Locally Adaptive Metric Nearest Neighbor Classification

Carlotta Domeniconi
Computer Science Department
University of California
Riverside, CA 92521
carlotta@cs.ucr.edu

Jing Peng
Computer Science Department
Oklahoma State University
Stillwater, OK 74078
jpeng@cs.okstate.edu

Dimitrios Gunopulos
Computer Science Department
University of California
Riverside, CA 92521
dg@cs.ucr.edu

Technical Report UCR-CSE-00-01

August 10, 2000

Abstract

Nearest neighbor classification assumes locally constant class conditional probabilities. This assumption becomes invalid in high dimensions with finite samples due to the curse of dimensionality. Severe bias can be introduced under these conditions when using the nearest neighbor rule. We propose a locally adaptive nearest neighbor classification method to try to minimize bias. We use a *Chi-squared* distance analysis to compute a flexible metric for producing neighborhoods that are highly adaptive to query locations. Neighborhoods are elongated along less relevant feature dimensions and constricted along most influential ones. As a result, the class conditional probabilities tend to be smoother in the modified neighborhoods, whereby better classification performance can be achieved. The efficacy of our method is validated and compared against other techniques using a variety of simulated and real world data.

Index terms: Chi-squared distance, classification, feature relevance, nearest neighbors.

1 Introduction

In a classification problem, we are given J classes and N training observations. The training observations consist of q feature measurements $\mathbf{x} = (x_1, \cdots, x_q) \in \Re^q$ and the known class labels, L_j, $j = 1, \ldots, J$. The goal is to predict the class label of a given query $\mathbf{x}_0$.

The K nearest neighbor classification method [6, 11, 12, 13, 16, 17] is a simple and appealing approach to this problem: it finds the K nearest neighbors of $\mathbf{x}_0$ in the training set, and

then predicts the class label of $\mathbf{x}_0$ as the most frequent one occurring in the K neighbors. Such a method produces continuous and overlapping, rather than fixed, neighborhoods and uses a different neighborhood for each individual query so that all points in the neighborhood are close to the query, to the extent possible. In addition, it has been shown [7, 8] that the one nearest neighbor rule has asymptotic error rate that is at most twice the Bayes error rate, independent of the distance metric used.

The nearest neighbor rule becomes less appealing with finite training samples, however. This is due to the curse-of-dimensionality [3]. Severe bias can be introduced in the nearest neighbor rule in a high dimensional input feature space with finite samples. As such, the choice of a distance measure becomes crucial in determining the outcome of nearest neighbor classification. The commonly used Euclidean distance measure, while simple computationally, implies that the input space is isotropic or homogeneous. However, the assumption for isotropy is often invalid and generally undesirable in many practical applications. Figure 1 illustrates a case in point, where class boundaries are parallel to the coordinate axes. For query a, dimension X is more relevant, because a slight move along the X axis may change the class label, while for query b, dimension Y is more relevant. For query c, however, both dimensions are equally relevant. This implies that distance computation does not vary with equal strength or in the same proportion in all directions in the feature space emanating from the input query. Capturing such information, therefore, is of great importance to any classification procedure in high dimensional settings.

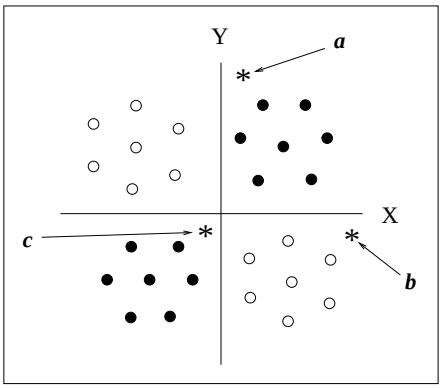


Figure 1: Feature relevance varies with query locations.

In this paper we propose an adaptive nearest neighbor classification method to try to minimize bias in high dimensions. We estimate a flexible metric for computing neighborhoods based on *Chi-squared* distance analysis. The resulting neighborhoods are highly adaptive to query locations. Moreover, the neighborhoods are elongated along less relevant feature dimensions and constricted along most influential ones. As a result, the class conditional probabilities tend to be constant in the modified neighborhoods, whereby better classification performance can be obtained.

Figure 2 shows an example. There are two classes and the data for both classes are generated from a bivariate standard normal distribution. The data for class one have the radius less than or equal to 1.15, while the data for class two have the radius greater than 1.15. As a result, class one is surrounded by class two. Figure 2(a) shows the nearest neighborhood of size 50 of a query located at (0, -1) near the class boundary. This neighborhood is computed using the Euclidian distance metric. Figure 2(b) shows the same size neighborhood computed by using our adaptive nearest neighbor classification algorithm. Note how the modified neighborhood is elongated along the direction of the true decision boundary and constricted along the direction orthogonal to it, which is the most relevant direction for the given query.

The paper is organized as follows. In section 2 we motivate and present our approach for measuring local feature relevance. Section 3 describes how to estimate the quantities involved in our local feature relevance measure. In section 4 we formally present our algorithm and the parameters involved. Section 5 shows that the averaging process performed by our approach can reduce the mean-squared error for feature relevance estimation. Section 6 presents the methods we consider for comparison in our experiments. Section 6.1 compares the methods through a set of simulated examples while section 6.2 uses real data examples. Section 7 is a discussion of related work and a concluding summary is given in section 8.

2 Local Feature Relevance Measure

Kernel methods are based on the assumption of smoothness of the target functions, which translates to locally constant class posterior probabilities for a classification problem. This

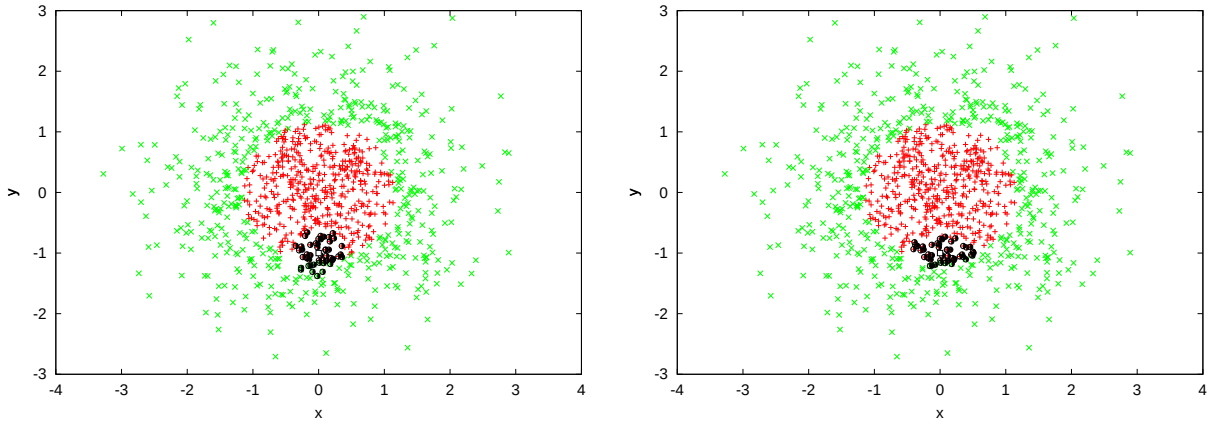


Figure 2: Plot (a) shows the spherical neighborhood of the query point $(0, -1)$ containing 50 points (shown as darker circles). Plot (b) shows the corresponding neighborhood found by our algorithm to be described in this paper, also containing 50 points. After applying our adaptive procedure the neighborhood is constricted along the most relevant dimension and elongated along the less important one.

assumption, however, becomes invalid for any fixed distance metric when the input observation $\mathbf{x}_0$ approaches class boundaries, as illustrated in Figure 1. In the following, we describe a nearest neighbor classification technique that is capable of producing a local neighborhood in which the posterior probabilities are approximately constant, and that is highly adaptive to query locations.

