BoostML: An Adaptive Metric Learning for Nearest Neighbor Classification

Nayyar Abbas Zaidi[1], David McG. Squire[1], and David Suter[2]

[1] Clayton School of IT Monash University, Clayton VIC 3800, Australia
[2] School of Computer Science University of Adelaide,
North Terrace SA 5005, Australia
{nayyar.zaidi,david.squire}@infotech.monash.edu.au,
david.suter@adelaide.edu.au

Abstract. A Nearest Neighbor (NN) classifier assumes class conditional probabilities to be locally smooth. This assumption is often invalid in high dimensions and significant bias can be introduced when using the nearest neighbor rule. This effect can be mitigated to some extent by using a locally adaptive metric. In this work we propose an adaptive metric learning algorithm that learns an optimal metric at the query point. We learn a distance metric using a feature relevance measure inspired by boosting. The modified metric results in a smooth neighborhood that leads to better classification results. We tested our technique on major UCI machine learning databases and compared the results to state of the art techniques. Our method resulted in significant improvements in the performance of the K-NN classifier and also performed better than other techniques on major databases.

Keywords: Adaptive Metric Learning, Nearest Neighbor, Bias-Variance analysis, Curse-of-Dimensionality, Feature Relevance Index.

1 Introduction

Nearest Neighbor (NN) methods for pattern recognition are widely applicable and have proven to be very useful in machine learning. Despite their simplicity, their performance is comparable to other, more sophisticated, classification and regression techniques. A nearest neighbor classifier works by assigning to a query point the label of the majority class in its neighborhood. Their only assumption is smoothness of the target function to be estimated. Therefore each point in the neighborhood votes for the prediction based on its distance from the query point.

The performance of a nearest neighbor classifier depends critically on two major factors: (a) the distance metric used and (b) K, size of the neighborhood. K denotes the number of nearest neighbors. The size of the neighborhood controls the smoothness of the predicted function and is usually tuned through cross-validation. A small K implies small bias but high variance, and vice-versa. Typical 'Metric Learning' algorithms aim at finding a metric that results in small intra-class and large inter-class distances [1,2]. 'Metric Learning' has also been

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 142–149, 2010.

introduced as a bias reduction strategy in high dimensions [3]. In this paper we focus on the latter version, that is optimizing a distance metric to reduce bias. Our goal is an optimal metric that depends on the problem at hand, as characterized by the respective class distribution and, within a given problem, on the location of the query point in that space.

Bias can be introduced due to a variety of reasons. The primary reason for the introduction of bias is the 'curse-of-dimensionality' (COD) effect. Let us consider training data of size N drawn from a uniform distribution in a p-dimensional unit hypercube. The expected diameter of a $K = 1$ neighborhood using Euclidean distance is $d_1(p, N) = 2\left(\frac{p\Gamma(p/2)}{2\pi^{p/2}N}\right)^{1/p}$. It can be seen that even for a moderate number of dimensions, a very large amount of training data is required to make even a $K = 1$ nearest neighborhood relatively small. This has the consequence that the bias can be large even for the smallest possible value of K. Bias can be reduced by learning a metric that gives no influence to the irrelevant features (feature selection). This removes irrelevant features, thereby reducing the dimensionality. This in turn reduces the diameter of the K-NN neighborhood hence lowering the bias.

In order to understand why metric learning can improve classification performance, we need to analyze the classification performance itself. In a classification scenario, all that is required to obtain an optimal decision is that largest estimated class probability correspond to the class with the largest true probability, irrespective of its actual value or the values (or the order) of the estimated probabilities for the other classes, as long as the following equation holds: $\forall j : \max \hat{f}_j(x) = \max f_j(x)$ where j is class index, $\hat{f}$ is the estimated probability, and f is the unknown true probability. It can often be the case that bias, though very large, affects each of $\hat{f}_j(x)$ in same way so that their order relation is similar enough to that of true underlying probabilities $\{f_j(x)\}_1^J$ to realize an optimal class assignment [3]. But the requirement of a large neighborhood for high dimensions and the presence of irrelevant features can affect bias differentially for the respective class probability estimates enough to cause non-optimal assignment, therefore decreasing classification performance. This differential bias can be reduced by taking advantage of the fact that the class probability functions may not vary with equal strength or in the same manner in all directions in the measurement space. Elongating a neighborhood in the directions in which class probabilities do not change and constricting along those dimensions where class probabilities do change—by choosing an appropriate metric—not only reduces bias, but will also result in smoother class conditional probabilities in the modified neighborhood, resulting in better classification performance (refer to figure 1).

