. Section 3.3.2 discusses this further.

Salzberg (1991) tested NGE on three domains: Breast Cancer, Iris Flowers and Echo cardiogram. In all three cases it performed well, producing results comparable to other machine learning algorithms. Additionally, the amount of memory required was also quite small, typically storing about 10% of the total number of examples, showing that NGE can compress datasets quite well.

3.3 Non-nested hyperrectangles

The results presented by Salzberg (1991) suggest that NGE is capable of learning a variety of different problems, and of classifying new examples with a high degree of accuracy. However, a later study shows that these results were fortuitous, and that for many other domains the performance of NGE is poor (Wettschereck and Dietterich, 1994). This Section explores the idea that overgeneralisation caused by allowing hyperrectangles to nest or overlap is the reason for NGE's poor performance on many domains. The solution proposed is to avoid all forms of overgeneralisation by never allowing exemplars to nest or overlap.

3.3.1 The case for preventing nesting

One of the key features of NGE is the nesting of generalisations. This is equivalent to representing concepts by default and exception conjunctive rules, nested within each other to any number of levels. An example that falls into both a default and an exception rule takes the class of the exception. NGE implements this by returning the class of the hyperrectangle that covers the smallest area.

Representing rules with exceptions is not new. As described earlier, CYRUS (Kolodner, 1984) and UNIMEM (Lebowitz, 1987) both represent knowledge as rules with exceptions. EDAGs (Gaines, 1994) are another representation of exception rules.

PROTOS (Bareiss and Porter, 1988) uses difference links to represent exceptions. The success of these systems suggests that exception rules are useful.

The problem with NGE is that the generation of exceptions is completely uncontrolled. In the absence of seed exemplars, NGE may generalise the entire problem into a single hyperrectangle per class. Even with seed exemplars, the set of rectangles obtained depends heavily on the order of the examples; in the worst case NGE will generalise two very distant examples into a single large rule with many exceptions, even though this representation may be inappropriate.

CYRUS and UNIMEM overcome this problem by determining a point at which the default-with-exceptions model no longer fits, and performing local reorganisation. If the proportion of examples contained within a generalisation that are also contained within exceptions grows too high, the generalisation is dropped. For generalised exemplars this is the same as dividing a hyperrectangle into many smaller ones when the number of examples contained within nested rectangles grows too large. This approach does not entirely cure the problem, however, because it is a local solution; CYRUS and UNIMEM may divide a rule into many smaller ones to cure overgeneralisation, when in fact a better solution might be to specialise the local rule while generalising some other rule to encompass the removed portion. It may be necessary to rebuild the generalisation hierarchy completely from the top down.

Overgeneralisation of conjunctive rules has an adverse effect on classification because early in the learning process default rules may be generated for the wrong class, and so the model will perform very poorly. If this situation is not corrected, the model will always generalise new cases to the wrong class, and so performance will never improve. CYRUS and UNIMEM correct this problem to some degree, but NGE does not. Instead, the model improves very slowly as the number of exceptions rises.

Overgeneralisation is more serious in nearest neighbour systems such as NGE than in rule-based systems. A considerable advantage of nearest neighbour over rule inducing systems is that for areas of the problem space for which no examples have yet been observed, the nearest neighbour representation retains information about the similarity between the new example and its neighbours, and is able to make an informed prediction of the class. For non-nested rules, if an example falls outside the current rule set, at best a rough prediction is made based on global probability of class. It is in these situations that nearest neighbour can significantly outperform rule-based systems.

Generalised exemplars are really embedded conjunctive rules. Once an example falls inside a hyperrectangle, the distance function no longer holds, because the distance to the hyperrectangle is zero. If an example falls inside more than one hyperrectangle, the distance to all of these is zero. Deciding which rectangle to use becomes arbitrary, a popular method being to choose the smallest rectangle, which is analogous to choosing the exception over the default rule.

The biggest problem occurs within the default rectangle. Here the distance function is zero to the rectangle, and so points that fall within it but not within an exception always classify to the default rule. The nearest neighbour representation therefore no longer applies. This is desirable if most points within the default rule are of the default class, but classification performance is adversely affected if the system has arrived at this representation arbitrarily due to the order of the input data. In this case it has arbitrarily dropped the nearest neighbour representation, even though it may be the best one for the problem.

Figure 3-3(a) illustrates a set of examples for which nesting does not appear to be an appropriate representation. The shaded area represents a rule for the class *white*. Given the small number of exemplars of this class, it is over-represented by the rule, and non-nested exemplars, as in Figure 3-3(b), appear to be a better representation.

The extreme case is when a very small disjunct is nested within a large rule. Suppose a disjunct is represented by a single exemplar. Before being generalised, the example is stored verbatim within the larger rectangle. Because the distance within the rectangle is always zero with respect to it, all points within the default rule (including the exception point) will have a distance of zero with respect to the default rule. The single exception point, on the other hand, has size zero, and so the only point that will has a distance of zero with respect to this one is the example itself. Points in the area immediately surrounding it will therefore classify to the default rule no matter how close they are. In the absence of duplicate examples or missing values this point will never classify new examples, and so the small disjunct that it represents

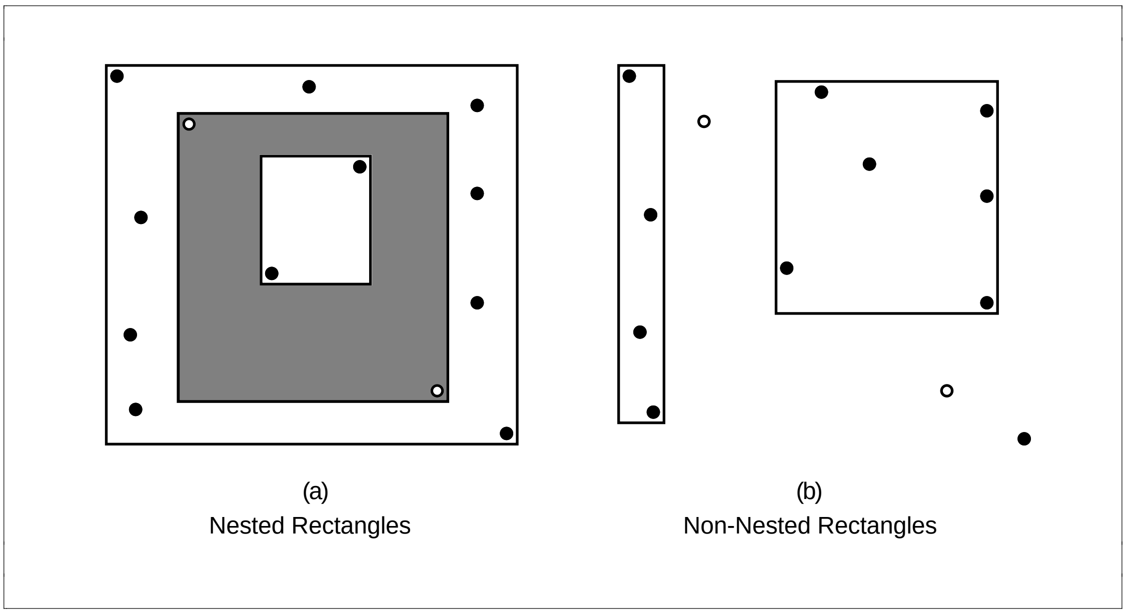

Figure 3-3. Nested vs non-nested exemplar representations

has been effectively discarded. Figure 3-4(a) illustrates this case. The new example X, although very close to a point of class *black*, is still classified as *white* because it lies within a rule for that class. Preventing nesting of rectangles, as in Figure 3-4(b), overcomes this problem. By not allowing exemplars to nest, single-point disjuncts do not fall within default rectangles, and so can compete with larger disjuncts using the distance function.

Salzberg(1991) argues that exceptions are necessary, and that the absence of the ability to store exceptions is a serious handicap to learning. While this may be true for some machine learning methods, nearest neighbour methods are able to represent exceptions without doing so explicitly. In particular, non-nested generalised exemplars have no problem representing exemplars implicitly. In Figure 3-5, nested generalised exemplars (a), non-nested generalised exemplars (b), and ungeneralised exemplars (c) are all able to represent the default and exception rules implicit in the data.

The arguments given in this section suggest the following hypothesis.

> *Hypothesis 4: A learning system using non-nested generalised exemplars shows better classification performance than one using nested generalised exemplars.*

3.3.2 Nesting vs overlap

A detailed study of NGE found that in general it does not classify new examples very well, and attempts to find the reasons for its poor performance (Wettschereck and Dietterich, 1994). Although the results of Salzberg's testing look promising, when tried on a greater variety of datasets it is found that NGE performs considerably worse than simple nearest neighbour in most cases.

Wettschereck and Dietterich tested three different configurations of NGE; NGE_3, NGE_{cv} and NGE_{limit}. Each uses a different number of seed exemplars: NGE_3 uses three

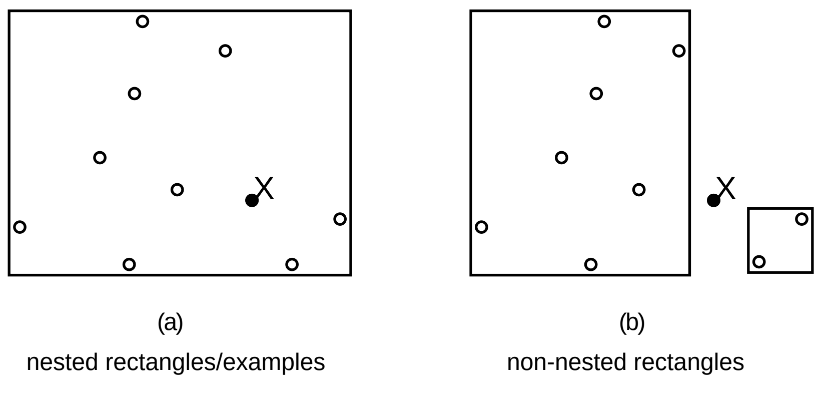

Figure 3-4. Small disjunct within a large rule

seed exemplars, NGE$_{CV}$ uses a variable number of seeds determined by cross-validation, with candidate numbers of seeds being 3, 5, 7, 10, 15, 20 and 25, and NGE$_{limit}$ uses seeds whose number equals half the number of training examples. The latter is included because a hyperrectangle requires twice the memory of a single example, and so with the number of seeds set to half, the rest should generalise into the seed examples. The overall memory requirement will be the same as if no generalisation had been performed. This result reflects the performance of NGE when it uses the same amount of memory as standard nearest neighbour but includes generalisation.

Wettschereck and Dietterich note that in contrast to Salzberg's reported findings, the performance of NGE depends strongly on the number of seeds, and that when this becomes large compared to the overall number of training examples, NGE begins to approximate a pure nearest neighbour system.

Limiting the generalisation that NGE can perform by adding more seeds deviates from the philosophy of creating the minimum number of hyperrectangles that classify the training set correctly. Therefore, the results for NGE$_{limit}$ and NGE$_{cv}$ do not reflect the performance of nested generalised exemplars. If nested generalisation is desirable, NGE should not require any seeds at all. NGE$_3$ is therefore the best representative of the model. This being the case, the results indicate that it could be improved, as the performance of NGE$_3$ is significantly inferior to standard nearest neighbour in all cases.

Wettschereck and Dietterich propose the following three hypotheses for why NGE preforms so badly:

- H1: nested rectangles cause poor performance;
- H2: overlapping rectangles cause poor performance;
- H3: poor performance is caused by deficiencies in the search algorithm.