2.1 Chi-Squared Distance

Our technique is motivated as follows. Let $\mathbf{x}_0$ be the test point whose class membership we are predicting. In the one nearest neighbor (NN) classification rule, a single nearest neighbor $\mathbf{x}$ is found according to a distance metric $D(\mathbf{x}, \mathbf{x}_0)$. Let $\Pr(j|\mathbf{x})$ be the class conditional probability at point $\mathbf{x}$. Consider the weighted *Chi-squared* distance [10, 14]

$$D(\mathbf{x}, \mathbf{x}_0) = \sum_{j=1}^{J} \frac{[\Pr(j|\mathbf{x}) - \Pr(j|\mathbf{x}_0)]^2}{\Pr(j|\mathbf{x}_0)}, \tag{1}$$

which measures the distance between the test point $\mathbf{x}_0$ and the point $\mathbf{x}$, in terms of the difference between the class posterior probabilities at the two points. Small $D(\mathbf{x}, \mathbf{x}_0)$ indicates that the classification error rate will be close to the asymptotic error rate for the one nearest

neighbor rule. In general, this can be achieved when $\Pr(j|\mathbf{x}) = \Pr(j|\mathbf{x}_0)$, which states that if $\Pr(j|\mathbf{x})$ can be sufficiently well approximated at $\mathbf{x}_0$, the asymptotic 1-NN error rate might result in finite sample settings.

Note that in comparison to the *Chi-squared* distance

$$d(\mathbf{x}, \mathbf{x}_0) = \sum_{j=1}^{J} [\Pr(j|\mathbf{x}) - \Pr(j|\mathbf{x}_0)]^2,$$

the weights, $1/\Pr(j|\mathbf{x}_0)$, in (1) have the effect of increasing the distance of $\mathbf{x}_0$ to any point $\mathbf{x}$ whose likely class is unlikely to include $\mathbf{x}_0$. That is, if $j^* = \arg\max_j \Pr(j|\mathbf{x})$, we have

$$\Pr(j^*|\mathbf{x}_0) \approx 0.$$

As a consequence, it becomes highly improbable for any such point to be a nearest neighbor candidate. In general, such a weighting benefits any nearest neighbor classifiers whose distance metric approximates the *Chi-squared* distance.

Equation (1) computes the distance between the true and estimated posteriors. Now, imagine we replace $\Pr(j|\mathbf{x}_0)$ with a quantity that attempts to predict $\Pr(j|\mathbf{x})$ under the constraint that the quantity is conditioned at a location along a particular feature dimension. Then, the *Chi-squared* distance (1) tells us the extent to which that dimension can be relied on to predict $\Pr(j|\mathbf{x})$. Thus, Equation (1) provides us with a foundation upon which to develop a theory of feature relevance in the context of pattern classification.

2.2 Local Feature Relevance

Based on the above discussion, our proposal is the following. We first notice that $\Pr(j|\mathbf{x})$ is a function of $\mathbf{x}$. Therefore, we can compute the conditional expectation of $\Pr(j|\mathbf{x})$, denoted by $\overline{\Pr}(j|x_i = z)$, given that x_i assumes value z, where x_i represents the ith component of $\mathbf{x}$. That is,

$$\begin{aligned} \overline{\Pr}(j|x_i = z) &= E[\Pr(j|\mathbf{x})|x_i = z] \\ &= \int \Pr(j|\mathbf{x}) p(\mathbf{x}|x_i = z) d\mathbf{x} \end{aligned} \tag{2}$$

Here $p(\mathbf{x}|x_i = z)$ is the conditional density of the other input variables defined as

$$p(\mathbf{x}|x_i = z) = p(\mathbf{x})\delta(x_i - z) / \int p(\mathbf{x})\delta(x_i - z) d\mathbf{x}, \tag{3}$$

where $\delta(x-z)$ is the Dirac delta function having the properties

$$\delta(x-z) = 0 \quad if \quad x \neq z$$

and

$$\int_{-\infty}^{\infty} \delta(x-z)dx = 1.$$

Let

$$r_i(\mathbf{z}) = \sum_{j=1}^{J} \frac{[\Pr(j|\mathbf{z}) - \overline{\Pr}(j|x_i = z_i)]^2}{\overline{\Pr}(j|x_i = z_i)}. \tag{4}$$

$r_i(\mathbf{z})$ represents the ability of feature i to predict the $\Pr(j|\mathbf{z})$s at $x_i = z_i$. The closer $\overline{\Pr}(j|x_i = z_i)$ is to $\Pr(j|\mathbf{z})$, the more information feature i carries for predicting the class posterior probabilities locally at $\mathbf{z}$.

We can now define a measure of feature relevance for $\mathbf{x}_0$ as

$$\bar{r}_i(\mathbf{x}_0) = \frac{1}{K} \sum_{\mathbf{z} \in N(\mathbf{x}_0)} r_i(\mathbf{z}), \tag{5}$$

where $N(\mathbf{x}_0)$ denotes the neighborhood of $\mathbf{x}_0$ containing the K nearest training points, according to a given metric. $\bar{r}_i$ measures how well on average the class posterior probabilities can be approximated along input feature i within a local neighborhood of $\mathbf{x}_0$. Small $\bar{r}_i$ implies that the class posterior probabilities will be well captured along dimension i in the vicinity of $\mathbf{x}_0$. Note that $\bar{r}_i(\mathbf{x}_0)$ is a function of both the test point $\mathbf{x}_0$ and the dimension i, thereby making $\bar{r}_i(\mathbf{x}_0)$ a local relevance measure.