In this paper we propose a technique for local adaptive metric learning to reduce bias in high dimensions. As will be discussed in section 2, current work in adaptive metric learning determines feature relevance at a query point using some numerical index. This index gauges the relevance of a feature and controls the form of the metric around the query point. Our proposed index is inspired by work in the field of boosting [4], where at each iteration data is partitioned across

the most discriminative dimension. The index is based on the logit-transform of the class probability estimate. In our work using this index, we pick the dimension that is most discriminative. This is similar to 'boosting classifiers' where at each iteration a feature is selected on which the data can be classified most accurately based on the weight distribution.

2 Related Work

Friedman in [3] proposed a technique for reducing bias in high dimensional machine learning problems. Our work is inspired by this paper. The main difference of our work from [3] is feature relevance determination at each step. We have used a measure inspired by the boosting literature, whereas in [3] a GINI-like (entropy-based) index is used for feature relevance. In our work a feature is deemed more relevant if it is more discriminatory, but in [3] a feature is considered relevant if the class label varies the most.

In [5], Hastie and et al. proposed an adaptive metric learning algorithm (DANN) based on linear discriminant analysis. A distance metric is computed as a product of properly weighted within- and between-class sum-of-squares matrices. The major limitation of their method is that, in high dimensions there may not be sufficient data to fill in $p \times p$ within-class sum-of-square matrices (due to sparsity). In our work we estimate only the diagonal terms of the matrix.

Some other notable techniques for adaptive metric learning are proposed in [6,7,8,9]. In [6] an algorithm is proposed for adaptive metric learning based on analysis of the chi-squared distance. Similarly, an algorithm for metric learning has been proposed in [7] that uses SVM-based analysis for feature relevance. A similar, but slightly modified, method for metric learning based on SVMs is proposed in [9]. As will be discussed in section 3, our method differs from these methods in the sense that it is recursive. We recursively home in around the query point and the estimated metric is modified iteratively. In the above mentioned methods, however, a metric is estimated in a single cycle.

Other research on query-sensitive metric methods includes Zhou et al. [10], who investigated a query-sensitive metric for content-based image retrieval.

3 Approach

In this section we describe our two algorithms, BoostML1 and BoostML2, for adaptive metric learning. BoostML2 is a variant of BoostML1. In the following discussion we will denote the query point by x_0 and training points by x_n, where $n = [1, \ldots, N]$, and N is the number of training data. P denotes the number of features, and x_{0p} and x_{np} denote the value at the pth feature of the x_0 and x_n data points respectively.

3.1 Feature Relevance

We start by describing our local feature relevance measure technique. The feature used for splitting is the one that maximizes the estimated relevance score

$p(x_0)$ as evaluated at query point x_0. The estimate of relevance is: $p^*(x_0) = \text{argmax}_{1 \le p \le P}\ c_p(x_0)$ where $c_p(x_0)$ is defined as:

$$c_p(x_0) = I_p(x_{0p}) / \sum_{p=1}^{P} I_p(x_{0p}) \tag{1}$$

and $I_p(x_0)$ is defined in the following equation:

$$I_p(x_0) = \sum_{c=1}^{C} \text{abs}\left(\frac{1}{2}\ln\left(\frac{\Pr(c|x_{np} = x_{0p}) + \epsilon}{\Pr(c|x_{np} \neq x_{0p}) + \epsilon}\right)\right) \tag{2}$$

