Learning Weighted Metrics to Minimize Nearest-Neighbor Classification Error

Roberto Paredes and Enrique Vidal, *Member, IEEE Computer Society*

Abstract—In order to optimize the accuracy of the Nearest-Neighbor classification rule, a weighted distance is proposed, along with algorithms to automatically learn the corresponding weights. These weights may be specific for each class and feature, for each individual prototype, or for both. The learning algorithms are derived by (approximately) minimizing the Leaving-One-Out classification error of the given training set. The proposed approach is assessed through a series of experiments with UCI/STATLOG corpora, as well as with a more specific task of text classification which entails very sparse data representation and huge dimensionality. In all these experiments, the proposed approach shows a uniformly good behavior, with results comparable to or better than state-of-the-art results published with the same data so far.

Index Terms—Weighted distances, nearest neighbor, leaving-one-out, error minimization, gradient descent.

1 Introduction

The Nearest-Neighbor (NN) rule is among the most popular and successful pattern classification techniques. NN classification generally achieves good results when the available number of prototypes is (very) large, relative to the intrinsic dimensionality of the data involved. However, in most real situations, the number of available prototypes is usually very small, which often leads to dramatic degradations of (k-)NN classification accuracy. This behavior is explained by the following finite-sample theoretical result: Let $T_n = \{(\mathbf{x}^1, l^1), \ldots, (\mathbf{x}^n, l^n)\}$ be a training data set of independent, identically distributed random variable pairs, where $l^i \in \{0, 1\}, 1 \leq i \leq n$ are classification labels, and let $g_n(\cdot)$ be a classification rule based on T_n. Let $\mathbf{x}$ be an observation from the same distribution and let l be the true label of $\mathbf{x}$. The probability of error is $R_n = P\{l \neq g_n(\mathbf{x})\}$. Devroye et al. show that, for any finite integer n and classification rule g_n, there exists a distribution of $(\mathbf{x}, l)$ with Bayes risk $R^* = 0$ such that the expectation of R_n is $E(R_n) \geq \frac{1}{2} - \varepsilon$, where $\varepsilon > 0$ is an arbitrary small number [7]. This theorem states that, even though we have rules, such as the k-NN rule, that are universally consistent (that is, they *asymptotically* provide optimal performance for any distribution), their *finite* sample performance can be extremely poor for some distributions. This clearly explains the growing interest in finding variants of the k-NN rule and adequate distance measures that help improve the k-NN classification performance in small data set situations. Most of these variants rely on using appropriately trained distance measures or metrics [31], [32], [13], [37], [10], [11], [12], [28], [20], [8], [3], [25], [4] or prototype *editing* techniques [36], [34], [26], [14], [5], [9], [18].

Here, we focus on distance training, aiming to achieve good performance with *given* prototype sets. More specifically, for any classification task, we assume that a set of raw supervised examples is given and our aim is to find a good metric that will lead to high classification accuracy with these raw prototypes. To this end, a distance weighting scheme is proposed which can independently emphasize prototypes and/or features in a class-dependent manner. Using the given prototypes as training data, the weights are learned by a *gradient-descent* algorithm based on update equations which are explicitly derived by (approximately) minimizing the *leaving-one-out classification error* of the training set.

The work presented here is based on our previous studies in this direction [20], [18], [17], [22], [21] which, in turn, follow the general ideas and concepts of other works such as [31], [10], [8], [28], [29], [3], [25], [4].

In [31], Short and Fukunaga presented a locally adapted distance based on the neighborhood of the query point. Their algorithm uses the *Euclidean* distance to obtain a neighborhood of the query point and, after that, a new local distance is defined based on the class means computed within this neighborhood. In [10], Hastie and Tibshirani presented a more general model based on a local *Linear Discriminant Analysis*, called the DANN algorithm. This local distance is related, under some restrictive assumptions, to the *weighted Chi-squared* distance of the class posterior probabilities between the query point and the training points. The local distance presented by Short and Fukunaga can be seen as an example of this local distance. Following the idea underlying the DANN algorithm, Domenicone et al. presented the ADAMENN algorithm [8] in which the weights associated with each feature are computed in a neighborhood of the query point by means of a *Local Feature Relevance* factor. A drawback of this otherwise interesting algorithm is the large number of parameters which need to be tuned (four neighborhood sizes, K_0, K_1, K_2, and K, a fixed number, L, of points within a defined interval, and a positive factor, c, for the exponential weighting scheme).

A different point of view is presented by Ricci and Avesani in [28], where a weighted distance is defined for each training point and a "data compression" approach is proposed. This

- *The authors are with the Departamento de Sistemas Informáticos y Computación, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Spain. E-mail: {rparedes, evidal}@iti.upv.es.*

Manuscript received 10 Dec. 2004; revised 24 Oct. 2005; accepted 9 Nov. 2005; published online 11 May 2006.
Recommended for acceptance by S.K. Pal.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0658-1204.

0162-8828/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

general idea has been further pursued in our recent work discussed in [22], [23].

Finally, a number of recent works, based on *kernel methods* and *linear embedding*, are worth mentioning. Among others, we can cite the *Adaptive Quasiconformal Kernel Nearest Neighbors* algorithm proposed in [25] by Peng et al. The quasiconformal kernel aims at expanding or shrinking the spatial resolution around prototypes whose class posterior probabilities are different from or similar to those of the query point, respectively. Among the linear embedding approaches, we can cite the *Local Linear Embedding* [29], a supervised version called SLLE [3], and the *Local Fisher Embedding* proposed in [4]. SLLE artificially increases the precalculated distances between samples belonging to different classes, but leaves them unchanged if samples are from the same class. This distance increase is controlled by a tunable parameter. The *Local Fisher Embedding* combines the LLE and the Fisher mapping approaches by means of another parameter that controls a trade-off between preserving local geometry and maximizing class separability.

With respect to the above works, the approach we propose here is discriminative in that it emphasizes the importance of the prototypes lying close to the class boundaries. On the other hand, it is fully nonparametric and explicitly aims at minimizing the same (error) criterion that will be used to measure the classifier performance in the test phase. This approach has been assessed through a series of benchmark experiments with UCI/STATLOG corpora, as well as with a more specific task of text classification which entails very large dimensionality and highly sparse data representation. In all these tests, the proposed approach exhibits a uniformly good behavior with results comparable to or better than other state-of-the-art results published on the same data sets.

The rest of this paper is organized as follows: Section 2 establishes background concepts and notation. In Section 3, the proposed optimization criterion is discussed in relation to the leaving-one-out error estimate, and the corresponding gradient descent update equations are derived. In Section 4, this criterion and learning equations are revised under different weighting schemes, aimed at reducing the overall number of parameters to be learned. Experiments are presented in Section 5, followed by conclusions drawn in the final section.

2 Preliminaries and Notation

- *Representation space*. Objects of interest are given as elements of a suitable representation space, E. Unless noted otherwise, it will be assumed that E is an m-dimensional vector space, i.e., $E = \Re^m$.
- A *training set* T is a collection of *prototypes* or *class-labeled* points of E: $T = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$, $\mathbf{x}^i \in E, 1 \le i \le n$. Without loss of generality, we will assume that all prototypes in T are different. Properly denoting repetitions in T would entail cumbersome formulation in some of the developments to appear throughout this paper. A generic prototype in T will be denoted either "$\mathbf{x} \in T$" or "$\mathbf{x}^i, 1 \le i \le n$."
- The index of a prototype $\mathbf{x} \in T$ is denoted as $index(\mathbf{x})$, defined as: $index(\mathbf{x}) = i$ *iff* $\mathbf{x} = \mathbf{x}^i$.
- The *class* of a prototype $\mathbf{x} \in T$ is an integer denoted as $class(\mathbf{x})$. The sets $T_c = \{\mathbf{x} \in T \mid class(\mathbf{x}) = c\}$ and $\bar{T}_c = \{\mathbf{x} \in T \mid class(\mathbf{x}) \neq c\}$ will denote the prototypes of class c or those of a class different from c, respectively.
- A *dissimilarity* is a function $d : E \times E \to \Re^{\geq 0}$ such that $d(\mathbf{y}', \mathbf{y}) = 0$ *iff* $\mathbf{y}' = \mathbf{y}$. Abusing the language, we will often use the words *distance* and *metric* instead of dissimilarity.
- A prototype $\hat{\mathbf{x}} \in T$ is a *d-Nearest-Neighbor* (d-NN$_T$) of $\mathbf{y} \in E$ *iff* $d(\mathbf{y}, \hat{\mathbf{x}}) \le d(\mathbf{y}, \mathbf{x})\ \forall \mathbf{x} \in T$. When talking about NNs, both d and T will be omitted if there is no ambiguity.
- *Same-class and different-class NN*. Let $\mathbf{x}$ be a prototype of class c and let $T'_c = T_c - \{\mathbf{x}\}$. The d-NN$_{T'_c}$ and the d-NN$_{\bar{T}_c}$ of $\mathbf{x}$ will be denoted as $\mathbf{x}^=$ and $\mathbf{x}^{\neq}$, respectively.
- The *step* function, centered at $z = 1$, is defined as:

$$step(z) = \begin{cases} 0 & \text{if } z < 1 \\ 1 & \text{if } z \ge 1. \end{cases} \tag{1}$$