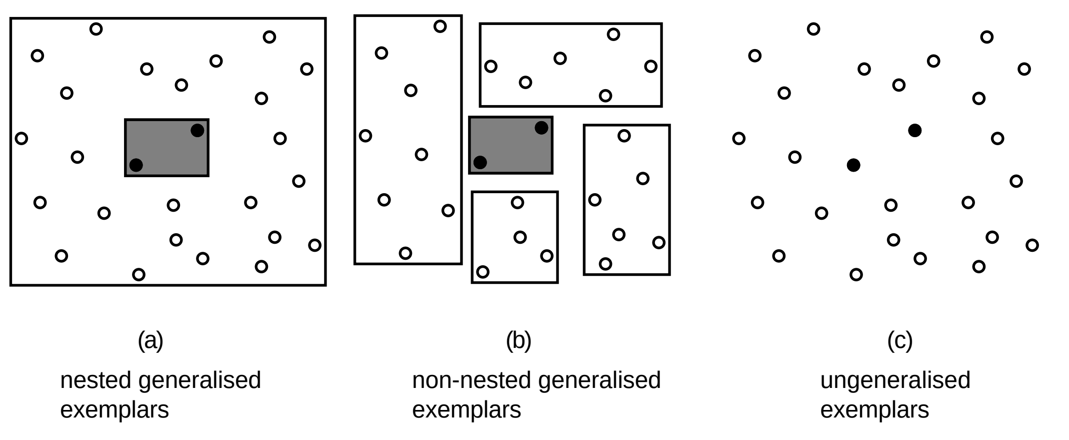
(a)
nested generalised exemplars
(b)
non-nested generalised exemplars
(c)
ungeneralised exemplars

Figure 3-5. Three representations of an exception

To test hypothesis H1, they use Salzberg's greedy version of NGE, which does not look for the second-best exemplar if the first proves to be incorrect. This avoids a second hyperrectangle growing within the first which, it is claimed, prevents nesting. However, Wettschereck notes that greedy NGE does produce nesting from time to time.

If a single exemplar is considered a trivial hyperrectangle, nesting occurs when a new example falls inside a hyperrectangle of a different class. Standard NGE generalises this example to the nearest exemplar of the same class, which may also exist within the conflicting hyperrectangle, while greedy NGE stores this new exemplar ungeneralised within the conflicting hyperrectangle. It therefore still produces nesting, because single exemplars are stored within conflicting hyperrectangles.

In fact, this means that the effects of nesting in greedy NGE can be worse. Since a single exemplar is the same as a hyperrectangle with size zero, an exemplar stored verbatim within a conflicting hyperrectangle is effectively discarded. In Figure 3-6(a) NGE correctly classifies the new example (X) to *black* while greedy NGE (b) misclassifies the new point to *white*. An exception to this is when missing values are present, because a new example with missing values may match a single exemplar. Salzberg uses greedy NGE to improve classification performance for the Echo cardiogram database, which is the only one tested that contains missing values.

Wettschereck reports that there is no support for hypothesis H1. Given that the version of NGE used to test it does not exclude nesting, and that in some cases it makes the consequences of nesting more serious, this result is not valid.

The main conclusion reached by Wettschereck is that overlapping rectangles cause NGE's poor performance. Hypothesis H2 is tested using NONGE, a version of

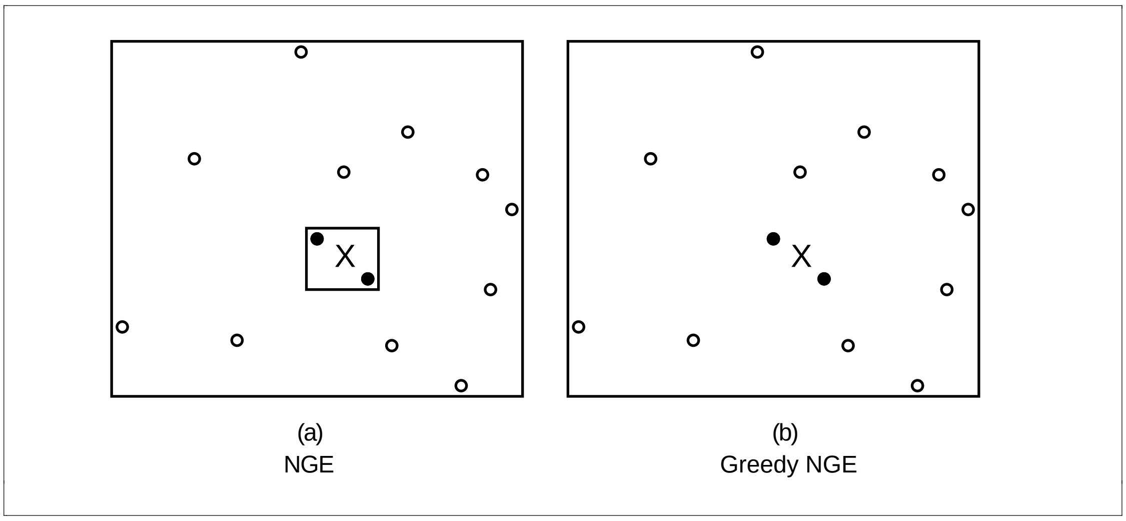

Figure 3-6. Standard vs greedy NGE

NGE that allows nesting but not overlap. The description of this algorithm indicates that a generalisation is accepted only if the new generalised exemplar would be either exclusive of other exemplars or contained within another generalised exemplar. It is not clear what happens when a newly generalised exemplar may cover another exemplar. If NONGE does not permit this case, one element of nesting is being prevented.

When obtaining the results for NONGE, Wettschereck applies cross-validation to determine the number of seeds to use. Given that overlapping has been removed, the only type of overgeneralisation possible is nesting. Since the number of seeds determines the extent to which overgeneralisation can result, this optimises the amount of nesting that occurs. The results gained from testing NONGE are therefore inconclusive.

Finally, nesting is merely a special case of overlapping (see Figure 3-7). We would therefore expect NONGE to perform much better than NGE. Overall Wettschereck's results indicate that reducing the amount of overgeneralisation increases the performance of NGE, which supports this thesis. Nesting default and exception conjuncts does not necessarily reduce classification performance, but overgeneralisation caused by *ad hoc* nesting of rectangles has an adverse effect. The best representation of a particular problem may contain nested conjuncts, but NGE is unlikely to find it.

3.4 Other generalisation methods

Creating hyperrectangles is just one possible method for generalising exemplars. In practice, we may use any generalisation technique. For example, an instance-based learner might use ID3 (Quinlan, 1986) to produce a decision tree, and convert each leaf node into a generalised exemplar of the form shown in Figure 3-8. A_1 and A_3 are decision nodes in the induced tree.

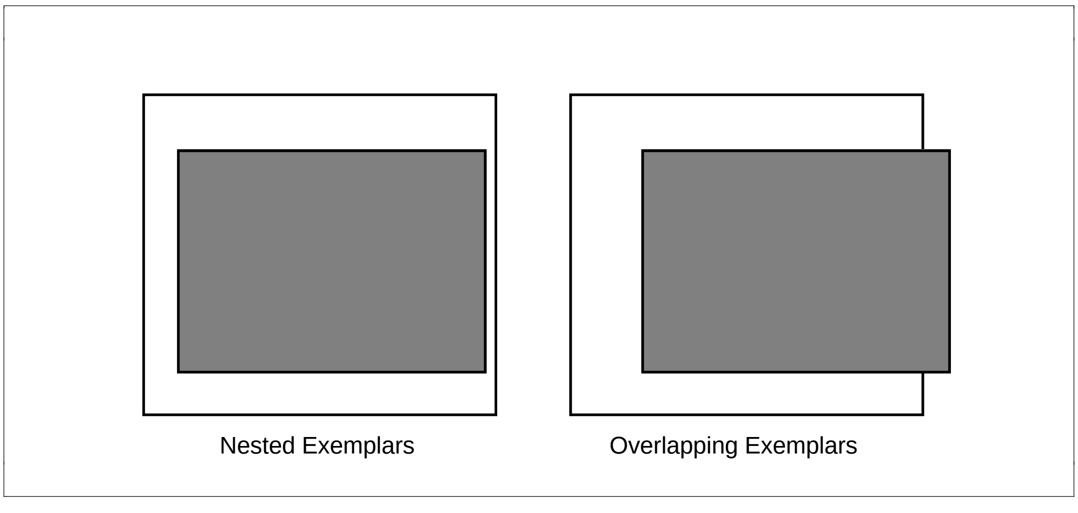

Figure 3-7. Nested vs overlapping exemplars

The problem is that this still adopts the generality bias of ID3, and so does not solve the problem of representing small disjuncts. However, it does solve the problem of what to do when a new example does not satisfy any rules, by applying the distance function.

Conceptual clustering algorithms are also possible candidates. The most general clusters that contain examples of a single class may be used as generalised exemplars. Like ID3, the problem is still that the bias of this type of learning scheme is too general.

An interesting alternative is hypersphere generation (Smith *et al*, 1990). Instead of recording a range of values for each feature, a hypersphere has a centre *c* and a radius *r*. Examples whose distanced from the centre is less than *r* fall within the hypersphere. The centre and radius are determined by computing the mean of each feature and the standard deviation in values for all features. While this approach may work well for some domains it has not been extensively tested, and is restricted to numeric domains.

In summary, we have chosen hyperrectangle generation as described as the method for exemplar generalisation because it represents a bias somewhere between the specificity of nearest neighbour and the generality of rule induction. As with rule induction methods it uses the well understood technique of partitioning the problem space using axis-parallel hyperplanes, allowing the results to be compared to other popular machine learning techniques without the added complication of a different concept representation.

Class = p if	A_1 = 1	AND
	A_2 = any value	AND
	A_3 = 4	AND
	A_4 = any value	AND
	A_5 = any value	

Figure 3-8. Exemplar generalised by ID3

4 Generalising using non-nested hyperrectangles

This chapter introduces Non-Nested Generalised Exemplars (NNGE), a novel algorithm that generalises exemplars without nesting or overlap. NNGE is an extension of NGE (Salzberg, 1991), which performs generalisation by merging exemplars, forming hyperrectangles in feature space that represent conjunctive rules with internal disjunction. NNGE forms a generalisation each time a new example is added to the database, by joining it to its nearest neighbour of the same class. Unlike NGE, it does not allow hyperrectangles to nest or overlap. This is prevented by testing each prospective new generalisation to ensure that it does not cover any negative examples, and by modifying any generalisations that are later found to do so. NNGE adopts a heuristic that perfoms this post-processing in a uniform fashion.

In contrast, NGE (Salzberg, 1991) generalises when the nearest exemplar has the same class as the new example. If it does not, the next closest exemplar is examined. If this one has the correct class, it is generalised. This heuristic permits a reasonable level of generalisation while preventing the gross overgeneralisation that would occur if every example was generalised to its nearest neighbour of the same class, even though that exemplar may be far away in feature space with many intervening negative examples. Because NNGE prevents overgeneralisation by correcting any overlap or nesting, it does not require this "second chance" heuristic. Instead it always tries to generalise new examples to their nearest neighbour of the same class, but if this is immediately impossible due to intervening negative examples, no generalisation is performed. If a generalisation later conflicts with a negative example, it is modified to maintain consistency.

4.1 Algorithm summary

NNGE learns incrementally by first classifying, then generalising each new example. It uses a modified Euclidean distance function that handles hyperrectangles, symbolic features, and exemplar and feature weights. Numeric feature values are normalised by dividing each value by the range of values observed. The class predicted is that of the single nearest neighbour. NNGE uses dynamic feedback to adjust exemplar and feature weights after each new example is classified. When classifying an example, one or more hyperrectangles may be found that the new example is a member of, but which are of the wrong class. NNGE prunes these so that the new example is no longer a member.

Once classified, the new example is generalised by merging it with the nearest exemplar of the same class, which may be either a single example or a hyperrectangle. In the former case NNGE creates a new hyperrectangle, wheras in the latter it grows the nearest neighbour to encompass the new example.

Overgeneralisation, caused by nesting or overlapping hyperrectangles, is not permitted. Before NNGE generalises a new example, it checks to see if there are any examples in the affected area of feature space that conflict with the proposed new hyperrectangle. If so, the generalisation is aborted and the example is stored verbatim. Figure 4-1 summarises the complete algorithm.