To formulate the measure of feature relevance as a weighting scheme, we first define

$$R_i(\mathbf{x}_0) = \max_j \{\bar{r}_j(\mathbf{x}_0)\} - \bar{r}_i(\mathbf{x}_0).$$

A weighting scheme can then be given by

$$w_i(\mathbf{x}_0) = (R_i(\mathbf{x}_0))^t / \sum_{l=1}^{q} (R_l(\mathbf{x}_0))^t, \tag{6}$$

where $t = 1, 2$, giving rise to linear and quadratic weightings, respectively. In this paper we propose the following exponential weighting scheme

$$w_i(\mathbf{x}_0) = \exp(cR_i(\mathbf{x}_0)) / \sum_{l=1}^{q} \exp(cR_l(\mathbf{x}_0)) \tag{7}$$

where c is a parameter that can be chosen to maximize (minimize) the influence of $\bar{r}_i$ on w_i. When $c = 0$ we have $w_i = 1/q$, thereby ignoring any difference between the $\bar{r}_i$'s. On the other hand, when c is large a change in $\bar{r}_i$ will be exponentially reflected in w_i. In this case, w_i is said to follow the Boltzmann distribution. The exponential weighting is more sensitive to changes in local feature relevance (5) and gives rise to better performance improvement. Thus, (7) can be used as weights associated with features for weighted distance computation

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{q} w_i (x_i - y_i)^2}. \tag{8}$$

These weights enable the neighborhood to elongate less important feature dimensions, and, at the same time, to constrict the most influential ones. Note that the technique is *query-based* because weightings depend on the query [1, 2].

A justification for (4) and, hence, (5), may go as follows. Suppose that the value of $r_i(\mathbf{z})$ is small, which implies a large weight along dimension i. Consequently, the neighborhood gets shrinked along that direction. This, in turn, penalizes points along dimension i that are moving away from z_i. Now, $r_i(\mathbf{z})$ can be small only if the subspace spanned by the other input dimensions at $x_i = z_i$ likely contains samples similar to $\mathbf{z}$ in terms of the class conditional probabilities. Then, a large weight assigned to dimension i based on (7) says that moving away from the subspace, hence from the data similar to $\mathbf{z}$, is not a good thing to do. Similarly, a large value of $r_i(\mathbf{z})$, hence a small weight, indicates that in the vicinity of z_i along dimension i one is unlikely to find samples similar to $\mathbf{z}$. This corresponds to an elongation of the neighborhood along dimension i. Therefore, in this situation in order to better predict the query, one must look farther away from z_i.

So far we have considered estimating feature relevance along each individual dimension, one at a time. However, there are situations where feature relevance can only be captured by examining several feature variables simultaneously. That is, feature variables are not independent, and there is a degree of correlation among them. It should be clear that, in the absence of any other information, determining which feature(s) should be examined to estimate local relevance adds considerable complexity to feature relevance computation. One way to decorrelate association among the features is to rotate the feature dimensions so that they coincide with the eigenvectors of a sample covariance matrix, as in [10]. Note

that, even without such a transformation to the eigenspace, the technique described here can be readily extended to estimating local relevance, conditioned on multiple feature variables simultaneously . We do not address this issue further in the rest of this paper.

3 Estimation

Since both $\Pr(j|\mathbf{z})$ and $\overline{\Pr}(j|x_i = z_i)$ in (4) are unknown, we must estimate them using the training data

$$\{\mathbf{x}_n, y_n\}_{n=1}^N$$

in order for the relevance measure (5) to be useful in practice. Here $y_n \in \{1, \cdots, J\}$. The quantity $\Pr(j|\mathbf{z})$ is estimated by considering a neighborhood $N_1(\mathbf{z})$ centered at $\mathbf{z}$:

$$\hat{\Pr}(j|\mathbf{z}) = \frac{\sum_{n=1}^N 1(\mathbf{x}_n \in N_1(\mathbf{z}))1(y_n = j)}{\sum_{n=1}^N 1(\mathbf{x}_n \in N_1(\mathbf{z}))}, \tag{9}$$

where $1(\cdot)$ is an indicator function such that it returns 1 when its argument is true, and 0 otherwise.

To compute $\overline{\Pr}(j|x_i = z) = E[\Pr(j|\mathbf{x})|x_i = z]$, we introduce an additional variable g_j such that

$$g_j|\mathbf{x} = \begin{cases} 1 & \text{if } y = j \\ 0 & \text{otherwise} \end{cases}$$

where $j = 1, \cdots, J$. We then have $\Pr(j|\mathbf{x}) = E[g_j|\mathbf{x}]$, from which it is not hard to show that

$$\overline{\Pr}(j|x_i = z) = E[g_j|x_i = z].$$

However, since there may not be any data at $x_i = z$, the data from the neighborhood of z along dimension i are used to estimate $E[g_j|x_i = z]$, a strategy suggested in [9]. In detail, by noticing $g_j = 1(y = j)$ the estimate can be computed from

$$\hat{\overline{\Pr}}(j|x_i = z_i) = \frac{\sum_{\mathbf{x}_n \in N_2(\mathbf{z})} 1(|x_{ni} - z_i| \leq \Delta_i)1(y_n = j)}{\sum_{\mathbf{x}_n \in N_2(\mathbf{z})} 1(|x_{ni} - z_i| \leq \Delta_i)}, \tag{10}$$

where $N_2(\mathbf{z})$ is a neighborhood centered at $\mathbf{z}$ (larger than $N_1(\mathbf{z})$), and the value of Δ_i is chosen so that the interval contains a fixed number L of points:

$$\sum_{n=1}^N 1(|x_{ni} - z_i| \leq \Delta_i)1(\mathbf{x}_n \in N_2(\mathbf{z})) = L. \tag{11}$$

Using the estimates in (9) and in (10), we obtain an empirical measure of the relevance (5) for each input variable i.

Given a test point $\mathbf{x}_0$, and input parameters K_0, K_1, K_2, L, K, and c:

1. Initialize $\mathbf{w}$ in (8) to 1;
2. Compute the K_0 nearest neighbors of $\mathbf{x}_0$ using the weighted distance metric (8);
3. For each dimension i, $i = 1, \ldots, q$, compute relevance estimate $\bar{r}_i(\mathbf{x}_0)$ (5) through Equations (9) and (10);
4. Update $\mathbf{w}$ according to (6) or (7);
5. Iterate steps 2, 3, and 4;
6. At completion, use $\mathbf{w}$, hence (8), for K-nearest neighbor classification at the test point $\mathbf{x}_0$.

Figure 3: The ADAMENN algorithm

4 Adaptive Metric Nearest Neighbor Algorithm

The adaptive metric nearest neighbor algorithm (ADAMENN) has six adjustable tuning parameters:

- K_0: the number of neighbors of the test point;
- K_1: the number of neighbors in $N_1(\mathbf{z})$ for estimation (9);
- K_2: the size of the neighborhood $N_2(\mathbf{z})$ for each of the K_0 neighbors for estimation (10);
- L: the number of points within the Δ intervals;
- K: the number of neighbors in the final nearest neighbor rule;

- c: the positive factor for the exponential weighting scheme (7).

At the beginning, the estimation of the $\bar{r}_i$ values in (5) is accomplished by using a weighted distance metric (8) with $\mathbf{w}$ being initialized to 1. Then, the elements w_i of $\mathbf{w}$ are updated according to $\bar{r}_i$ values via (6) or (7). In our experiments, we tested both a linear and an exponential weighting scheme. We obtained better results using the exponential scheme, therefore we present the results for this case. The update of $\mathbf{w}$ can be iterated. At completion, the resulting $\mathbf{w}$ is plugged in (8) to compute nearest neighbors at the test point $\mathbf{x}_0$.

