A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features

Scott Cost and Steven Salzberg
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
Internet: cost@cs.jhu.edu, salzberg@cs.jhu.edu

Abstract

In the past, nearest neighbor algorithms for learning from examples have worked best in domains in which all features had numeric values. In such domains, the examples can be treated as points and distance metrics can use standard definitions. In symbolic domains, a more sophisticated treatment of the feature space is required. We introduce a nearest neighbor algorithm for learning in domains with symbolic features. Our algorithm calculates distance tables that allow it to produce real-valued distances between instances, and attaches weights to the instances to further modify the structure of feature space. We show that this technique produces excellent classification accuracy on three problems that have been studied by machine learning researchers: predicting protein secondary structure, identifying DNA promoter sequences, and pronouncing English text. Direct experimental comparisons with the other learning algorithms show that our nearest neighbor algorithm is comparable or superior in all three domains. In addition, our algorithm has advantages in training speed, simplicity, and perspicuity. We conclude that experimental evidence favors the use and continued development of nearest neighbor algorithms for domains such as the ones studied here.

Running head: **Learning with Symbolic Features**
Keywords: nearest neighbor, exemplar-based learning, protein structure, text pronunciation, instance-based learning

1 Introduction

Learning to classify objects is fundamental problem in artificial intelligence and other fields, one which has been attacked from many angles. Despite many successes, there are some domains in which the task has proven very difficult, due either to the inherent difficulty of the domain or to the lack of sufficient data for learning. For example, instance-based learning programs (also called *exemplar-based* (Salzberg, 1990) or nearest neighbor (Cover and Hart, 1967) methods), which learn by storing examples as points in a feature space, require some means of measuring distance between examples (Aha, 1989; Aha and Kibler, 1989; Salzberg, 1989; Cost and Salzberg, 1990). An example is usually a vector of feature values plus a category label. When the features are numeric, normalized Euclidean distance can be used to compare examples. However, when the feature values have symbolic, unordered values (e.g., the letters of the alphabet, which have no natural inter-letter "distance"), nearest neighbor methods typically resort to much simpler metrics, such as counting the features that match. (Towell et al. (1990) recently used this metric for the nearest neighbor algorithm in their comparative study.) Simpler metrics may fail to capture the complexity of the problem domains, and as a result may not perform well.

In this paper, we present a more sophisticated instance-based algorithm designed for domains in which some or all of the feature values are symbolic. Our algorithm constructs modified "value difference" tables (in the style of (Stanfill and Waltz, 1986)) to produce a non-Euclidean distance metric, and we introduce the idea of "exception spaces" that result when weights are attached to individual examples. The combination of these two techniques results is a robust instance-based learning algorithm that works for any domain with symbolic feature values. We describe a series of experiments demonstrating that our algorithm, PEBLS, performs well on three important practical classification problems. Comparisons given below show that our algorithm's accuracy is comparable to back propagation, decision trees, and other learning algorithms. These results support the claim that nearest neighbor algorithms are powerful classifiers even when all features are symbolic.

1.1 Instance-based learning versus other models

The power of instance-based methods has been demonstrated in a number of important real world domains, such as prediction of cancer recurrence, diagnosis of heart disease, and classification of congressional voting records (Aha and Kibler, 1989; Salzberg, 1989). Our experiments demonstrate that instance-based learning (IBL) can be applied effectively in three other domains, all of which have features with unordered symbolic values: (1) prediction of protein secondary structure, (2) word pronunciation, and (3) prediction of DNA promoter sequences. These domains have received considerable attention from connectionist researchers who employed the back propagation learning algorithm (Sejnowski and Rosenberg, 1986; Qian and Sejnowski, 1988; Towell et al., 1990). In addition, the word pronunciation problem has been the subject of a number of comparisons using other machine learning algorithms (Stanfill and Waltz, 1986; Shavlik et al., 1989; Dietterich et al., 1990). All of these domains represent problems of considerable practical importance, and all have symbolic feature values, which makes them difficult for conventional nearest neighbor algorithms. We will show how our nearest neighbor algorithm, PEBLS, which is based on Stanfill and Waltz's (1986) "value difference" method, can produce highly accurate predictive models in each of these domains.

Our intent is to compare IBL to other learning methods in three respects; classification accuracy, speed of training, and perspicuity (i.e., the ease with which the algorithm and its representation can be understood). Because of its comparable performance in the first respect, and its superiority in the latter two, we argue that IBL is often preferable to other learning algorithms for the types of problem domains considered in this paper. Instance-based learning has been shown to compare favorably to other algorithms, (e.g., decision trees and rules), on a wide range of domains in which feature values were either numeric or binary (e.g., Aha, 1989; Aha and Kibler, 1989; Aha et al., 1991; Salzberg, 1989). This paper presents similar evidence in terms of classification accuracy for domains with symbolic feature values. However, before describing these domains, we should consider other advantages that instance-based learning algorithms can provide.

Training time. Most neural net learning algorithms requires vastly more time for training than other machine learning methods. Training is normally performed by repeatedly presenting the network with instances from a training set, and allowing it gradually to converge on the best set of weights for the task, using (for example) the back propagation algorithm. Weiss and Kapouleas (1989), Mooney et al. (1989), and Shavlik et al. (1989) report that back propagation's training time is many orders of magnitude greater than training time for algorithms such as ID3, frequently by factors of 100 or more. In addition, neural net algorithms have a number of parameters (e.g., the "momentum" parameter) that need to be tweaked by the programmer, and may require much additional time. The Weiss and Kapouleas experiments required many months of the experimenters' time to produce results for back propagation, while the other algorithms typically required only a few hours. The only parameter that might be adjusted for our algorithm is the value of r in our distance metric (see below), and we only consider two possible values, $r = 1$ and $r = 2$.

Our nearest neighbor algorithm requires very little training time, both in terms of experimenter's time and processing time. Below we present two versions of the PEBLS algorithm, one slightly more complex than the other. The simpler version, which is called the "unweighted" version in our experimental section below, requires $O(dn + dv^2)$ time for training, where n is the number of examples, d is the number of features (dimensions) per example, and v is the number of values that a feature may have. (In general n is much larger than v^2, hence the complexity is usually $O(dn)$.) The more complex version of PEBLS incrementally computes weights for exemplars, and requires $O(dn^2)$ time for n training instances[1]. After training, classification time for a nearest neighbor system is at worst $O(dn)$, which is admittedly slow compared to other algorithms.