4.2 Classifying a new example

NNGE classifies new examples by determining the nearest neighbour in the exemplar/hyperrectangle database using a Euclidean distance function. This function is modified slightly to enable it to compute the distance from hyperrectangles. The function (identical to that used by NGE) is:

$$D_{EH} = W_H \sqrt{\sum_{i=1}^{m} \left(W_i \frac{E_i - H_i}{\max_i - \min_i} \right)^2}$$

where E_i is the i th feature value in the example, H_i is the i th feature value in the exemplar, and W_H and W_i are exemplar and feature weights. The feature difference $E_i - H_i$ for ungeneralised exemplars is the difference between the feature value of the example and that of the exemplar, while for hyperrectangles it is defined as

$$E_i - H_i = \begin{cases} E_i - H_{upper} & \text{when} \quad E_i > H_{upper} \\ H_{lower} - E_i & \text{when} \quad E_i < H_{lower} \\ 0 & \text{otherwise} \end{cases}$$

where H_{upper} and H_{lower} are the boundaries of the hyperrectangle for this feature. For symbolic attributes, the distance is trivialised to

$$(E_i - H_i) = 0 \text{ if } E_i \text{ is in the exemplar/hyperrectangle}$$
$$(E_i - H_i) = 1 \text{ if } E_i \text{ is not in the exemplar/hyperrectangle}$$

Having computed the distance between the new example and all exemplars and hyperrectangles, NNGE chooses the class of the closest one. In the event of a tie, it chooses the class with the most exemplars at the minimum distance.

NNGE treats missing attributes by ignoring them; if the attribute is missing for either the example or the exemplar against which it is being compared, it does not contribute to the distance function. The final distance is divided by the number of non-missing attributes so that exemplars with all attribute values present are not penalised (Salzberg does not describe how NGE treats missing values).

```
While (more examples)
    read example
    store example
    adjust attribute ranges

    classify example:
        while (more rectangles)
            compute distance from new example to rectangle
            if (distance < min distance so far for this rectangle’s class)
                set class minimum distance to this distance
                set class count to 1
            if (distance = min distance so far for this rectangle’s class)
                increment class count by 1

        while (more ungeneralised exemplars)
            compute distance from new example to exemplar
            if (distance < min distance so far for this exemplar’s class)
                set class minimum distance to this distance
                set class count to 1
            if (distance = min distance so far for this exemplar’s class)
                increment class count by 1

        return class with lowest distance and highest count
        return first exemplar/hyperrectangle found with this class/distance

    adjust model:
        if (correct prediction)
            increment positive count for this exemplar/hyperrectangle
        else
            increment negative count for this exemplar/hyperrectangle
            if (exemplar falls inside a hyperrectangle of another class)
                prune this overgeneralised hyperrectangle
            else
                adjust weights for attributes with differing values

    generalise the new example:
        if (nearest neighbour was a hyperrectangle)
            extend each feature range to include the new example
            if (extended rectangle covers conflicting examples/rectangles)
                restore hyperrectangle to original size
                store the new example verbatim
            else
                retain modifications to the hyperrectangle
                discard example

        if (nearest neighbour was a single example)
            create a hyperrectangle that covers the two examples
            if (new rectangle covers conflicting examples/rectangles)
                discard the new hyperrectangle
                store the new example verbatim
            else
                retain the new hyperrectangle
                discard example
```

Figure 4-1. NNGE algorithm

4.3 Exemplar generalisation

NNGE always generalises an example to its nearest neighbour of the same class if possible. If this generalisation fails because the new rectangle covers a conflicting exemplar, no further attempt to generalise is made. In contrast, NGE adopts the second chance heuristic if the nearest exemplar does not have the correct class.

4.3.1 Creating /growing hyperrectangles

NNGE stores hyperrectangles as exemplars with an extended format for storing feature values. For continuous features, minimum and maximum values are stored which describe the range of values covered by the hyperrectangle. For symbolic features a linked list is maintained for each feature listing the feature values covered by the hyperrectangle. Figure 4-2 presents the data structure for a hyperrectangle. *Symbolic_attributes* is an array containing a pointer for each symbolic attribute to its linked list of feature values, and *numeric_attributes* is an array of minimum and maximum values for numeric attributes.

To add a new exemplar to a hyperrectangle, the features that do not already overlap with the new example are extended. For numeric features the minimum is decreased or the maximum increased so that the range of values now covers the new value. For symbolic features the new feature-value is added to the list of values if it is not present already.

4.3.2 Preventing overgeneralisation

Overgeneralisation is prevented in two ways: by checking all generalisations for conflict before implementing them, and by detecting unavoidable conflicts and remedying them.

```
struct rectangle
{
        char                            class;
        int                             positive_count;
        int                             negative_count;
        struct symbolic_attribute
        {
                char                            value;
                struct  symbolic_attribute      *next;
        } *symbolic_attributes[MAX_ATT];

        struct numeric_attribute
        {
                int                             min;
                int                             max;
        } numeric_attributes[MAX_ATT];
} ;
```

Figure 4-2. Hyperrectangle data structure

Before growing a hyperrectangle, NNGE checks the new part to see if it covers any conflicting examples or overlaps any other rectangles. If either of these conditions is met, it abandons the generalisation and stores the new example as a single exemplar. When a hyperrectangles is created, it is checked in the same way. If a conflict exists, NNGE rejects it in favour of retaining two exemplars.

The incremental nature of NNGE means that examples may be introduced to the system that conflict with one or more existing hyperrectangles. NNGE checks this condition as each new exemplar is presented. If a hyperrectangle of the wrong class covers the new example, it is shrunk so that it no longer does. This operation is performed before the new example is generalised.

To exclude the new example from the offending hyperrectangle, it is necessary only to reduce the size of the hyperrectangle along a single feature axis. For symbolic features NNGE adopts the following heuristic to decide which feature to trim:

> *Select the feature for which the feature-value to be removed predicts the class of the hyperrectangle the least well.*

In other words, NNGE checks each feature-value of the conflicting new example to see what proportion of examples with that feature-value have the same class as the hyperrectangle, and removes the one with the smallest count.

For continuous domains it is not practical to check individual values. Instead, the range between the two adjacent examples along the axis under test is used, since this will determine the two new boundaries of the shrunken hyperrectangle. Note that if the conflicting point is in the middle of the hyperrectangle along this axis, NNGE will divide the hyperrectangle rather than shrink it. Figure 4-3 illustrates an example of splitting along a numeric feature. The dashed rectangle represents the class *black*. When the *white* example is introduced, the generalisation becomes inconsistent. To overcome this, the rectangle is divided, the inner boundaries are set at the next closest examples along the *x* axis (labelled *a* and *b*), and the resulting pair of hyperrectangles, shown in solid lines, are resized to fit the examples that they cover exactly.

4.4 The distance function

Like many nearest neighbour systems, NNGE uses dynamic feedback to alter its view of the exemplar database. This is mainly to overcome problems of the exemplar database not accurately representing the underlying information, such as noisy data, missing data, and insufficient examples to represent accurately actual population densities. Modifying the Euclidean distance function to cope with variable exemplar reliability and feature importance leads to an overall improvement in classification performance. There are many ways of achieving these aims. The methods chosen are now discussed.

4.4.1 Dynamic feature weighting

A major problem with systems that use a Euclidean distance function to determine similarity is that the scale of each attribute is unknown. A difference of 0.1 of the range of possible values may be insignificant in attribute *a*, but very important in attribute *b*. Combining symbolic and numeric attributes exacerbates this problem. How far apart must two numeric values be before the difference is as important as two dissimilar symbolic values?

NNGE, like NGE, uses dynamic feature weighting to overcome this problem. Each time it classifies a new example, the weight for each attribute is modified according to whether or not the classification was correct, and whether or not this attribute had the same value for the new example and the nearest neighbour. This is a common method of scaling attributes.

NGE (Salzberg, 1991) uses a very simple dynamic weighting scheme to alter the scale of each attribute in relation to the others. If the new example classified correctly, it increases the weight of like attributes, and decreases that of unlike ones. If it classified incorrectly, the reverse is true. While this scheme is better than not weighting attributes at all, it has drawbacks. If the classification performed was correct, it is reasonable to assume that the weights are correct, so there is no reason to change them. Furthermore, if an attribute has like values, altering its weight will have no effect on the classification decision. Altering unlike values, on the other hand, will alter the distance, and so may influence the classification decision.

NNGE adopts a weighting scheme that takes these two points into account. If it classified a new example correctly, it leaves the weights unchanged. If it classified incorrectly, it increases the weights for those attributes that differed, accentuating the difference.

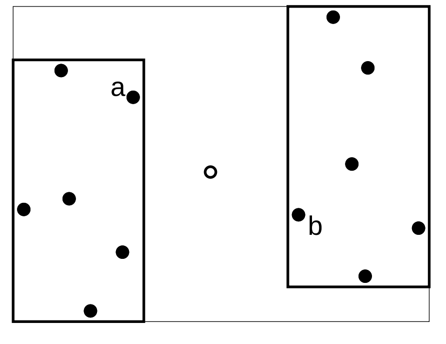

Figure 4-3. Splitting a hyperrectangle along a numeric feature

NGE (Salzberg, 1991) is very sensitive to the amount *d* that it alters the weights at each step. For example, for the Iris domain 1.0 is the best weight for each attribute, and performance degrades markedly as *d* increases. NNGE performs very well for any reasonable value of *d*, regardless of domain. This suggests that NGE performs unnecessary weight adjustments that have a detrimental effect on classification. In particular, for the Iris domain, classification performance is very high from the beginning. NGE continuously modifies the weights even when the system is classifying correctly. In contrast, NNGE modifies them only when a mistake is made, and so the weights for the Iris domain remain more stable over the learning period. NNGE performs very well for the Iris domain with any value of *d* between 1.0 and 1.5, supporting the validity of the modify-if-wrong approach to weight adjustment.

Both schemes assume that the scale for each attribute is linear, which may not be the case. For a numeric feature, the distance between any two values might have a hyperbolic effect on the similarity measure, for example. Also, along any particular axis there may be collections of values that can be considered similar for the purpose of classification, while others are very different. There may be some areas where the value of this attribute is completely irrelevant.

This problem is solved in part by the nature of the instance-based approach. Given a sufficient number of examples, an area where the values of a feature can be considered the same for classification should produce a cluster of points of the same class. An area where the attribute is completely irrelevant should produce an even spread of points of all classes. Performance in these regions will therefore depend on how well the distribution of exemplars models the spread of examples over the entire domain. We would therefore expect classification performance in these two cases to improve as the number of exemplars increases.

Generalised exemplars help the first case, by producing rectangles that cover areas of like classification, thereby increasing the significance of the attribute over the range where points are clustered. During the early stages of learning when the number of points is low, gaps in these areas will be filled in by generalisation. On the other hand, overgeneralisation will almost certainly occur due to the absence of negative examples to prevent the growth of over-sized rectangles.

For symbolic attributes the problem is more complex. Given possible values of A, B, and C, the relative distances between A and B, B and C, and A and C are unknown. Generalising the exemplars does not address this problem, although it reduces its effect by reducing the proportion of cases in which the distance function determines the class.

One solution is to construct a value difference metric (Stanfill and Waltz, 1986), which records a computed difference between each value and each other value. PEBLS (Cost and Salzberg, 1994) achieves this by making two passes over the data.

During the first pass, it uses Bayesian probability to compute the value distance metric for all possible pairs of attribute values. During the second, it adopts these metrics to aid classification and learning. This approach is very successful, but suffers from a lack of incrementality, one of the major advantages of instance-based learning.

In a complex problem space requiring many rules to represent the learned concept, the importance of each attribute will vary depending on the rule. A system that assumes that attribute importance is constant imposes a restriction upon the rule sets that it can learn (Cendrowska, 1987), and will perform poorly on any problem for which this criteria is not met. Similarly, a weighting system that assigns a single weight to each attribute will suffer from this restriction.