- The *sigmoid* function with slope β, centered at $z = 1$, is defined as:

$$\mathcal{S}_\beta(z) = \frac{1}{1 + e^{\beta(1-z)}}. \tag{2}$$

  Note that, if β is large, then

$$\mathcal{S}_\beta(z) \approx step(z), \forall z \in \Re, z \neq 1.$$

- The derivative of $\mathcal{S}_\beta(\cdot)$ will often be needed throughout the paper:

$$\mathcal{S}'_\beta(z) = \frac{d\,\mathcal{S}_\beta(z)}{d\,z} = \frac{\beta e^{\beta(1-z)}}{(1 + e^{\beta(1-z)})^2}. \tag{3}$$

  $\mathcal{S}'_\beta(z)$ is a "windowing" function which is maximum for $z = 1$ and vanishes for $|z - 1| >> 0$. If β is large, then $\mathcal{S}'_\beta(z)$ approaches the Dirac delta function; conversely, if β is small, then $\mathcal{S}'_\beta(z)$ is approximately constant for a wide range of values of z.

3 Distance Definition and Weight Learning

Let $T = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ be a set of *training vectors* or *prototypes*, each of which may belong to one of C classes. A fairly general *weighted distance* from an arbitrary vector $\mathbf{y} \in E$ to a prototype $\mathbf{x} \in T$ can be defined as:

$$d(\mathbf{y}, \mathbf{x}) = \sqrt{\sum_{j=1}^{m} w^2_{\mathbf{xy}j} (y_j - x_j)^2}, \tag{4}$$

where $w_{\mathbf{xy}j}$ is a weight associated with the jth component of vectors $\mathbf{x}$ and $\mathbf{y}$.

Note that this definition can assign independent weights to the different dimensions or *features* of the representation space. Note also that it is fully *local* in that it depends on the exact positions of the two vectors being compared. This definition includes, as particular cases, the weighting schemes adopted in many papers on this topic. In particular, the *Euclidean distance* (L_2) corresponds to $w_{\mathbf{xy}j} = 1$ for all $\mathbf{x}, \mathbf{y}$, and j, while the so-called *Class-Dependent (diagonal) Mahalanobis distance* (CDM) corresponds to $w_{\mathbf{xy}j} = 1/\sigma_{cj}$, where $c = class(\mathbf{x})$ and σ_{cj} is the standard deviation of $x_j\ \forall \mathbf{x} \in c$.

3.1 Finite Parameter Formulation

The number of parameters, $w_{\mathbf{x}\mathbf{y}j}$, needed for the distance definition (4) is infinite. Therefore, in order to allow for a proper formulation of the estimation of these parameters, some simplifications are needed.

A first step is to ignore the dependence of $w_{\mathbf{x}\mathbf{y}j}$ on $\mathbf{y}$, i.e.,

$$d^2(\mathbf{y},\mathbf{x}) = \sum_{j=1}^{m} w_{ij}^2 (y_j - x_j)^2, \tag{5}$$

where $i = index(\mathbf{x}), \mathbf{x} \in T$. This way, the number of parameters to learn, $w_{ij}, 1 \le i \le n, 1 \le j \le m$, becomes finite.

Note that the weighting scheme underlying this dissimilarity is *asymmetric* in that weights are associated only with the right-hand vector (the prototype $\mathbf{x}$) of the two being compared. On the other hand, when used for NN classification, it is expected that test points generally fall close to some prototype. Correspondingly, the weights assigned to a given prototype properly determine how the dissimilarity will behave in the neighborhood of this prototype and this distance definition can be considered *local* in a similar sense as the word *local* is used in other works such as [31], [28], [10], [8]. This remark also applies to the simpler weighting schemes (17) and (20) that will be introduced in Section 4.

3.2 Learning the Weights

Our proposal for weight learning is to minimize a criterion index which is closely related with the *leaving-one-out* (LOO) NN estimate of the probability of classification error. Let W be the set of weights to be learned. The LOO NN error estimate can be written as:

$$J_T(W) = \frac{1}{n}\sum_{x\in T} step\left(\frac{d(\mathbf{x},\mathbf{x}^{=})}{d(\mathbf{x},\mathbf{x}^{\neq})}\right), \tag{6}$$

where $\mathbf{x}^{=}$ and $\mathbf{x}^{\neq}$ are the *same-class* and *different-class* NNs of $\mathbf{x}$, as defined in Section 2.

If $\mathbf{x}$ is closer to some prototype of its own class than to any other from a different class, the NN rule classifies $\mathbf{x}$ without error. In this case, $d(\mathbf{x},\mathbf{x}^{=}) < d(\mathbf{x},\mathbf{x}^{\neq})$ and the argument of *step* is smaller than 1. On the contrary, if $\mathbf{x}$ is closer to some prototype of a different class than to any other from its own class, the NN rule classifies $\mathbf{x}$ with error and the argument of *step* is greater than 1. Correspondingly, $J_T(W)$ is in fact the LOO NN estimate of the misclassification probability over the training set T.

This index is related to the theory of margin maximization and boosting [30]. In [30], the classification margin is defined as the difference between a weight assigned to the correct label and the *maximal* weight assigned to any single incorrect label. A test point is classified correctly if and only if its margin is positive. Conceptually speaking, these weights are in close relation to our distances $d(\mathbf{x},\mathbf{x}^{=})$ and $d(\mathbf{x},\mathbf{x}^{\neq})$, and the classification rule is similar to ours, where a test point is correctly classified if the relation between both distances satisfies a suitable condition.

Throughout this paper, *gradient descent* optimization will be used. This requires the functions to be minimized to be differentiable with respect to the corresponding parameters ($w_{ij}, 1 \le i \le n, 1 \le j \le m$). Therefore, some approximations are needed. First, the *step* function will be approximated by using the *sigmoid* function, $\mathcal{S}_\beta$:

$$J_T(W) \approx \frac{1}{n}\sum_{x\in T} \mathcal{S}_\beta(r(\mathbf{x})), \tag{7}$$

$$\text{where} \quad r(\mathbf{x}) = \frac{d(\mathbf{x},\mathbf{x}^{=})}{d(\mathbf{x},\mathbf{x}^{\neq})}. \tag{8}$$

Clearly, if β is large, this approximation is very accurate. On the other hand, if it is small, the contribution of each LOO NN classification error (or success) to the index J_T is more or less important depending on the corresponding quotient of the distances responsible of the error (or the success). In some cases, this can be a desirable property which may make the *sigmoid* approximation preferable to the exact *step* function.

The minimization of $J_T(W)$ by gradient descent consists in an iterative procedure which, at each step t, updates the weights w_{ij} by a small amount, μ_{ij}, in the negative direction of the gradient of J_T:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \mu_{ij}\left(\frac{\partial J_T(W)}{\partial w_{ij}}\right)^{(t)}. \tag{9}$$

The values of μ_{ij} are refered to as *learning rates* or *learning step factors*. They can take just a fixed value for all i, j or may depend on i, j following simple rules; for instance, they may be inversely proportional to the variance of each feature j.