Nearest neighbor methods lend themselves well to parallelization, which can produce significantly faster classification times. Each example can be assigned to a separate processor in a parallel architecture, or, if there are not enough processors, the examples can be divided among them. When the number of processors is as large as the training set, classification time may be reduced to $O(d \log n)$. We implemented our system on a set of four loosely-coupled transputers as well as on a conventional architecture, and other recent efforts such as FGP (Fertig and Gelernter, 1991) use larger numbers of parallel processors. The MBRtalk system of Waltz and Stanfill (1986) implemented a form of k-nearest-neighbor learning using a tightly-coupled massively parallel architecture, the 64,000-processor Connection Machine^{tm}.

Perspicuity. Our instance-based learning algorithm is also more transparent in its operation than other learning methods. The algorithm itself is nothing if not straightforward: define a measure of distance between values, compare each new instance to all instances in memory, and classify according to the category of the closest. Decision tree algorithms are perhaps equally straightforward at classification time – just pass an example down through the tree to get a decision. Neural net algorithms are fast at classification time, but are not as transparent – weights must be propagated through the net, summed, and passed through a filter (e.g., a threshold) at each layer. The only complicated part of our method is the computation of our distance tables, which are computed via a fixed statistical technique based on frequency of occurrence of values.

[1] For reference, the system takes about 30 minutes of real time on a DECstation 3100 train on 17,142 instances. The experimenter's time is limited to a few minutes defining the data set.

However, this is certainly no more complicated than the entropy calculations of decision tree methods (e.g., Quinlan, 1986) or the weight adjustment routines used in back propagation. The tables themselves provide some insight into the relative importance of different features – we noticed, for instance, that several of the tables for the protein folding data were almost identical (indicating that the features were very similar). The instance memory of an IBL system is readily accessible, and can be examined in numerous ways. For example, if a human wants to know why a particular classification was made, the system can simply present the instance from its memory that was used for classification. Human experts commonly use such explanations; for example, when asked to justify a prediction about the economy, experts typically produce another, similar economic situation from the past. Neural nets do not yet provide any insight into why they made the classification they did, although some recent efforts have explored new methods for understanding the content of a trained network (Hanson and Burr, 1990). It is also relatively easy to modify our algorithm to include domain specific knowledge: if the relative importance of the features is known, the features may be weighted accordingly in the distance formula (Salzberg, 1989).

Taken together, the advantages listed above make it clear that IBL algorithms have a number of benefits with respect to competing models. However, in order to be considered a realistic practical learning technique, IBL must still demonstrate good classification accuracy. When the problem domain has symbolic features, the obvious distance metric for IBL, counting the number of features that differ, does not work well. (This metric is called the "overlap" metric.) Our experimental results show that a modified value-difference metric can process symbolic values exceptionally well. These results, taken together with other results on domains with numeric features (e.g., Aha, 1990; Aha et al., 1991), show that IBL algorithms perform quite well on a wide range of problems.

2 Learning algorithms

Back propagation is the most widely used and understood neural net learning algorithm, and will be used as the basis of comparison for the experiments in this paper. Although earlier approaches, most notably the perceptron learning model, were unable to classify groups of concepts that are not linearly separable, back propagation can overcome this problem. Back propagation is a gradient descent method that propagates error signals back through a multi-layer network. It has been described in many places (e.g., Rumelhart et al., 1986; Rumelhart and McClelland, 1986), and readers interested in a more detailed description should look there. We will use the decision tree algorithm ID3 (Quinlan, 1986) as the basis for comparison with decision tree algorithms. In addition, we have compared the performance our algorithm to other methods used on the same data for each of the domains described in section 3. Where appropriate, we present results from domain-specific classification methods.

2.1 Instance-based learning

Our instance-based learning algorithm, like all such algorithms, stores a series of training instances in its memory, and uses a distance metric to compare new instances to those stored. New instances are classified according to the closest exemplar from memory. Our algorithm is implemented in a program called PEBLS, which stands for **P**arallel **E**xemplar–**B**ased **L**earning **S**ystem.[2] For clarity, we use the term "example" to mean a training or test example being shown to the system for the first time. We use the term "exemplar" (following the usage of (Salzberg, 1991)) to refer specifically to an instance that has been previously stored in computer memory. Such exemplars may have additional information attached to them (e.g., weights). The term "instance" covers both examples and exemplars.

[2]The parallelization of the algorithm was developed to speed up experimentation, and is of no theoretical importance to our learning model.

PEBLS was designed to process instances that have symbolic feature values. The heart of the PEBLS algorithm is the way in which it measures distance between two examples. This consists of essentially three components. The first is a modification of Stanfill and Waltz's (1986) Value Difference Metric (VDM), which defines the distance between different values of a given feature. We call our method MVDM, for Modified Value Difference Metric. Our second component is a standard distance metric for measuring the distance between two examples in a multi-dimensional feature space. Finally, the distance is modified by a weighting scheme that weights instances in memory according to their performance history (Salzberg 1989, 1990). These components of the distance calculation are described in sections 2.2 and 2.3.

PEBLS requires two passes through the training set. During the first pass, feature value difference tables are constructed from the instances in the training set, according to the equations for the Stanfill Waltz VDM. In the second pass, the system attempts to classify each instance, by computing the distance between the new instance and previously stored ones. The new instance is then assigned the classification of the nearest stored instance. The system then checks to see if the classification is correct, and uses this feedback to adjust a weight on the old instance (this weight is described in detail in section 2.3). Finally, the new instance is stored in memory. During testing, examples are classified in the same manner, but no modifications are made to memory or to the distance tables.

2.2 The Stanfill-Waltz VDM

In 1986, Stanfill and Waltz presented a powerful new method for measuring the distance between values of features in domains with symbolic feature values. They applied their technique to the English pronunciation problem with impressive initial results (Stanfill and Waltz, 1986). Their Value Difference Metric (VDM) takes into account the overall similarity of classification of all instances for each possible value of each feature. Using this method, a matrix defining the distance between all values of a feature is derived statistically, based on the examples in the training set. The distance δ between two values (e.g., two amino acids) for a specific feature is defined in Equation 1:

$$\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1_i}}{C_1} - \frac{C_{2_i}}{C_2} \right|^k \tag{1}$$

In the equation, V_1 and V_2 are two possible values for the feature, e.g., for the protein data these would be two amino acids. The distance between the values is a sum over all n classes. For example, the protein folding experiments in section 4.1 had three categories, so $n = 3$ for that data. C_{1_i} is the number of times V_1 was classified into category i, C_1 is the total number of times value 1 occurred, and k is a constant, usually set to 1.

Using Equation 1, we compute a matrix of value differences for each feature in the input data. It is interesting to note that the value difference matrices computed in the experiments below are quite similar overall for different features, although they differ significantly for some value pairs.

The idea behind this metric is that we wish to establish that values are similar if they occur with the same relative frequency for all classifications. The term $\frac{C_{1_i}}{C_1}$ represents the likelihood that the central residue will be classified as i given that the feature in question has value V_1. Thus we say that two values are similar if they give similar likelihoods for all possible classifications. Equation 1 computes overall similarity between two values by finding the sum of the differences of these likelihoods over all classifications.

