LDA/SVM Driven Nearest Neighbor Classification

Jing Peng
Electrical Engr. & Computer Sci. Dept.
Tulane University
New Orleans, LA 70118
jp@eecs.tulane.edu

Douglas R. Heisterkamp & H.K. Dai
Computer Science Dept.
Oklahoma State University
Stillwater, OK 74078
{doug,dai}@cs.okstate.edu

Abstract

Nearest neighbor classification relies on the assumption that class conditional probabilities are locally constant. This assumption becomes false in high dimensions with finite samples due to the curse-of-dimensionality. The nearest neighbor rule introduces severe bias under these conditions. We propose a locally adaptive neighborhood morphing classification method to try to minimize bias. We use local support vector machine learning to estimate an effective metric for producing neighborhoods that are elongated along less discriminant feature dimensions and constricted along most discriminant ones. As a result, the class conditional probabilities can be expected to be approximately constant in the modified neighborhoods, whereby better classification performance can be achieved. The efficacy of our method is validated and compared against other competing techniques using a number of data sets.

Keywords: Classification, Nearest Neighbor, LDA, SVM

1 Introduction

In classification, a feature vector $\mathbf{x} = (x_1, \cdots, x_n)^t \in \Re^n$, representing an object, is assumed to be in one of J classes $\{i\}_{i=1}^J$, and the objective is to build classifier machines that assign $\mathbf{x}$ to the correct class from a given set of l training samples.

A simple and attractive approach to this problem is the K nearest neighbor (NN) classification method [4, 6, 10, 11, 13, 16, 18]. 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. Furthermore, empirical evaluation to date shows that the KNN rule is a rather robust method.

In addition, it has been shown [7, 9] that the 1NN rule has asymptotic error rate that is at most twice the Bayes error rate, independent of the distance metric used.

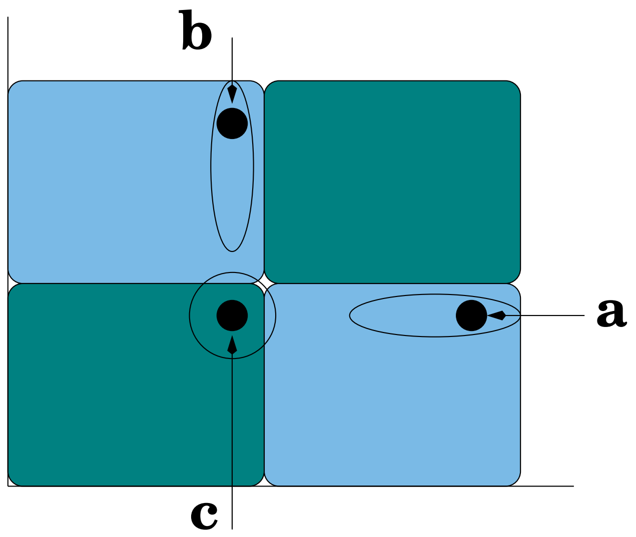

Figure 1: Feature relevance varies with query locations.

The nearest neighbor rule becomes less appealing in finite training samples, however. This is due to the curse-of-dimensionality [3]. Severe bias can be introduced in the NN 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. 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, the vertical coordinate is more relevant, because a slight move along the that axis may change the class label, while for query b, the horizontal coordinate is more relevant. For query c, however, both coordinates 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.

In this paper we propose an adaptive neighborhood morphing classification method to try to minimize bias in high dimensions. We estimate an effective local metric for computing neighborhoods based on local Support Vector Machines (SVMs), which have been successfully used as a classification tool in a number of areas, ranging from object recognition to classification of cancer morphologies. The resulting neighborhoods are highly adaptive to query locations. Moreover, the neighborhoods are elongated along less relevant (discriminant) 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.

The rest of the paper is organized as follows. Section 2 describes related work addressing issues of feature relevance computation and nearest neighbor classification. Section 3 presents our approach to measuring local feature relevance based on linear discriminant analysis and support vector machines. Section 4 describes our neighborhood morphing procedure based on local feature relevance. After that, we present in Section 5 experimental results demonstrating the efficacy of our technique using a number of real-world data. Finally, Section 6 concludes this paper by pointing out possible extensions to the current work and future research directions.

2 Related Work

Friedman [10] describes an approach to learning local feature relevance that 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 local relevance of ith coordinate/feature axis can be described by

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

where $\overline{\Pr}(j)$ represents the expected value of $\Pr(j|\mathbf{x})$, and $\overline{\Pr}(j|x_i = z_i)$ the conditional expectation of $\Pr(j|\mathbf{x})$, given that x_i assumes value z_i. This measure reflects the influence of the ith input variable on the variation of $\Pr(j|\mathbf{x})$ at the particular point $x = z$. In this case, the most informative dimension is the one that deviates the most from $\overline{\Pr}(j)$.

In contrast, we measure feature relevance (section 3.2) based on discriminant analysis. We say a feature dimension is more relevant if it provides more class discriminanting information, whereby a large weight can be associated with that dimension. As a result, our relevance measure is more discriminant than Friedman's.

Hastie and Tibshirani [11] propose an adaptive nearest neighbor classification method based on linear discriminant analysis (LDA). 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 *chi-squared* distance by a Taylor series expansion, given that class densities are Gaussian and have the same covariance matrix. While sound in theory, the method has limitations. The main concern is that in high dimensions we may never have sufficient data to locally fill in $n \times n$ within and between sum-of-squares matrices.

Amari and Wu [1] describe a method for improving SVM performance by increasing spatial resolution around the decision boundary surface based on the Riemannian geometry. The method first trains a SVM with an initial kernel that is then modified from the resulting set of support vectors and a qausiconformal mapping. A new SVM is built using the new kernel. Viewed under the same light, our technique can be regarded as a way to increase spatial resolution around the separating hyperplane in a local fashion. However, our technique varies spatial resolution judiciously in that it increases spatial resolution along discriminanting directions, while decreasing spatial resolution along less discriminant ones.