The ϵ in equation 2 is introduced for numerical tractability. Small I_p (close to zero) implies that there is an equal split of positive and negative training data points in the neighborhood of x_0, whereas large I_p implies that one class dominates the other class. The computation of $\Pr(c|x_{np} \neq x_{0p})$ in equation 2 is not trivial, as we may not have sufficient data in the neighborhood of the query point to accurately define the probability. The probabilities in equation 2 are computed as in equation 3.

$$\Pr(c|x_{np} = x_{0p}) = \frac{\sum_{x_n \in N(x_0)} 1(|x_{np} - x_{0p}| \le \delta_p) 1(y_n = c)}{\sum_{x_n \in N(x_0)} 1(|x_{np} - x_{0p}| \le \delta_p)} \tag{3}$$

A small neighborhood around query point x_0 denoted by $N(x_0)$ is defined and a value of δ_p is chosen to make sure that the neighborhood contains L points:

$$\sum_{n=1}^{N} 1(|x_{np} - x_{0p}| \le \delta_p) = L \tag{4}$$

In other words, we look for L points that are close to the query point on feature p and compute the probabilities in equation 2 using these points. The output of feature relevance analysis is a $p \times p$ diagonal matrix, the diagonal terms of which are the estimated relevances of the features. Based on equation 1 we can write the distance metric as a matrix A for local relevance (equation 5). This local metric is used to measure distances.

$$A(x_0) = \begin{pmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_p \end{pmatrix} \tag{5}$$

3.2 Details of the Algorithm

Given a query point x_0 and training data $\{(x_n, y_n)\}_{n=1}^{N}$, the goal is to estimate the label of the query point. Our method starts by initializing the neighborhood of the query point to be the entire measurement space (R_0). The neighborhood

is split in two based on the feature that maximizes the relevance score. Thus for the same training data, different features can be selected for this split for different query points x_0 based on the relevance of that feature at that location in the input measurement space. The split divides the input measurement space into two regions: $R_1(x_0)$, that contains the query point and the M_1 training points that are closest to it on the chosen feature, and other (complement) region $R_2(x_0)$, that contains the $N - M_1$ points that are farthest from x_0 on that feature. $R_2(x_0)$ is removed from further consideration. Thus the result of the split is just one region, $R_1(x_0)$. The above procedure is then applied again on region $R_1(x_0)$. We have named this method BoostML1 and its outline is given in algorithm 1. Refer to figure 1 for an illustration of BoostML1 algorithm.

Algorithm 1. BoostML1: Local Adaptive Metric Learning Algorithm

Require: Testing data: x_0, Training data: $\{x_n, y_n\}_{n=1}^{N}$, k : Number of elements in final neighborhood, α: Stepping size. Initialize $K = N$ and A as a p dimensional unit matrix. $N_K(x_0)$ denotes neighborhood of x_0 consisting of K points.

while flag **do**
- Find all x in $N_K(x_0)$.
- Get Feature Relevance index $c_p(x_0)$ at x_0 (equation 1), update A (equation 5).
- Choose feature $r = \text{argmax}_p c$.
- Modify neighborhood by setting $K = \alpha K$.
- Find all x in $N_K(x_0)$ using feature r. (Note: In case of BoostML2, metric A is used to find K neighbors)

if $N_K(x_0) < k$ **then**
flag = false.
end if
end while
- Do K-NN classification in the final neighborhood of k points using metric A.

As can be seen in algorithm 1, the splitting procedure is recursively applied until there are only k training observations left in the final neighborhood. The metric A (equation 5) obtained at the final step is used to measure the distance to the k nearest neighbors that predict the label of query point. At each step, a region is split on the feature that is estimated to be most relevant in terms of capturing the variation of the target functions within that region. All diagonal terms of the A matrix in equation 5 are ignored except the one with the maximum value, which is retained to split the region at each step. This is a greedy approach which is not necessarily effective all the time. BoostML2 is a variant of the above method, but at every iteration it splits the region based on the metric defined by matrix A in equation 5, as computed in the current iteration. As will be shown in section 4, BoostML2 is an improvement on algorithm 1 and results in an improvement in classification performance.