Consider the following example. Say we have a pool of instances for which we examine a single feature that takes one of three values, A, B, and C. Two classifications, α and β, are possible. From the data we construct Table 1, in which the table entries represent the number of times an instance had a given feature value and classification. From this information we construct a table of distances as follows. The frequency of occurrence of A for class α is 57.1%, since there were 4 instances classified as α out

Table 1: Number of occurrences of each value for each class

	Classes	
Feature values	α	β
A	4	3
B	2	5
C	4	2

Table 2: Value difference table

	Feature values		
	A	B	C
A	0.000	0.571	0.191
B	0.571	0.000	0.762
C	0.191	0.762	0.000

of 7 instances with value A. Similarly, the frequencies of occurrence for B and C are 28.6% and 66.7% respectively. The frequency of occurrnce of A for class β is 42.9%, and so on. To find the distance between A and B, we use Equation 1, which yields $|4/7 - 2/7| + |3/7 - 5/7| = 0.571$ The complete table of distances is shown in table 2. Note that we construct a different value difference table for each feature; if there are 10 features, we will construct 10 tables.

Equation 1 defines a geometric distance on a fixed, finite set of values. That is, the property that a value has distance zero to itself, that it has a positive distance to all other values, that distances are symmetric, and that distances obey the triangle inequality. We can summarize these properties as follows:

i. $\delta(a,b) > 0, a \neq b$

ii. $\delta(a,b) = \delta(b,a)$

iii. $\delta(a,a) = 0$

iv. $\delta(a,b) + \delta(b,c) \geq \delta(a,c)$

Stanfill and Waltz's original VDM also used a weight term, w_f^g, which makes their version of δ non-symmetric; e.g., $\delta(a,b) \neq \delta(b,a)$. A major difference between their metric (VDM) and ours (MVDM) is that we omit this term, which makes δ symmetric.

The total distance Δ between two instances is given by:

$$\Delta(X,Y) = w_X w_Y \sum_{i=1}^{N} \delta(x_i, y_i)^r \tag{2}$$

where X and Y represent two instances (e.g., two windows for the protein folding domain), with X being an exemplar in memory and Y a new example. The variables x_i and y_i are values of the i^{th} feature for X and Y, where each example has N features. w_X and w_Y are weights assigned to exemplars, described in the following section. For a new example Y, $w_Y = 1$. (In domains with numeric features, $r = 1$ yields Manhattan distance and $r = 2$ produces Euclidean distance.) For most of our experiments, we used $r = 2$; however, we used $r = 1$ for the protein secondary structure task.

In summary, there are four major differences between our MVDM and the Stanfill-Waltz VDM.

1. We omit the weight term w_f^g, which makes the Stanfill-Waltz VDM non-symmetric. In our formulation, δ and Δ are symmetric.

2. Stanfill and Waltz (1986) used the value of $k = 2$ in their version of Equation 1. Our preliminary experiments indicated that equally good performance is achieved when $k = 1$, so we chose that value for reasons of simplicity.

3. We have added exemplar weights to our distance formula, as described in section 2.3.

4. Stanfill and Waltz used the 10 closest exemplars for classification, whereas PEBLS uses only the nearest neighbor. (This is really a difference between the learning algorithms rather than the value difference metrics.)

2.3 Weighted exemplars and exception spaces

Some stored instances are more reliable classifiers than others. Intuitively, one would like these trustworthy exemplars to have more "drawing power" than others. The final difference between our MVDM metric and the original VDM is a capacity in the metric for treating more reliable instances differently. We accomplish this with the weight w_X in our distance formula: reliable exemplars are given smaller weights, making them appear closer to a new example. Our weighting scheme was first adopted in the EACH system (Salzberg 1989, 1990), which assigned weights to exemplars according to their performance history. w_X is the ratio of the number of uses of an exemplar to the number of *correct* uses of the exemplar; thus, accurate exemplars will have $w_X \approx 1$. Unreliable exemplars will have $w_X > 1$, making them appear further away from a new example. These unreliable exemplars may represent either noise or "exceptions" – small areas of feature space in which the normal rule does not apply. The more times an exemplar is incorrectly used for classification, the larger its weight grows. An alternative scheme for handling noisy or exceptional instances in the IBL framework is discussed by Aha and Kibler (1989), and elaborated further in Aha (1990). In their scheme, an instance is not used in the nearest-neighbor computation until it has proven itself to be an "acceptable" classifier. Acceptable instances are those whose classification accuracies exceed the baseline frequency for a class by a fixed amount. (For example, if the baseline frequency of a class is 30%, an instance that was correct 80% of the time would be acceptable, whereas if the baseline frequency was 90%, the same instance would not be acceptable.) We should note that our technique is not designed primarily to filter out noisy instances, but rather to identify *exceptional* instances. The difference is that noisy instances should probably be ignored or discarded, whereas exceptional instances should be retained, but used relatively infrequently.

We differ from Salzberg's original exemplar weighting scheme in one significant as-

Figure 1: Two unweighted points in instance space

pect: the way in which exemplars (points) are weighted initially. The original scheme stored points with initial weights of 1/1. The effect which this has on the feature space is significant. Consider an instance space containing two points, classified as α and β. Unweighted, these two points define a hyperplane that divides the n-dimensional space into an α and a β region, as shown in Figure 1. Any point located on the left side of the plane will be classified as α, and likewise for β.

When PEBLS computes distance from a new instance to a weighted exemplar, that distance is multiplied by the exemplar's weight. Intuitively, that makes it less likely for a new instance to appear near an exemplar as the exemplar's weight grows. Figure 2 shows that, geometrically, the use of weights creates a circular envelope around the exemplar with the larger weight, defining an "exception space" that shrinks as the weight difference increases. Only points *inside* the circle will match the point with the larger weight.

When the weights are equal, we have the special case of the hyperplane given above. More generally, given a space with many exemplars, the exemplars with the smallest weights (or best classification performance) partition the space with a set of hyperplanes. If the weights of these "best" exemplars are not identical, the partitioning uses very large circles. Each exemplar is effectively the "rule" for its region of space. Exemplars with larger weights define exception spaces around themselves. Figure 3 shows that

Figure 2: Two weighted points in instance space

Figure 3: Partitions with exception spaces

within each exception space, this process may recur if other groups of exemplars have approximately equal weights.

The ability to partition space into large, general "rules" with pockets of exceptions is important in domains that contain many exceptions. Without this capability, many more points are required for learning, as it is necessary to surround exceptions with a set of non-exception points to define the edge of the space. Here, only two points are required to define a rule and an exception. The capability becomes even more important for IBL models that store only a subset of the training examples, because it further reduces the number of points which must be stored (Cost and Salzberg, 1990).