Weston et al. [20] propose a technique for feature selection for SVMs to improve generalization performance. In their technique, a feature is either completely relevant or completely irrelevant. Clearly, feature importance as such is non-local, and therefore, insensitive to query locations. In addition, these global relevance techniques usually do not work well on tasks that exhibit local feature differential relevance, as evidenced by the example shown in Figure 1.

3 Feature Relevance

Our technique is motivated as follows. In linear discriminant analysis (for $J = 2$), data are projected onto a single dimension where class label assignment is made for a given input query. From a set of training data

$$\{\mathbf{x}_i, y_i\}_1^l$$

where $y_i \in \{0, 1\}$, this dimension is computed according to

$$\mathbf{w} = \mathbf{W}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \tag{2}$$

where

$$\mathbf{W} = \sum_{j=0}^{1} \sum_{y_i=j} p_i(\mathbf{x}_i - \bar{\mathbf{x}}_j)(\mathbf{x}_i - \bar{\mathbf{x}}_j)^t \tag{3}$$

denotes the within sum-of-squares matrix, $\bar{\mathbf{x}}_j$ the class means, and p_i the relative occurrence of $\mathbf{x}_i$ in class j. The vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)^t$ represents the same direction as the discriminant in the Bayes classifier along which the data has the maximum separation. Furthermore, any direction, $\mathbf{\Theta}$, whose dot product with $\mathbf{w}$ is large, also carries discriminant information. The larger $|\mathbf{w} \cdot \mathbf{\Theta}|$ is, the more discriminant information that $\mathbf{\Theta}$ captures. State it differently, if we transform $\mathbf{\Theta}$ via

$$\tilde{\mathbf{\Theta}} = \mathbf{W}^{-1/2}\mathbf{\Theta}, \tag{4}$$

then in the transformed space, any direction $\tilde{\mathbf{\Theta}}$ close to $\mathbf{W}^{-1/2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$ carries discriminant information. More formally, let

$$J(\mathbf{w}) = \frac{\mathbf{w}^t\mathbf{B}\mathbf{w}}{\mathbf{w}^t\mathbf{W}\mathbf{w}}, \tag{5}$$

be the LDA criterion function maximized by $\mathbf{w}$ (2), where $\mathbf{B}$ is the between sum-of-squares matrix and computed according to

$$\mathbf{B} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^t. \tag{6}$$

If we let

$$\mathbf{B}^* = \mathbf{W}^{-1/2}\mathbf{B}\mathbf{W}^{-1/2} \tag{7}$$

be the between sum-of-squares matrix in the transformed space, then the criterion function (5) in the transformed space becomes

$$\begin{aligned} J^*(\tilde{\mathbf{\Theta}}) &= \frac{\tilde{\mathbf{\Theta}}^t \mathbf{B}^* \tilde{\mathbf{\Theta}}}{\tilde{\mathbf{\Theta}}^t \tilde{\mathbf{\Theta}}} \\ &= \frac{(\tilde{\mathbf{w}}^t \tilde{\mathbf{\Theta}})^2}{\tilde{\mathbf{\Theta}}^t \tilde{\mathbf{\Theta}}}, \end{aligned} \tag{8}$$

where

$$\tilde{\mathbf{w}} = \mathbf{W}^{-1/2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$$

and $\tilde{\mathbf{\Theta}}$ by (4). Therefore, any direction $\tilde{\mathbf{\Theta}}$ that is close to $\mathbf{W}^{-1/2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$ in the transformed space computes higher values in J^*, thereby capturing discriminant information.

In particular, when $\mathbf{\Theta}$ is restricted to the feature axes, i.e.,

$$\mathbf{\Theta} \in \{\mathbf{e}_1, \cdots, \mathbf{e}_n\}, \tag{9}$$

where $\mathbf{e}_i$ is a unit vector along the ith feature, the value of $|\mathbf{w} \cdot \Theta|$, which is the magnitude of the projection of $\mathbf{w}$ along Θ, measures the degree of relevance of feature dimension $\mathbf{\Theta}$ in providing class discriminant information. It thus seems natural to associate, when $\Theta = \mathbf{e}_i$ (hence $|\mathbf{w} \cdot \Theta| = w_i$),

$$r_i = \frac{|w_i|}{\sum_{j=1}^n |w_j|} \tag{10}$$

as a weight, with each dimension $\mathbf{\Theta}$ in a weighted nearest neighbor rule

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n r_i (x_i - y_i)^2}. \tag{11}$$

Now imagine for each input query we compute $\mathbf{w}$ locally, from which to induce a new neighborhood for the final classification of the query. In this case, large $|\mathbf{w} \cdot \mathbf{\Theta}|$ forces the shape of neighborhood to constrict along $\mathbf{\Theta}$, while small $|\mathbf{w} \cdot \mathbf{\Theta}|$ enlongates the neighborhood along the $\mathbf{\Theta}$ direction. Figure 1 illustrates a case in point, where for query a the discriminant direction is parallel to the vertical axis, and as such, the shape of the neighborhood is squashed along that direction and enlongated along the horizontal axis.

We use two-dimensional Gaussian data with two classes and substantial correlation, shown in Figure 2, to illustrate neighborhood computation based on LDA. The number of

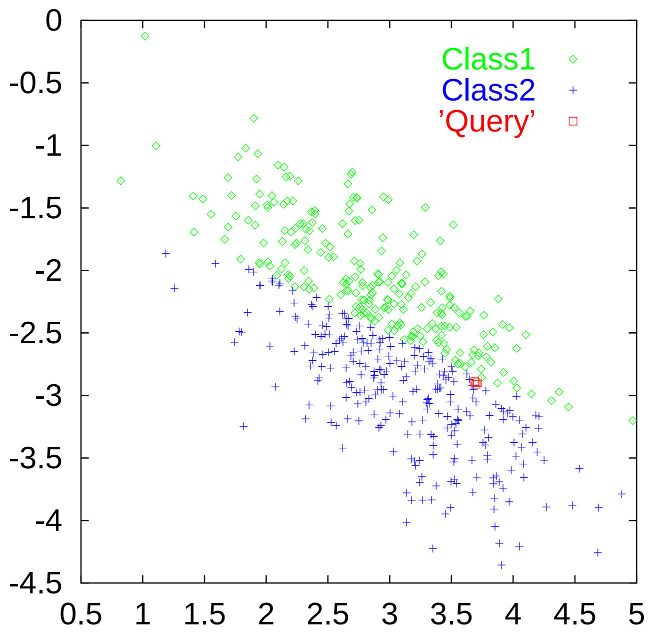

Figure 2: Two-dimensional Gaussian data with two classes and substantial correlation. The (red) square indicates the query.

