A Gradient-based Metric Learning Algorithm for k-NN Classifiers

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

[1] Clayton School of Information Technology, Monash University, 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 (NN) classification/regression techniques, besides their simplicity, are amongst the most widely applied and well studied techniques for pattern recognition in machine learning. A drawback, however, is the assumption of the availability of a suitable metric to measure distances to the k nearest neighbors. It has been shown that k-NN classifiers with a suitable distance metric can perform better than other, more sophisticated, alternatives such as Support Vector Machines and Gaussian Process classifiers. For this reason, much recent research in k-NN methods has focused on metric learning, i.e. finding an optimized metric. In this paper we propose a simple gradient-based algorithm for metric learning. We discuss in detail the motivations behind metric learning, i.e. error minimization and margin maximization. Our formulation differs from the prevalent techniques in metric learning, where the goal is to maximize the classifier's margin. Instead our proposed technique (MEGM) finds an optimal metric by directly minimizing the mean square error. Our technique not only results in greatly improved k-NN performance, but also performs better than competing metric learning techniques. Promising results are reported on major UCIML databases.

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 sophisticated classification and regression techniques, such as Support Vector Machines (SVM) and Guassian Processes (GP), and they have been applied to a wide variety of problems. For a given query point, a nearest neighbor classifier works by assigning it the label of the majority class in its neighborhood.

It is evident that the k-NN classifier's simplicity is one of its major advantages. A k-NN classifier deals with multi-class classification scenario effortlessly. In contrast, one needs one-versus-one and one-versus-all techniques to deal with multi-class scenarios when using binary classifiers such as SVM. This makes them computationally expensive. As k-NN classifiers need no training, they are computational efficient. Nevertheless, the effectiveness of k-NN methods relies

on their asymptotic properties. The asymptotic results in [1–3] suggest that a 1-NN method based on a simple Euclidean distance will perform well provided the number of training samples is not too small. Indeed 1-NN will approach the performance of a Bayes optimal classifier as the number of training data becomes very large. 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 compared to the number of features p i.e., $N \gg p$. Typical machine learning data, however, has large numbers of features, and the amount of data required to achieve these asymptotic results is unfeasibly large. This is known as the Curse-of-Dimensionality (COD). Another interpretation of the COD is that, in high dimensions, most of the data points are very far apart and k-NN neighborhoods are no longer 'local' [4, section 2.5]. Modifying distances in high dimensions can help to alleviate the COD, reduce bias and make neighborhoods local. This requires a tuned metric—and hence metric learning.

As discussed above, the performance of a nearest neighbor classifier depends critically on two factors: the distance metric used, and size of the neighborhood (specified by k, which denotes the number of nearest neighbors). The value of k controls the Mean Square Error (MSE) which is defined as MSE = bias^2 + variance. Small k implies small bias but high variance, and vice-versa. Since k is specified in terms of the number of nearest neighbors of a query point x, which implicitly depends on a distance measure, MSE can be controlled by estimating a distance metric (a metric is generally specified through a norm and a positive semi-definite matrix). Typically we estimate the inverse square root of the metric. That is, we learn a matrix parameterizing the linear transformation of the input space such that in the transformed space k-NN performs well. If we denote such a transformation by a matrix A, we are effectively learning a metric defined by A^TA such that $d(x, y) = (x - y)^TA^TA(x - y) = (Ax - Ay)^T(Ax - Ay)$.

In the current research on nearest neighbor methods, a dichotomy exists between metric learning methods in terms of their goals. Most 'Metric Learning' algorithms aim to find a metric that results in small intra-class and large inter-class distances [5–10]. This results in maximizing the margin.[3]

Figure 1 depicts a simple contrived example of data belonging to two classes represented by red and blue dots. As can be seen, the classes are linearly separable. A hyperplane is indicated by a dark black line. In this scenario, the margin can be maximized in two ways: either we modify the hyper-plane to better fit the training data, or we transform the training data to maximize the margin with respect to a certain hyperplane. The latter has been the goal of most metric learning algorithms. SVMs, on the other hand, optimize the margin by finding an optimal hyperplane. They are designed to minimize empirical risk with a bound on generalization error. Metric learning can also be used to minimize empirical risk i.e., maximize the margin by transforming the training data. Such a strategy has been introduced in [11], where metric learning was introduced as a bias reduction strategy and to reduce MSE to better fit the training data.

[3] The margin of a point is defined as the distance between the point and the closest point on the classification boundary.

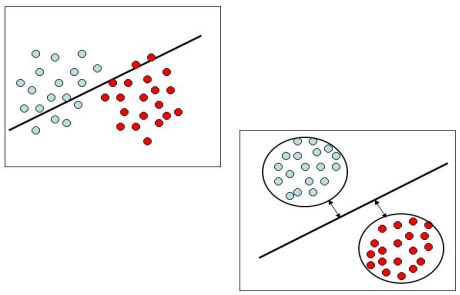

Fig. 1. Contrived example demonstrating the impact of metric on margin.

In this paper we present a novel metric learning algorithm with the goals of maximizing the margin by reducing MSE directly. We propose a simple MSE gradient minimization (MEGM - Mean square Error Gradient Minimization) approach to improve the performance of the k-NN neighbor classifier. Our method is based on gradient descent on the MSE objective function. We compare MEGM performance with other metric learning approaches for margin maximization, e.g. neighborhood component analysis (NCA). As shown in section 4, our method not only results in significant improvement in the performance of the k-NN classifier, but also outperforms other metric learning algorithm on most data-sets. As we discuss in section 5, unlike SVM, we minimize the empirical risk only. We do not address generalization in our algorithm, but in our experiments we did not experience any over-fitting. A regularization term can be easily introduced into our framework. This is left as a future work.