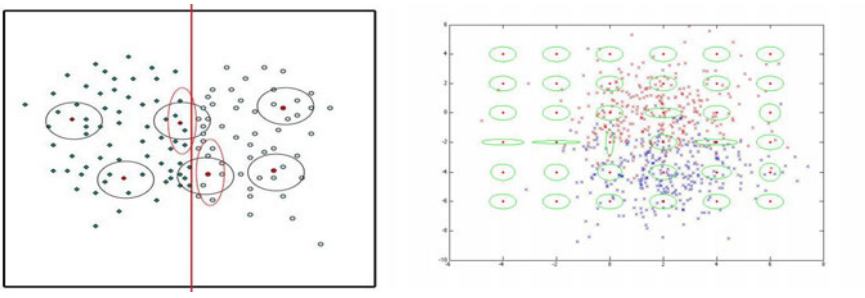

Fig. 1. Left: Illustration of Adaptive Metric Learning on synthetic two class 2-D data. Round curves depicts the Euclidean metric. The assumption of isotropy is not valid near class boundaries and a modified metric depicted as elliptical curves seems more accurate, Right: Demonstration of our algorithm BoostML on synthetic two class 2-D data. Green ellipses shows the learnt metric at different points in the measurement space.

4 Experimental Results

In this section we show the results of our adaptive metric learning algorithm on some well known databases from UCI Machine Learning Repository [11]. Databases were selected such that the competing techniques perform best on at least one of the databases. The other competing local adaptive metric learning techniques against which we tested our algorithms are as follows:

- **k-NN:** k nearest neighbor classifier with Euclidean distance.
- **DANN:** Discriminative Adaptive Nearest Neighbor classifier based on [5] as described in section 2.
- **ADAMENN:** Adaptive metric nearest neighbor classification technique based on chi-squared analysis as implemented in [6].
- **Machette:** Recursive partitioning algorithm as described in [3].
- **Scythe:** This is a generalization of the Machette algorithm in which features influence each split in proportion to their estimated local relevance, in contrast to the 'winner-takes-all' strategy of Machette.
- **BoostML1:** Algorithm 1. The implementation details regarding tuning of input parameters are described in following discussion.
- **BoostML2:** Variant of BoostML1 as described in section 3.2.

To obtain error rates, we used leave-one-out cross-validation for the Iris, Ionosphere, Dermatology, Echocardiogram and Heart data sets. 10 rounds of two-fold cross-validation were used for the Credit and Diabetes data sets.

Our metric learning algorithm results in an improvement of k-NN classification. This improvement, however, does come at an extra cost. BoostML1 has introduced two new tuning parameters. The value of L in equation 4 determines bias-variance trade-off but does not effect the performance provided it is neither too small nor too large. A value of 20 for L was used in our experiments. The

Table 1. Average classification error rates for different techniques across various databases, $k = 10$

	Iris 150,4,3	Ionosphere 351,34 2	Dermatology 358,34,6	Credit 653,15,2	Echocardiogram 61,12,2	Heart 270,13,2	Sonar 208,60,2	Diabetes 768,8,2
K-NN	4.66	16.52	3.35	13.47	8.19	20	23.07	25.91
DANN	4.66	12.53	3.35	14.09	6.55	**17.4**	**13.46**	26.17
ADAMENN	4.66	15.95	5.58	15.15	10.11	21.48	18.75	26.56
Machete	4	12.53	3.07	16.23	24.59	24.81	21.15	29.55
Scythe	4	16.8	**2.51**	15.62	9.83	19.62	19.23	**23.43**
BoostML1	**3.333**	**8.83**	2.79	15.62	18.03	23.33	20.67	29.16
BoostML2	**3.333**	11.68	3.07	**13.32**	**4.91**	19.25	18.75	25.13