In all our experiments we obtained optimal performance for small values (one or three) of parameters K_1 and K. Optimal values for parameters K_0 and K_2 are in a range close to respectively 10% and 15% of the number of training points. The value for L is usually set to be roughly half the value of K_2. Different values of the c factor turned out to be optimal for different problems (5, 11, and 16). An outline of the ADAMENN algorithm is shown in Figure 3.

5 Why Averaging

In this section we show more formally that averaging in (5) potentially reduces overall mean-squared estimation error, thereby improving classification performance. Let $\mathbf{x}_0$ be a given query point. For each given dimension i, our goal is to estimate R_i:

$$R_i(\mathbf{x}_0) = \sum_{j=1}^{J} \frac{[\Pr(j|\mathbf{x}_0) - \overline{\Pr}(j|x_i = x_{0i})]^2}{\overline{\Pr}(j|x_i = x_{0i})}. \tag{12}$$

Let $r_i(\mathbf{x}_0, \mathbf{z})$ be the estimator as defined by Equation (4), where $\mathbf{z}$ is in $N(\mathbf{x}_0)$. Then the aggregated estimator

$$\bar{r}_i(\mathbf{x}_0) = E_{\mathbf{z}} r_i(\mathbf{x}_0, \mathbf{z})$$

is the average over $\mathbf{z}$ of $r_i(\mathbf{x}_0, \mathbf{z})$ in a neighborhood $N(\mathbf{x}_0)$ of $\mathbf{x}_0$.

Assume $\mathbf{x}_0$ is fixed and $R_i(\mathbf{x}_0)$ is the relevance value for dimension i at $\mathbf{x}_0$. Then the combined mean-squared error of these estimates for all q dimensions is

$$\sum_{i=1}^{q} E_{\mathbf{z}}[(R_i(\mathbf{x}_0) - r_i(\mathbf{x}_0, \mathbf{z}))^2] = \sum_{i=1}^{q} (R_i^2(\mathbf{x}_0) - 2R_i(\mathbf{x}_0) E_{\mathbf{z}}[r_i(\mathbf{x}_0, \mathbf{z})] + E_{\mathbf{z}}[r_i^2(\mathbf{x}_0, \mathbf{z})]). \tag{13}$$

Applying $E[X^2] \geq E^2[X]$ to the third term in (13) gives

$$\begin{aligned}\sum_{i=1}^{q} E_{\mathbf{z}}[(R_i(\mathbf{x}_0) - r_i(\mathbf{x}_0, \mathbf{z}))^2] &\geq \sum_{i=1}^{q} (R_i^2(\mathbf{x}_0) - 2R_i(\mathbf{x}_0)E_{\mathbf{z}}[r_i(\mathbf{x}_0, \mathbf{z})] + E_{\mathbf{z}}^2[r_i(\mathbf{x}_0, \mathbf{z})]) \\ &= \sum_{i=1}^{q} (R_i(\mathbf{x}_0) - E_{\mathbf{z}}[r_i(\mathbf{x}_0, \mathbf{z})])^2 \\ &= \sum_{i=1}^{q} (R_i(\mathbf{x}_0) - \bar{r}_i(\mathbf{x}_0))^2 \qquad (14)\end{aligned}$$

Integrating both sides of (14) over the joint distribution of $R_i(\mathbf{x}_0)$ and $\mathbf{x}_0$, we can conclude that the mean-squared error of $\bar{r}_i(\mathbf{x}_0)$ is lower than the mean-squared error of $r_i(\mathbf{x}_0, \mathbf{z})$ averaged over $\mathbf{z}$.

We note that $\bar{r}_i(\mathbf{x}_0)$ is a function of both $\mathbf{x}_0$ and the probability distribution P from which the training data are drawn. Of course, our estimate (5) is not $E_{\mathbf{z}} r_i(\mathbf{x}_0, \mathbf{z})$. Instead, it follows the distribution that allocates $1/K$ to each $\mathbf{z} \in N(\mathbf{x}_0)$. The gain in error reduction depends on how unequal the two sides of (14) are. This is in direct analogy to improvement in performance that can be achieved by bagging predictors [5].

6 Empirical Results

In the following we compare several classification methods using both simulated and real data. The simulated data experiments allow us to reliably predict the strengths and limitations of algorithms because the precise nature of the problem the algorithms are facing is known. We compare the following classification approaches:

- ADAMENN-adaptive metric nearest neighbor described in Figure 3 (one iteration), coupled with the exponential weighting scheme (7).
- i-ADAMENN-adaptive metric nearest neighbor with five iterations.
- Simple K-NN method using the Euclidean distance measure.
- C4.5 decision tree method [15].

- Machete [9]. It is a recursive partitioning procedure, in which the input variable used for splitting at each step is the one that maximizes the estimated local relevance (normalized) described in equation (15).

- Scythe [9]. It is a generalization of the machete algorithm, in which the input variables influence each split in proportion to their estimated local relevance, rather than the winner-take-all strategy of the machete.

- DANN-discriminant adaptive nearest neighbor classification [10]. It is an adaptive nearest neighbor classification method based on linear discriminant analysis. It computes a distance metric as a product of properly weighted within and between sum of squares matrices.

- i-DANN-discriminant adaptive nearest neighbor classification [10] with five iterations.

In all the experiments, the features are first normalized over the training data to have zero mean and unit variance, and the test data features are normalized using the corresponding training mean and variance. Procedural parameters for each method were determined empirically through cross-validation.

6.1 Experiments on Simulated Data

For all simulated data, 20 independent training samples (of size N) were generated. For each of these, an additional independent test sample consisting of 500 observations was generated. These test data were classified by each competing method using the respective training data set. Error rates computed over all 10,000 such classifications are reported in Table 1.

6.1.1 The Problems

1. This problem is taken from [9], and designed to be favorable to the adaptive methods (ADAMENN/DANN/scythe/machete/C4.5), and unfavorable to the regular K-NN procedure. There are $q = 10$ input features, $N = 200$ training data, and $J = 2$ classes. The data for the first class were generated from a standard normal distribution $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$. The

data for the second class were also generated from a normal distribution $\mathbf{x}_n \sim N(\mathbf{m}, \mathbf{C})$, with the coordinate mean values and covariance matrix given by

$$\{m_i = \sqrt{i}/2\}_{i=1}^{q}, \;\; \mathbf{C} = diag\{1/\sqrt{i}\}_{i=1}^{q}.$$

Although all input variables are relevant, the ones with higher coordinate number i are more so. Also, since only the diagonal elements of the covariance matrix are non zeros, much of the discriminating information is axis oriented. The first column of Table 1 shows the results for the eight methods under comparison, with standard deviations: 1.86, 1.36, 1.98, 1.78, 1.19, 1.62, 1.16 and 1.18 respectively. i-DANN had the lowest error rate, with DANN exhibiting similar performance. As expected, the K-NN procedure had the poorest performance for this problem.

