BoostML: An Adaptive Metric Learning for Nearest Neighbor Classification

Nayyar A. 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. The nearest neighbor classification/regression technique, besides its simplicity, is one of the most widely applied and well studied techniques for pattern recognition in machine learning. A nearest neighbor 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 present a detailed analysis of the introduction of bias in high dimensional machine learning data and propose an adaptive metric learning algorithm that learns an optimal metric at the query point using respective class distributions on the input measurement space. 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.

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

1 Introduction

Nearest neighbor methods for pattern recognition have proven to be very useful in machine learning. Despite their simplicity, their performance is comparable to other, more sophisticated, classification and regression techniques and they have been applied to a great variety of problems. Computer vision research has benefited greatly from advancements in nearest neighbor methods, for example some state of the art techniques for object recognition are based on nearest neighbor analysis [1, 2]. Given a query point, a nearest neighbor classifier works by assigning to it the label of the majority class in its neighborhood. In order to obtain a smooth boundary, each point in the neighborhood votes for the prediction based on its distance from the query point [3].

The performance of a nearest neighbor classifier depends critically on two major factors: (a) the distance metric used and (b) size of the neighborhood K. The size of the neighborhood that controls the smoothness of the predicted function is usually tuned through cross-validation [3].[3] In the current research on nearest neighbor methods, a dichotomy exists here. Typical 'Metric Learning' algorithms aim at finding a metric that results in small intra-class and large inter-class distances [4–7]. 'Metric Learning' has also been introduced as a bias reduction strategy in high dimensions [8]. 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 on the input measurement space and, within a given problem, on the location of the query point in that space.

The effectiveness of nearest neighbor-based methods stems from their asymptotic properties. The asymptotic results in [9, 10] suggests that a 1-NN method based on simple Euclidean distance will perform well provided the training sample is not too small. These asymptotic results are based on the fact that bias in the prediction of function $f(x)$ becomes vanishingly small. If the number of training data N is large with few features p, these asymptotic results hold. Typical machine learning data sets, however, have a large p and the N required to validate these asymptotic results is not feasible because of the 'curse-of-dimensionality'.

Equation 1 shows the well-know phenomenon of the 'curse-of-dimensionality' 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 shown. It can be seen that even for moderate number of dimensions, a very large number 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 thus smallest possible value of K, invalidating the asymptotic results.

$$d_1(p, N) = 2\left(\frac{p\Gamma(p/2)}{2\pi^{p/2}N}\right)^{1/p} \tag{1}$$

Metric learning can be employed to mitigate the effects of the 'curse-of-dimensionality'. There are several motivating reasons for using metric learning for this purpose:

– Bias can be reduced by learning a metric that gives no influence to the 'irrelevant' features. This removes irrelevant features thereby reducing the dimensionality. This in turn reduces the diameter in equation 1 of the K-NN neighborhood, hence lowering the bias.

[3] The size of the neighborhood around query point x is specified by K, which denotes the number of nearest neighbors of x, and a choice of distance measure which in turn is determined by a norm and a metric defined by a positive semi-definite matrix. A small K implies small bias but high variance, and vice-versa.

One of the other reasons for optimizing the distance metric has to do with the heterogeneous nature of features. Since different features come from different sources and are of different natures, their input/votes into the classification procedure may not all be equally relevant for classifying a new object. It is important to note that this relevance may depend on the location of object in the input measurement space. A feature relevant at one location may be irrelevant at an other location. It is well known that features having low relevance can degrade performance if they are allowed to be equally influential with those of high relevance in defining the distance from the point to be classified [11].

The requirement of a large neighborhood (equation 1) can affect bias differentially for the respective class probability estimates. This differential bias can be reduced by taking advantage of the fact that the class probability functions may not vary locally with equal strength, or in the same manner, across all features in the measurement space around the prediction point x_0. Bias can be reduced by choosing a metric so that the resulting neighborhood elongates in directions for which class probabilities do not change much and is constricted along dimensions where class probabilities changes. This will make class conditional probabilities smooth in the modified neighborhood and will result in better classification performance (refer to figure 1).

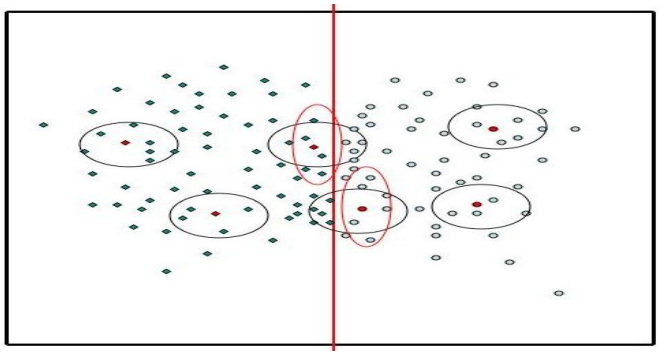

Fig. 1. Illustration of Adaptive Metric Learning on 2-D data. Data belongs to two classes which are well separated by a linear classifier. Round curves depicts the Euclidean metric. As can be seen, this assumption of isotropy is not valid near class boundaries and a modified metric depicted as elliptical curves seems more accurate.