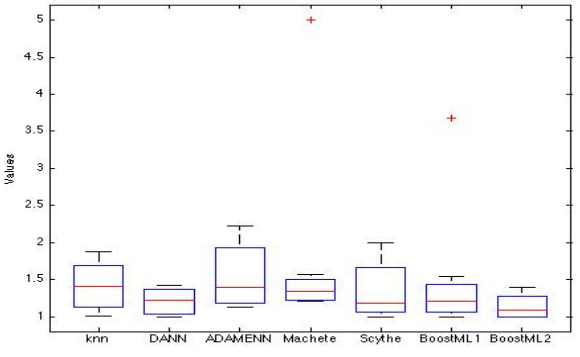

Fig. 2. Box plots for various techniques

α parameter which controls the size of the neighborhood at each step is critical to the performance. A large value of α results in a better performance at an increased computational cost. A small value of α results in poorer performance, but will be faster. A tradeoff has to be achieved between computational cost and performance. A value of 0.8 is used in all our experiments.

Table 1 shows the average classification error rates for different techniques. Each database's name is followed by number of data, features and classes. It can be seen that BoostML1 and BoostML2 perform well on the majority of data sets. It results in significant improvement over the performance of the basic k-NN classifier, and also performs better than the competing algorithms in some cases.

To compare the robustness of our algorithm with other algorithms we used the technique described in [3]. This test measures how well a particular method m performs on average in situations that are most favorable to other procedures. Robustness can be measured by computing the ratio b_m of its error rate e_m and the smallest error rate over all other methods that are compared in that example. That is:

$$b_m = \frac{e_m}{\min_{1 \leq k \leq 7} e_k} \tag{6}$$

The best method m^* will have $b_m^* = 1$ and all other methods will have values larger than 1. The larger the value of b_m the worse the performance is of the m^{th} method in relation to the best one for that data set. Figure 2 shows the distribution of b_m for each method over all data sets considered. BoostML2 turned out to be most robust among all the methods, with DANN coming second.

5 Conclusion

In this work we introduced an adaptive metric learning algorithm based on an index inspired by work on boosting for reducing bias in high dimensional spaces. We tested our algorithm on a variety of well-known machine learning databases and found that our system performs better than several well known techniques for adaptive metric learning. This improvement, however, comes at an extra cost. Our algorithm is computationally expensive compared to simple k-NN. We had to introduce two new parameters, the values of which should be optimized. Though this complicates matters, other competing algorithms also have one or more tuning parameters, so it should not be taken as a major drawback of our algorithm.

References

1. Davis, J., Dhillon, I.: Structured metric learning for high dimensional problems. In: KDD (2008)
2. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: NIPS (2005)
3. Friedman, J.: Flexible metric nearest neighbor classification. Tech Report, Dept. of Statistics, Stanford University, Tech. Rep. (1994)
4. Shapire, R., Singer, Y.: Improved boosting algorithms using confidence rated predictions. In: Conf. on Computational Learning Theory (1998)
5. Hastie, T., Tibshirani, R.: Discriminative adaptive nearest neighbor classification. IEEE transactions on Pattern Analysis and Machine Intelligence (1996)
6. Domenciconi, C., Peng, J., Gunopulos, D.: An adaptive metric machine for pattern classification. In: NIPS (2000)
7. Peng, J., Heisterkamp, D., Dai, H.: Lda/svm driven nearest neighbor classification. In: CVPR (2001)
8. Janusz, K. (ed.): Support Vector Machines: Theory and Applications. Springer, Heidelberg (2005); ch. Adaptive Discriminant and Quasiconformal Kernel Nearest Neighbor Classification
9. Domenciconi, C., Peng, J., Gunopulos, D.: Large margin nearest neighbor classifiers. IEEE transactions on Pattern Analysis and Machine Intelligence (2005)
10. Zhou, Z., Dai, H.: Query-sensitive similarity measure for content-based image retrieval. In: ICDM (2006)
11. Mertz, C., Murphy, P.: Machine learning repository (2005), `http://archive.ics.uci.edu/ml/`