2. This problem is adapted from [10], and consists of four dimensional spheres with 6 noise features. There are $q = 10$ input features, $N = 200$ training data, and $J = 2$ classes. The last 6 features are noise variables, with standard Gaussian distributions, independent of each other and the class membership. The data for both classes are generated from a standard normal distribution. The data for class one have the property that the radius, computed from the first four features, is greater than 1.85 while the data for class two do not have such restriction. Class one basically surrounds class two in the subspace spanned by the first four features. Results are shown in the second column of Table 1. The standard deviations are: 2.30, 2.83, 2.73, 1.56, 2.44, 2.59, 3.17, and 2.26 respectively. C4.5 is by far the best performer in this case. Machete gives the second best performance, with i-DANN and i-ADAMENN being close to it. K-NN performs very poorly on this problem.

3. This example is also taken from [9]. It is designed to be more favorable to the K-NN procedure, since all the input variables have the same global relevance. As before there are $q = 10$ input features and $J = 2$ classes, but $N = 500$ training data. The data for both classes are generated from a standard normal distribution $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$, and the classes are defined by

$$\sum_{i=1}^{10} x_i^2 \leq 9.8 \Rightarrow class1, \quad otherwise \Rightarrow class2.$$

The third column of Table 1 shows the results for this example, with the standard deviations:

2.09, 2.01, 2.35, 2.74, 2.71, 1.97, 2.36 and 1.73 respectively. The K-NN procedure exhibits a more competitive performance with the adaptive techniques, even though it still has the worst error rate. i-DANN shows the best performance for this problem. DANN gives a similar result, with C4.5 being the closest to it. Note that for this example, the performance of ADAMENN doesn't improve by performing five iterations.

4. This example is again taken from [9]. It is constructed so that all input variables have equal local relevance everywhere in the input space. However, there is a single direction in the space that contains all the discriminant information. There are $q = 10$ input features, $N = 200$ training data, and $J = 2$ classes. The data for both classes are generated from a standard normal distribution $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$, and the classes are defined by

$$\sum_{i=1}^{10} x_i \leq 0 \Rightarrow class1, \quad otherwise \Rightarrow class2.$$

Results are shown in the fourth column of Table 1. The standard deviations are: 2.32, 1.83, 1.62, 2.06, 1.92, 1.99, 2.20 and 1.86. i-DANN gives the best performance, with DANN being very close to it. K-NN performs well because all variables are equally locally relevant everywhere. ADAMENN and i-ADAMENN come close to it, showing that with our adaptive method we don't loose much when all variables are equally relevant. For this example, the performance of ADAMENN improves only slightly by performing 5 iterations. C4.5 is the worst performer in this case.

5. This problem is adapted from [10]. There are $q = 2$ input features, $N = 200$ training data, and $J = 2$ classes. Each class contains six spherical bivariate normal subclasses, having standard deviation 0.25. The means of the 12 subclasses are chosen at random without replacement from the integers $[1, 2, \ldots, 5] \times [1, 2, \ldots, 5]$. For each class, data are evenly drawn from each of the six normal subclasses. The fifth column of Table 1 shows the results for this problem, with standard deviations: 0.83, 0.78, 0.86, 6.01, 0.82, 0.76, 1.11 and 1.20, respectively. ADAMENN, i-ADAMENN and scythe show the best performance for this problem. K-NN and machete give similar results. C4.5 is again the worst performer.

6. This problem is taken from [10]. There are $q = 2$ input features, $N = 200$ training data, and $J = 4$ classes. Each class contains three spherical bivariate normal subclasses, having

standard deviation 0.25. As in the previous example, the means of the 12 subclasses are chosen at random without replacement from the integers $[1, 2, \ldots, 5] \times [1, 2, \ldots, 5]$. For each class, data are evenly drawn from each of the three normal subclasses. Results are shown in the sixth column of Table 1. The standard deviations are: 1.46, 1.37, 1.19, 8.57, 5.04, 5.36, 1.69 and 1.75, respectively. For this problem ADAMENN, i-ADAMENN and K-NN give the best performance. DANN shows a similar result. Again, the worst performer is C4.5.

7. This problem is also taken from [10]. There are $q = 10$ input features, $N = 200$ training data, and $J = 4$ classes. The data for this problem are generated as in the previous example, but augmented with eight predictors having independent standard Gaussian distributions. They serve as noise. The seventh column of Table 1 shows the results, with standard deviations: 3.93, 3.66, 8.02, 16.84, 5.31, 4.96, 8.31 and 7.74, respectively. ADAMENN is by far the best performer in this case, with only i-ADAMENN coming close to it. K-NN gives the worst performance in this case.

6.1.2 Results

Table 1 shows that, for each method, there is at least one example for which it has the best performance, or close to the best. Therefore, it seems natural to ask the question of robustness. That is, how well a particular method m performs on average in situations that are most favorable to other procedures. Following Friedman [9], we capture robustness by computing the ratio b_m of its error rate e_m and the smallest error rate over all methods being compared in a particular example:

$$b_m = e_m / \min_{1 \leq k \leq 8} e_k.$$

Thus, the best method m^* for that example has $b_{m^*} = 1$, and all other methods have larger values $b_m \geq 1$, for $m \neq m^*$. The larger the value of b_m, the worse the performance of the mth method is in relation to the best one for that example, among the methods being compared. The distribution of the b_m values for each method m over all the examples, therefore, seems to be a good indicator concerning its robustness. For example, if a particular method has an error rate close to the best in every problem, its b_m values should be densely

distributed around the value 1. Any method whose b value distribution deviates from this ideal distribution reflect its lack of robustness.

Table 1: Average classification error rates for simulated data.

	Ex1	Ex2	Ex3	Ex4	Ex5	Ex6	Ex7
ADAMENN	9.9	23.9	33.7	20.8	2.4	3.3	12.8
i-ADAMENN	8.3	23.1	33.7	20.3	2.4	3.3	14.2
K-NN	14.7	33.9	36.1	18.1	2.5	3.3	50.7
C4.5	10.3	14.6	30.6	30.1	25.5	31.7	38.2
Machete	7.1	21.7	33.0	25.7	2.6	14.5	20.1
Scythe	7.9	25.6	32.7	22.2	2.4	21.3	38.1
DANN	6.2	25.3	26.7	13.3	2.8	4.2	37.6
i-DANN	5.3	22.8	25.4	13.2	3.1	6.1	26.7

Figure 4 plots the distribution of b_m for each method over the seven simulated data sets. The dark area represents the lower and upper quartiles of the distribution that are separated by the median. The outer vertical lines show the entire range of values for the distribution. It is clear that the most robust method over the simulated data is i-ADAMENN. In 4/7 of the data its error rate was no worse than 33% higher than the best error rate. In the worst case it was 87%. In contrast, C4.5 has the worst distribution, where the corresponding numbers are 128% and 962%.