Generalised exemplars reduce this problem. Consider the two-attribute example illustrated in Figure 4-4(a). In the absence of any weighting scheme, the new point appears closest to a point of class *b*. The layout of the points, however, suggests that the correct classification for the new point is *a*. Weighting of the attributes will probably produce good results in this simple case. Generalisation of the points, however, can exactly capture the desired concept, leaving only the area between the *b* and *c* clusters and the area to the left of *b*—about which nothing is known—as areas of unknown class, as shown in Figure 4-4(b). The need to weight attributes differently by rule is now less important.

The effect of a single incorrect classification on the weighting scheme should reduce as more examples are introduced into the exemplar database. NNGE and NGE (Salzberg, 1992) both adjust the weights by a fixed amount each time, so the effect of a single incorrect classification late in learning is as drastic as one early on. Salzberg notes that this can lead to oscillation, and suggests altering weights for a short period only, after which they remain fixed. In practice, the feature weight methods adopted by NNGE and NGE both perform well on some domains, and poorly on others, indicating that neither are ideal. More elaborate dynamic feature-adjustment methods

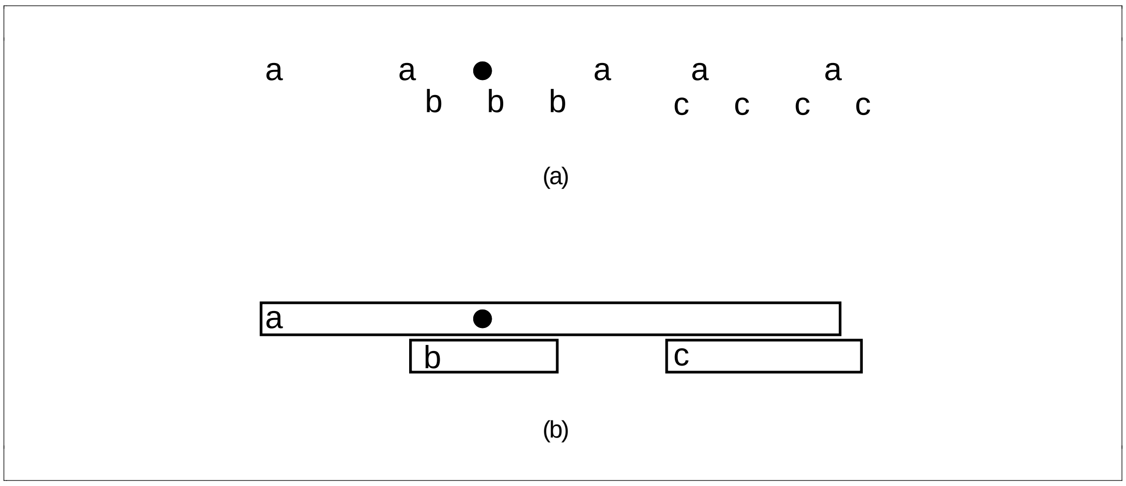

Figure 4-4. A two-attribute example

perform better than either of those described above (Aha 1992, Wettschereck *et al*, 1994).

4.4.2 Exemplar reliability

Exemplar weighting provides a method for weeding out those exemplars containing noise. Noisy examples degrade classification performance by matching incorrectly to new examples, or by returning an erroneous class.

Also, the examples given to the system will often not represent the underlying information in terms of what proportion of examples lies within each region of the problem space. In particular, the input set is often devoid of duplicates, even though duplicates may occur frequently. If an example being classified is equally close to two exemplars, it is reasonable to choose the more common one to determine the new example’s class.

NGE (Salzberg, 1991) handles this phenomenon, sometimes called *prototypicality*, by weighting the distance function according to the number of correct and incorrect predictions made by each exemplar. Bareiss and Porter (1988) determined experimentally that prototypicality is important to classification in instance-based learning systems.

A commonly used solution for screening out noise is to keep track of the number of correct and incorrect predictions made by each exemplar. This rests on two assumptions. First, the more common an exemplar is in real life, the more likely it is to lead to a correct prediction. Conversely, if an exemplar is very often the closest to the example being classified and if it leads to a large number of correct predictions, it is likely that this is because a large number of examples of this class lie within this region of problem space. Second, by tracking both the correct and incorrect predictions, exemplars that lead to incorrect predictions can be penalised. Such exemplars are likely to be either noisy or very atypical.

When generalising exemplars, this weighting scheme also gives an idea of the relative density of points within the generalisation space. The scheme employed in NGE (Salzberg, 1991) is

$$W_i = \frac{p+n}{p}$$

where p is the number of correct predictions made by this exemplar, and n is the number of incorrect predictions. The calculated distance is therefore divided by the ratio of correct to total predictions.

This weighting scheme has problems. In particular, if p is very small compared to n, as may occur at the start of the learning session, W_i rapidly becomes very large and so the exemplar is unlikely to ever make another prediction. NGE therefore considers only those exemplars that start well. Only when the weights for all of the others grow similarly large will the weight of a penalised exemplar be corrected again. The following scheme is proposed as an alternative:

$$W_i' = \frac{n}{p+n}$$

In the event of an exemplar performing badly at the start of the learning cycle, W_i' remains around one, while for those that classify well W_i' approaches zero. These exemplars, if erroneously rewarded, soon begin to accumulate negative predictions, and so return to a more realistic value.

In summary, NGE's weighting scheme penalises heavily, while NNGE's rewards heavily. A heavily penalised exemplar drops out of consideration and therefore is never corrected, while a heavily rewarded exemplar is considered more often, and so its weight is continuously refined.

New exemplars and generalisations must be given an initial value for *p* and *n*. NGE uses initial values of *p*=1 and *n*=0, giving a default weighting of 1/1. This gives each new exemplar the best possible weighting. It will therefore favour the new example over all previous exemplars until it has accumulated an appreciable number of positive and negative predictions to weight it more fairly.

A better method might be to give each new example an "average" weighting, say the average of all current exemplar weights. Cost and Salzberg (1994) suggest that the new exemplar be given the weighting of its nearest neighbour of the same class. The new exemplar therefore begins with a weighting from a point nearby in the problem space. This assumes that if a point in space has a particular prototypicality, the area close to it will be similar. NNGE adopts this initialisation method.

4.5 Summary

NNGE is a novel algorithm for generalisation of exemplars in nearest neighbour that implements several new methods. They are:

- it always tries to generalise new examples to their closest neighbour of the same class;
- if a new generalisation would conflict with other examples or hyperrectangles, it is not performed;
- if an existing hyperrectangle conflicts with a new example, it is pruned;
- it uses a new dynamic exemplar weight formula that rewards, rather than penalises, exemplars in memory;

- it uses a new dynamic feature weighting method that only alters the weights of incorrectly classifying exemplars.

The next chapter describes testing of the NNGE algorithm.

5 Experiments

In the field of machine learning, it is often difficult to compare results to those published by others because crucial details regarding the test environment are missing. The proportion of a dataset used to train and test the system can have a large effect on the result. Cross validation is a common form of testing, where many runs are performed, partitioning the input data set into various random training and validation sets. Leave-one-out testing has recently gained popularity as a method of doing this, whereas previously a 90%–10% split was more common. Results obtained from these two testing regimes are not directly comparable, because a system tested using the leave-one-out strategy has the advantage of a larger input dataset, and so can be expected to perform better. Often the proportion of examples used to train the system is not mentioned at all, and so the tests cannot be replicated.

The dataset itself may also have been modified. Classes may be removed because the tester feels that they are not adequately represented by the data, or they may be merged to reduce the problem to a two-class variety. The tester may merge, partition, or remove feature values in the interests of improving the performance of the classification system under test.

Finally, the learning system itself may be subject to tuning. C4.5, for example (Quinlan, 1993), prunes the resultant decision tree, this action being controlled by a set of parameters. A recent survey reports several published results for C4.5 for various standard datasets, the difference between the lowest and highest scores being typically 2% (Holte, 1993). While such results are useful for comparing the approximate performance of different learning systems, systems that perform similarly cannot be compared accurately.

5.1 Test method

NNGE was tested by validating the following four hypotheses:

- **Hypothesis 1:** Generalised exemplars increase the performance of nearest neighbour systems by improving the representation of large disjuncts;

- **Hypothesis 2:** Producing exclusive generalised exemplars results in a useful set of rules that may be compared with those produced by other rule induction methods;

- **Hypothesis 3:** Generalised exemplars reduce classification time without sacrificing accuracy;
- **Hypothesis 4:** A learning system using non-nested generalised exemplars shows better classification performance than one using nested generalised exemplars.

NNGE was tested empirically by using a selection of commonly used datasets and comparing the results obtained to those published for NGE (Salzberg, 1991), C4.5 (Quinlan, 1993), and the composite learner (Ting, 1994). For hypotheses one, two and four classification performance is being tested, while for hypothesis three it is classification speed. The learning system was trained on a subset of the total dataset, and then tested against the remaining examples, ensuring that the test set contained only previously unseen examples. Testing was repeated a fixed number of times, with a different split of the dataset being used each time.

The datasets used to test the hypotheses were all obtained from the UCI machine learning data repository[1]. Table 5-1 summarises the testing details for the domains used. Some domains were used twice because they were compared to the results reported in both (Ting, 1994) and (Wettschereck and Dietterich, 1994), under different conditions. Columns four and five give the number of training and test examples used when comparing results with NGE (Wettschereck and Dietterich, 1994). Columns six and seven give the same information for comparing results with the composite learner (CL) and C4.5 (Ting, 1994). The data contained within the examples was not altered in any way.

A set of 25 input files was produced from each data file. This was done to ensure that any differences in performance observed during testing were purely because of changes in the system under test, and not because of differences in the input data. Unfortunately, this is not the case when comparing with published results, as the actual datasets used are not known. The first file in each case is the orignal dataset from the UCI repository. The second is produced by taking the last 1/25 of the datafile and moving it to the top of the file. This process is repeated 25 times, giving 25 different input datasets.

NNGE was then run using each datafile. The first *t* examples were used to train the system, where *t* is given in columns four and six of Table 5-1, depending on which system the results are to be compared with. The system was then tested on the remaining examples. The number correctly classified was summed over the 25 runs, and divided by the total number of examples tested to give the classification performance for each domain.

[1] This repository resides at the University of California, Irvine. The data files may be obtained via anonymous FTP from ics.uci.edu, in directory ~/pub/machine-learning-databases.

Database Name	# classes	# features	NGE # train	NGE # test	CL # train	CL # test
Iris	3	4	105	45		
Led-7	10	7	180	20	180	20
Led-24	10	24	180	20	180	20
Waveform-21	3	21	300	100	270	30
Waveform-40	3	40	300	100	270	30
Cleveland heart disease	2	13	212	91		
Hungary heart disease	2	13	206	88		
Voting records	2	16	305	130		
Breast cancer-wisconsin	2	9			629	70
Promoter	2	57			96	10
Monks-2	2	6			169	432
Diabetes	2	8			691	77
Hepatitis	2	19			140	15

Table 5-1. Test domain details

5.2 Test domains

The datasets used to test the hypotheses presented in this thesis were all obtained from the UCI machine learning data repository. The following paragraphs summarise each dataset used. This information was also obtained from the data repository.

Iris The examples in this dataset represent 150 iris flowers, where each is described by four measurements of its dimensions, and is classified into one of three classes according to its variety. One of the classes, Setosa, is linearly separable, and the other two almost so. This domain tests the ability of the system to detect simple concepts. It is an exceedingly easy domain to learn, with many systems achieving around 95% accuracy.

Led-7 This artificial domain contains examples representing the set of ten decimal digits on an LED display. Each example contains seven boolean attributes—the seven segments of an LED display—and the class. The problem would be easy with one example per class being sufficient to describe the ten concepts, if it were not for the introduction of noise. Each attribute, excluding the class, has a 10% chance of being inverted. There are no missing values. The optimal Bayes classification rate for this database is 74%, compared to 71% for nearest neighbour and 72.6% for C4.5. Learning methods such as IB3 (Aha, 1992) that are very sensitive to noise perform poorly on this domain.