data points for both classes is roughly the same (about 250). The (red) square, located at $(3.7, -2.9)$, represents the query. Figure 3(a) shows the 100 nearest neighbors (red squares) of the query found by the unweighted KNN method (simple Euclidean distance). The resulting shape of the neighborhood is circular, as expected. In contrast, Figure 3(b) shows the 100 nearest neighbors of the query, computed by the technique described above. That is, the nearest neighbors shown in Figure 3(a) are used to compute (2) and, hence, (10) and (11), with estimated new (normalized) weights: $r_1 = 0.3$ and $r_2 = 0.7$. As a result, the new (elliptical) neighborhood is elongated along the horizontal axis (the less important one) and constricted along the vertical axis (the more important one). The effect is that there is a sharp increase in the retrieved nearest neighbors that are in the same class as the query.

This example demonstrates that even a simple problem in which a linear boundary roughly separates two classes can benefit from the feature relevance learning technique just described, especially when the query approaches the class boundary. It is important to note that for a given distance metric the shape of a neighborhood is fixed, independent of query locations. Furthermore, any distance calculation with equal contribution from each feature variable will always produce spherical neighborhoods. Only by capturing the relevant contribution of the feature variables can a desired neighborhood be realized that is highly customized to query locations.

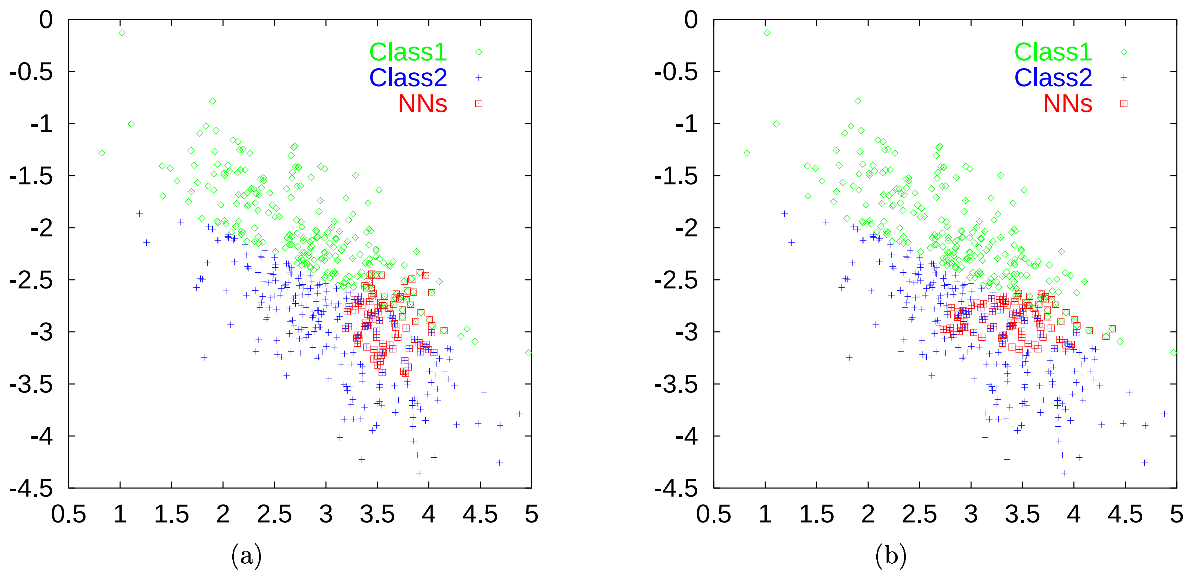

Figure 3: Effect of feature relevance learning on neighborhood shapes. (a) Circular neighborhood (Euclidean distance). (b) Elliptical neighborhood, where features are weighted from LDA.

While **w** points to a direction along which projected data can be well separated, the corresponding hyperplane may be far from optimal with respect to margin maximization. In general, such a hyperplane does not yield the maximum margin of separation between the data, which has direct bearing on its generalization performance. In terms of weighted nearest neighbor computation discussed above, this implies that the class (conditional) probability tends to vary in the neighborhood induced by **w**. Furthermore, the assumption on equal covariance structures for all classes is often invalid in practice. Computationally, if the dimension of the feature space is large, there will be insufficient data to locally estimate the $\Omega(n^2)$ elements of the within sum-of-squares matrices, thereby making them highly biased. Moreover, in high dimensions the within sum-of-squares matrices tend to be ill-conditioned. In such situations, one must decide at what threshold to zero small singular values in order to obtain better solutions. In addition, very often features tend to be independent locally. Without local clustering weighting based on local LDA will be less effective, as we shall see later. This motivates us to consider the SVM approach to feature relevance estimation.

3.1 Support Vector Machines

Let

$$R(\mathbf{w}) = \int \frac{1}{2}|y - f(\mathbf{x}, \mathbf{w})| dP(\mathbf{x}, y) \tag{12}$$

be the expected generalization error (risk) for a learning machine $f(\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is an adjustable parameter that determines f, and $P(\mathbf{x}, y)$ the (unknown) probability distribution [19]. The empirical risk is defined as

$$R_{emp}(\mathbf{w}) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(\mathbf{x}_i, \mathbf{w})| \tag{13}$$

for a set of training data

$$\{\mathbf{x}_i, y_i\}_{i=1}^{l}. \tag{14}$$

In the SVM framework, unlike typical classification methods that simply minimize $R_{emp}(\mathbf{w})$, SVMs minimize the following upper bound of the expected generalization error

$$R(\mathbf{w}) \leq R_{emp}(\mathbf{w}) + C(h), \tag{15}$$

where C represents the "VC confidence," and h the VC dimension. This can be accomplished by maximizing the margin between the separating plane and the data, which can be viewed as realizing the Structure Risk Minimization principle [19].