DANN and i-DANN performed well in examples 1, 3 and 4, where the data were generated from Gaussian distributions. This might be attributed to the fact that the distance metric computed by DANN approximates the weighted *Chi-squared* distance (1), only when class densities are Gaussian and have the same covariance matrix. This may also explain DANN's performance degradation in those examples where data do not follow Gaussian distributions or are corrupted by noise.

6.2 Experiments on Real Data

While simulated data are informative for comparison studies, it is highly likely that artificially constructed examples will not correspond to situations that are likely to occur in practise. Thus, in this section we examine the performance of the competing classification methods using real world data. One of the advantages of real data is that they are generated without any knowledge of the classification procedures that it will be used to test.

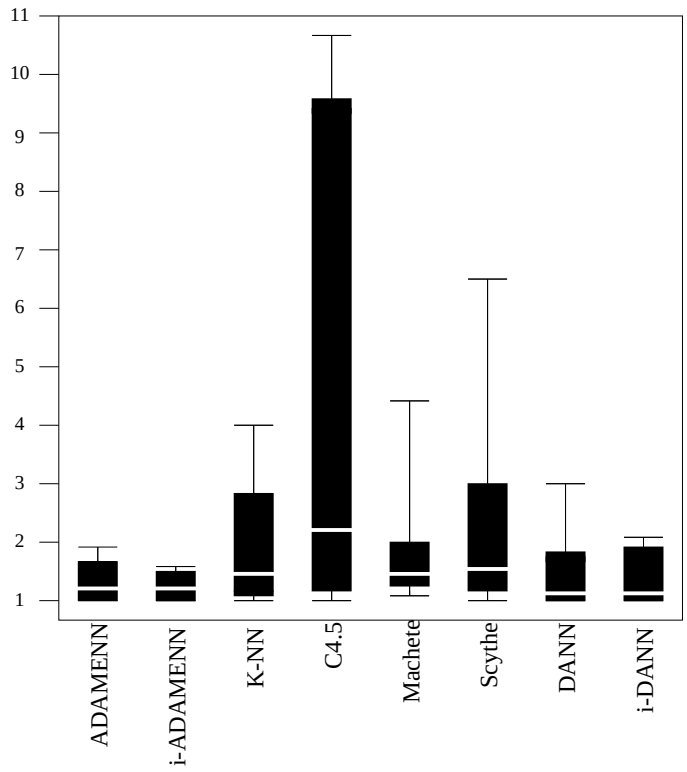


Figure 4: Performance distributions for simulated data.

In our experiments we used seven different real data sets. The Iris, Sonar, Vowel, Glass, Segmentation and Letter data are taken from UCI Machine Learning Repository at http://www.cs.uci.edu/~mlearn/MLRepository.html. The Image data are obtained from MIT Media Lab at ftp://whitechapel.media.mit.edu/pub/VisTex. For the Iris, Sonar, and Glass data we perform leave-one-out cross-validation to measure performance. For the Vowel and Image data we randomly divide the data into a training set of 200 data points and a test set consisting of the remaining data points (320 for the Vowel data and 440 for the

Image data). We repeat this process 10 times independently, and report the average cross-validation error rates for these two data sets. On the Segmentation and Letter data we perform two 10-fold cross-validation. We randomly divide the data into 10 sets of equal size and use one of them in turn as a test set and the remaining nine as a training set. We repeat this process two times independently and report the two 10-fold cross-validation error rates for these two data sets. Table 2 shows the cross-validated error rates for the eight methods under consideration on the seven real data.

6.2.1 The Problems

1. Iris data. This data set consists of $q = 4$ measurements made on each of $N = 100$ iris plants of $J = 2$ species. The two species are iris versicolor and iris virginica. The problem is to classify each test point to its correct species based on the four measurements. The results on this data set are shown in the first column of Table 2.

2. Sonar data. This data set consists of $q = 60$ frequency measurements made on each of $N = 208$ data of $J = 2$ classes ("mines" and "rocks"). The problem is to classify each test point in the 60-dimensional feature space to its correct class. The results on this data set are shown in the second column of Table 2.

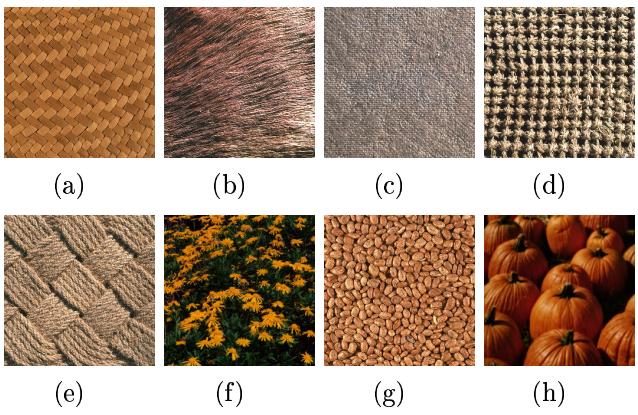


Figure 5: Sample images taken from the Image database.

3. Vowel data. This example has $q = 10$ measurements and $J = 11$ classes. There are

$N = 528$ samples in this example. Results are shown in the third column of Table 2, having standard deviations: 2.82, 3.06, 2.56, 3.68, 2.82, 2.30, 3.06 and 2.93 respectively.

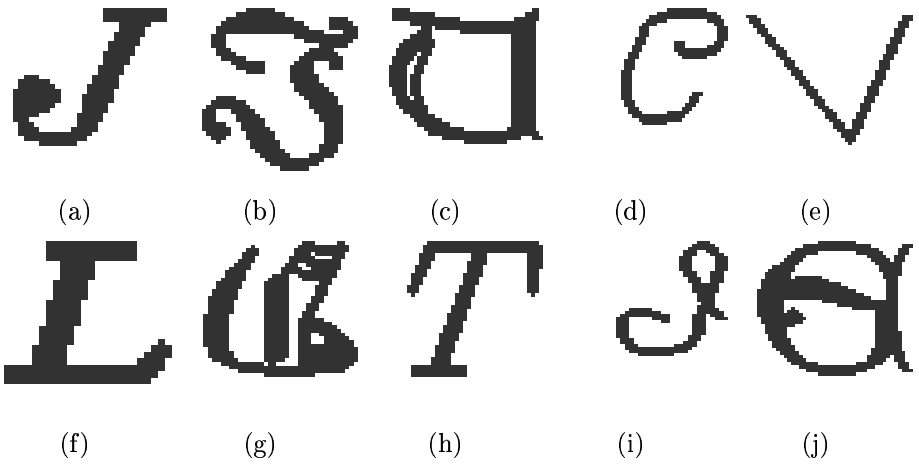
4. Glass data. This data set consists of $q = 9$ chemical attributes measured for each of $N = 214$ data of $J = 6$ classes. The problem is to classify each test point in the 9-dimensional space to its correct class. Results are shown in the fourth column of Table 2.