Led-24 This is the same as the above domain but with seventeen irrelevant attributes added, making it a much more difficult domain to learn. It tests the ability of the system to determine the usefulness of each attribute. Pure nearest neighbour systems do not do this, and so perform very poorly on this domain. IB1, IB2, and IB3

(Aha, 1992) for example, achieve a classification accuracy of 42 to 47% compared to 72% for C4.5, and the optimal Bayes classification rate of 74%.

Waveform-21 This dataset is similar to LED-7 but for a numeric domain. Each input example represents an artificially created waveform described by 21 numeric values, with 10% noise added. Two of three possible base waves are combined to form the initial wave, to which the noise is then added. Learning systems must be able to tolerate noise in numeric features to do well on this domain. Table 5-2 lists some reported classification performances (Breiman *et al*, 1984).

System	**Accuracy (%)**
CART	72
NN	78
Bayes	86

Table 5-2. Waveform-21 performance

Waveform-40 This domain is the same as waveform-21 but with nineteen irrelevant attributes added, making the problem much more difficult. Table 5-3 reports some previous results (Breiman *et al*, 1984).

System	**Accuracy (%)**
CART	72
NN	38
Bayes	86
C4.5	69

Table 5-3. Waveform-40 performance

Cleveland heart disease The examples in this database represent heart disease patients described by 75 attributes, but all published experiments use a subset of thirteen of them. Attributes are both continuous and nominal. The class identifies the presence of heart disease in the patient; it is an integer valued from 0 to 4. Experiments with the Cleveland database have concentrated on attempting to distinguish presence from absence, and so reduce the problem to a two-class variety. This is true of the data files used to test NNGE. The dataset contains several missing values. Table 5-4 lists some previously published results.

System	Accuracy (%)	Reference
NTgrowth	77.9	Aha, 1991
C4.5	75.4	Aha, 1991
CLASSIT	78.9	Gennari *et al*, 1989

Table 5-4. Cleveland performance

Hungarian heart disease This database is for the same problem as the Cleveland heart disease database, but with the data being provided from a different clinic. More values are missing than in the Cleveland database.

Voting records This dataset contains examples representing the votes cast for each of the US House of Representatives Congressmen on the sixteen key voting issues identified by the Congressional Quarterly Almanac, 1984. The CQA lists nine different types of votes: voted for, paired for, and announced for (simplified to *yea*), voted against, paired against, and announced against (simplified to *nay*), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (simplified to *unknown disposition*). The class identifies Democrat or Republican party affiliation. Approximately 4% of attribute values are missing. A typical classification performance for this database is 90 to 95%.

Breast cancer-wisconsin (bcw) The examples in this dataset represent 699 cancer sufferers. Instances are described by nine attributes, all of which are medical test results. Each instance has one of two possible classes, benign or malignant. Samples arrive periodically as clinical cases are reported. The database therefore reflects this chronological grouping of the data. This dataset can be partly represented by a single rule involving three attributes, which predicts unseen examples to around 95% accuracy. Failure to capture this rule results in very low classification accuracy. This rule was first reported by Mangasarian and Wolberg (1990), who noted that three pairs of parallel hyperplanes give a predictive accuracy of 95.9%.

Promoter The instances in this dataset represent E. coli promoter gene sequences. Each instance contains 57 nucleotides representing a portion of a DNA gene sequence. Each nucleotide may be one of four types, represented by a nominal value. Learning systems usually attempt to determine the length of gene sequence required to predict whether the gene sequence centred about a particular gene is E. coli promoting. Table 5-5 lists some previously reported classification performances for this dataset (Towell *et al*, 1990).

System	Accuracy (%)
KBANN	96.2
BP	92.5
O'Neill	88.7
NN	87.7
ID3	82.1

Table 5-5. Promoter performance

Monks-2 The Monks' problems were the basis of an international comparison of learning algorithms, summarised by Thrun *et al* (1991). One significant characteristic of this comparison is that it was performed by a collection of researchers, each of who was an advocate of the technique they tested, and often was the creator of the method. In this sense, the results are less biased than in comparisons performed by a single person advocating a specific learning method, and more accurately reflect the generalisation behaviour of the learning techniques as applied by knowledgeable users. There are three Monks' problems, the domains for all of which are artificial, containing six nominal attributes. Each domain contains a concept that requires a more complex representation than the usual conjunction of attribute values. For each problem the domain has been partitioned into a train and test set, the training set being a small subset of the test set. MONKS-2 is interesting because the target concept is one that cannot be represented by a small conjunctive rule set. The target concept is:

exactly two of {a1 = 1, a2 = 1, a3 = 1, a4 = 1, a5 = 1, a6 = 1}

The equivalent concept in conjunctive rules is:

{a1 = 1 ^ a2 = 1 ^ a3 ≠ 1 ^ a4 ≠ 1 ^ a5 ≠ 1 ^ a6 ≠ 1 } v
{a1 = 1 ^ a2 ≠ 1 ^ a3 = 1 ^ a4 ≠ 1 ^ a5 ≠ 1 ^ a6 ≠ 1 } v
{a1 = 1 ^ a2 ≠ 1 ^ a3 ≠ 1 ^ a4 = 1 ^ a5 ≠ 1 ^ a6 ≠ 1 } v
{a1 = 1 ^ a2 ≠ 1 ^ a3 ≠ 1 ^ a4 ≠ 1 ^ a5 = 1 ^ a6 ≠ 1 } v
{a1 = 1 ^ a2 ≠ 1 ^ a3 ≠ 1 ^ a4 ≠ 1 ^ a5 ≠ 1 ^ a6 = 1 } v
{a1 ≠ 1 ^ a2 = 1 ^ a3 = 1 ^ a4 ≠ 1 ^ a5 ≠ 1 ^ a6 ≠ 1 } v
{a1 ≠ 1 ^ a2 = 1 ^ a3 ≠ 1 ^ a4 = 1 ^ a5 ≠ 1 ^ a6 ≠ 1 } v
etc

Accordingly, rule inducers that attempt to discriminate attributes such as C4.5 (Quinlan, 1993) do not perform very well with this dataset, while nearest neighbour methods fare better.

Pima Indian diabetes Originally owned by the National Institute of Diabetes and Digestive and Kidney Diseases, this domain represents test results for 768 patients. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organisation criteria. The population lives near Phoenix, Arizona, USA. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. All attributes are numeric.

Hepatitis The original source of this database is unknown. It documents 155 hepatitis sufferers, classifying them according to whether they lived or died. Patients are represented by nineteen nominal valued attributes, one of which has been derived from a continuous value. This domain contains missing values.

5.3 Test results

This section reports the test results.

5.3.1 Hypothesis 1: Improved classification performance

To verify that nested generalisation leads to significant performance improvements, NNGE was tested against standard nearest neighbour, C4.5 (Quinlan, 1993) and the composite learner (Ting, 1994).

5.3.1.1 Generalised vs non-generalised exemplars

NNGE can be run with generalisation disabled. This provides a nearest neighbour system with identical characteristics. Doing so isolates the performance difference due to exemplar generalisation, because all other variables, such as exemplar and feature weights, are held constant. Thirteen domains from the UCI database were used for this purpose, giving wide variations in the number, type, and spread of attribute values. Some of the domains are artificial, and give insight into how NNGE performs when faced with problems such as noisy or irrelevant attributes. The Monks-2 domain illustrates the ability of both ungeneralised and generalised exemplars to represent a complex concept.

A batch version of NNGE is used for this and all subsequent tests except for comparisons with NGE. This is identical to the incremental version except that it considers all training exemplars before making each generalisation decision, rather than just those classified so far. Table 5-6 summarises the classification performance observed, and the same results are presented graphically in Figure 5-1.

The following points summarise the observed performance of NNGE.

- classification performance is improved in eight of the thirteen domains;
- classification performance is unaffected in one domain (Iris);
- performance is reduced for four of the domains, only one of which—Waveform-24—shows a substantial decrease in classification performance;
- performance is substantially improved for four of the domains.

Domain	NNGE	nearest neighbour
Iris	**94.7**	**94.7**
Led-7	69.4	**70.2**
Led-24	**55.2**	39.2
Waveform-21	68.6	**69.4**
Waveform-40	64.6	**70.2**
Cleveland	**80.7**	80.0
Hungary	**81.5**	77.8
Voting	**92.9**	85.7
Bcw	**95.4**	94.0
Promoter	**78.0**	77.6
Monks-2	**82.2**	81.7
Diabetes	71.6	**72.1**
Hepatitis	**83.2**	78.4

Table 5-6. NNGE vs nearest neighbour

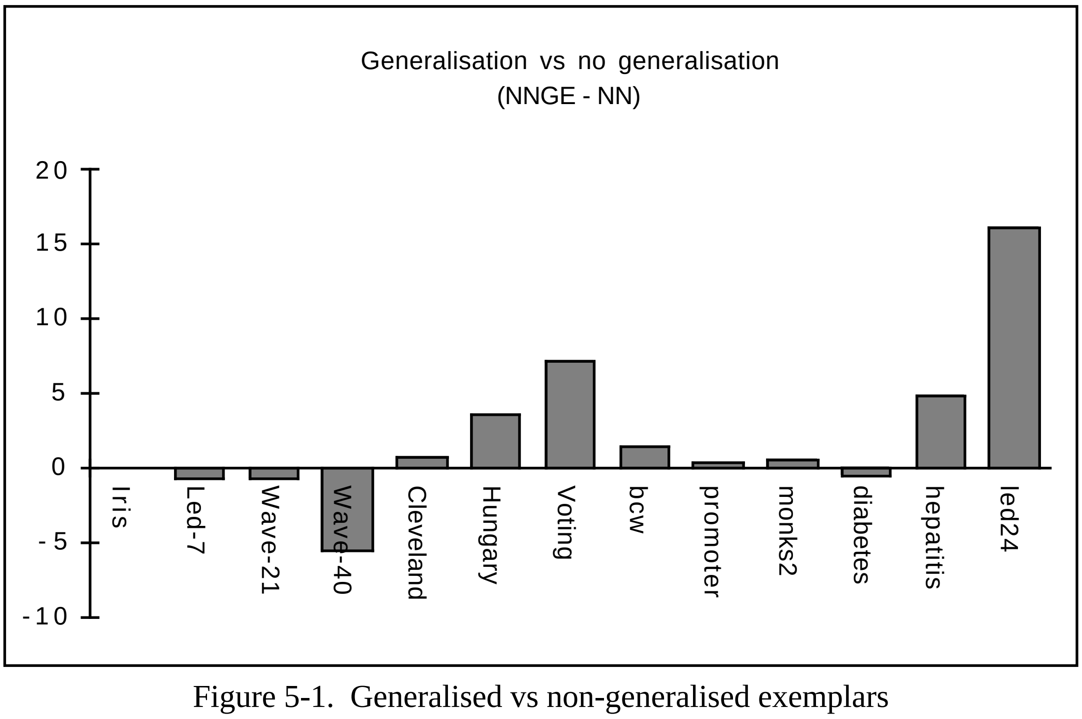

Figure 5-1. Generalised vs non-generalised exemplars

Of the four domains showing a decrease in performance, three of them—waveform-21, waveform-40 and led-7—are artificial domains that test for robustness against noise. This result suggests that the NNGE method is problematic when a significant level of noise is present. LED-24 is a symbolic domain containing many irrelevant features. The large improvement in classification performance of this domain—over 15%—suggests that NNGE is able to reduce the effect of irrelevant attributes.

5.3.1.2 The small disjuncts problem

As discussed earlier, nearest neighbour methods perform well at representing small disjuncts, while rule inducers such as C4.5 perform better when the dataset represents large disjuncts. Section 3.1.1 proposed that generalised exemplars can represent both, because they combine the specificity bias of nearest neighbour with the generality bias of rule induction. If this is the case, we would expect NNGE to perform better than both nearest neighbour and the popular rule inducer C4.5 (Quinlan, 1993) and to perform comparably to, if not better than, the composite learner (CL) described in Section 2.5.1 (Ting, 1994). Because both C4.5 and the composite learner are non-incremental, a batch version of NNGE was used to compare results with these two methods. This algorithm is the same as incremental NNGE, except it considers all training exemplars each time it classifies and generalises a training example, rather than just those observed so far. NNGE was tested against eight of the domains used to test the composite learner, with the number of runs and test to training data ratio being the same as used by Ting (1994). We can therefore compare the results directly to those of the composite learner.