Given the above discussion, it should be clear that all instances should not be initialized with weights of 1. Consider a system trained on $n-1$ instances, now training on the n^{th}. A hierarchy of instance weights has already been constructed through training to represent the structure of the domain. An instance entered with a weight of 1 would immediately become one of the most influential classifiers in the space. We have found that a better strategy is to initialize a new instance with a weight equal to that of its matching exemplar. We have adopted this weighting strategy in the experiments described below. This weighting scheme completes the Modified Value Difference Metric.

3 Domains

We chose for our comparisons three domains that have received considerable attention from the machine learning research community: the word pronunciation task (Sejnowski and Rosenberg, 1986; Shavlik et al., 1989), the prediction of protein secondary structure (Qian and Sejnowski, 1988; Holley and Karplus, 1989), and the prediction of DNA promoter sequences (Towell et al., 1989). Each domain has only symbolic-valued features; thus, our MVDM is applicable whereas standard Euclidean distance is not. Sections 3.1–3.3 describe the three databases and the problems they present for learning.

3.1 Protein secondary structure

Accurate techniques for predicting the folded structure of proteins do not yet exist, despite increasingly numerous attempts to solve this problem. Most techniques depend in part on prediction of the secondary structure from the primary sequence of amino acids. The secondary structure and other information can then be used to construct the final, tertiary structure. Tertiary structure is very difficult to derive directly, requiring expensive methods of X-ray crystallography. The primary sequence, or sequence of amino acids which constitute a protein, is relatively easy to discover. Attempts to predict secondary structure involve the classification of residues into three categories:

α helix, β sheet, and coil. Three of the most widely used approaches to this problem are those of Robson (Garnier et al., 1978), Chou and Fasman (1978), and Lim (1974), which produce classification accuracies ranging from 48% to 58%. Other, more accurate techniques have been developed for predicting tertiary from secondary structure (e.g., Cohen et al., 1986; Lathrop et al., 1987), but the accurate prediction of secondary structure has proven to be an extremely difficult task.

The learning problem can be described as follows. A protein consists of a sequence of amino acids bonded together as a chain. This sequence is known as the primary structure. Each amino acid in the chain can be one of twenty different acids. At the point at which two acids join in the chain, various factors including their own chemical properties determine the angle of the molecular bond between them. This angle, for our purposes, is characterized as one of three different types of "fold": α helix, β sheet, or coil. In other words, if a certain number of consecutive acids (hereafter residues) in the chain join in a manner which we call α, that segment of the chain is an α helix. This characterization of fold types for a protein is known as the secondary structure. The learning problem, then, is: given a sequence of residues from a fixed length window from a protein chain, classify the central residue in the window as α helix, β sheet, or coil. The setup is simply:

$$\text{....TDYGNDVEY}\overbrace{\text{XGQVT}\underbrace{\text{ E }}_{\text{central residue}}\text{GTPGK}}^{\text{window}}\text{SFNLNFDTG....}$$

Qian and Sejnowski (1988) and Holley and Karplus (1989) formulated the problem in exactly the same manner. Both of these studies found the optimal window size to be approximately 17 residues (21 was the largest window tested in either study). In a separate statistical study, Cost (1990) found that a window of size five or six is nearly sufficient for uniquely identifying all residues in our data set, as is indicated by Table 3.

Table 3: Percent unique patterns by window size

	Skew						
Window size	-3	-2	-1	0	1	2	3
1	00.00	00.00	04.76	00.00	04.76	00.00	00.00
2	01.43	02.38	03.09	00.25	04.77	01.67	00.71
3	45.44	45.64	47.33	48.81	47.69	46.06	45.62
4	91.39	91.75	91.95	92.01	92.11	91.90	91.61
5	98.73	98.78	98.78	98.75	98.69	98.65	98.53
6	99.35	99.41	99.41	99.41	99.36	99.29	99.23
7	99.50	99.53	99.54	99.52	99.48	99.42	99.36
8	99.55	99.58	99.60	99.59	99.59	99.54	99.49
9	99.62	99.63	99.63	99.64	99.62	99.58	99.54
10	99.63	99.65	99.65	99.67	99.66	99.64	99.60
11	99.66	99.68	99.69	99.68	99.67	99.65	99.62
12	99.68	99.68	99.70	99.70	99.69	99.68	99.68
13	99.71	99.71	99.71	99.70	99.69	99.70	99.69
14	99.72	99.73	99.72	99.72	99.71	99.71	99.71
15	99.75	99.74	99.72	99.72	99.73	99.72	99.73
16		99.75	99.74	99.73	99.74	99.74	99.74
17		99.76	99.75	99.75	99.74	99.75	
18			99.76	99.76	99.75	99.75	
19			99.76	99.76	99.75		
20				99.76	99.76		
21				99.77			

This table shows the percentage of sequences of a given size which unambiguously (for the entire data set) determine a fold classification for a protein segment. For example, if we consider a window size of six centered on the residue being classified, we found that 99.41% of the patterns in the data set were unique. In addition, we found that there is slightly more information contained in residues to the left of the point of prediction than to the right. (The point of prediction is the residue for which the secondary structure must be predicted.) The "skew" at the top of each column in the table indicates the left or right shift of the pattern with respect to the center of the window; e.g., a skew of -2 means the pattern was centered two positions to the left of the point of prediction. The table shows quite clearly that, if one stored *all* patterns of length 6 in the data set, one could then classify the data set with better than 99% accuracy.

Some of the obstacles to good performance in this domain include under-sampling and non-local effects. Considering only a window of size five and the database that we are using, at most only about 21,618 of 3.2 million possible segments are represented in the database, or 0.68%. Also, proteins in solution form globular structures, the net result of which is that residues which are sequentially very far from each other may be physically quite close, and have significant effects on each other. For this reason, secondary structure probably cannot be completely determined from primary structure. Qian and Sejnowski (1988) claim that no method incorporating only local information will perform much better than current results in the 60-70% range (for non-homologous proteins).

3.2 Promoter sequences

The promoter sequence database was the subject of several recent experiments by Towell et al. (1990). Related to the protein folding task, it involves predicting whether or not a given subsequence of a DNA sequence is a promoter — a sequence of genes that initates a process called transcription, the expression of an adjacent gene. This data set contains 106 examples, 53 of which are positive examples (promoters). The negative examples were generated from larger DNA sequences that are believed to contain no promoters. See Towell et al. (1990) for more detail on the construction of the data set. An instance consists of a sequence of 57 nucleotides from the alphabet a, c, g and t, and a classification of $+$ or $-$. For learning, the 57 nucleotides are treated as 57 features, each with one of four symbolic values.