To obtain the required partial derivatives from (7)-(8), it should be noted that J_T depends on W through the distances $d(\cdot,\cdot)$ in two different ways. First, it depends directly through the weights involved in the definition of $d(\cdot,\cdot)$ (5). The second, more subtle dependence is due to the fact that, for each $\mathbf{x} \in T$, $\mathbf{x}^{=}$ and $\mathbf{x}^{\neq}$ may change as the weights W are varied. Correspondingly, we can write:

$$J_T(W) = J_T(W, \mathcal{H}(W)), \tag{10}$$

where $\mathcal{H}$ is an abstract selection function which determines which prototypes are *same-class* and *different-class* NNs of the others. Therefore, the partial derivatives of $J_T(W)$ involve *primary* terms, $\partial J_T/\partial w_{ij}$, plus *secondary* terms which depend on the derivatives of $\mathcal{H}$, $\partial\mathcal{H}/\partial w_{ij}$.

As we will see below, the primary terms can be directly developed from (7)-(8). The secondary terms are more problematic. $\mathcal{H}(W)$ is not a continuous function of W and, moreover, the dependence of $\mathcal{H}$ on W is quite complex. While this formulation can still be followed to some extent [17], the development becomes rather cumbersome and, in the end, it does not really lead to useful approximations. Therefore, a simpler approach will be followed here which just ignores the secondary dependence of J_T on W through $\mathcal{H}(W)$. In other words, we will assume that, for sufficiently small variations of the weights, the prototype neighborhoods remain unchanged.[1] Under this assumption, we can derive from (5) and (7)-(8):

$$\begin{aligned}\frac{\partial J_T}{\partial w_{ij}} \approx & \frac{1}{n}\sum_{\substack{\forall \mathbf{x}\in T:\\ index(\mathbf{x}^{=})=i}} \mathcal{S}'_\beta(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x},\mathbf{x}^{=}) w_{ij} \\ & - \frac{1}{n}\sum_{\substack{\forall \mathbf{x}\in T:\\ index(\mathbf{x}^{\neq})=i}} \mathcal{S}'_\beta(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x},\mathbf{x}^{\neq}) w_{ij},\end{aligned} \tag{11}$$

[1] This *assumption* can certainly be inappropriate for specific prototype and metric configurations. We hope, however, that the effects of the secondary terms will be negligible in many practical situations, thereby making the proposed approximation adequate in these cases.

where $r(\mathbf{x})$ and $\mathcal{S}'_\beta(\cdot)$ are as in (8) and (3), respectively, and, for $\tilde{\mathbf{x}} \in \{\mathbf{x}^=, \mathbf{x}^{\neq}\}$:

$$R_j(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{(x_j - \tilde{x}_j)^2}{d^2(\mathbf{x}, \tilde{\mathbf{x}})}. \qquad (12)$$

Using these derivatives in (9) leads to the following update equations:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \mu_{ij}\, w_{ij}^{(t)} \left(\sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^=)=i}} \mathcal{S}'_\beta(r(\mathbf{x}))\; r(\mathbf{x})\; R_j(\mathbf{x}, \mathbf{x}^=) \right. \\ \left. - \sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^{\neq})=i}} \mathcal{S}'_\beta(r(\mathbf{x}))\; r(\mathbf{x})\; R_j(\mathbf{x}, \mathbf{x}^{\neq}) \right). \qquad (13)$$

The effects of these equations are intuitively clear. For each prototype $\mathbf{x}$, the weight associated with its *same-class* NN, $\mathbf{x}^=$, is modified so as to make it appear closer to $\mathbf{x}$, while that of its *different-class* NN, $\mathbf{x}^{\neq}$, is modified so that it will appear farther from $\mathbf{x}$.

All the update equations are affected by the windowing factor $\mathcal{S}'_\beta(r(\mathbf{x}))$ (the derivative of the sigmoid function). The argument of S'_β is the distance ratio (8) between the $\mathbf{x}^=$ and $\mathbf{x}^{\neq}$ for each training vector $\mathbf{x} \in T$. For large values of β, learning only happens when the distance ratio is (very) close to 1, maybe never if β is very large. On the other hand, for small values of β, the sigmoid derivative is almost constant and the algorithm would learn almost the same regardless of the value of $r(\mathbf{x})$. That is, the same importance would be given to those training vectors $\mathbf{x}$ that are safely well-classified (with $r(\mathbf{x}) \ll 1.0$) as to other vectors $\tilde{\mathbf{x}}$ that lie close to the class decision boundaries ($r(\tilde{\mathbf{x}}) \approx 1.0$) or are plainly misclassified ($r(\tilde{\mathbf{x}}) > 1.0$). In this case, as the number of correctly classified vectors becomes much larger than the number of errors, after some iterations, the algorithm can become reluctant to learn more. A suitable β value should allow the proposed algorithms to learn from the prototypes that lie near the class decision boundaries or are misclassified, but, moreover, the windowing effect of the sigmoid derivative should prevent learning from outliers whose $r(\mathbf{x})$ value is too large.

This property reminds us of the effects of boosting techniques. Boosting is known to be particularly good at finding classifiers with large margins because it focuses on those points whose margins are small (or negative) and forces the base learning algorithm to generate good classifications for those points. In our case, the effect of the derivative of the sigmoid function enforces this same behavior. An empirical study of these effects will be presented in Section 5.3.

Note that, in general, gradient descent does not guarantee global optimization. Moreover, the descent equations (13) have been obtained thanks to several approximations to the original LOO error estimation criterion (6). As a consequence, the weights obtained at the end of the gradient descent are not necessarily an optimal solution for NN classification. In some cases, however, the partial solutions (weights) available at some intermediate steps of the descent process may happen to be better solutions than those obtained at the end. Since a true LOO NN error estimation is available at each step as a byproduct of computing (13), this suggests selecting a set of weights $\hat{W}$ whose error estimation is the lowest among all Ws obtained throughout the descent process. This guarantees that, despite all the approximations, the resulting weights will always provide a LOO NN error estimation better than or at least as good as that provided by the weights used to initialize the descent procedure.

3.3 Asymptotic Behavior

The simple weight selection technique mentioned above allows us to characterize the asymptotic behavior of the classification error of the proposed approach.

Devroye et al. show that, for any finite training set of size n, $E\{|\hat{\epsilon}(n) - \epsilon(n)|\} \leq \sqrt{7/n}$, where $\hat{\epsilon}(n)$ is the LOO error estimation for the nearest neighbor classifier and $\epsilon(n)$ is the probability of error of this classifier. This upper bound is metric-independent and distribution-free.[2] Consequently, when n tends to infinity, the LOO error estimation of an NN classifier tends to the expected error rate of this classifier.

As discussed above, the weights obtained by the proposed weight selection technique always provide a LOO NN error estimation, $\hat{\epsilon}_W(n)$, better than or at least as good as that provided by the weights used to initialize the descent procedure, $\hat{\epsilon}_D(n)$; i.e., $\hat{\epsilon}_W(n) \leq \hat{\epsilon}_D(n)$. Let ϵ_W be the asymptotical error of the NN with the optimal weights provided by the proposed approach and ϵ_D be the asymptotical error of the plain NN classifier with the metric used to initialize the proposed weight learning algorithm. Then:

$$\left.\begin{array}{l} \hat{\epsilon}_W(n) \leq \hat{\epsilon}_D(n) \\ \lim_{n\to\infty} \hat{\epsilon}_D(n) = \epsilon_D \\ \lim_{n\to\infty} \hat{\epsilon}_W(n) = \epsilon_W \end{array}\right\} \rightarrow \epsilon_W \leq \epsilon_D. \qquad (14)$$

In conclusion, the algorithm proposed here guarantees an asymptotical classification error which is equal to or lower than that of the original NN classifier with the initial distance.

4 Reducing the Number of Parameters to Be Learned

The number of parameters involved in the distance (5) defined in Section 3 is exceedingly large for practical purposes: There are $n \cdot m$ parameters, i.e., as many as scalar data available in T. In order to render the parameter learning problem tractable, several approaches to reduce this number will be discussed in the following sections.