Furthermore, this implementation adopts the feature weighting described in Section 4.5, with the value of d_f set at 1.2, this being the weighting that gave the best overall results. In practice, any value of d_f between 1.05 and 1.5 produces similar results. Table 5-7 lists the observed classification performance of NNGE, C4.5, IBL (selective use of IB1, IB2 or IB3 depending on which performs best), and CL. All results except those for NNGE are taken from Ting (1994).

Figure 5-2 plots the difference in classification performance between NNGE and CL, and C4.5. This graph shows that:

- NNGE outperforms C4.5 in six of the eight domains tested. The improvement is greater than 5% for three domains and greater than 15% for one of these domains;
- NNGE is less accurate than C4.5 for two of the eight domains;
- performance of NNGE is better than the composite learner for half of the domains and worse for the other half;
- in five of the eight cases NNGE is better than both IBL and C4.5.

NNGE often performs better than both IBL and C4.5, suggesting that it is better at handling small disjuncts than either system. The two domains for which performance is worse than C4.5 are again the two artificial domains with a high level of noise added. Performance was about the same overall as the composite learner.

Domain	NNGE	CL	IBL	C4.5
Bcw	95.4	**96.1**	95.4	94.6
Promoter	**84.4**	84.3	80.7	77.4
Monks-2	**81.9**	71.0	70.4	65.0
Diabetes	71.2	**74.2**	70.2	70.7
Hepatitis	**84.8**	81.1	80.7	76.5
Led-7	67.8	**72.9**	69.8	71.4
Led-24	**64.4**	62.1	60.6	62.1
Waveform-40	67.0	**71.1**	63.5	69.2

Table 5-7. NNGE vs CL, IBL, and C4.5

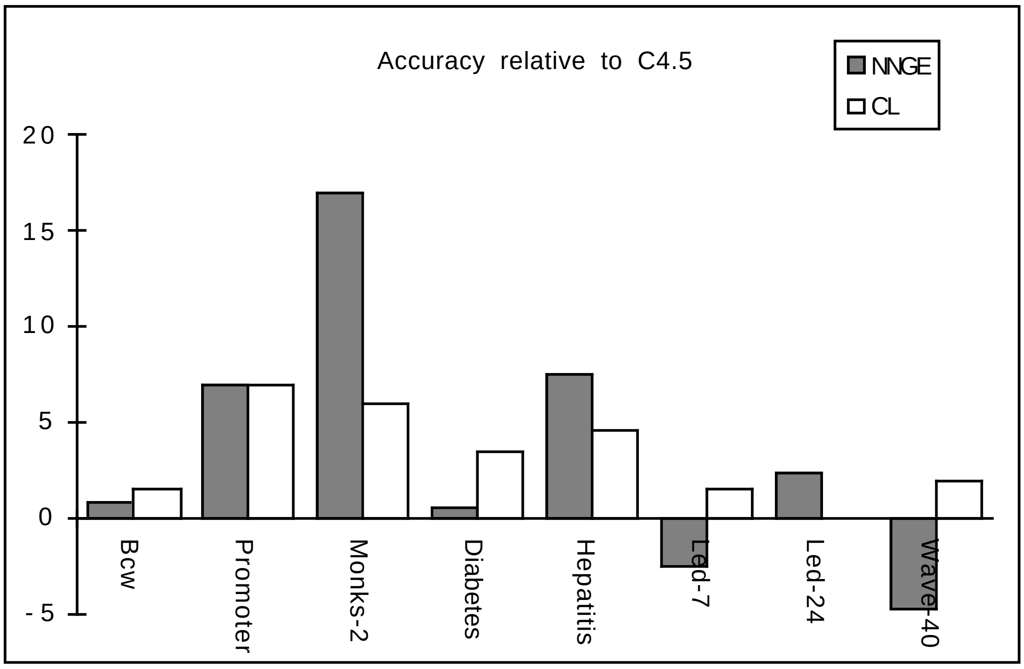

Figure 5-2. NNGE and CL vs C4.5

5.3.2 Hypothesis 2: Useful rules

The rule induction capabilities of NNGE were evaluated by comparing it to ID3 (Quinlan, 1986). The coverage of the training set by the largest hyperrectangle per class was measured for several domains. The hyperrectangle generation procedure adopted by NNGE does not try to produce the smallest number possible. Rectangles can therefore be created that were not joined together by NNGE but are capable of being merged. A post-processor was added to the batch version of NNGE that merges any hyperrectangles where this is possible without losing consistency. Rules were then generated by training NNGE on the entire dataset, and converting each hyperrectangle into a rule of the form shown in Figure 3-2. ID3 was trained using the same dataset, and each leaf node converted into a rule. This is the simplest way that rules can be induced from a tree. Better results may be obtained using more complex methods. The results for the two systems were compared to determine how well NNGE produces useful rules compared to the tree inducer.

High coverage indicates that the rules are very general, and will be applicable to many new examples. Table 5-8 lists the percentage of the training set covered by the largest hyperrectangle per class.

Figure 5-3 plots the percentage of the training set covered by the largest rule for each class compared to ID3. An ideal result would be a 100%, indicating that NNGE has created a single rule for each class. This result is unlikely in practice, as the concepts to be learned may not be representable by a single axis-parallel hyperrectangle.

Domain	**NNGE coverage (%)**	**ID3 coverage (%)**
Monks-2	**17.0**	8.2
Voting	81.2	**81.4**
Bcw	**70.2**	67.5
Promoter	**46.7**	26.2

Table 5-8. Coverage of largest rule

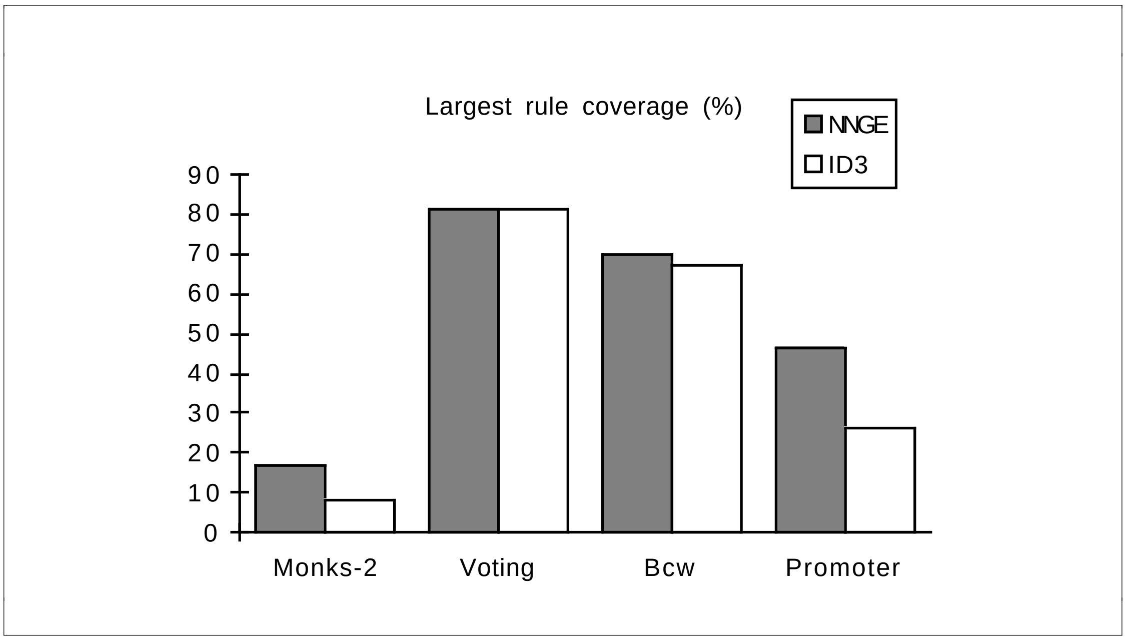

Figure 5-3. Dataset coverage by largest rule

In all cases tested, the coverage by the largest hyperrectangle is comparable to, if not better than, that of ID3. Significant improvements are observed for the Monks-2 and Promoter domains, both of which contain concepts that are not easily represented by single attribute-value tests. Here the hyperrectangle representation fares better because it is more flexible. Table 5-9 lists the largest rule represented by both systems for these two domains.

The voting domain is the only one where both systems produce the same main rule. In particular, the main rules induced by both systems for the Bcw domain have aproximately the same level of generality and yet are completely different. In all but the voting domain, the rule induced by NNGE tests more attributes than that induced by ID3, but each attribute test is more general. This probably reflects the different bias adopted by each system

Domain	**NNGE**	**ID3**

Monks-2	A1 ≠ 1 ^
	A2 ≠ 1 ^
	A3 ≠ 1 ^
	A4 ≠ 1 ^
	A5 ≠ 1 ^
	A6 ≠ 1
		A3 = 1 ^
		A4 = 1 ^
		A6 = 1

Voting	A3 = 1 ^
	A4 = 2
		A3 = 1 ^
		A4 = 2

Bcw	A1 = (1,2,3,4,5,6) ^
	A2 = (1,2,3,4,8) ^
	A3 = (1,2,3,4,8) ^
	A4 = (1,2,3,4,6) ^
	A5 = (1,2,3,4,5) ^
	A6 = (1,2,3,4) ^
	A7 = (1,2,3,6,7) ^
	A8 = (1,2,3,6,7) ^
	A9 = (1,8)
		A2 = 1 ^
		A6 = (1,2,3,4)

Promoter	A4 ≠ 4 ^
	A15 ≠ 3 ^
	A16 ≠ 2 ^
	A17 ≠ 1 ^
	A38 ≠ 4 ^
	A39 ≠ 2 ^
	A49 ≠ 3
		A15 = 3 ^
		A39 = 1

Table 5-9. Induced rules

5.3.3 Hypothesis 3: Reduced classification time

To determine the extent to which generalised exemplars improve the speed of nearest neighbour systems, the time taken to classify each test set was measured with generalisation first disabled and then enabled. Of interest also is the reduction in the total number of exemplars, as this gives an indication of the method's ability to compress large datasets. This compression is especially desirable if the learning system is to be run indefinitely. Table 5-10 lists the speedup and compression ratios observed for each domain.

Figure 5-4 plots the ratio of time taken for NNGE to classify the instances against no generalisation. A speedup of four, for example, indicates that NNGE took one quarter the time to classify the examples as nearest neighbour with no exemplar generalisation. The degree of compression is a measure of the number of exemplars (including rectangles) stored by NNGE compared to those stored by non-generalised nearest neighbour. A compression value of two indicates that NNGE required only half the number of exemplars.

With the exception of the Iris domain, generalised exemplars resulted in a similar amount of compression for all domains. The average factor was 2.6, indicating that NNGE stored only 38% of the original instances.