3.3 Pronunciation of English text

The word pronunciation problem presents interesting challenges for machine learning, although effective practical algorithms have been developed for this task. Given a relatively small sequence of letters, the objective is to learn the sound and stress required to pronounce each part of a given word. Sejnowski and Rosenberg (1987) introduced this task to the learning community with their NETtalk program. NETtalk, which used the back propagation learning method, performed well on this task when pronouncing both words and continuous spoken text, although it could not match the performance of current speech synthesis programs.

The instance representation for text pronunciation is very similar to the previous problems. Instances are sequences of letters which make up a word, and the task is to classify the central letter in the sequence with its correct phoneme. We used a fixed window of seven characters for our experiments, as did Sejnowski and Rosenberg. (Stanfill and Waltz (1986) used a window of size 15.) The classes include 54 phonemes plus 5 stress classifications. When phoneme and stress are predicted, there are $5 \times 54 = 270$ possible classes, although only 115 actually occur in the dictionary. Our experiments emphasized prediction of phonemes only.

The difficulties in this domain arise from the irregularity of natural language, and the English language in particular. Few rules exist that do not have exceptions. Better performance on the same data set can been obtained with (non-learning) rule-based approaches (Kontogiorgios, 1988); however, learning algorithms have trouble finding the best set of rules.

4 Experimental results

In this section, we describe our experiments and results on each of the three test domains. For comparison, we use previously published results for other learning methods. In order

Table 4: Composition of training and test sets

	# of protein segments	# of residues	% α	% β	% coil
Train	100	17142	26.1	19.5	54.4
Test	28	4476	21.8	23.1	55.1

to make the comparisons valid, we attempted to duplicate the experimental design of earlier studies as closely as possible, and we used the same data as was used by those studies.

4.1 Protein secondary structure

The protein sequences used for our experiments were originally from the Brookhaven National Laboratory. Secondary structure assignments of α-helix, β-sheet, and coil were made based on atomic coordinates using the method of Kabsch and Sander (1983). Qian and Sejnowski (1988) collected a database of 106 proteins, containing 128 protein segments (which they called "subunits"). We used the same set of proteins and segments that they used. A parallel experiment by Sigillito (1989), using back propagation on the identical data, reproduced the classification accuracy results of Qian and Sejnowski.

For our initial experiment, we divided the data into a training set containing 100 protein segments and a test set containing 28 segments. There was no overlap between the two sets. Table 4 shows the composition of the two sets. Table 4 shows that the percentages of the three categories were approximately the same in the test set as in the training set.[3] Protein segments were not separated for the main experiments; i.e., all instances drawn from one segment resided together either in the training or the testing set.

[3]Qian and Sejnowski carefully balanced the overall frequencies of the three categories in the training and test sets, and we attempted to do the same. In addition, they used a training set with 18,105 residues, while ours was slightly smaller. Although our databases were identical, we did not have access to the specific partitioning into training and test sets used by Qian and Sejnowski.

Table 5: Classification accuracy (%) by window size

Window size	Unweighted PEBLS	PEBLS	Holley & Karplus	Qian & Sejnowski
3	57.5	57.6	60.0	57.7
5	60.9	61.4	60.6	60.5
7	62.7	63.8	59.6	61.9
9	64.7	65.6	62.3	62.3
11	66.0	67.2	61.6	62.1
13	66.5	68.1	62.7	62.7
15	67.6	70.0	62.9	62.2
17	68.0	70.8	63.2	61.5
19	69.2	71.0	62.6	—
21	69.1	69.8	62.9	61.6

PEBLS was trained as described above on the training set, using $r = 1$ for Equation 2. (We found in preliminary experiments that this produced slightly improved accuracy for this domain.) We repeated the main experiment for a variety of different window sizes, ranging from 3 to 21. For this domain, PEBLS included a post-processing algorithm based on the minimal sequence length restrictions used by Holley and Karplus (1989). These restrictions stated that a β-sheet must consist of a contiguous sequence of no fewer than two such residues, and an α-helix no fewer than four. Where subsequences were predicted that did not conform to these restrictions, the individual residues were re-classified as *coil.* Qian and Sejnowski (1988) used a different form of post-processing, which they called a "cascaded" neural net. They fed the output of one net into another network, which then attempted to re-classify some of the residues. The second network was designed to "take advantage of ... correlations between neighboring secondary structure assignments."

Our results on classification accuracy are given in Table 5. The "unweighted" PEBLS column shows results using PEBLS without the weights w_X on exemplars. The entries in Table 5 are the percentages of correct predictions for the test set. As the table shows, the highest accuracy was produced by PEBLS, which achieved 71.0% with a window of size 19. Qian and Sejnowski obtained their best result, 64.3%, using a *cascaded* network architecture. When they used a single network design (similar to that of Holley and Karplus), their best result was 62.7%. The best performance of PEBLS without post-processing was 67.8%. The best conventional technique, as reported by Holley and Karplus, produced accuracies of only 55%. We also performed experiments using the overlap metric, which produced accuracies in the 55–60% range for different window sizes.

A matched pairs analysis reveals that the weighted version of PEBLS performs significantly better than the unweighted version. In particular, a t-test shows the weighted version to be better at the 99.95% confidence level ($t = 5.48$,d.f.$= 9$). Thus the exemplar weights did improve performance significantly.

Another frequently used measure of performance in this domain are the correlation coefficients, which provide a measure of accuracy for each of the categories. They are defined by the following equation, from (Mathews, 1975):

$$C_\alpha = \frac{p_\alpha \times n_\alpha - u_\alpha \times o_\alpha}{\sqrt{(n_\alpha + u_\alpha)(n_\alpha + o_\alpha)(p_\alpha + u_\alpha)(p_\alpha + o_\alpha)}}$$

where p_α is the number times α was correctly predicted, n_α is the number of times α was correctly rejected, o_α is the number of false positives for α, and u_α is the number of misses (α was correct but not predicted). Similar definitions were used for C_β and C_{coil}. These coefficients for PEBLS and for the two back propagation experiments appear in Table 6.