4.1 Sharing All the Weights within Each Class

A natural way to reduce the number of parameters is to assume that all the prototypes of the same class share the same weights. This *Class-dependent Weighting* (CW) scheme leads to a dissimilarity defined as:

$$d^2_{CW}(\mathbf{y}, \mathbf{x}) = \sum_{j=1}^{m} w^2_{cj}(y_j - x_j)^2, \qquad (15)$$

[2] The bound is obtained under the assumption that the same randomized tie-breaking criterion is used both for LOO error estimation and for the final classifier (see [6] and [7] for details). Note that randomized tie-breaking is the usual way to deal with equal-distance situations, though it is seldom necessary because features are continuous in most cases.

where $c = class(\mathbf{x}), \mathbf{x} \in T$. The number of parameters, $w_{cj}, 1 \le c \le C, 1 \le j \le m$, is now $C \cdot m$; i.e., n/C times less than the amount of scalar data in T.

Departing from (15) and (7)-(8), a similar development as in Section 3.2 easily yields the following update equations:

$$\begin{aligned} w_{cj}^{(t+1)} = w_{cj}^{(t)} &- \sum_{\substack{\forall \mathbf{x} \in T: \\ class(\mathbf{x}^{=})=c}} \mu_{cj}\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x}, \mathbf{x}^{=}) w_{cj}^{(t)} \\ &+ \sum_{\substack{\forall \mathbf{x} \in T: \\ class(\mathbf{x}^{\neq})=c}} \mu_{cj}\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x}, \mathbf{x}^{\neq}) w_{cj}^{(t)}, \end{aligned} \tag{16}$$

where $r(\mathbf{x})$, $\mathcal{S}'_{\beta}(\cdot)$, and $R_j(\cdot,\cdot)$ are as in (8), (3), and (12), respectively, and μ_{cj} are adequate learning step factors. Note that, by the definition of $\mathbf{x}^{=}$, the condition of the first sum in (16) can be equivalently written as $\forall \mathbf{x} \in T : class(\mathbf{x}) = c$.

As in the general case (13), these equations modify the weights associated with $\mathbf{x}^{=}$ and $\mathbf{x}^{\neq}$ so as to make them appear closer or farther, respectively, from each $\mathbf{x} \in T$. Here, however, the same modifications affect globally to all prototypes in each class.

4.2 Sharing Weights for Each Prototype

A different way to reduce the number of parameters in definition (5) is to assume that all data features (dimensions) are equally "important" (so they have the same weight), but distances are measured differently depending on the (positions of the) specific prototypes involved. Such a *Prototype-dependent Weighting* (PW) scheme leads to a *local* dissimilarity defined as:

$$d^2_{PW}(\mathbf{y}, \mathbf{x}) = \sum_{j=1}^{m} v_i^2 (y_j - x_j)^2, \tag{17}$$

where $i = index(\mathbf{x}), \mathbf{x} \in T$. The number of parameters, $v_i, 1 \le i \le n$, is now n, i.e., m times less than the amount of scalar data in T.

This weighting scheme is particularly interesting because it can be applied to any kind of dissimilarity, even to *nonvector* space dissimilarities. Let δ be any dissimilarity defined in an arbitrary representation space E. Then, a PW dissimilarity is defined as:

$$d_{PW}(y, x) = v_i\ \delta(y, x). \tag{18}$$

Obviously, if $E = \Re^m$ and δ is the Euclidean metric in E, then (18) reduces to (17).

Now, let us use the dissimilarity d_{PW} defined by (18) in (7)-(8). In this case, the same dependence assumptions as in the introduction to Section 3.2 can be made to obtain the derivatives $\partial J_T / \partial v_i$, leading to the following update equations:

$$\begin{aligned} v_i^{(t+1)} = v_i^{(t)} &- \sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^{=})=i}} \rho_i\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, \frac{1}{v_i^{(t)}} \\ &+ \sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^{\neq})=i}} \rho_i\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, \frac{1}{v_i^{(t)}}, \end{aligned} \tag{19}$$

where $r(\mathbf{x})$ and $\mathcal{S}'_{\beta}(\cdot)$ are as in (8) and (3), respectively, and ρ_i are adequate learning step factors.

As in the general case (13), these equations modify the weights associated with $\mathbf{x}^{=}$ and $\mathbf{x}^{\neq}$ so as to make them appear closer or farther, respectively, from each $\mathbf{x} \in T$. Here, however, the modifications are *local*, as they only affect the position (or neighborhood) of each prototype involved.

On the other hand, in contrast with (13) and (16), the update equations (19) do not depend on $R_j(\cdot,\cdot)$. That is, they are independent on the features (j) of the objects being compared. As mentioned above, since only full interobject distances are involved, this weight learning technique can be applied to any arbitrary base dissimilarity, δ, even in cases where it is not defined in a vector space.

4.3 Combining Class-Dependent and Prototype-Dependent Weighting

Definitions (15) and (17) can be combined into a weighting scheme that assumes both the *Class* and the *Prototype* dependencies, but in an independent manner, that is:

$$d^2_{CPW}(\mathbf{y}, \mathbf{x}) = v_i^2 \sum_{j=1}^{m} w_{cj}^2 (y_j - x_j)^2, \tag{20}$$

where $i = index(\mathbf{x})$ and $c = class(\mathbf{x}), \mathbf{x} \in T$.

Now, we have two sets of parameters, $v_i, 1 \le i \le n$ and w_{cj}, $1 \le c \le C, 1 \le j \le m$, which amounts to $n + C \cdot m$ parameters—still generally much less than the total amount of scalar data in T. Finally, using (20) in (7)-(8), yields the following update equations for the combined d_{CPW} dissimilarity:

$$\begin{aligned} w_{cj}^{(t+1)} = w_{cj}^{(t)} &- \sum_{\substack{\forall \mathbf{x} \in T: \\ class(\mathbf{x}^{=})=c}} \mu_{cj}\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x}, \mathbf{x}^{=}) w_{cj}^{(t)} \\ &+ \sum_{\substack{\forall \mathbf{x} \in T: \\ class(\mathbf{x}^{\neq})=c}} \mu_{cj}\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, R_j(\mathbf{x}, \mathbf{x}^{\neq}) w_{cj}^{(t)} \\ v_i^{(t+1)} = v_i^{(t)} &- \sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^{=})=i}} \rho_i\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, \frac{1}{v_i^{(t)}} \\ &+ \sum_{\substack{\forall \mathbf{x} \in T: \\ index(\mathbf{x}^{\neq})=i}} \rho_i\, \mathcal{S}'_{\beta}(r(\mathbf{x}))\, r(\mathbf{x})\, \frac{1}{v_i^{(t)}}, \end{aligned} \tag{21}$$

where $r(\mathbf{x})$, $\mathcal{S}'_{\beta}(\cdot)$ and $R_j(\cdot,\cdot)$ are as in (8), (3), and (12), respectively, and μ_{cj}, ρ_i are adequate learning step factors.

As in the general case (13), these equations modify the weights associated with $\mathbf{x}^{=}$ and $\mathbf{x}^{\neq}$ so as to make them appear closer or farther, respectively, from each $\mathbf{x} \in T$. Here, however, the different types of weights are expected to account both for class (and dimension)-dependent effects and/or for local effects which only depend on the position of the prototypes involved.

Note that, as defined in (8), $r(\mathbf{x})$ involves two different subsets of weights, one associated with $\mathbf{x}^{=}$ and the other with $\mathbf{x}^{\neq}$. Therefore, a simple manner to implement the update (21) is by visiting each prototype $\mathbf{x}$ in T and updating the weights associated with the *same-class* and *different-class* NNs of $\mathbf{x}$. This is shown in the iterative procedure presented in Fig. 1. To initialize this procedure, a set of *initial weights* (V, W) is needed. Typically, the weights in V are simply initialized to 1, while either the Euclidean or the CDM weights can be used to initialize W.