Domain	speedup	compression
Iris	12.3	21.0
Led-7	1.4	1.4
Promoter	3.5	3.1
Monks-2	2.1	2.3
Hepatitis	2.9	2.4
Glass	4.4	3.3
Cleveland	3.3	3.1

Table 5-10. Speedup and compression

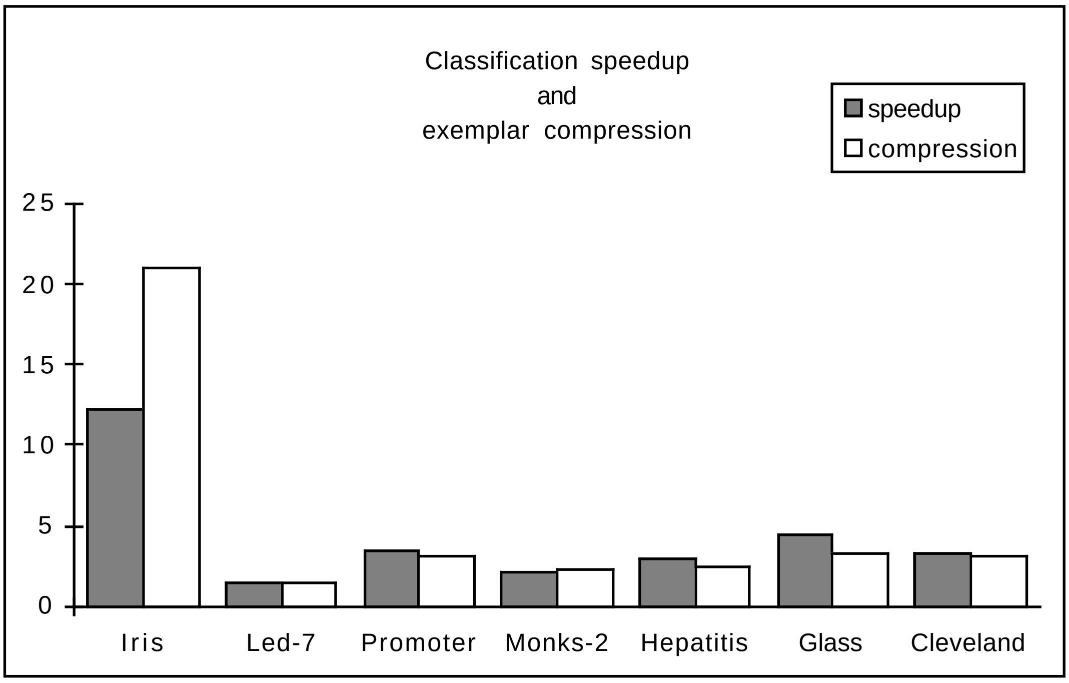

Figure 5-4. Speed and compression compared to non-generalised exemplars

5.3.4 Hypothesis 4: Nested vs non-nested hyperrectangles

Wettschereck and Dietterich (1994) report classification results for NGE for three different numbers of seed exemplars, using eight different UCI databases. The report gives sufficiently detailed information to allow replication of the experimental setup, except that it does not specify the actual examples used for test and training. Given that they perform 25 different random runs on each dataset, tests performed using the same datasets and the same training to testing data ratio should yield similar results. To verify this assumption, NNGE was run with nest-prevention disabled. The results obtained were very similar to those reported for NGE with three seeds.

To test NNGE against NGE it is necessary to remove as many unknowns as possible. To achieve this, the versions of NGE and NNGE tested were both incremental, contained no feature weighting, and used the same exemplar weighting formula. The results for NGE are taken from Wettschereck and Dietterich (1994). All three reported versions of NGE—three seeds, cross-validation, and at the limit—are considered. Section 3.3.2 describes each of these. The first is a fair test of the nested generalised exemplars paradigm, because it maximises nesting by using a small number of seed exemplars. The other two reduce the amount of generalisation performed by increasing the number of seeds, and so are a test of the possible capabilities of NGE, given that we have relaxed the nested generalised exemplars paradigm somewhat.

Domain	**NNGE**	**NGE$_3$**	**NGE$_{cv}$**	**NGE$_{limit}$**
Iris	**94.7**	92.0	93.7	94.4
Led-7	**66.6**	56.0	59.8	64.2
Waveform-21	67.2	69.3	70.0	**71.5**
Waveform-40	60.0	64.2	64.6	**64.9**
Cleveland	**80.2**	55.0	66.9	76.3
Hungary	**80.4**	61.3	76.5	79.3
Voting	**90.4**	64.5	88.4	84.8

Table 5-11. NNGE vs NGE

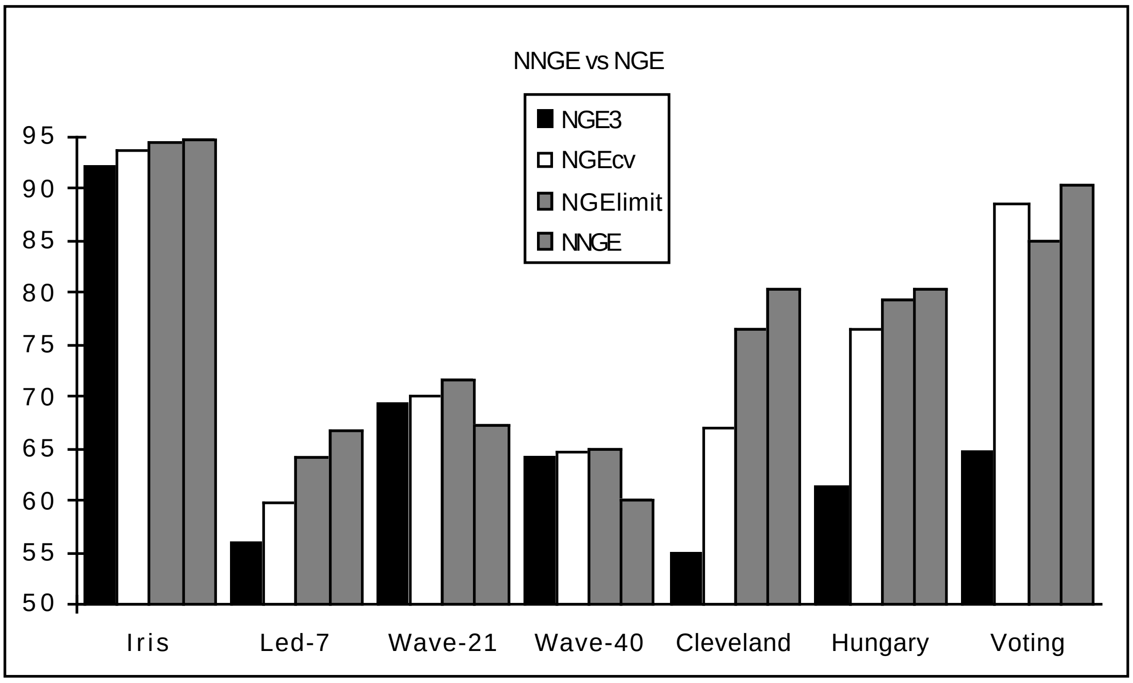

Figure 5-5. Classification accuracy of NNGE vs NGE

Table 5-11 lists classification performance of NNGE, NGE_3, NGE_{cv}, and $\text{NGE}_{\text{limit}}$, and Figure 5-5 presents these results graphically. In five cases, NNGE provides better classification accuracy than any variant of NGE, and the difference between NNGE and NGE_3 is large. NNGE is worse than NGE for the two waveform domains. These results also show that increasing the number of seeds improves classification accuracy of NGE uniformly—in some cases by a very large margin.

5.3.5 Improvements to dynamic feedback

The validity of the new dynamic feedback mechanism was established by running NNGE both with and without feature weighting. The results were then compared to those obtained for NGE, as reported by Wettschereck and Dietterich (1994). Table 5-12 lists these results. Figure 5-6 plots the difference in performance resulting from the application of dynamic feedback. For both systems, the amount d_f that the weights were altered for each domain was determined by cross-validation of several possible values. These results show that neither feature weighting system improves classification performance across all domains, and that the domains for which each scheme performs well are different. Overall, the performance of NNGE's weighting scheme was slightly better than that of NGE's.

Domain	**NNGE (no feedback)**	**NNGE (feedback)**	**df**	**NGE (no feedback)**	**NGE (feedback)**
Iris	95.7	96.0	1.05	93.7	94.0
Led-7	69.2	68.6	1.5	59.8	61.9
Waveform-21	66.2	70.0	1.4	70.0	70.5
Waveform-40	63.2	67.0	1.3	64.6	62.6
Cleveland	78.6	79.5	1.05	66.9	69.0
Hungary	73.7	74.5	1.02	76.5	77.8
Voting	93.0	95.0	1.2	88.4	88.7

Table 5-12. Dynamic feedback performance

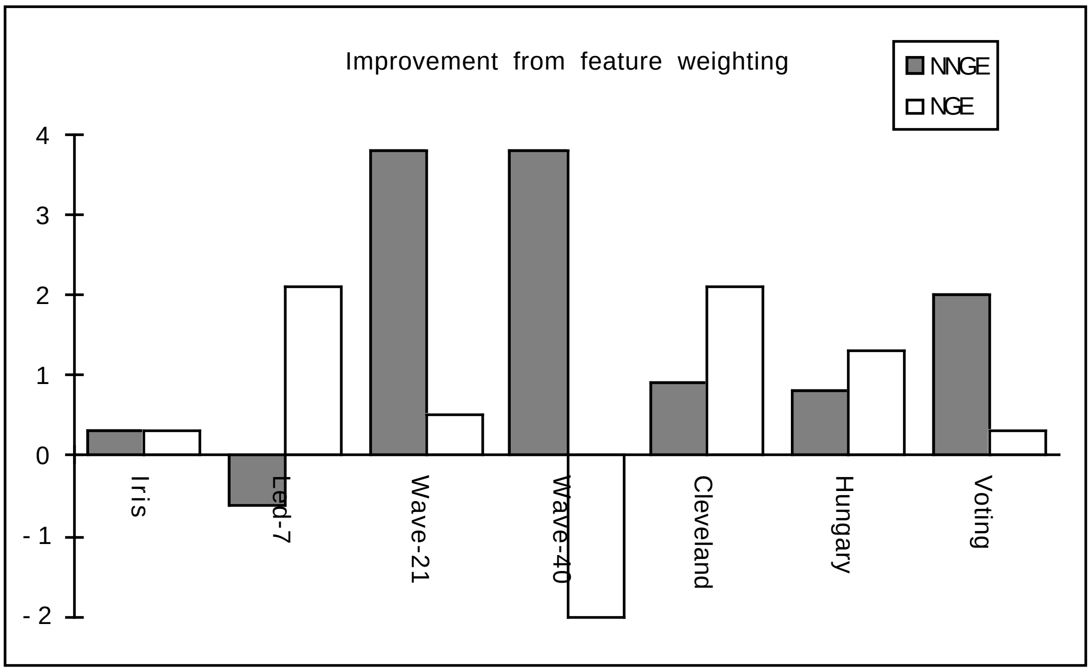

Figure 5-6. Feature weighting improvements for NNGE and NGE

6 Summary and conclusions

The first part of this chapter reviews earlier chapters, highlighting the contribution of each. The second summarises NNGE's performance, noting strengths and weaknesses, and suggesting further research.

6.1 Review

Chapter One discussed the problem of using historic data to make decisions in the future. It introduced instance-based learning and rule induction by generalisation as two very different approaches to machine learning. Both have their strengths and weaknesses, making neither approach ideal for all problem domains. This thesis proposed that generalised exemplars are a way of combining the two, and sought a better method than NGE (Salzberg, 1991). Four specific hypotheses about the performance advantages offered by NNGE over both NGE and standard nearest neighbour methods were put forward:

> **Hypothesis 1:** Generalised exemplars increase the performance of nearest neighbour systems by improving the representation of large disjuncts;
>
> **Hypothesis 2:** Producing exclusive generalised exemplars results in a useful set of rules that may be compared with those produced by other rule induction methods;
>
> **Hypothesis 3:** Generalised exemplars reduce classification time without sacrificing accuracy;
>
> **Hypothesis 4:** A learning system using non-nested generalised exemplars shows better classification performance than one using nested generalised exemplars.

Chapter Two described three methods of machine learning, namely nearest neighbour, case-based learning and rule induction. Nearest neighbour methods form the basis of NNGE, while case-based learning is an example of generalisation in instance-based learning. Rule induction is an alternative approach that also uses generalisation. The notion of a generality versus specificity bias was described and the problem of small disjuncts was introduced. It has been suggested that small disjuncts are the cause of poor performance of generalisation-based methods for some domains, and that instance-based learners perform well where small disjuncts are present (Holte *et al*, 1989). This thesis proposed that nearest neighbour systems often represent large disjuncts poorly, and that this leads to their poor performance for some domains that are well represented by generalisation based learning systems. A motive for combining generalisation with nearest neighbour is therefore to try to accurately represent both large and small disjuncts.