Variations in training set size A third measure of classification performance involves repeated testing of *randomly* selected test sets. Table 7 shows the performance of PEBLS (weighted) when trained on varying percentages of randomly selected instances from the entire data set, using a window of size 19. In each trial here, a set of examples

Table 6: Comparison of correlation coefficients

Algorithm	% correct	C_α	C_β	C_{coil}
PEBLS	71.0	0.47	0.45	0.40
Qian and Sejnowski	64.3	0.41	0.31	0.41
Holley and Karplus	63.2	0.41	0.32	0.46

Table 7: Training PEBLS on varying percentages of the data set

Training set size (%)	Percent correct on test set
10	56.1
20	57.1
30	58.7
40	59.7
50	60.2
60	60.9
70	62.3
80	63.4
90	65.1

was chosen at random for training, and these examples were removed from the data set. The test phase then used the remaining examples. (Since each protein comprises many examples, different parts of a single protein could appear in both the training and test set on a given trial.) Classification accuracy in Table 7 is averaged over ten runs for each training set size. Note that the numbers reported in Table 7 reflect the classification performance of PEBLS *without* the post processing for minimal sequence length restrictions (as explained above, this post processing was part of our experiments and of Holley and Karplus' experiments). Thus, the performance should be compared with the weighted algorithm using the same window size and no post-processing. For weighted PEBLS, this figure was 67.8 (with post processing the accuracy improves to 71.0%). Thus we see that the particular composition of the training and testing sets in the earlier experiment – which was constructed to mimic the design of earlier experiments – improved the accuracy of the learning algorithm.[4]

4.2 Promoter sequences

The experiments run by Towell et al. (1990) on the promoter sequence database were leave-one-out trials. This methodology involves removing one element from the data, training on the all the remaining data, and testing on the one element. Thus, with 106 instances in the database, PEBLS was trained on 105, and tested on the remaining 1. This was performed for each instance in the database, and the entire procedure was repeated 10 times, each time using a different random order of the instances. (Towell et al. also repeated the entire leave-one-out experiment 10 times, using different randomized initial states of their neural nets each time.)

The results are shown in Table 8, which compares PEBLS to Towell et al.'s KBANN algorithm. In addition, we report numbers obtained by Towell et al. for several other machine learning algorithms, including back propagation, ID3, nearest neighbor with the overlap metric, and the best method reported in the biological literature (O'Neill, 1989). Recall that the overlap metric measures distance as the number of features with different values. It is also worth noting that in each of the 10 test runs of PEBLS, the same four instances caused the errors, and that three of these four were negative instances. Towell notes that the negative examples in his database (the same data as used here) were derived by selecting substrings from a fragment of *E. coli* bacteriophage that is "believed not to contain any promoter sites" (Towell et al., 1990, p. 865). We would suggest, based on our results, that four of the examples be re-examined. These four examples might be interesting exceptions to the general patterns for DNA promoters.

[4] One likely source of variation in classification accuracy is homologies between the training and test sets. Homologous proteins are structurally very similar, and an algorithm may be much more accurate at predicting the structure of a protein once it has been trained on a homologous one.

Table 8: Promoter sequence prediction

Algorithm	Error rate
PEBLS	4/106
KBANN	4/106
PEBLS (unweighted)	6/106
Back propagation	8/106
ID3	19/106
Nearest Neighbor (overlap)	13/106
O'Neill	12/106

Table 9: English text pronunciation

Algorithm	Phoneme accuracy	Phoneme/stress
PEBLS	78.2	69.2
PEBLS (unweighted)	79.1	67.2
Back propagation	—	77.0

4.3 English text pronunciation

For the English pronunciation task, we used the training set defined by Sejnowki and Rosenberg (1987) for their NETtalk program. This set consists of all instances drawn from the Brown Corpus, or the 1000 most commonly used words of the English language. We were unable to discern a difference between that training set and the somewhat more restricted set of Shavlik (Shavlik et al., 1989), so only one experimental design was used. After training on the Brown Corpus, PEBLS was tested on the entire 20,012 word Merriam Webster Pocket Dictionary. Results are presented in Table 9 for weighted and unweighted versions of the PEBLS algorithm. For comparison, we give results from the NETtalk program, which used the back propagation learning algorithm.

Shavlik et al. (1989) replicated Sejnowski and Rosenberg's methodology as part of their work, and although their results differ from Sejnowski and Rosenberg's (not surprisingly, since back propagation networks require much tuning), they make for easier

Table 10: Phoneme/stress accuracy and output encoding

Algorithm	Local encoding (% correct)	Distributed encoding (% correct)
PEBLS	69.2	—
Back propagation	63.0	72.3
ID3	64.2	69.3
Perceptron	49.2	42.1

comparison with ours. This property follows from the fact the the original Sejnowski and Rosenberg study used a *distributed* output encoding; that is, their system produced a 26-bit sequence (rather than one bit for each of the 115 phoneme/stress combinations). The first 21 bits were a distributed encoding of the 51 phonemes, and the remaining 5 bits were a local encoding of the stress types. The 21-bit output vector is then matched to the closest of the descriptive vectors for the 51 phonemes. Shavlik et al. explicitly compared this encoding to a purely local encoding. Since the output of PEBLS was always local (i.e., the output was a specific phoneme or phoneme/stress combination), it is more appropriate to compare it to other methods that produced the same output. Table 10 shows our results[5] compared to back propagation, perceptron, and ID3 (Quinlan, 1986), where the latter three results are all from Shavlik et al. (1989). The table shows that PEBLS performed slightly better than the other learning methods when the output was a local encoding. Distributed encoding improved the results of both ID3 and back propagation, but comparable experiments with PEBLS, which would require significant changes to the output function, have not yet been performed.

Shavlik et al. also tested performance of back propagation, ID3, and perceptron learning as a function of the size of the training set. We performed a similar experiment, increasing the show our results in Table 11, and graphically in Figure 4 for comparison.

[5]In our preliminary experiments, we used the overlap metric on this database, with abysmal results. Our desire to improve these results was one of the reasons we developed the MVDM.

Table 11: PEBLS performance on varying training set sizes

Percentage of Brown Corpus for training	% phonemes correct in full dictionary
5	60.1
10	66.2
25	72.1
50	75.8
75	77.4
100	78.2

Figure 4: Classification accuracy as a function of training set size

Results are averaged over 10 runs, with different randomly-chosen training sets on each run. Not surprisingly, performance improves steadily as the size of the training set increases. What was surprising, though, was how good performance was with even very small training sets.

5 Discussion

5.1 Classification accuracy

Our studies show that the classification accuracy of PEBLS is, in general, equal or slightly superior to that of other learning methods for domains with symbolic features. Most notably, on the protein structure prediction task, PEBLS gives considerably better classification results than back propagation, both with and without weighted exemplars. It should be noted that in what should be considered the most fair test of performance, the random selection of residues for varying percentages of the dataset, the performance figures for our algorithm are slightly worse, albeit still quite good. It would be informative to see a similar experiment run with a neural network learning algorithm. Recently, Zhang and Waltz have investigated a hybrid learning method for protein structure prediction, combining nearest neighbor with neural net learning and statistical information. Figures have not yet been published, but their method also outperforms previous methods (Waltz, 1990), although its accuracy does not exceed that of PEBLS.