Now we examine general SVMs having basis functions $\Phi : \Re^n \rightarrow \Re^N$, where $n \leq N$. SVMs search for a linear function

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b \tag{16}$$

in space $\Re^N$. An input query $\mathbf{x}$ is classified according to the algebraic sign of $f(\mathbf{x})$. The SVM solution produces a hyerplane having the maximum margin, where the margin is defined as

$2/\|\mathbf{w}\|$. It is shown [19] that this hyperplane is optimum with respect to the maximum margin. The hyperplane, determined by its normal vector $\|\mathbf{w}\|$, can be explicitly written as

$$\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \Phi(\mathbf{x}_i), \tag{17}$$

where α_i's are Langrange coefficients that maximize

$$\mathbf{L}_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{18}$$

and SV is the set of support vectors determined by the SVM.

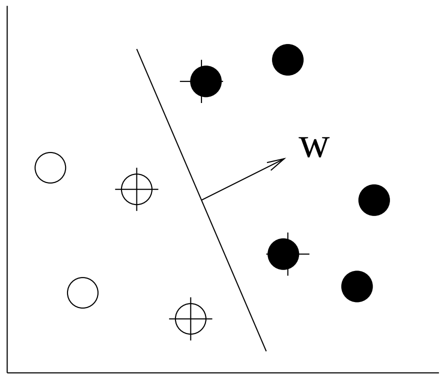

Figure 4: Separating hyperplane and its norm.

In a simple case in which Φ is the identity function on $\Re^n$

$$\Phi(\mathbf{x}) = \mathbf{x}, \tag{19}$$

we have

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{20}$$

and

$$\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i. \tag{21}$$

Here the normal $\mathbf{w}$ is perpendicular to the separating hyperplane. Similar to linear discriminant analysis, the normal $\mathbf{w}$ points to the direction that is more discriminant and yields the maximum margin of separation between the data. This situation is illustrated in Figure 4.

When dealing with non-separable data, the algorithm using the identity mapping (19) cannot find feasible solutions. To construct the optimal margin hyperplane when the data are non-separable, positive slack variables $\xi_i \geq 0$ are introduced and the primal problem becomes minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + c(\sum_i \xi_i) \tag{22}$$

subject to

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 - \xi_i \tag{23}$$

where c is a procedural parameter to be selected by the user [5]. It can be shown that the dual problem (18) remains the same but now subject to

$$0 \leq \alpha_i \leq c. \tag{24}$$

The solution is again given by (21). Solving separable data in the feature space is called hard margin SVMs, while solving non-separable data in the feature space is called soft margin SVMs.

Let us revisit the normal computed by LDA (2). This normal is optimal (the same Bayes discriminant) under the assumption that the two classes follow the same distribution. Optimality breaks down, however, when the assumption is violated, which is often the case in practice. In contrast, SVMs compute the optimal (maximum margin) hyperplane (17) without such an assumption. This spells out the difference between the directions pointed to by the two normals, which has important implications on generalization performance. Figure 5 illustrates a case in point. Here we have two-dimensional data with two classes that follow different Gaussian distributions. Each class has 100 data points. The two lines represent the hyperplanes computed by LDA and a soft margin SVM, respectively. In this simple example, the SVM hyperplane has a better separation of the data than the LDA hyperplane. In fact, the SVM linear classifier has an error rate of 3.3% while the LDA classifier has an error rate of 3.7% over 2000 independently generated test points.

Finally, we note that real data are often highly non-linear. In such situations, linear machines cannot be expected to work well. As such, $\mathbf{w}$ is unlikely to provide any useful

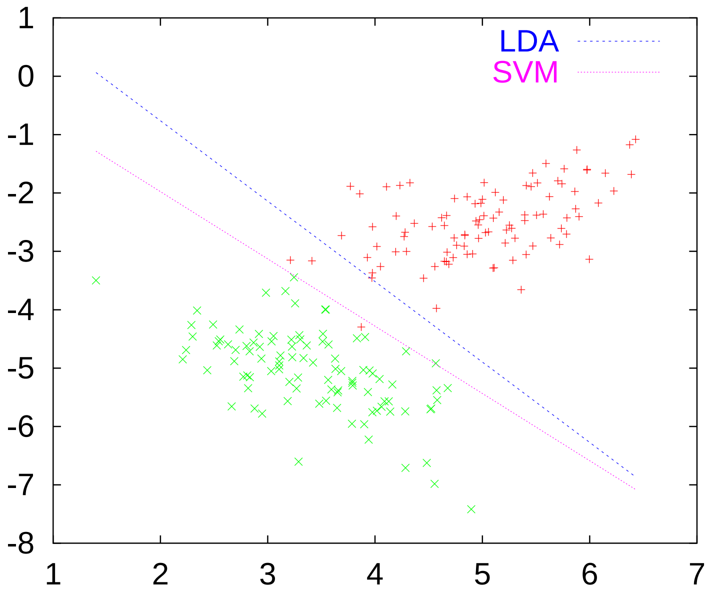

Figure 5: Two-dimensional data with two classes having different Gaussian distributions. Two lines indicate the hyperplanes computed by LDA and SVMs, respectively.

discriminant information. On the other hand, piecewise local hyperplanes can approximate any decision boundaries, thereby enabling **w** to capture local discriminant information.

3.2 Discriminant Feature Relevance

Based on the above discussion, we now propose a measure of feature relevance for an input query $\mathbf{x}_0$ as

$$\Upsilon_i(\mathbf{x}_0) = |w_i| \tag{25}$$

where w_i denotes the ith component of $\mathbf{w}$ in (21) computed locally at $\mathbf{x}_0$. One attractive property of (25) is that $\mathbf{w}$ enables Υ_i's to capture relevance information that may not otherwise be attainable should relevance estimates been conducted along each individual dimension one at a time, as in [8, 10].