5. Image data. This data set consists of 40 texture images that are manually classified into 15 classes. Each of these images is then cut into 16 non-overlapping images of 128×128, giving rise to a total of 640 images in the database. Sample images are shown in Figure 5. The number of images in each class varies from 16 to 80. The images in this database are represented by $q = 16$ dimensional feature vectors (8 Gabor filters: 2 scales and 4 orientations). The mean and the standard deviation of the magnitude of the transform coefficients are used as feature components, after being normalized by the standard deviations of the respective features, over the entire set of images in the database. Results are shown in the fifth column of Table 2. The standard deviations are: 0.78, 1.31, 1.21, 3.0, 1.69, 1.57, 2.23 and 3.35 respectively.

6. Segmentation data. This data set consists of images that were drawn randomly from a database of 7 outdoor images. The images were hand segmented by the creators of the database to classify each pixel. Each image is a region. There are $J = 7$ classes, each of which has 330 instances. Thus, there are $N = 2,310$ images in the database. These images are represented by $q = 19$ real valued attributes. Results are shown in the sixth column of Table 2. The standard deviations are: 0.91, 1.07, 1.27, 1.15, 1.17, 1.28, 0.91 and 1.10, respectively.

7. Letter Image Recognition data. This data set consists of $q = 16$ numerical attributes and $J = 26$ classes. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. Sample images are shown in Figure 6. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and

edge counts) which were then scaled to fit into a range of integer values from 0 through 15. Results are shown in the seventh column of Table 2. The standard deviations are: 0.78, 0.71, 0.91, 0.88, 0.87, 0.87, 1.14 and 0.86, respectively.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 6: Sample letter images.

6.2.2 Results

Table 2 shows that ADAMENN achieved the best performance in 4/7 of the real data sets, followed closely by i-ADAMENN. For the remaining three data sets, ADAMENN has the second best performance. As shown in Figure 8, the spread of the error distribution for ADAMENN is narrow and close to 1. The spread for i-ADAMENN has a similar behavior. The results clearly demonstrate that they obtained the most robust performance over these data sets. Similar characteristics were also observed for the two methods over the simulated data sets. This could be attributed to the fact that local feature relevance estimate in ADAMENN is conducted over regions in the feature space instead of using individual points, as is done in machete and scythe [9]. This observation is corroborated by our discussion in section 5.

Table 2: Average classification error rates for real data.

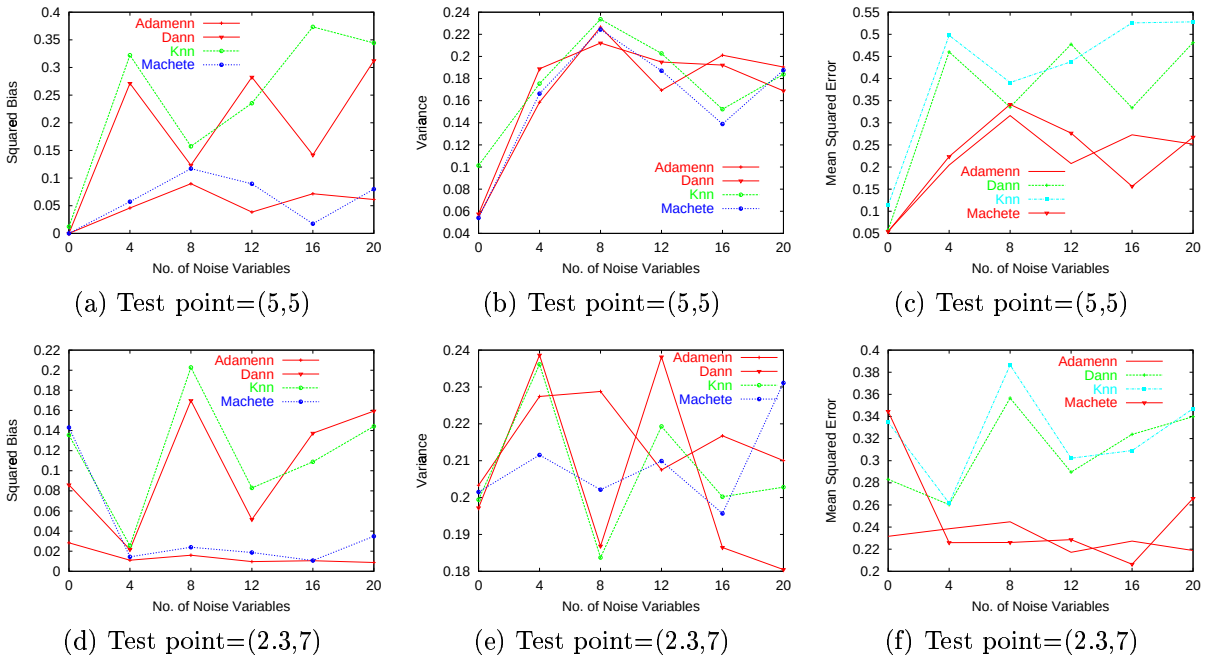
	Iris	Sonar	Vowel	Glass	Image	Seg	Letter
ADAMENN	3.0	9.1	10.7	24.8	5.2	2.4	5.1
i-ADAMENN	5.0	9.6	10.9	24.8	5.2	2.5	5.3
K-NN	6.0	12.5	11.8	28.0	6.1	3.6	6.9
C4.5	8.0	23.1	36.7	31.8	21.6	3.7	16.4
Machete	5.0	21.2	20.2	28.0	12.3	3.2	9.1
Scythe	4.0	16.3	15.5	27.1	5.0	3.3	7.2
DANN	6.0	7.7	12.5	27.1	12.9	2.5	3.1
i-DANN	6.0	9.1	21.8	26.6	18.1	3.7	6.1

6.3 Bias and Variance Calculations

For a two-class problem with $\Pr(Y = 1|\mathbf{x}) = p(\mathbf{x})$, we compute a nearest neighborhood at a query $\mathbf{x}_0$ and find the nearest neighbor $\mathbf{X}$ having class label $Y(\mathbf{X})$ (random variable). The estimate of $p(\mathbf{x}_0)$ is $Y(\mathbf{X})$. The bias and variance of $Y(\mathbf{X})$ are: $Bias = Ep(\mathbf{X}) - p(\mathbf{x}_0)$ and $Var = Ep(\mathbf{X})(1 - Ep(\mathbf{X}))$, where the expectation is computed over the distribution of the nearest neighbor $\mathbf{X}$ [10].

We performed simulations to estimate the bias and variance of ADAMENN, KNN, DANN and Machete on the following two-class problem. There are $q = 2$ input features and 180 training data. Each class contains three spherical bivariate normal subclasses, having standard deviation 0.75. The means of the 6 subclasses are chosen at random without replacement from the integers $[1, 2, \ldots, 8] \times [1, 2, \ldots, 8]$. For each class, data are evenly drawn from each of the normal subclasses. Fig. 7 shows the bias and variance estimates from each method at locations $(5, 5, 0, \cdots, 0)$ and $(2.3, 7, 0, \cdots, 0)$, as a function of the number of noise variables over five independently generated training sets. Here the noise variables have independent standard Gaussian distributions. The true probability of class 1 for $(5, 5, 0, \cdots, 0)$ and $(2.3, 7, 0, \cdots, 0)$ are 0.943 and 0.747, respectively.