A similar procedure can be written for all the update equations discussed in the previous sections.

```
ClassPrototypeWeightLearning $(T, V, W, \beta, \mu, \rho, \varepsilon)$ {
    // T: training set; V, W: initial weights;
    // $\beta$: sigmoid slope; $\mu, \rho$: learning factors; $\varepsilon$: small constant
    $\lambda' = \infty;\ \lambda = J_T(V, W);\ V' = V;\ W' = W$
    while($|\lambda' - \lambda| > \varepsilon$) {
        $\lambda' = \lambda$
        for all $\mathbf{x} \in T$ {
            $\mathbf{x}^{=} = $ findNNsameClass$(T, V, W, \mathbf{x})$
            $\mathbf{x}^{\neq} = $ findNNdiffClass$(T, V, W, \mathbf{x})$
            $i = index(\mathbf{x}^{=});\ k = index(\mathbf{x}^{\neq})$
            $c = class(\mathbf{x}^{=});\ l = class(\mathbf{x}^{\neq})$
            $Q = \mathcal{S}'_\beta(r(\mathbf{x})) \cdot r(\mathbf{x})$
            $v'_i = v'_i - \rho_i \cdot Q / v_i$
            $v'_k = v'_k + \rho_k \cdot Q / v_k$
            for $j = 1 \ldots m$ {
                $w'_{cj} = w'_{cj} - \mu_{cj} \cdot Q \cdot R_j(\mathbf{x}, \mathbf{x}^{=}) \cdot w_{cj}$
                $w'_{lj} = w'_{lj} + \mu_{lj} \cdot Q \cdot R_j(\mathbf{x}, \mathbf{x}^{\neq}) \cdot w_{lj}$
            }
        }
        $V = V';\ W = W';\ \lambda = J_T(V, W)$
    }
    return$(V, W)$
}
```

Fig. 1. Class and Prototype Weight Learning Algorithm (CPW).

5 Experiments

The capabilities of the proposed distance learning techniques have been empirically assessed through three different types of experiments. In the first one, a synthetic data set was used to show the behavior of the proposed approach in a controlled setting. In the second experiment, several *standard benchmark corpora* from the well-known UCI Repository of Machine Learning Databases and Domain Theories [1] and the STATLOG Project [33] were considered. The last experiment corresponds to a more specific task of *text classification*, which will be fully described in Section 5.5.

5.1 Synthetic Data

The following class-conditional normal distributions with identical priors were assumed. Class A: $\mu = (2, 0.5)^t$, $\Sigma = (1, 0; 0, 1)$ (identity matrix). Class B: $\mu = (0, 2)^t$, $\Sigma = (1, 0.5; 0.5, 1)$. Class C: $\mu = (0, -1)^t$, $\Sigma = (1, -0.5; -0.5, 1)$. See Fig. 2.

The standard technique to achieve good classification boundaries in this task would be *editing* [18]. Alternatively, we can reduce the importance of the prototypes that are in the class overlapping regions by increasing the associated weights using the proposed PW approach. Given the symmetries underlying the proposed task, it seems clear that class-dependent feature weighting (CW) could hardly help to improve the boundaries in this case.

Fig. 2 shows classification error rates for different training set sizes, ranging from 8 to 256 prototypes per class. For each size, the algorithm was run 100 times with different training sets randomly drawn from the above distribution. A fixed test set of 5,000 vectors, independently drawn from the same distribution, was used for error estimation. Each point of the figure is the error averaged over the 100 runs. The class and prototype dependent weights, w_{cj} and v_i, respectively, were initialized to 1.0. The learning rates μ_{cj} and ρ_i were set to 0.001 and 0.01, respectively. The sigmoid slope β was set to 10.

The results agree with the above discussion. CW is slightly worse than the Euclidean distance for very small training sets due to the biased LOO error estimation. However, as the amount of training data increases, CW becomes as good as the original Euclidean distance. This tendency is in clear agreement with the asymptotical behavior discussed in Section 3.3. On the other hand, both PW and CPW are better than the Euclidean distance, even for small training sets and the improvement increases with the amount of training data.

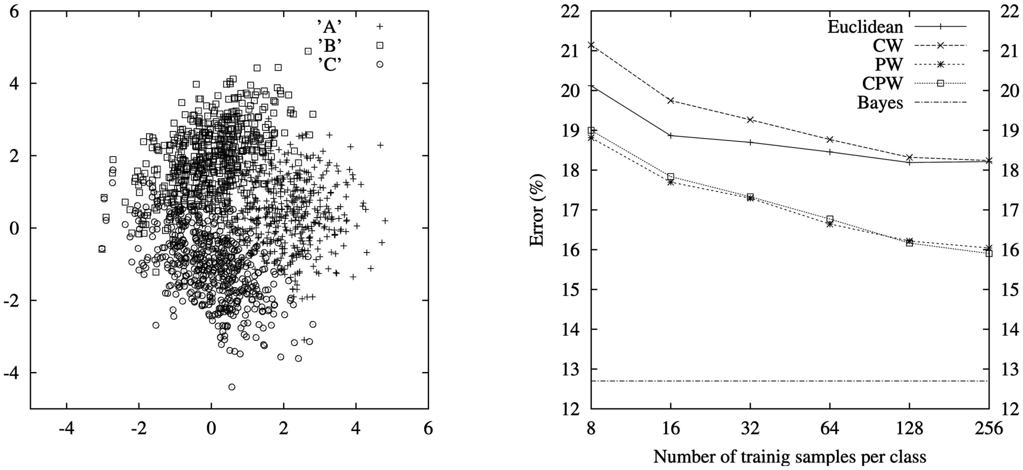

Fig. 2. Left, class distributions. Right, comparison of the Bayes risk and the Nearest neighbor error with the Euclidean, CW, PW, and CPW dissimilarities.

TABLE 1
Benchmark Data Sets Used in the Experiments

Task	N	C	m
Australian	690	2	42
Balance	625	3	4
Cancer	685	2	9
Diabetes	768	2	8
DNA	3,186	3	180
German	1,000	2	24
Glass	214	6	9
Heart	270	2	25
Letter	20,000	26	16
Liver	345	2	6
Satimage	6,435	6	36
Vehicle	846	4	18
Vote	435	2	10
Vowel	440	11	16
Wine	178	3	13

N, C, and m are, respectively, the total number of vectors, the number of classes, and the dimension of each data set. In two data sets, Australian and Heart, m is the dimension after expanding categorical features (the corresponding original dimensions were 14 and 13, respectively).

TABLE 2
Error Rate (%) Obtained Using CW for Different Values of β

	0.125	0.5	2.0	8.0	32	128	Avge	StdD
Australian	18.10	17.64	16.79	17.37	17.62	18.06	17.6	0.18
Balance	17.00	17.03	17.25	17.98	17.75	25.26	18.70	1.20
Cancer	3.77	3.83	4.09	3.69	3.73	4.75	3.97	0.15
Diabetes	30.67	30.82	30.92	30.23	30.44	32.60	30.95	0.32
German	27.68	27.77	28.30	27.99	27.74	32.15	28.61	0.65
Glass	28.09	28.41	28.20	28.52	28.37	27.23	28.13	0.17
Heart	23.44	23.60	22.78	22.34	22.77	22.55	22.91	0.18
Liver	40.05	40.20	40.39	40.22	39.57	39.42	39.98	0.15
Vehicle	30.59	30.54	30.25	29.38	30.43	32.10	30.55	0.33
Vote	7.05	7.03	7.03	6.61	6.25	6.97	6.82	0.12
Vowel	1.55	1.60	1.51	1.36	1.64	1.67	1.56	0.04
Wine	2.15	2.15	2.06	1.44	2.43	2.60	2.13	0.14
Average	19.18	19.22	19.13	**18.93**	19.10	20.45		

5.2 UCI and Statlog Corpora

A short description of the selected UCI/STATLOG corpora is given in Table 1. Some of these data sets involve both *numeric* and *categorical* features. In our experiments, each categorical feature has been replaced by as many binary features as different values are allowed for this feature. Many UCI and STATLOG data sets are small. In these cases, *B-Fold Cross-Validation* [27] (B-CV) has been applied to estimate error rates. Each corpus is divided into B blocks using $B-1$ blocks as a training set and the remaining block as a test set. Therefore, each block is used exactly once as a test set. In all the experiments with UCI/STATLOG data, B is fixed to 5, except for *DNA*, *Letter*, and *Satimage*. In these relatively larger corpora, the single partition into training and test sets specified in the UCI repository was adopted.