The relative relevance, as a weighting scheme, can then be given by

$$r_i(\mathbf{x}_0) = (\Upsilon_i(\mathbf{x}_0))^t / \sum_{j=1}^{n} (\Upsilon_j(\mathbf{x}_0))^t. \tag{26}$$

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

$$r_i(\mathbf{x}_0) = \exp(C\Upsilon_i(\mathbf{x}_0)) / \sum_{j=1}^{n} \exp(C\Upsilon_j(\mathbf{x}_0)) \tag{27}$$

where C is a parameter that can be chosen to maximize (minimize) the influence of Υ_i on r_i. When $C = 0$ we have $r_i = 1/n$, thereby ignoring any difference between the Υ_i's. On the other hand, when C is large a change in Υ_i will be exponentially reflected in r_i. The exponential weighting is more sensitive to changes in local feature relevance (25) and gives rise to better performance improvement. Moreover, exponential weighting is more stable because it prevents neighborhoods from extending infinitely in any direction, i.e., zero weight. This, however, can occur when either linear or quadratic weighting is used. Thus, (27) can be used as weights associated with features for weighted distance computation

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} r_i (x_i - y_i)^2}. \tag{28}$$

These weights enable the neighborhood to elongate along feature dimensions that run more or less parallel to the separating hyperplane, and, at the same time, to constrict along feature coordinates that have small angles with $\mathbf{w}$. This can be considered highly desirable in nearest neighbor search. Note that the technique is *query-based* because weightings depend on the query [2].

We desire that the parameter C in (27) increases with decreasing perpendicular distance between the input query and the decision boundary in an adaptive fashion. The advantage of doing so is that any difference among w_i's will be magnified exponentially in $\mathbf{r}$, thereby making the neighborhood highly elliptical as the input query approaches the decision boundary. Figure 6 illustrates this situation.

In general, however, the boundary is unknown. By using the knowledge that Equation (20) computes an approximate locally linear boundary, we can potentially solve the problem by computing the following:

$$|\mathbf{w} \cdot \mathbf{x} + b|. \tag{29}$$

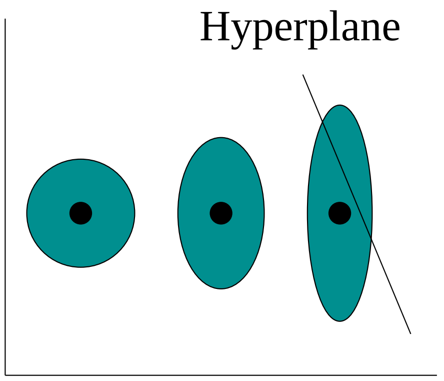

Figure 6: Neighborhood changes with decreasing distance between the query and the decision boundary.

After normalizing $\mathbf{w}$ to unit length, the above equation returns the perpendicular distance between $\mathbf{x}$ and the local separating hyperplance. We can set C to be inversely proportional to

$$\frac{|\mathbf{w} \cdot \mathbf{x} + b|}{||\mathbf{w}||} = |\mathbf{w} \cdot \mathbf{x} + b|. \tag{30}$$

In practice, we find it more effective to set C to a fixed constant. In the experiments reported here, C is determined through cross-validation.

Instead of axis parallel elongation and constriction, one attempts to use general Mahalanobis distance and have an ellipsoid whose main axis is parallel to the separating hyperplane, and whose width in other dimensions is determined by the distance of $\mathbf{x}$ from the hyperplane. The main concern with such an approach is that in high dimensions there may not be sufficient data to locally fill in $n \times n$ within sum-of-squares matrices. Moreover, very often features may be locally independent. Therefore, to effectively compute general Mahalanobis distance some sort of local clustering has to be done. In such situations, without local clustering, general Mahalanobis distance reduces to weighted Euclidean distance.

Let us examine the relevance measure (25) in the context of the Riemannian geometry proposed by Amari and Wu [1]. A large component of $\mathbf{w}$ along a direction $\mathbf{\Theta}$, i.e., a large value of $\mathbf{\Theta} \cdot \mathbf{w}$, implies that data points along that direction become far apart in terms of Equation (28). Likewise, data points are moving closer to each other along directions that have a small dot product with $\mathbf{w}$. That is, (25) and (28) can be viewed as approximating a local qausiconformal transformation around the separating boundary surface. This transformation is more judicious than that proposed by Amari and Wu [1], because this local mapping increases spatial resolution along discriminant directions around the separating boundary. In contrast, the qausiconformal mapping introduced by Amari and Wu [1] does not attend to directions.

4 Neighborhood Morphing Algorithm

The neighborhood *mor*phing nearest neighbor algorithm (MORF) has three adjustable procedural (meta-)parameters: K: the number of nearest neighbors in the final nearest neighbor rule; K_L: the number of nearest neighbors in the neighborhood N_{K_l} for local SVM computation; and C: the positive factor for the exponential weighting scheme (27).

We note that the parameter K is common to all nearest neighbor rules. Our algorithm however has added two new parameters. Arguably, there is no strong theoretic foundation upon which to determine their selection. The value of K_L should be a reasonable number to support local SVM computation. To be consistent, K_L has to be a diminishing fraction of l, the number of training points. The value of C should increase as the input query moves close to the decision boundary, so that highly stretched neighborhoods will result. We have empirically tested different ranges of values. Alternatively, cross validation can be used to choose best values for these parameters, which is what we do in the examples in the next section.

At the beginning, a nearest neighborhood of K_L points around the query $\mathbf{x}_0$ is computed using the simple Euclidean distance. From these K_L points a local linear SVM is built (i.e., the mapping from the input space to the feature space is the identity mapping (19)), whose $\mathbf{w}$ (normal to the separating hyperplance) is employed in (25) and (27) to obtain an exponential feature weighting scheme r_i. Finally, the resulting r_i (27) are used in (28) to compute K nearest neighbors at the query point $\mathbf{x}_0$ to classify $\mathbf{x}_0$. An outline of the MORF algorithm is shown in Figure 7.

The bulk of the computational cost associated with the MORF algorithm is incurred by solving the quadratic programming problem to obtain local linear SVMs. This optimization problem can be bounded by $O(nK_L^2)$ citeburges. Note that throughout computation K_L remains fixed and is (usually) less than or equal to half the number of training points (l). For large but practical n, K_L can be viewed as a constant. While there is a cost associated with building local linear SVMs, the gain in performance over simple KNN outweighs this extra cost, as we shall see in the next section.