The four methods have similar variance, since they all use three neighbors for classification. While the bias of KNN and DANN increases with increasing number of noise variables, ADAMENN retains a low bias by averaging out noise.



(a) Test point=(5,5) (b) Test point=(5,5) (c) Test point=(5,5)

(d) Test point=(2.3,7) (e) Test point=(2.3,7) (f) Test point=(2.3,7)

Figure 7: Bias and variance estimates.

7 Related Work

Friedman [9] describes an approach for learning local feature relevance that combines some of the best features of K-NN learning and recursive partitioning. This approach recursively homes in on a query along the most (locally) relevant dimension, where local relevance is computed from a reduction in prediction error given the query's value along that dimension. This method performs well on a number of classification tasks. In our notations, the reduction in prediction error can be described by

$$I_i^2(\mathbf{z}) = \sum_{j=1}^{J} (\overline{\Pr}(j) - \overline{\Pr}(j|x_i = z_i)])^2, \tag{15}$$

where $\overline{\Pr}(j)$ represents the expected value of $\Pr(j|\mathbf{x})$. This measure reflects the influence of the ith input variable on the variation of $\Pr(j|\mathbf{x})$ at the particular point $x_i = z$. In this

case, the most informative input variable is the one that gives the largest deviation from the average value of $\Pr(j|\mathbf{x})$.

The main difference, however, between our relevance measure (5) and Friedman's (15) is the first term in the squared difference. While the class conditional probability is used in our relevance measure, its expectation is used in Friedman's. As a result, a feature dimension is more relevant than others when it minimizes (4) in case of our relevance measure, whereas when it maximizes (15) in case of Friedman's. Furthermore, we take into account not only the test point $\mathbf{x}_0$ itself, but also its K nearest neighbors, resulting in a relevance measure (5) that is in general more robust, as shown in our experiments.

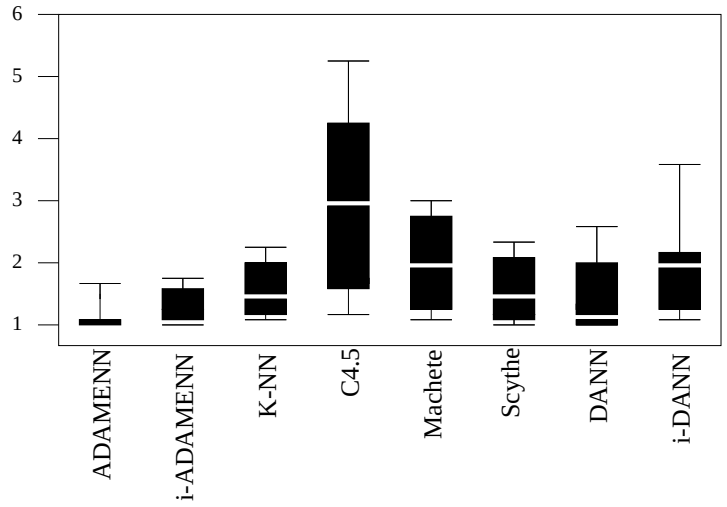


Figure 8: Performance distributions for real data.

In [10], Hastie and Tibshirani propose an adaptive nearest neighbor classification method based on linear discriminant analysis. The method computes a distance metric as a product of properly weighted within and between sum of squares matrices. They show that the resulting metric approximates the weighted *Chi-squared* distance (1) by a Taylor series expansion, given that class densities are Gaussian and have the same covariance matrix. In contrast, our method does not make such assumptions, which are unlikely in real world applications. Instead, our method attempts to approximate the weighted *Chi-Squared* distance (1) directly. While sound in theory, DANN may be limited in practice. The main concern is that in high dimensions we may never have sufficient data to fill in $q \times q$ matrices. It is

interesting to note that our work can potentially serve as a general framework upon which to develop a unified adaptive metric theory that encompasses both Friedman's work and that of Hastie and Tibshirani.

8 Summary and Conclusions

This paper presents an adaptive nearest neighbor method for effective pattern classification. This method estimates a flexible metric for producing neighborhoods that are elongated along less relevant feature dimensions and constricted along most influential ones. As a result, the class conditional probabilities tend to be more homogeneous in the modified neighborhoods. The experimental results using both simulated and real data show clearly that the ADAMENN algorithm can potentially improve the performance of K-NN and recursive partitioning methods in some classification problems, especially when the relative influence of input features changes with the location of the query to be classified in the input feature space. The results are also in favor of ADAMENN over other adaptive methods such as machete and DANN.

A potential extension to the technique described in this paper is to consider additional derived variables (features) for local relevance estimate, thereby contributing to the distance calculation. When the derived features are more informative, huge gains may be expected. On the other hand, if they are not informative enough, they may cause classification performance to degrade since they add to the dimensionality count. The challenge is to be able to have a mechanism that computes such informative derived features efficiently.

Acknowledgments

This research has been supported in part by the National Science Foundation under grant number IIS-9907477, and by the US Department of Defense.

References

[1] D. Aha, "Lazy Learning," *Artificial Intelligence Review*. 11:1-5. 1997.

[2] C. Atkeson, A.W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*. 11:11-73. 1997.

[3] R.E. Bellman, *Adaptive Control Processes*. Princeton Univ. Press, 1961.

[4] L. Bottou and V. Vapnik, Local learning algorithms. *Neural Computation*, **4**(6), 888-900, 1992.

[5] L. Breiman, "Bagging Predictors," *Machine Learning* **24**:123-140, 1996.

[6] W.S. Cleveland and S.J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *J. Amer. Statist. Assoc.* **83**, 596-610, 1988

[7] T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. on Information Theory*, pp. 21-27, 1967.

[8] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, Inc., 1973.

[9] J.H. Friedman "Flexible Metric Nearest Neighbor Classification," Tech. Report, Dept. of Statistics, Stanford University, 1994.

[10] T. Hastie and R. Tibshirani, "Discriminant Adaptive Nearest Neighbor Classification", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 6, pp. 607-615, 1996.

[11] T.K. Ho, "Nearest Neighbors in Random Subspaces," *Lecture Notes in Computer Science: Advances in Pattern Recognition*, pp. 640-648, 1998.

[12] D.G. Lowe, "Similarity Metric Learning for a Variable-Kernel Classifier," *Neural Computation* **7**(1):72-85, 1995.

[13] G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 1992.

[14] J.P. Myles and D.J. Hand, "The Multi-Class Metric Problem in Nearest Neighbor Discrimination Rules," *Pattern Recognition*, Vol. 23, pp. 1291-1297, 1990.

[15] J.R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan-Kaufmann Publishers, Inc., 1993.

[16] S. Salzberg, A Nearest Hyperrectangle Learning Method. *Machine Learning* **6**:251-276, 1991.

[17] C.J. Stone, Nonparametric regression and its applications (with discussion). *Ann. Statist.* **5**, 595, 1977.