5.3 Dependence on the Sigmoid Slope

As previously discussed, the slope of the sigmoid function, β, may affect the learning performance of the proposed techniques. This section is devoted to studying this dependence and to determine adequate values to be used in further experiments. Only the results for the CW dissimilarity measure will be reported. Similar behavior was observed for both PW and CPW. The experiments were performed with the "small" selected data sets from the UCI/STATLOG repository, using 5-CV to estimate the error rate, as mentioned above. In order to clearly show the tendencies we are interested in, sufficiently smooth results are needed. To achieve this goal, in these (relatively small) experiments, each training-testing experiment was run 100 times using different random 5-CV partitions and the results were averaged over the 100 runs.

The weights of the CW dissimilarity were initialized according to the following simple rule, which is based on LOO NN performance of conventional methods on the training data: If the raw Euclidean (L_2) metric outperforms Class Dependent Mahalanobis (CDM), then set all initial $w_{ij} = 1$; otherwise, set them to the inverse of the corresponding training data standard deviations. Similarly, the step factors, μ_{ij}, were set to a small constant (0.001) in the former case and to the inverse of the variance in the latter. In the case of CDM, computation singularities can appear when dealing with categorical features, which often exhibit *null* class-dependent variances. This problem was solved by using the overall variance as a "back-off" smoothing for the null values.

Table 2 shows the results obtained for a range of values of β.

A fairly stable CW behavior is observed for all the values of β up to 32, with better overall results around $\beta = 8$. Accuracy tends to worsen significantly using $\beta = 128$ for several tasks (Balance, German, and Vehicle). This is consistent with our discussion about the update equations of the proposed gradient descent algorithm (Section 3.2).

For one of the tasks studied in Table 2, *Vehicle*, Fig. 3 plots the CW results as a function of β, along with the results obtained using the Euclidean and CDM distances. For $\beta = 128$, the error rate obtained is the same as that of the Euclidean distance, which corresponds to the weights used to initialize the CW. Clearly, for such a large β value, the descent algorithm was not able to learn the appropriate class dependent weights for this task.

5.4 Experiments with CW, PW, and CPW

The experiments in this section were carried out to compare the results obtained using the baseline distances (L_2, CDM) and the three trained dissimilarities (CW, PW, CPW) proposed here. In all the cases, the $1-NN$ classification rule was used. Following the results of the previous section, the sigmoid slope was set to $\beta = 8.0$ in all the cases. The results are reported in Table 3. For the small data sets, these results

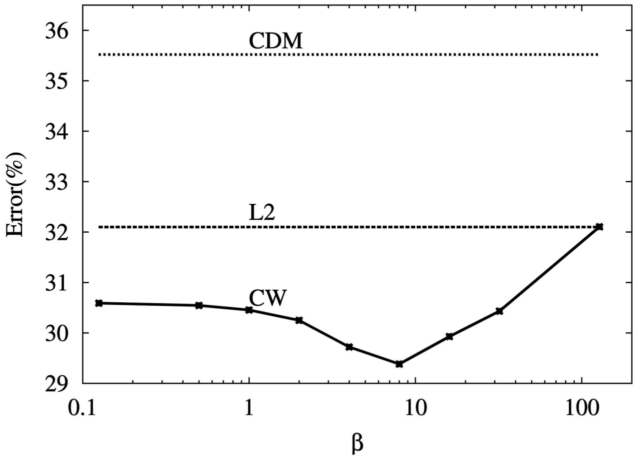

Fig. 3. Results for the *Vehicle* corpus, using Euclidean (L2), Class Dependent Mahalanobis, and $CW(\beta)$ dissimilarities.

were obtained, as in the previous subsection, by averaging 100 5-CV runs on the available data. For the larger corpora (*Letter*, *Dna*, and *Satimage*), the standard training/test partition specified in the UCI/STATLOG repository was adopted.

Initial values of the class dependent weights w_{ij} and the corresponding learning rates μ_{ij} for CW were selected using the same simple rule described in the previous subsection. In the case of PW, the prototype weights v_i were initialized to 1.0 and the corresponding learning rates ρ_i were set to 0.001. Finally, the combined CPW class and prototypes dependent weights (w_{ij}, v_i) were initialized as for CW and PW. In this last case, using values of μ significantly higher than those of ρ amounts to give more emphasis to weight the features than the prototypes, while, if the values of ρ are higher that those of μ, weighting the prototypes is more important. Therefore, several combinations of learning factors $\mu_{ij} \in [0.0, 0.01]$ and $\rho_i \in [0.0, 0.01]$ were considered during the first five iterations of the gradient descent algorithm. The combination with the best LOO error estimation after these initial iterations was adopted for the remaining gradient descent process.

Results are shown in Table 3. Most of the proposed learned dissimilarities achieved better results than the baseline Euclidean or CDM distances (which were used to initialize the learning algorithms) and many of these improvements are statistically significant assuming 95 percent confidence intervals.

Taking into account that PW consists just in weighting the baseline (Euclidean or CDM) distance by a weight v_i learned for each prototype $\mathbf{x}_i$, the important accuracy gain of this dissimilarity with respect to the baselines is remarkable. Clearly, the algorithm learns large weights for outliers and/or prototypes that are not useful for the classification, while small weights are obtained for those prototypes which are important to define class boundaries. This explains the very good behavior of the *editing* technique presented in [18], [24], which consisted in pruning out those prototypes $\mathbf{x}_i$ for which v_i is sufficiently large.

PW results are also generally better than those of CW. Finally, CPW, by combining feature/class and prototype weights, generally achieves some improvements over CW or/and PW. Moreover, CPW outperforms both baseline Euclidean and CDM in all the cases, except *Glass*.

TABLE 3
Nearest Neighbor Error Rates (%) for the Different Dissimilarities

	L_2	CDM	CW	PW	CPW
Australian	34.37	18.19	17.37	16.95	16.83
Balance	25.26	35.15	**17.98**	**13.44**	**17.60**
Cancer	4.75	8.76	3.69	3.32	3.53
Diabetes	32.25	32.47	30.23	**27.39**	**27.33**
Dna	23.44	15.01	**4.72**	**6.49**	**4.21**
German	33.85	32.15	**27.99**	**28.32**	**27.29**
Glass	27.23	32.90	28.52	26.28	27.48
Heart	42.18	22.55	22.34	18.94	19.82
Letter	4.35	6.30	**3.15**	4.6	4.2
Liver	37.7	39.32	40.22	36.22	36.95
Satimage	10.55	14.70	11.70	**8.80**	9.05
Vehicle	35.52	32.11	29.38	29.31	**28.09**
Vote	8.79	6.97	6.61	5.51	5.26
Vowel	1.52	1.67	1.36	1.68	1.48
Wine	24.14	2.60	1.44	1.35	1.24

Baseline: Euclidean (L_2) and Class-Dependent Mahalanobis (CDM); Learned: Class Weighted (CW), Prototype Weighted (PW), and Class and Prototype Weighted (CPW). The results typeset in boldface are significantly better (with 95 percent confidence intervals) than the best of the baseline distances (Euclidean or CDM).

Generally speaking, these results are comparable to or better than those obtained by other state-of-the-art methods recently published on the same tasks [22], [21], [4], [25], [15].

5.5 Text Classification

The capabilities of the proposed distance learning techniques have been further assessed in a number of more specific classification tasks including OCR [17], [24], face recognition [22], confidence measures for Speech Recognition [17], [19], and text classification [17]. In order to provide further insight into the proposed techniques, an additional task of text classification is considered here which is known as *"4 Universities WebKb."*

The WebKb data set [2] contains Web pages gathered from university computer science departments. The pages are divided into seven categories: *student*, *faculty*, *staff*, *course*, *project*, *department*, and *other*. Most works carried out on this corpus have focused on the four most populous entity-representing categories: *student*, *faculty*, *course*, and *project*, all together containing 4,199 documents. In the present work, we also adopt this standard setting. To estimate error rates, we adopted the defacto standard hold-out partition generally used for 4-Univ WebKb corpus, where 70 percent of the data is used for training and 30 percent for testing.