Given a query point $\mathbf{x}_0$, and procedural parameters K, K_L and C:

1. Initialize $\mathbf{r}$ in (28) to 1;
2. Compute the K_L nearest neighbors around $\mathbf{x}_0$ using the weighted distance metric (28);
3. Build a local linear SVM from the K_L neighbors.
4. Update $\mathbf{r}$ according to (21) and (27).
5. At completion, use $\mathbf{r}$, hence (28), for K-nearest neighbor classification at the test point $\mathbf{x}_0$.

Figure 7: The MORF algorithm

5 Empirical Evaluation

In the following we compare several competing classification methods using a number of data sets:

- MORF - boundary adjusted local metric method described above, coupled with the exponential weighting scheme (27); SVMlight [12] was used to build local SVMs.
- LDAW - locally weighted Euclidean metric coupled with the exponential weighting

scheme (27). For each input query, we first use K_L nearest neighbors to the query to compute (3), i.e.,

$$\mathbf{W} = \sum_{j=1}^{J} \sum_{y_i = j} p_i (\mathbf{x}_i - \bar{\mathbf{x}}_j)(\mathbf{x}_i - \bar{\mathbf{x}}_j)^t, \tag{31}$$

from which to compute

$$\mathbf{w} = \mathbf{W}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2).$$

In case $\mathbf{W}$ is singular, we compute the pseudo-inverse of $\mathbf{W}$ by

$$\mathbf{W}^{+} = \mathbf{V} \begin{pmatrix} \Lambda^{-1} & 0 \\ 0 & 0 \end{pmatrix} \mathbf{V}^t \tag{32}$$

where $\Lambda = diag(\sigma_1, \sigma_2, \cdots, \sigma_r)$. In case $\mathbf{W}$ is ill-conditioned, we set a threshold to 0.05 to zero small singular values. We then normalize $\mathbf{w}$ using (27) to compute (11), from which the final neighborhood for classification is obtained.

- SVM-R - SVM classifier using using radial basis kernels. Again we used SVMlight [12].

- KNN - simple K nearest neighbor method using the Euclidean distance.

- C4.5 - decision tree method [17].

- MACHETE - an adaptive NN procedure [10], in which the input variable used for splitting at each step is the one that maximizes the estimated local relevance (1).

- SCYTHE - a generalization of the Machete algorithm [10], in which the input variables influence each split in proportion to their estimated local relevance, rather than the winner-take-all strategy of Machete.

- DANN - discriminant adaptive nearest neighbor classification [11].

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. Also, in all the experiments where SVMlight was involved, we did not attempt to estimate optimal values for *eps*. We instead used its default value (0.001). The values of γ in the radial basis kernel

$$\exp(-\gamma\|\mathbf{x}-\mathbf{x}'\|)$$

and c (trade-off between training error and margin) that affect the performance of SVM-R were chosen through cross-validation for each problem. Similarly, optimal procedural parameters for each method are selected through experiments for each problem. For C4.5 we used default parameters.

5.1 The Problems

The first 9 data sets are taken from UCI Repository of Machine Learning Database (Merz & Murphy, 1996). The last one is a simulated data set that is created according the given distribution. For the first six examples we randomly split the data sets into 60% training and 40% testing. For the remaining four examples we first randomly select 200 points as training data and then randomly select 200 from the remaining data as testing data. The average error rates over 20 independent runs are reported in Table 1. The procedural parameters for each method used for the data sets are listed in the Appendix.

1. Iris data (**Iris**). This data set consists of $n = 4$ measurements made on each of 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 average error rates are shown in the first row of Table 1.

2. Vote data (**Vote**). This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The data set consists of 232 instances after removing missing values, and two classes (democrat and republican). The instances are represented by 16 Boolean valued features. The average error rates are shown in the second row of Table 1.

3. Sonar data (**Sonar**). This data set consists of $n = 60$ frequency measurements made on each of 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 average error rates are shown in the third column of Table 1.

4. Ionosphere data (**Ion**). The data consists of 34 electromagnetic features that are used to determine "good" or "bad" ($J = 2$) radar returns characterizing evidence of some type of structure in the ionosphere. The data set of 351 instances. Average error rates computed are reported in the 4th row of Table 1.

5. Liver data (**Liver**). This example has $n = 6$ numerical attributes and $J = 2$ classes. There are 345 samples in this example. The average error rates over 20 independent runs are given in in the fifth row of Table 1.

6. Hepatitis data (**Hep**). The data set consists of 155 instances of 19 numerical features and $J = 2$ classes. The average error rates are reported in the sixth row of Table 1.

7. Wisconsin breast cancer data (**Cancer**). The data consists of 9 medical input features that are used to make a binary decision on the medical condition: determining whether the cancer is malignant or benign. The data set consists of 683 instances after removing missing values. The average error rates computed over all 2000 such classifications are reported in the seventh row of Table 1.

8. Pima Indians Diabete data (**Pima**). This data set consists of $n = 8$ numerical attributes measured for each of 768 samples of $J = 2$ classes. The problem is to classify each test point in the 8-dimensional space to its correct class. The average error rates over 20 independent runs are given in in the eighth row of Table 1.

9. OQ data (**OQ**). This data set consists of $n = 16$ numerical attributes and $J = 2$ classes. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the two capital letters (O and Q) in the English alphabet (they are randomly selected from 26 letter classes). Sample images are shown in Figure 8. There are total 1536 instances in this data set. Letter images are represented by 16 numerical