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 relevancy of a feature and controls the form of metric around the query point. Our proposed index is inspired from work in the area of boosting [12], 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.

The rest of the paper is organized as follows. In section 2 we discuss related work and compare our method with related techniques. We discuss our algorithm in section 3. Experimental results are given in section 4. We conclude in section 5, and suggest directions for future work.

2 Related Work

The earliest work in adaptive metric learning can be traced back to Friedman [8]. In this work Friedman proposed a technique for reducing bias in high dimensional machine learning problems. It is based on learning a local metric around a query point by recursively splitting the neighborhood based on a feature relevance index and learning a new metric on this modified neighborhood. Our work is inspired by this paper. The main difference of our work from [8] is feature relevance determination at each step. We have used a measure inspired by the boosting literature, whereas in [8] a GINI-like (entropy-based) index is used for feature relevance. Another difference is that in our work a feature is deemed more relevant if it is more discriminatory but in [8] a feature is considered relevant if the class label varies the most.

In [13], Hastie and Tibshirani propose an adaptive metric learning algorithm based on linear discriminant analysis (LDA). A distance metric is computed as a product of properly weighted within and between sum-of-squares matrices. The authors also revealed the connection of their resulting method to the approximation of the chi-squared distance by a Taylor series expansion. Though sound in theory, the method has limitations. The major limitation is that in high dimensions we may not have sufficient data to fill in $p \times p$ within class sum-of-square matrices (due to sparsity). They found it more effective to estimate only the diagonal terms of within class sum-of-square matrices and assume that the off-diagonal terms are zero. This is especially true if the dimensionality of the input space is large, as there will be insufficient data locally to estimate the $\Theta(p^2)$ elements of within class sum of square matrices. In our work, similar to [8], we estimate only the diagonal terms of metric.

Some other notable techniques for adaptive metric learning are proposed in [14–17]. In [14] an algorithm is proposed for adaptive metric learning based on the analysis of the chi-squared distance. An algorithm for metric learning has been proposed in [15] which uses SVM-based analysis for feature relevance. A similar but slightly modified method for metric learning based on SVMs is proposed in [17]. 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 above mentioned methods, however, a metric is estimated in a single cycle.

3 Approach

In this section we describe our two algorithms, BoostML1 and BoostML2, for adaptive metric learning. BoostML2 is a variant of BoostML1.

3.1 Feature Relevance

We start by describing our local feature relevance measure technique. In the following discussion we will denote the query point by x_0 and the training points by x_n where $n = [1, ...N]$, N is the number of training data, while P denotes number of features. Also x_{0p} or x_{np} denotes the value at the pth feature of the x_0 and x_n data points respectively. The feature used for splitting is the one that maximizes the estimated relevance score (equation 2) as evaluated at query point x_0. The estimate of relevance is:

$$p^*(x_0) = \text{argmax}_{1 \leq p \leq P} \; c_p(x_0), \tag{2}$$

where $c_p(x_0)$ is defined as

$$c_p(x_0) = \frac{I_p(x_{0p})}{\sum_{p=1}^{P} I_p(x_{0p})}. \tag{3}$$

Equation 3 normalizes the weights $I_p(x_0)$ defined in equation 4.

$$I_p(x_0) = \sum_{c=1}^{C} \text{abs}\left(\frac{1}{2}\ln\left(\frac{\Pr(c|x_{np} = x_{0p}) + \epsilon}{\Pr(c|x_{np} \neq x_{0p}) + \epsilon}\right)\right) \tag{4}$$

The ϵ in equation 4 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 4 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 4 are computed as in equation 6. We define a small neighborhood around query point x_0 denoted by $N(x_0)$ and make sure that it contains some number δ_p of points (refer to equation 5). In other words we look for δ_p points that are close to the query point on feature p and compute the probabilities in equation 4 on these points.

$$\sum_{n=1}^{N} 1(|x_{np} - x_{0p}| \leq \delta_p) 1(y_n = c) = L \tag{5}$$

$$\Pr(c|x_{np} = x_{0p}) = \frac{\sum_{x_n \in N(x_0)} 1(|x_{np} - x_{0p}| \leq \delta_p) 1(y_n = c)}{\sum_{x_n \in N(x_0)} 1(|x_{np} - x_{0p}| \leq \delta_p)} \tag{6}$$