Documents are represented using the popular *bag-of-words* approach. An m-dimensional vector of word counts, $\mathbf{x}$, is assigned to each document, where m is the size of a given vocabulary. Each feature $j, 1 \leq j \leq m$, corresponds to a word of the vocabulary and x_j is the number of times that the jth word appears in the document.

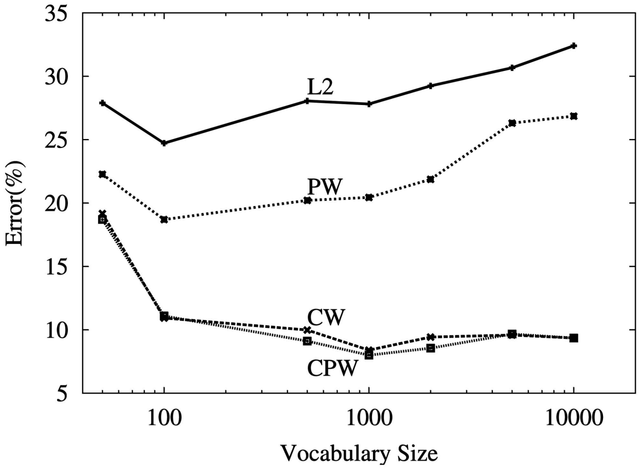

Fig. 4. WebKb Nearest Neighbor classification results using the Euclidean (L2) distance, as well as the CW, PW, and CPW learned distances proposed here.

Vocabulary words are selected following basic ideas commonly applied in the field of text classification. All the words apearing in the given document collection are sorted according to a mutual information criterion. The selected vocabulary is then determined by picking the top m words from the full vocabulary sorted in this way [38]. It should be noted that m can be huge, which makes data representation very sparse. For instance, an average WebKb document contains about 80 different words, out of a vocabulary of 10^4 words. Therefore, for this largest m, each document is represented as a 10^4-dimensional vector and, on the average documents, 99 percent of the features are *zero*.

The experiments compare the results obtained with the NN rule using a conventional baseline distance and the here proposed CW, PW, and CPW learned dissimilarities. The Euclidean distance has been selected as the baseline because it always outperforms the CDM distance in this task. In this case, the sigmoid slope was set to $\beta = 10$ and the learning factors to $\mu = 0.01$ and $\rho = 0.001$. Fig. 4 shows the results obtained for increasing vocabulary sizes.

The Euclidean distance achieves its best result (24.7 percent) for a vocabulary size as small as 100 words. For larger sizes, errors tend to increase monotonically. This is certainly due to the fact that not all the vocabulary words share the same class-discriminating power. In this kind of problem with a word-count representation, it is important to adequately enhance the class-dependent influence of the most important words and to lower the impact of the irrelevant words in each class. By adequately weighting each individual reference document (prototype), the PW metric notably improves the unweighted L2 accuracy. However, a similar tendency to degradation with increassing vocabulary size is observed. Clearly, these distances cannot account for the (often very large) word discriminating power differences and using more features only tends to add noise to the representation.

The other two learned dissimilarities proposed here, CW and CPW, easily overcome the problem. Results are much less sensible to vocabulary sizes, with a general tendency to improve accuracy with increasing sizes. The best results are now obtained for a 1,000-words vocabulary, with 8.4 percent and 8.0 percent error rates for CW and CPW, respectively. The error rate was cut to a third of that of the original Euclidean distance, which was used to initialize the CW/CPW gradient descent algorithms.

TABLE 4
Error Rates Achieved by CW, PW, and CPW Compared with Those Reported for Other Techniques [16] on the Same 4-Univ WebKb Corpus and Experimental Setup

Method	Error rate (%)
Naive Bayes	13.7
Scaled NBayes	13.1
Max Entropy w/Prior	8.1
Max Entropy	7.9
1–NN Euclidean	24.7
1–NN PW	18.7
1–NN CW	8.4
1–NN CPW	8.0

Table 4 compares these results with state-of-the-art results obtained using other techniques on the 4-Univ WebKb corpus under the same experimental setup [16] (see also [35] for additional results on this corpus). Naive Bayes and Maximum Entropy are two commonly used techniques for document classification tasks. Maximum Entropy can suffer from overfitting. By introducing a prior on the model, overfitting can be reduced [16]. According to the results in Table 4, our 1-NN CPW classifier constitutes a competitive approach for this task.

6 Conclusion

From the results reported in the last section, we can conclude that the proposed techniques achieved a uniformly good performance when applied to a great variety of classification tasks, including those involving categorical data, as well as others with huge dimensionality and a highly sparse object representation. In all the cases, the very same algorithms were used and only a few parameters needed some simple adjustments in order to provide the high degree of accuracy achieved.

The impact of one of these parameters, the slope of the sigmoid function (β), has been studied in Section 5.3. It has been found that the algorithm performs reasonably well for a wide range of values around $\beta = 8$ and, in fact, this value has been generally adopted in all further experiments (except WebKB). The other tunable parameters are the learning rates μ_{ij} and ρ_i. In all the experiments presented here, these parameters have been tuned using very simple rules, based only on training-data observations, and just *two* "metaparameters" (overall learning rates for μ_{ij} and ρ_i). Overall learning rates were not observed to significantly affect the results when used separately for CW or PW. However, when used together in CPW, the relation between μ_{ij} and ρ_i may notably impact the results for tasks where it is important to properly balance prototype and class/feature weights. This dependence stems from the fact that the proposed methods only guarantee finding an (approximate) local minimum of the leaving-one-out error criterion function. Clearly, a higher learning rate for the prototypes tends to bias the minimization process toward local minima which are close to local minima of the prototype-weights manifold of the search space. Similarly, a higher learning rate for the class/feature weights may lead to different results, closer to local minima of the class/feature-weights manifold.

Therefore, this μ/ρ balance is the only significant tuning which has been proven necessary in some cases, such as in the CPW experiments reported in Section 5.5. Future research should study this issue in more detail and should investigate adequate techniques to automatically optimize the balance. Other future works should study alternative weight initialization and optimization techniques. In addition, it could be interesting to study the benefits of using a small initial β value (allowing us to learn the class distributions), and to increase this value along the succesive algorithm iterations (to finally model the class boundaries in a discriminative way). Also, suitable extensions of the approaches discussed here to learn optimal weights for $k-NN$ classifiers, rather than plain NN, could be worth exploring. Some steps in this direction appear in [17], but additional research is needed. On the other hand, other error estimator indexes can be studied, for instance, M-fold cross-validation instead of Leaving One Out.

Finally, closely related with the ideas discussed here, another promising approach we have recently been working with is worth mentioning [22], [23]. In this approach, called *Learning prototypes and Distances* (LPD), rather than using all the training data available, T, a small subset P is selected (at random, as in [28]). Then, T is used to gradient-descent train, for every $\mathbf{x} \in P$, both its *feature-and-prototype dependent weights* and the corresponding *positions (features)* themselves. As compared with the methods discussed in this paper, LPD has one more parameter to tune (the size of P), but, otherwise, it has also shown uniformly good performance in many classification tasks, with results generally similar to those reported here.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their careful reading and in-depth criticisms and suggestions. This work was partially supported by the Spanish "Ministerio de Ciencia y Tecnología" under grants TIC2003-08496-C04-02 and DPI2004-08279-C02-02.

REFERENCES

[1] C.J. Merz, P.M. Murphy, and D.W. Aha, "UCI Repository of Machine Learning Databases," Dept. of Information and Computer Science, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html. 1997.

[2] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, "Learning to Extract Symbolic Knowledge from the World Wide Web," *Proc. 15th Nat'l Conf. Artificial Intelligence,* pp. 509-516, 1998.

[3] D. de Ridder, O. Kouropteva, O. Okun, M. Pietikäinen, and R.P.W. Duin, "Supervised Locally Linear Embedding," *Proc. Joint Conf. Artificial Neural Networks and Neural Information Processing,* 2003.