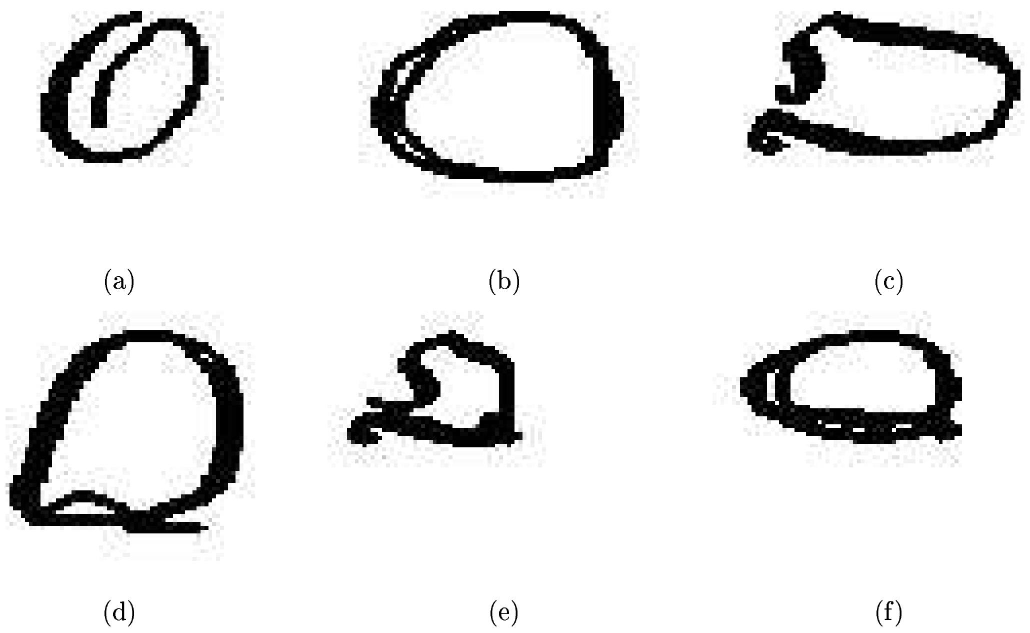

Figure 8: Sample letter images. First row: letter O. Second row: letter Q.

attributes (statistical moments and edge counts). The average error rates by each method over 20 independent runs are shown in the nineth row of Table 1.

10. Unstructured with eight noise (**Unstruct**). This problem is taken from [11]. There are $n = 10$ input features, and $J = 2$ classes. Each class contains six spherical bivariate normal subclasses (in first two dimensions), having standard deviation 0.25. The rest of eight feature dimensions follow independent standard Gaussian distributions. They serve as noise. 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 average error rates over 20 independent runs are reported in the last row of Table 1.

5.2 Results

Table 1 shows clearly that MORF achieved the best or near best performance over the ten data sets, followed by SVM-R. While performance improvement may not be significant

Table 1: Average classification error rates.

	SVM-R		MORF		LDAW		KNN		DANN		MACHETE		SCYTHE		C4.5	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Iris	4.9	3.11	**4.6**	2.88	5.4	2.88	4.9	3.58	6.4	3.83	6.0	4.21	4.8	2.95	8.3	3.54
Vote	3.7	1.78	**3.5**	1.96	7.6	3.79	8.4	2.35	3.9	1.90	5.4	2.12	5.4	1.65	**3.5**	1.60
Sonar	14.4	5.06	13.4	4.24	16.0	3.42	16.0	3.44	**12.9**	4.02	21.0	3.81	18.0	3.87	30.1	4.69
Ion	**5.4**	0.96	7.2	1.98	11.4	2.82	12.59	2.24	10.4	2.48	11.5	2.29	13.5	2.54	10.7	2.60
Liver	**28.0**	2.46	30.3	2.63	36.3	4.46	36.4	33.96	32.6	4.36	36.1	3.02	36.8	4.53	37.6	3.12
Hep	15.2	3.93	14.5	3.95	14.4	4.13	14.8	4.80	**13.6**	3.90	17.4	4.35	16.9	4.13	19.6	4.21
Cancer	2.98	0.62	2.9	0.89	3.2	0.89	3.1	0.91	**2.8**	0.78	3.6	1.04	3.2	0.95	4.0	1.31
Pima	23.5	2.41	24.6	2.57	26.6	2.91	27.1	3.35	26.4	2.47	25.7	3.00	25.7	3.00	**19.1**	1.33
OQ	**3.1**	1.35	4.3	2.11	6.1	2.16	6.4	2.04	4.5	1.88	8.0	1.69	6.3	2.01	3.5	0.81
Unstruct	29.6	2.70	**7.0**	5.92	26.1	11.78	34.0	3.56	30.0	3.39	9.0	2.25	13.6	3.07	10.1	7.40

in many cases, the results nonetheless indicate the pratical potential of MORF over other competing methods.

Clearly each method works well on some problems, while not on others. 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 algorithms. Following [10], 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:

$$b_m = \frac{e_m}{\min_{1 \le k \le 8} e_k}.$$

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 problems, therefore, seems to be a good indicator of robustness.

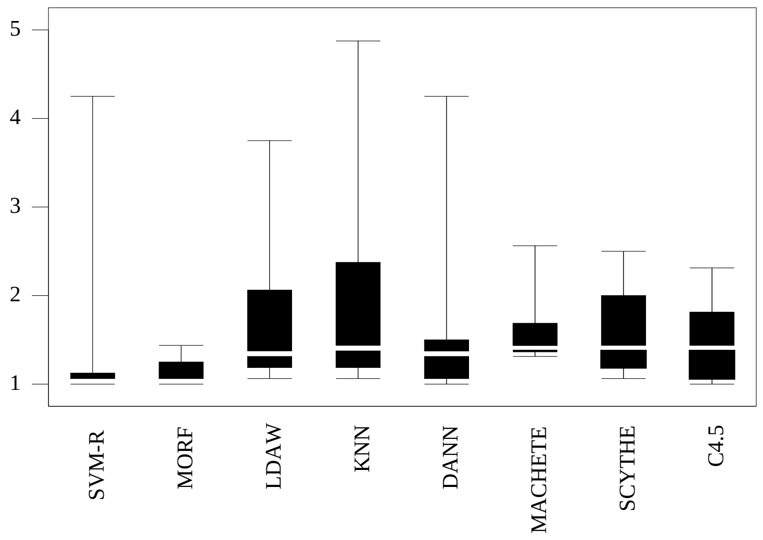

Figure 9: Performance distributions.

Fig. 9 plots the distribution of b_m for each method over the ten 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 data sets is MORF. In 3/10 of the problems its error rate was the best (median = 1.07). In 8/10 of them it was no worse than 30% higher than the best error rate. In the worst case it was 40%. In contrast, KNN has the worst distribution, where the corresponding numbers are 1.42, 234% and 488%.