For the DNA promoter sequence prediction, Towell et al. (1990) report that KBANN, a technique that integrates neural nets and domain knowledge, is superior to standard back propagation with 99.95% certainty ($t = 5.29$, d.f.= 18). KBANN was designed specifically to show how adding domain knowledge could improve the performance of a neural net learning algorithm. In addition, KBANN outperformed ID3 and nearest neighbor, when nearest neighbor was using the overlap metric. Using the same experimental design, PEBLS exactly matched the performance of KBANN, and by the same measures was superior to back propagation, ID3, and the O'Neill method on this data. The strong performance of PEBLS on this data set demonstrates that nearest neighbor can perform well using both large (protein folding) and small (promoter sequences) training sets. It is especially significant that PEBLS, using a "weak" general method, was able to match the performance of KBANN's knowledge-rich approach.

In the English pronunciation domain, the results are mixed. The best result of Sejnowski and Rosenberg, 77%, is superior to the phoneme/stress accuracy of PEBLS, 69.2%. However, when Shavlik et al. replicated the experiment, their best result was 72.3%, and as we note above, both the neural net results reflect a distributed output encoding. With a local encoding – which is what PEBLS produces – back propagation's classification accuracy is 63.0%, and ID3 is 64.2%, both somewhat lower than PEBLS. Our conclusion is that all techniques perform similarly, and that no learning technique yet comes close to the performance of good commercial systems, much less native speakers of English. Clearly, there is still room for considerable progress in this domain.

Shavlik et al. concluded, based on their experiments with classification accuracy versus number of training examples (see Figure 4 on the NETtalk data), that for small amounts of training data, back propagation was preferable to the decision trees constructed by ID3. However, our results indicate that nearest neighbor algorithms also work well when the training set is small. Our performance curve in Figure 4 shows that PEBLS needs very few examples to achieve relatively good performance.

5.2 Transparency of representation and operation

Once trained on a given domain, PEBLS contains in its memory a set of information that is relatively perspicuous in comparison to the weight assignments of a neural network. The exemplars themselves provide specific reference instances, or "case histories" as it were, which may be cited as support for a particular decision. Other information may easily be gathered during the training or even the testing phase that can shed additional light on the domain in question. For instance, consider attaching a counter to each exemplar and incrementing it each time the exemplar is used as an *exact* match. By comparing this with the number of times an exemplar was used, we can get a good idea as to whether the exemplar is a very specific exception, or part of a very general rule. By examining the weight w_X we attach to exemplars, we can determine whether the instance is a reliable classifier. The distance tables reveal an order on the set of symbolic values that is not apparent in the values alone. On the other hand, the derivation of these distances is not perspicuous, being derived from global characteristics of the training data.

For the English pronunciation task, distributed output encodings have been shown to produce superior performance to local encodings (Shavlik et al. 1989). This result points out a weakness of PEBLS, and of the 1-nearest-neighbor method, in that they do not allow for distributed output encodings. Neural nets can handle such encodings quite easily, and decision trees can handle them with some difficulty. (Shavlik et al. built a separate decision tree for each of the 26 bits in the distributed encoding of the phoneme/stress pairs in this task.) This raises the question of whether nearest neighbor methods can handle such encodings. One possibility is to use k-nearest neighbor, which would allow more than one exemplar to determine each of the output bits. E.g., if each exemplar contained the 26-bit encoding, the predicted value of each bit i for a new example would be determined by the majority vote of the k nearest neighbors for that bit. Further experiments are required to determine if such a strategy would be advantageous in general.

As for transparency of operation, the learning and classification algorithms of the nearest neighbor algorithm are very simple. The basic learning routine simply stores new examples in memory. In PEBLS, the computation of exemplar weights is nothing more than simple record-keeping based on the classification performance of the existing exemplars. Adding exemplars and changing weights change the way nearest neighbor algorithms partition a feature space, as we have illustrated above with our exception spaces. Although this may be hard to visualize in more than three dimensions, it is nonetheless straightforward. Compare this operation to the adjustment of weights by the back propagation algorithm; thus far, researchers have found it very difficult to characterize how classification performance is changed when connection weights are changed.

One minor drawback is that the PEBLS method is nonincremental, unlike back propagation and some versions of decision tree methods. An incremental extension to PEBLS would probably be quite expensive, since the value difference tables might have to be recomputed many times. On the other hand, extending PEBLS to handle mixed symbolic and numeric data is quite straightforward: the algorithm could use simple differences for numeric features, and value difference tables for symbolic ones.

Finally, our experiments in the protein domain demonstrated that the use of weights attached to exemplars can improve the accuracy of nearest neighbor algorithms. In other domains, such as English pronunciation, weights did not make a significant difference. Based on these results, and our earlier results on real-valued domains (Salzberg, 1990, 1991), we conclude that exemplar weights offer real potential for enhancing the power of practical learning algorithms.

6 Conclusion

We have demonstrated, through a series of experiments, that an instance-based learning algorithm can perform exceptionally well on domains in which features values are symbolic. In direct comparisons, our implementation (PEBLS) performed as well as (or better than) back propagation, ID3, and several domain-specific learning algorithms on several difficult classification tasks. In addition, nearest neighbor offers clear advantages in that it is much faster to train and its representation relatively easy to interpret. No one yet knows how to interpret the networks of weights learned by neural nets. Decision trees are somewhat easier to interpret, but it is hard to predict the impact of a new example on the structure of the tree. Sometimes one new example makes no difference at all, and at other times it may radically change a large portion of the tree. On the other hand, neural nets have a fixed size, and decision trees tend to be quite small, and in this respect both methods compress the data in a way that nearest neighbor does not. In addition, classification time is fast (dependent only on the depth of the net or tree, not on the size of the input). Based on classification accuracy, though, it is not clear that other learning techniques have an advantage over nearest-neighbor methods.

With respect to nearest neighbor learning per se, we have shown how weighting exemplars can improve performance by subdividing the instance space in a manner that reduces the impact of unreliable examples. The nearest neighbor algorithm is one of the simplest learning methods known, and yet no other algorithm has been shown to outperform it consistently. Taken together, these results indicate that continued research on extending and improving nearest neighbor learning algorithms should prove fruitful.

Acknowledgements. Thanks to Joanne Houlahan and David Aha for numerous insightful comments and suggestions. Thanks also to Richard Sutton and three anonymous reviewers for their detailed comments and ideas. This research was supported in part by the Air Force Office of Scientific Research under Grant AFOSR-89-0151, and by the National Science Foundation under Grant and IRI-9116843.

References

Aha, D. (1989) Incremental, instance-based learning of independent and graded concept descriptions. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 387-391). Ithaca, NY: Morgan Kaufmann.