[4] D. de Ridder, M. Loog, and M.J.T. Reinders, "Local Fisher Embedding," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 2, pp. 295-298, 2004.

[5] P. Devijver and J. Kittler, *Pattern Recognition. A Statistical Approach.* Prentice Hall, 1982.

[6] L. Devroye, L. Gyorfi, A. Krzyzak, and G. Lugosi, "On the Strong Universal Consistency of the Nearest Neighbor Regression Function Estimates," *Annals of Statistics,* vol. 22, pp. 1371-1385, 1994.

[7] L. Devroye, L. Gyorfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, 1996.

[8] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally Adaptive Metric Nearest Neighbor Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 9, pp. 1281-1285, Sept. 2002.

[9] F. Ferri, J. Albert, and E. Vidal, "Considerations about Sample-Size Sensitivity of a Family of Edited Nearest-Neighbor Rules," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 29, no. 4, pp. 667-672, Aug. 1999.

[10] T. Hastie and R. Tibshirani, "Discriminant Adaptive Nearest Neighbor Classification and Regression," *Advances in Neural Information Processing Systems,* vol. 8, pp. 409-415, 1996.

[11] N. Howe and C. Cardie, "Examining Locally Varying Weights for Nearest Neighbor Algorithms," *Proc. Second Int'l Conf. Case-Based Reasoning,* pp. 455-466, 1997.

[12] R. Kohavi, P. Langley, and Y. Yung, "The Utility of Feature Weighting in Nearest-Neighbor Algorithms," *Proc. Ninth European Conf. Machine Learning,* 1997.

[13] I. Kononenko, "Estimating Attributes: Analysis and Extensions of RELIEF," technical report, Faculty of Electrical Eng. and Computer Science, Univ. of Ljubjana, 1993.

[14] J. Koplowitz and T. Brown, "On the Relation of the Performance to Editing in Nearest Neighbor Rules," *Pattern Recognition,* vol. 13, no. 3, pp. 251-255, 1981.

[15] M. Loog and R.P.W. Duin, "Linear Dimensionality Reduction via a Heteroscedastic Extension of LDA: The Chernoff Criterion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 6, pp. 732-739, June 2004.

[16] K. Nigam, J. Lafferty, and A. McCallum, "Using Maximum Entropy for Text Classification," *Proc. IJCAI-99 Workshop Machine Learning for Information Filtering,* pp. 61-67, 1999.

[17] R. Paredes, "Técnicas para la Mejora de la Clasificación por el Vecino Más Cercano," PhD thesis, Dept. de Sistemas Informátics y Computación, Univ. Politécnica de València, Spain, 2003.

[18] R. Paredes and E. Vidal, "Weighting Prototypes. A New Editing Approach," *Proc. 15th Int'l Conf. Pattern Recognition,* vol. 2, pp. 25-28, Sept. 2000.

[19] R. Paredes, A. Sanchis, E. Vidal, and A. Juan, "Utterance Verification Using an Optimized k-Nearest Neighbor Classifier," *Proc. Eighth European Conf. Speech Comm. and Technology,* 2003.

[20] R. Paredes and E. Vidal, "A Class-Dependent Weighted Dissimilarity Measure for Nearest Neighbor Classification Problems," *Pattern Recognition Letters,* vol. 21, pp. 1027-1036, Nov. 2000.

[21] R. Paredes and E. Vidal, "Learning Prototypes and Distances (LPD). A Prototype Reduction Technique Based on Nearest Neighbor Error Minimization," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 3, pp. 442-445, 2004.

[22] R. Paredes and E. Vidal, "Learning Weighted Metrics to Minimize Nearest-Neighbor Error Estimation," technical report, Dept. de Sistemas Informáticos y Computación, Univ. Politécnica de Valencia, Spain, 2004.

[23] R. Paredes and E. Vidal, "Learning Prototypes and Distances: A Prototype Reduction Technique Based on Nearest Neighbor Error Minimization," *Pattern Recognition,* vol. 39, no. 2, pp. 180-188, 2006.

[24] R. Paredes, E. Vidal, and D. Keysers, "An Evaluation of the WPE Algorithm Using Tangent Distance," *Proc. Int'l Conf. Pattern Recognition,* pp. 48-51, 2002.

[25] J. Peng, D.R. Heisterkamp, and H. Dai, "Adaptive Quasiconformal Kernel Nearest Neighbor Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 5, May 2004.

[26] C. Penrod and T. Wagner, "Another Look at the Edited Nearest Neighbor Rule," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 7, pp. 92-94, 1977.

[27] S. Raudys and A. Jain, "Small Sample Effects in Statistical Pattern Recognition: Recommendations for Practitioners," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 3, pp. 252-264, Mar. 1991.

[28] F. Ricci and P. Avesani, "Data Compression and Local Metrics for Nearest Neighbor Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 4, pp. 380-384, Apr. 1999.

[29] L.K. Saul and S.T. Roweis, "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds," *J. Machine Learning Research,* vol. 4, pp. 119-155, 2003.

[30] R.E. Schapire, Y. Freund, P. Barlett, and W.S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *The Annals of Statistics,* vol. 26, no. 5, pp. 1651-1686, 1998.

[31] R. Short and K. Fukunaga, "A New Nearest Neighbor Distance Measure," *Proc. Fifth IEEE Int'l Conf. Pattern Recognition,* pp. 81-86, 1980.

[32] C. Stanfill and D. Waltz, "Toward Memory-Based Reasoning," *Comm. ACM,* vol. 29, pp. 1213-1228, 1986.

[33] *Machine Learning, Neural and Statistical Classification*, D. Michie, D.J. Spiegelhalter, C.C. Taylor, eds, Ellis Horwood, 1994, data sets available from http://www.liacc.up.pt/ML/statlog/datasets.html.

[34] I. Tomek, "An Experiment with the Edited Nearest Neighbor Rule," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 6, no. 2, pp. 121-126, 1976.

[35] D. Vilar, H. Ney, A. Juan, and E. Vidal, "Effect of Feature Smoothing Methods in Text Classification Tasks," *Proc. Fourth Int'l Workshop Pattern Recognition in Information Systems,* pp. 108-117, 2004.

[36] D. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 2, pp. 408-421, May/June 1972.

[37] D. Wilson and T.R. Martinez, "Value Difference Metrics for Continously Valued Attributes," *Proc. Nat'l Conf. Artificial Intelligence,* pp. 11-14, 1996.

[38] Y. Yang and J.O. Pederson, "Feature Selection in Statistical Learning of Text Categorization," *Machine Learning: Proc. 14th Int'l Conf. Machine Learning,* pp. 412-420, 1997.

Roberto Paredes received the PhD degree in computer science in 2003 from the Universidad Politécnica de Valencia, Spain. From 1998 to 2000, he was with the Instituto Tecnológico de Informática working on computer vision and pattern recognition projects. In 2000, he joined the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia (UPV), where he is an assistant professor on the Facultad de Informática. His current fields of interest include statistical pattern recognition and biometric identification. In these fields, he has published several papers in journals and conference proceedings. Dr. Paredes is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI).

Enrique Vidal received the Doctor en Ciencias Fisicas degree in 1985 from the Universidad de Valencia, Spain. From 1978 to 1986, he was with this university working on computer system programming and teaching positions. In the same period, he coordinated a research group in the fields of pattern recognition and automatic speech recognition. In 1986, he joined the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia (UPV), where he is a full professor on the Facultad de Informática. In 1995, he joined the Instituto Tecnológico de Informática, where he has been coordinating several projects on pattern recognition and machine translation. He is coleader of the Pattern Recognition and Human Language Technology Group of UPV. His current fields of interest include statistical and syntactic pattern recognition and their applications to language, speech, and image processing. In these fields, he has published more than 100 papers in journals, conference proceedings, and books. Dr. Vidal is a member of the IEEE Computer Society, the Spanish Society for Pattern Recognition and Image Analysis (AERFAI), and a fellow of the International Association for Pattern Recognition (IAPR).