Chapter Three described hyperrectangles, the method of exemplar generalisation investigated. The motives for generalising exemplars were discussed, giving rise to the hypotheses. These motives are: improved classification performance of nearest neighbour, induction of useful rules, and reduction in classification time. Nested Generalised Exemplars, a method of exemplar generalisation (Salzberg, 1991), was described. The problem with nesting generalisations—that the system will overgeneralise, reducing the effectiveness of the distance function—was then discussed, and a recent study of NGE reviewed. Shortcomings of that study were discussed, and it was concluded that the experimental results support the hypothesis that it is undesirable to nest generalised exemplars. Finally, Chapter Three described two other methods of generalising exemplars, namely rule induction and conceptual clustering.

Chapter Four described the NNGE implementation in detail. NNGE is similar to NGE except that it does not allow any instance to be a member of more than one generalisation, unless those generalisations are of the same class. Furthermore, it prevents the boundaries of generalised exemplars from overlapping. A batch version of NNGE was also implemented, for comparison with other non-incremental systems.

Chapter Five reported empirical testing of NNGE. Thirteen domains were used, providing a variety of testing conditions. Included in the test domains were four artificial domains that specifically test for robustness against noise and irrelevant attributes. Several contained missing values. One of the domains was artificially produced, and the target concept requires a complex representation. NNGE excelled in this domain. The four hypotheses were tested in isolation, and compared with reported results. The results used were sufficiently documented that test conditions could be adequately duplicated. The results of each test were presented, and the significance of each outcome was discussed.

6.2 Conclusions

The performance of NNGE exceeded expectations, satisfying all four hypotheses proposed in this thesis. The following sections describe the performance observed.

6.2.1 Hypothesis 1: Improved classification performance

For the thirteen domains tested, generalising exemplars results in an average classification performance improvement of 2.6% over standard nearest neighbour.

Four domains show a reduction in performance. Three of these are artificial domains containing noise, suggesting that the NNGE implementation is not very robust against noise. The domain showing the largest improvement in classification performance contains a large number of irrelevant attributes, suggesting that by generalising their values a nearest neighbour system can ignore irrelevant attributes.

When tested against domains containing both large and small disjuncts, NNGE performs better than C4.5, a generality-biased learning system, and than pure nearest neighbour, which is specificity-biased. Performance over these domains is comparable to Composite Learner, an algorithm that combines both rule induction and nearest neighbour.

These results provide strong evidence that generalised exemplars improve the classification performance of nearest neighbour learning, by providing a bias that is suitable for inducing both large and small disjuncts.

6.2.2 Hypothesis 2: Useful rules

For the domains tested, NNGE produces a single hyperrectangle per class that covers as much of the dataset as rules induced by ID3 (Quinlan, 1986). For two domains the rules induced by NNGE represent a much larger proportion of the input set. The rules produced are quite different to those induced by ID3 in all but one case, reflecting the different bias used.

NNGE tends to produce rules that test a large number of attributes. Because of this they are not very intelligible to people. This shortcoming is discussed in Section 6.4.

6.2.3 Hypothesis 3: Reduced classification time

Generalised exemplars speed up classification by an average of 2.6 times over standard nearest neighbour, reducing the number of exemplars by 62%. This seems fairly modest, but this is probably due to the nature of the training datasets. All of these are fairly small, so the number of instances representing each target concept is low. If the system were being used incrementally and exposed to input data on a continuous basis, the number of exemplars would be expected to become static once a sufficient number of instances had been incorporated to represent the target concepts.

The results can be further put into perspective by comparing them to the compression method implemented in IB2 (Aha, 1992). IB2 reduces the number of exemplars by 64% over standard nearest neighbour, but at the cost of a 4.7% reduction in average classification performance. For these domains NNGE reduces the number of exemplars by a similar amount, but improves classification accuracy by 4%.

Generalised exemplars are therefore as good at compressing the exemplar database as saving only misclassified instances, with the added advantage that this compression does not come at the cost of classification accuracy.

6.2.4 Hypothesis 4: Nested vs non-nested hyperrectangles

NNGE out-classifies NGE (Salzberg, 1991) in five of the seven domains tested, regardless of the variant of NGE that is used. NNGE is inferior in only the two artificial noisy numeric-attribute domains, where NGE's performance approximates that of pure nearest neighbour while NNGE's is worse. In all seven domains the performance of NGE declines as the number of seed instances is reduced, strongly suggesting that overgeneralisation resulting from nesting and overlapping of generalised exemplars degrades classification performance. In the extreme case, NNGE's classification performance is 25% better than NGE's.

6.3 Research contributions

This research has examined generalisation of exemplars in detail. The problem of overlapping generalisations has been explored and a novel approach to its prevention introduced.

The resulting implementation improves the classification performance of nearest neighbour learning by an average of almost 3%, and reduces classification time by over 60%. NNGE performs well on datasets that combine small and large disjuncts, outperforming a popular generalisation based learner, C4.5 (Quinlan, 1993), on all but the two artificially noisy domains, and performing comparably to a hybrid learner that combines C4.5 with instance-based learning (Ting, 1994).

Non-nested generalised exemplars is a technique that can be added to instance-based learners and applied to both discrete and continuous-values problems. It is independent of the distance function used and of other characteristics of the learning method, such as feature and exemplar weighting strategies.

Nearest neighbour learning is a practical method that offers respectable classification performance for minimal learning effort. The flexibility of its incremental learning capability allows it to be used for domains where continuous learning is required. Non-nested generalised exemplars improve classification performance, making nearest neighbour learning suitable for use on a wider set of domains.

6.4 Future research

One of the key problems with non-nested generalised exemplars is deciding how to reduce the size of a hyperrectangle that has become inconsistent due to the introduction of a conflicting example. The heuristic used in this study is a very simple method of discriminating the feature value or range of values to remove from the rectangle. Further work is needed in this area to determine if there is a better method.

NNGE performs poorly on domains with a high degree of noise. This is probably because it does not allow any conflict of class within a rectangle. Noisy examples may fall into a rectangle of the wrong class, requiring that the rectangle be pruned. This problem may be overcome by allowing a small amount of conflict of class within a rectangle, an approach similar to decision tree pruning. Determining how much conflict to allow is an interesting problem for further work.

The ability of NNGE to generate useful rules was tested by converting each hyperrectangle into a production rule. The resulting rules contained many terms. It is possible that some of these terms are redundant, and that increasing the size of the hyperrectangle along a particular axis would not produce any conflict. This might produce simpler rules because some terms may be generalised out. Simpler rules would be more understandable by people.

A batch version of NNGE was produced for testing against other non-incremental machine learning systems. This system is still affected by the input order of the examples. Some preliminary tests were performed where the order in which the examples are processed is determined by the distance between them, but the results were inconclusive. Further study into a non-incremental NNGE approach may yield interesting results.

Finally, axis-parallel hyperrectangles were chosen as the representation of a generalised exemplar due to their simplicity, and similarity to other rule representations. Other shapes may also prove useful. Convex hulls, for example, would produce more conservative generalisation. This is the smallest polygon that can fit around a selection of points. While more expensive to compute than hyperrectangles, recently developed algorithms make this idea possible.

References

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

Bareiss, E.R. and Porter, B.W. (1988) PROTOS: an exemplar-based learning apprentice. *International Journal of Man-Machine Studies*, 29: 549–561.

Breiman, L. Friedman, J.H. Olshen, R.A. & Stone, C.J. (1984) *Classification and regression trees*. Wadsworth International Group, Belmont, California.

Cendrowska, J. (1987) PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27: 349–370.

Clarke, P. and Niblett, T. (1987) Induction in noisy domains. *Progress in Machine Learning edited by Ivan Bratko and Nada Lavrac*, Sigma Press, Wilmslow, England.

Cost, C. and Salzberg, S. (1994) A weighted nearest neighbour algorithm for learning with symbolic features. *Unpublished paper.*

Fisher, D.H. and Schlimmer,J.C. (1988) Concept simplification and predictive accuracy. *Proceedings of the Fifth International Conference on Machine Learning*, 22–28.

Fix, E. and Hodges, J.L.Jr (1951) Discriminating analysis: non-parametric distribution. Technical Report 21-49-004(4), USAF School of Aviation Medicine, Randolph Field, Texas.

Fix, E. and Hodges, J.L.Jr (1952) Discriminating analysis: small sample performance. Technical Report 21-49-004(11), USAF School of Aviation Medicine, Randolph Field, Texas.

Frey, P.W. and Slate, D.J. (1991) Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2).

Gaines, B. (1994) Exception DAGs as knowledge structures. *AAAI-94 Workshop on Knowledge Discovery in Databases.*

Gennari, J.H. (1989). A survey of clustering methods. Technical Report 89-38, Department of Information and Computer Science, University of Irvine, California.

Gennari, J.H. Langley, P. and Fisher, D. (1989) Models of incremental concept formation. *Artificial Intelligence*, 40: 11–61.

Holte, R.C.; Acker L.E.; Porter, B.W. (1989) Concept learning and the problem of small disjuncts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 813–818.

Holte, R.C. (1993) Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11: 63–90.

Johns, M.V. (1961) An empirical Bayes approach to non-parametric two-way classification. *Studies in Item Analysis and Prediction edited by H. Solomon*, Stanford University Press.

Kanal, L.N. (1963) Statistical methods for pattern classification. Philco Report VO43-2 and VO43-3, U.S. Army Electronics Research and Development lab.

Kibler, D. and Aha, D. (1987) Learning representative exemplars of concepts: an initial case study. *Proceedings of the Fourth International Workshop on Machine Learning.*

Kolodner, J. L. (1984) *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates.

Lebowitz, M. (1980) Generalization and memory in an integrated understanding system. Research Report 186, Department of Computer Science, Yale University, New Haven, CT.

Lebowitz, M. (1987) Experiments with universal concept formation: UNIMEM. *Machine Learning*, 2: 103–138.

Mangasarian, O.L. and Wolberg, W.H. (1990) Cancer diagnosis via linear programming. *SIAM news,* 23(5).

Porter, B.W. Bariess, R. and Holte, R.C. (1990) Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45: 229–263.

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

Quinlan, J.R. (1993) *C4.5; Program for Machine Learning*, Morgan Kaufmann.

Salzberg, S. (1991) A nearest hyperrectangle learning method. *Machine Learning*, 6: 277–309.

Smith L.A., Scott B.L., Lin L.S., and Newell J.M (1990) Template adaption in a hypershere word classifier. *Proceedings of ICASSP.*

Stanfill, C. and Waltz, D. (1986) Toward memory-based reasoning. *Communications of the ACM*, 29(12): 1213–1228.

Thrun S.B., Bala J., Bloedorn E., Bratko I., Cestnik B., Cheng J., De Jong K., Dzeroski S., Fahlman S.E., Fisher D., Hamann R., Kaufman K., Keller S., Kononenko I., Kreuziger J., Michalski R.S., Mitchell T., Pachowicz P., Reich Y., Vafaie H., Van de Welde W., Wenzel W., Wnek J. and Zhang J. (1991) The MONK's problems - performance comparison of different learning algorithms, Technical Report CS-CMU-91-197, Carnegie Mellon University, Pittsburgh, PA.

Ting, K.M. (1994) The problem of small disjuncts: its remedy in decision trees. *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, 91–97.

Towell, G., Shavlik, J. and Noordewier, M. (1990) Refinement of approximate domain theories by knowledge-based artificial neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence.*

Utgoff, P.E. (1989) Incremental induction of decision trees. *Machine Learning*, 4: 161–186.

Wettschereck, D. and Dietterich, G. (1994) An experimental comparison of the nearest-neighbour and nearest-hyperrectangle algorithms. *Machine Learning*, (to appear).

Appendix Dataset acknowledgements

The Cleveland Heart Disease data was provided by Robert Detrano, M.D., Ph.D. of the V.A. Medical Center, Long Beach and Cleveland Clinic Foundation.

The Hungarian Heart Disease data was provided by Andras Janosi, M.D of the Hungarian Institute of Cardiology, Budapest.

The Wisconsin Breast Cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.