Aha, D. and Kibler, D. (1989) Noise-tolerant instace-based learning algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794-799). Detroit, MI: Morgan Kaufmann.

Aha, D. (1990) A Study of Instance-Based Algorithms for Supervised Learning Tasks. Ph.D. Thesis, Technical Report 90-42, Department of Information and Computer Science, University of California, Irvine.

Aha, D., Kibler, D., and Albert, M. (1991) Instance-Based Learning Algorithms. *Machine Learning*, 6:1, 37-66.

Chou, P. and Fasman, G. (1978) Prediction of the secondary structure of proteins from their amino acid sequence. *Advanced Enzymology, 47* (1978), 45-148. *Biochemistry, 13*, 222-245.

Cohen, F., Abarbanel, R., Kuntz, I., and Fletterick, R. (1986) Turn Prediction in Proteins Using a Pattern Matching Approach. *Biochemistry, 25*, 266-275.

Cost, S. (1990) M.S. Thesis, Department of Computer Science, Johns Hopkins University.

Cost, S. and Salzberg, S. (1990) Exemplar-based Learning to Predict Protein Folding. *Proceedings of the Symposium on Computer Applications to Medical Care*, Washington, D.C., November 1990.

Cover, T. and Hart, P. (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:1, 21-27.

Crick, F., and Asanuma, C. (1986) Certain aspects of the anatomy and physiology of the cerebral cortex. In *Parallel Distributed Processing: explorations in the microstructure of cognition*, vol II, J. McClelland, D. Rumelhart, and the PDP Research Group (Eds.), 170-215. Cambridge, MA: MIT Press.

Dietterich, T., Hild, H., and Bakiri, G. (1990) A comparative study of ID3 and backpropagation for English text-to-speech mapping. *Proceedings of the 7th International Conference on Machine Learning*, 24-31.

Fertig, S. and Gelernter, D. (1991) FGP: A virtual machine for acquiring knowledge from cases. *Proceedings of the 12th International Joint Conference on Artificial Intelligence* (pp. 796-802). Los Altos, CA: Morgan Kaufmann.

Fisher, D., and McKusick, K. (1989) An empirical comparison of ID3 and back-propagation. *Proceedings of the International Joint Conference on Artificial Intelligence*, 788-793. San Mateo, CA: Morgan Kaufmann.

Garnier, J., Osguthorpe, D., and Robson, B. (1978) *Journal of Molecular Biology, 120*, 97-120.

Hanson, S. and Burr, D. (1990) What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences, 13,* 471-518.

Holley, L. and Karplus, M. (1989) Protein Secondary Structure Prediction with a Neural Network. *Proceedings of the National Academy of Sciences USA*, 86, 152-156.

Kabsch, W. and Sander, C. (1983) Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometric features. *Biopolymers, 22* (1983), 2577-2637.

Kontogiorgis, S. (1988) Automatic Letter-to-Phoneme Transcription for Speech Synthesis. Technical Report JHU-88/22, Department of Computer Science, Johns Hopkins University.

Lathrop, R., Webster, T. and Smith, T. (1987) ARIADNE: Pattern-directed Inference and Hierarchical Abstraction in Protein Structure Recognition. *Communications of the ACM*, 30:11, 909-921.

Lim, V. (1974) Algorithms for prediction of α-helical and *beta*-structural regions in globular proteins. *Journal of Molecular Biology, 88*, 873-894.

Mathews, B. W. (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta., 405*, 442-451.

McClelland, J. and Rumelhart, D. (1986) A distributed model of human learning and memory. In *Parallel Distributed Processing: explorations in the microstructure of cognition*, vol II, J. McClelland, D. Rumelhart, and the PDP Research Group (Eds.), 170-215. Cambridge, MA: MIT Press.

Mooney, R., Shavlik, J., Towell, G., and Gove, A. (1989) An experimental comparison of symbolic and connectionist learning algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, 775-780. San Mateo, CA: Morgan Kaufmann.

Medin, D. and Schaffer, M. (1978) Context theory of classification learning. *Psychological Review, 85:3*, 207-238.

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

O'Neill, M. (1989) Escherichia coli promoters: I. Consensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organization. *Journal of Biological Chemistry, 264*, 5522-5530.

Preparata, F. and Shamos, M. (1985) *Computational Geometry: An Introduction.* New York: Springer-Verlag.

Qian, N. and Sejnowski, T. (1988) Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology, 202*, 865-884.

Reed, S. (1972) Pattern Recognition and Categorization. *Cognitive Psychology, 3*, 382-407.

Rumelhart, D., Hinton, G., and Williams, R. (1986) Learning representations by back-propagating errors. *Nature 323:9*, October 1986, 533-536.

Rumelhart, D., Smolensky, P., McClelland, J., and Hinton, G. (1986) Schemata and sequential thought processes in PDP models. In *Parallel Distributed Processing: explorations in the microstructure of cognition*, vol II, J. McClelland, D. Rumelhart, and the PDP Research Group (Eds.), 7-57. Cambridge, MA: MIT Press.

Rumelhart, D., McClelland, J., and the PDP Research Group (1986) *Parallel Distributed Processing: explorations in the microstructure of cognition*, vol I. Cambridge, MA: MIT Press.

Salzberg, S. (1989) Nested Hyper-rectangles for Exemplar-based Learning. In K.P. Jantke (ed.), *Analogical and Inductive Inference: International Workshop AII '89*, 184-201. Berlin: Springer- Verlag.

Salzberg, S. (1990) *Learning with Nested Generalized Exemplars.* Norwell, MA: Kluwer Academic Publishers.

Salzberg, S. (1991) A nearest hyperrectangle learning method. *Machine Learning,* 6:3, 251-276.

Sejnowski, T. and Rosenberg, C. (1987) NETtalk: A Parallel Network that Learns to Read Aloud. *Complex Systems, 1*, 145-168. Also Technical Report JHU/EECS-86/01, Johns Hopkins University, Baltimore, MD, 1986.

Shavlik, J., Mooney, R., and Towell, G. (1989) Symbolic and neural learning algorithms: an experimental comparison. Technical Report #857, Computer Sciences Department, University of Wisconsin, Madison, WI, June 1989.

Sigillito, V. (1989) Personal communication.

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

Towell, G., Shavlik, J., and Noordewier, M. (1990) Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings Eighth National Conference on Artificial Intelligence*, 861-866. Menlo Park, CA: AAAI Press.

Waltz, D. (1990) Massively parallel AI. *Proceedings Eighth National Conference on Artificial Intelligence*, 1117-1122. Menlo Park, CA: AAAI Press.

Weiss, S. and Kapouleas, I. (1989) An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. *Proceedings of the International Joint Conference on Artificial Intelligence*, 781-787. San Mateo, CA: Morgan Kaufmann.