It is important to consider the output of feature relevance analysis as a $p \times p$ diagonal matrix, the diagonal terms of which are the estimated relevances of the features. Based on equation 2 we can write the distance metric as a matrix A for local relevance (as shown in equation 7). This local metric is used to measure distances.

$$A(x_0) = \begin{pmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_p \end{pmatrix} \tag{7}$$

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 a query point that can belong to any one of C classes. Our method starts by initializing the neighborhood of the query point to be the entire measurement space (R_0). Similar in essence to decision trees, the neighborhood is split in two on one of the features. The feature used for splitting is the one that maximizes the relevance score as shown in equation 2. 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. The other (complement) region $R_2(x_0)$ contains the $N - M_1$ points that are farthest from x_0 on that feature. The complement region is removed from consideration by discarding the data contained within it. Thus the result of the split is just one region $R_1(x_0)$. The above procedure is applied on region $R_1(x_0)$. We have named this method BoostML1 and its outline is given in algorithm 1.

As can be seen in algorithm 1, the splitting procedure is recursively applied until there are only L training observations left in the final neighborhood. The metric (equation 7) obtained at the final step is used to measure the distance to the K nearest neighbors that predict the label of query point. Refer to section 5 for a discussion of a non-recursive version of BoostML.

At each step, a region is split on the feature that is estimated to be most relevant in terms of capturing the variation of target functions within that region. All diagonal terms of the A matrix in equation 7 are ignored except the one with the maximum value, which is retained to split 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 7 as computed in

the current iteration. As will be shown in section 4, BoostML2 is an improvement on algorithm 1 and results in an increase in classification performance. Refer to figure 2 for illustration of BoostML algorithm.

Algorithm 1 BoostML1: Iterative algorithm for learning local adaptive metric for nearest neighbor classification.

Require: – Testing data: x_0
- Training data: $\{x_n, y_n\}_{n=1}^{N}$ where x is a p dimensional feature vector. N is the number of training data. y is training label. $y = \{1, 2, ...C\}$ where C is the number of classes.
- k : Number of nearest neighbors, L : Number of elements in final neighborhood, α: Stepping size.

- $K = N$
- Find all x in $N_K(x_0)$, where $N_K(x_0)$ denotes neighborhood of x_0 consisting of K points.
- Initialize A as a p dimensional diagonal matrix

while flag **do**
  - $c_p(x_0)$ = Get Feature Relevance index at x_0
  - $r = \text{argmax}_p c$
  - Update A by setting all diagonal terms to zero except that of r.
  - Modify neighborhood by setting $K = \alpha K$
  - Find all x in $N_K(x_0)$ using metric A
  if $N_K(x_0) < L$ **then**
    flag = false
  end if
end while

- Find k closest elements to x_0 using metric A and returns respective probabilities of class labels in the neighborhood.

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 [18]. Details of the databases are given in table 1. 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** Simple k nearest neighbor classifier based on Euclidean distance between data points.

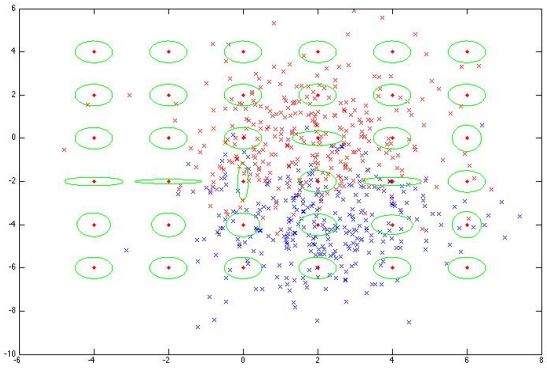

Fig. 2. Demonstration of our algorithm BoostML on synthetic data. Data consists of two classes shown in red and blue. Green ellipses shows the learnt metric at different points in the measurement space.

Databases Name	Data	Features	Classes
Iris	150	4	3
Ionosphere	351	34	2
Dermatology	358	34	6
Credit-Screeing	653	15	2
Echocardiogram	61	12	2
Statlog Heart	270	13	2
Sonar	208	60	2
Diabetes	768	8	2

Table 1. Database Details: Number of data, Number of Features and Number of Classes on Column 2,3 and 4 respectively

- **DANN** Discriminative Adaptive Nearest Neighbor classifier based on [13] as described in section 2.
- **ADAMENN** Adaptive metric nearest neighbor classification technique based on chi-squared analysis as implemented in [14].
- **Machette** Recursive partitioning algorithm as described in [8].
- **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** As described in section 3.2 and 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.