6 Summary and Conclusions

This paper presents an adaptive metric 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 show clearly that the MORF algorithm can potentially improve the performance of KNN 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 MORF over similar competing methods such as Machete and DANN.

Appendix A. Procedural Parameters for Each Method Used for the Data Sets

1. Iris data. Parameters: SVM-R($\gamma = 0.08, c = 2$), MORF($K = 11, K_L = 12, C = 10, c = 1$), LDAW($K = 9, K_L = 30, C = 1$), KNN($K = 9$), DANN($K = 5, K_L = 40$), MACHETE($K = 5, L = 20, L' = 15, r = 0.96$), and SCYTHE($K = 5, L = 20, L' = 15, r = 0.96$).

2. Vote data. Parameters: SVM-R($\gamma = 0.008, c = 4.5$), MORF($K = 39, K_L = 75, C = 15, c = 0.1$), LDAW($K = 3, K_L = 104, C = 24$), KNN($K = 5$), DANN($K = 5, K_L = 90$), MACHETE($K = 7, L = 30, L' = 15, r = 0.84$), and SCYTHE($K = 5, L = 32, L' = 15, r = 0.84$).

3. Sonar data. Parameters: SVM-R($\gamma = 0.02, c = 3.5$), MORF($K = 1, K_L = 90, C = 20, c = 0.005$), LDAW($K = 1, K_L = 110, C = 24$), KNN($K = 1$), DANN($K = 9, K_L = 34$), MACHETE($K = 1, L = 24, L' = 14, r = 0.8$), and SCYTHE($K = 1, L = 30, L' = 14, r = 0.85$).

4. Ionosphere data. Parameters: SVM-R($\gamma = 0.05, c = 5$), MORF($K = 3, K_L = 104, C = 31.5, c = 0.03$), LDAW($K = 1, K_L = 139, C = 20$), KNN($K = 1$), DANN($K = 3, K_L = 97$), MACHETE($K = 3, L = 29, L' = 13, r = 0.82$), and SCYTHE($K = 1, L = 46, L' = 19, r = 0.66$).

5. Liver data. Parameters: SVM-R($\gamma = 0.05, c = 7$), MORF($K = 25, K_L = 111, C = 2, c = 1.95$), LDAW($K = 19, K_L = 190, C = 5$), KNN($K = 15$), DANN($K = 13, K_L = 143$), MACHETE($K = 1, L = 63, L' = 36, r = 0.8$), and SCYTHE($K = 1, L = 46, L' = 20, r =$

0.85).

6. Hepatitis data. Parameters: SVM-R($\gamma = 0.1, c = 5$), MORF($K = 9, K_L = 31, C = 4.1, c = 0.001$), LDAW($K = 3, K_L = 60, C = 3$), KNN($K = 3$), DANN($K = 19, K_L = 67$), MACHETE($K = 5, L = 25, L' = 14, r = 0.97$), and SCYTHE($K = 7, L = 25, L' = 14, r = 0.99$).

7. Wisconsin breast cancer data. Parameters: SVM-R($\gamma = 0.009, c = 2$), MORF($K = 7, K_L = 4, C = 1, c = 1.6$), LDAW($K = 7, K_L = 165, C = 1$), KNN($K = 7$), DANN($K = 9, K_L = 160$), MACHETE($K = 7, L = 25, L' = 14, r = 0.95$), and SCYTHE($K = 7, L = 82, L' = 43, r = 0.99$).

8. Pima Indians Diabete data. Parameters: SVM-R($\gamma = 0.007, c = 6$), MORF($K = 23, K_L = 162, C = 2.77, c = 0.5$), LDAW($K = 17, K_L = 166, C = 3$), KNN($K = 17$), DANN($K = 13, K_L = 53$), MACHETE($K = 17, L = 85, L' = 40, r = 0.7$), and SCYTHE($K = 9, L = 76, L' = 37, r = 0.61$).

9. OQ data. Parameters: SVM-R($\gamma = 0.1, c = 6$), MORF($K = 1, K_L = 50, C = 3.8, c = 0.28$), LDAW($K = 1, K_L = 61, C = 8$), KNN($K = 1$), DANN($K = 9, K_L = 117$), MACHETE($K = 5, L = 29, L' = 14, r = 0.98$), and SCYTHE($K = 1, L = 32, L' = 17, r = 0.77$).

10. Unstructured with eight noise. Parameters: SVM-R($\gamma = 0.09, c = 11$), MORF($K = 3, K_L = 173, C = 9, c = 4$), LDAW($K = 11, K_L = 188, C = 101$), KNN($K = 5$), DANN($K = 1, K_L = 20$), MACHETE($K = 1, L = 68, L' = 34, r = 0.6$), and SCYTHE($K = 9, L = 80, L' = 25, r = 0.55$).

References

[1] S. Amari & S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, *12* , 783-789, 1999.

[2] C. Atkeson, A.W. Moore, & S. Schaal, "Locally weighted learning," *AI Review, 11*, 11-73, 1997.

[3] R.E. Bellman, *Adaptive control processes*. Princeton Univ. Press, 1961.

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

[5] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 2, 121-167, 1998.

[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] C. Domeniconi, J. Peng and D. Gunopulos, "An adaptive metric machine for pattern classification," *Advances in Neural Information Processing Systems*, 2000.

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

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

[11] 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.

[12] T. Joachims, "Making large-scale SVM learning practical," *Advances in Kernel Methods - Support Vector Learning*, B. Schvlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999. http://www-ai.informatik.uni-dortmund.de/throsten/svm_light.html.

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

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

[15] C. Merz and P. Murphy, UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html.

[16] 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.

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

[18] R.D. Short and K. Fukunaga, "Optimal Distance Measure for Nearest Neighbor Classification," *IEEE Transactions on Information Theory*, Vol. 27, pp. 622-627, 1981.

[19] V. Vapnik, *Statistical Learning Theory*, Wiley & Sons, Inc., 1998.

[20] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, & V. Vapnik, "Feature selection for SVMs," *Advances in Neural Information Processing Systems*, 2000.