	Iris	Ionosphere	Dermatology	Credit	Echocardiogram	Heart	Sonar	Diabetes
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

Table 2. Average classification error rates for different techniques across various databases, $K = 10$, refer to the text for details

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

As can be seen, our metric learning algorithm results in an improvement of k-NN classification. This improvement, however, does come at an extra cost. As can be seen in algorithm 1, BoostML1 has introduced two new tuning parameters. The value of L in all our experiments have been set to 20. We tested with other values but there wasn't any significant improvement in the results for setting the value of L less than 20.

The 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, but will increase the 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. In this work we have not optimized this value. A value of 0.8 is set in all experiments.

Table 2 shows the average classification error rates for the different techniques for the different databases. It can be seen that BoostML1 and BoostML2 perform well on the majority of data sets. It results in significant improvement on the classification 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 [8]. This test measures that 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:

$$b_m = \frac{e_m}{\min_{1 \leq k \leq 7} e_k} \tag{8}$$

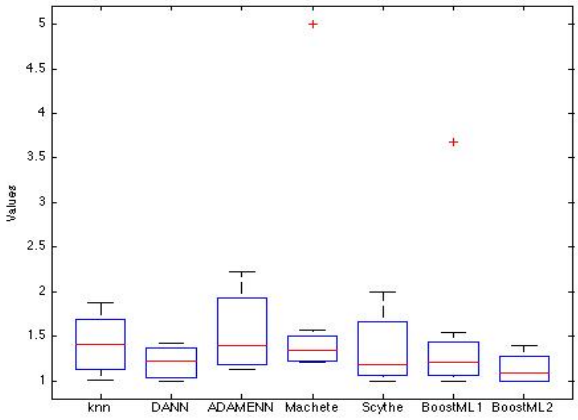

Fig. 3. Box plots for various techniques.

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 3 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 and Future Work

In this work we presented a study of distance measurement in high dimensional spaces. We showed that significant bias is introduced due to the 'curse-of-dimensionality' in high dimensions and proposed techniques to mitigate the effects of bias. We introduced an adaptive metric learning algorithm based on an index inspired by work on boosting. 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. Also, our system results in the improvement of the nearest neighbor classifier. This improvement, however, comes at an extra cost. Our algorithm is computationally expensive as 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.

Since our work is inspired by boosting, and boosting has been used for tuning distance metric [19], we designed an algorithm to use boosting for learning an adaptive distance metric. To improve the computational cost of our approach,

we are working on algorithm that, rather than recursively homing in around the query point to determine an adaptive metric, specifies a neighborhood around the query point (similar to [14]) and defines a metric by determining the relevance of features in that neighborhood alone.

References

1. A. Frome, Y. Singer, and J. Malik, "Image retrieval and classification using local functions," in *NIPS*, 2006.
2. M. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *CVPR*, 2006.
3. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.
4. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood component analysis," in *NIPS*, 2005.
5. J. Davis and I. Dhillon, "Structured metric learning for high dimensional problems," in *KDD*, 2008.
6. K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *NIPS*, 2005.
7. B. Sriperumbudar, O. Lang, and G. Lanckriet, "Metric embedding for kernel classification rules," in *ICML*, 2008.
8. J. Friedman, "Flexible metric nearest neighbor classification," Tech Report, Dept. of Statistics, Stanford University, Tech. Rep., 1994.
9. E. Fix and J. Hodges, "Discriminatory analysis - nonparameteric discrimination: consistency properties," Tech Report, Randolph Field Texas, US Airforce School of Aviation Medicine, Tech. Rep., 1951.
10. T. Cover, "Rates of convergence for nearest neighbor procedures," in *Inter. Conf. on Systems Sciences*, 1968.
11. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature Extraction, Foundation and Applications*. Springer, 2004.
12. R. Shapire and Y. Singer, "Improved boosting algorithms using confidence rated predictions," in *Conf. on Computational Learning Theory*, 1998.
13. T. Hastie and R. Tibshirani, "Discriminative adaptive nearest neighbor classification," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 1996.
14. C. Domenciconi, J. Peng, and D. Gunopulos, "An adaptivfe metric machine for pattern classification," in *NIPS*, 2000.
15. J. Peng, D. Heisterkamp, and H. Dai, "Lda/svm driven nearest neighbor classification," in *CVPR*, 2001.
16. K. Janusz, Ed., *Support Vector Machines: Theory and Applications*. Springer Berlin/ Heidelberg, 2005, ch. Adaptive Discriminant and Quasiconformal Kernel Nearest Neighbor Classification.
17. C. Domenciconi, J. Peng, and D. Gunopulos, "Large margin nearest neighbor classifiers," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2005.
18. "Machine learning repository," 2005. [Online]. Available: http://archive.ics.uci.edu/ml/
19. L. Yang, R. Suthankar, and S. Hoi, "A boosting framework for visuality-preserving distance metric learning and its applications to medical image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.