The rest of the paper is organized as follows: we discuss related work in section 2. Our proposed MEGM algorithm is described in detail in section 3. A detailed description of our experimental setup and comparitive results on UCIML data-sets are given in section 4. We conclude in section 5 with pointers to future work.

2 Related Work

Our proposed algorithm MEGM is very close in nature to [12] where a gradient based technique is used for selecting relevant features. That is, only diagonal terms of the covariance matrix are estimated. In our method we learn a full covariance matrix rather than estimating only diagonal terms. Thats why MEGM is superior to technique proposed in [12].

The other notable techniques for metric learning are LMNN [13], RCA [14] and NCA [5]. Relevant Component Analysis (RCA) [14] constructs a Mahalanobis distance metric from a weighted sum of in-class covariance matrices. It is similar to Principal Component Analysis (PCA) and Linear Discriminant

Analysis (LDA) in its reliance on second order statistics. Large Margin Nearest Neighbor (LMNN) algorithm in [13] is posed as a convex problem, and thus the reach of the global solution is guaranteed. However, a special optimization solver is needed for efficient implementation.

Neighborhood Component Analysis (NCA) [5] maximizes margin by minimizing the probability of error under stochastic neighborhood assignment. In particular each point i selects another point j as its neighbor with some probability p_{ij}, and inherits its class labels from the point it selects. p_{ij} is defined as a softmax over Euclidean distances in the transformed space, parameterized by matrix A:

$$p_{ij} = \frac{-\exp(\|Ax_i - Ax_j\|^2)}{\sum_{k \neq i} \exp(-\|Ax_i - Ax_k\|^2)} \tag{1}$$

NCA maximizes the p_{ij} in above equation by finding an optimal A matrix. That is the probability of the number of points correctly classified is maximized. The comparison of our proposed algorithm (MEGM) with NCA has been a major motivation of this work. Though NCA is sound in theory, our empirical results in section 4 suggests that MEGM performs better than NCA on most data-sets. We will mention in section 5 about an approach to combine both MEGM and NCA to improve MEGM's generalization capacity.

3 Approach

In a typical regression setting, an unknown function $f : R^D \rightarrow R$ is predicted from the training data $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ...(\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i$ is a data point and y is the corresponding target value. The predicted function $\hat{f}$ is chosen to be the one that minimizes some loss function such as 'mean squared error' (MSE) etc. The MSE for a data set containing N of points is given in the following equation:

$$\text{MSE}(\hat{f}) = \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - \hat{f}(\boldsymbol{x}_i))^2 \tag{2}$$

For classification task having T classes we can replace above error function as:

$$\text{MSE}(\hat{y}) = \sum_{t=1}^{T} \sum_{i=1}^{N} (y_{ti} - \hat{y}_{ti})^2 \tag{3}$$

where $\hat{y}_i$ denotes the predicted probability of point $\boldsymbol{x}_i$ and y_i denotes the true label (either 0 or 1) of point $\boldsymbol{x}_i$. For brevity we have denoted $\hat{y}(\boldsymbol{x_{ti}})$ with $\hat{y}_{ti}$ and $y(\boldsymbol{x_{ti}})$ with y_{ti}. In the following discussion we will assume that there are only two classes to make our derivations simple. For any query point $\boldsymbol{x}_i$, nearest neighbor methods work by predicting the value $\hat{y}_i$ by considering the labels of its k nearest neighbors. In order to have a smooth boundary, each neighbor votes

for the query label based on its distance from the query point (refer to [4] for details). Equation 4 shows the Nadaraya-Watson kernel for regression:

$$\hat{y}(\boldsymbol{x}) = \frac{\sum_j y_j V_j}{\sum_j V_j} \tag{4}$$

The vote V_j casted by each label around the query point $\boldsymbol{x}$ is usually chosen to be a function that decays exponentially as the distance from the query point increases, for example a Gaussian kernel:

$$V_j = \exp\left(\frac{-\mathrm{d}(\boldsymbol{x}, \boldsymbol{x_j})}{2\sigma^2}\right) \tag{5}$$

Determining votes using equation 5 assumes a well defined distance measure. This assumption, as discussed in the previous section, is not always true, due to the COD and irrelevant features, and can lead to bad results. $\mathrm{d}(\boldsymbol{x}, \boldsymbol{x_j})$ in equation 5 can be replaced by a more general metric: that is $\mathrm{d}_L(\boldsymbol{x}, \boldsymbol{x_j})$. If $L = A^T A$, then $\mathrm{d}_L(\boldsymbol{x}, \boldsymbol{x_j}) = (A\boldsymbol{x} - A\boldsymbol{x_j})^T(A\boldsymbol{x} - A\boldsymbol{x_j})$. Since MSE is a function of $\hat{y}$ and $\hat{y}$ depends on $||\boldsymbol{x} - \boldsymbol{x}_j||_L^2$, MSE can be minimized by selecting an optimal value of L. In other words, a change in the L induces a change in the distance, which can alter the votes. This alteration in the votes V_j triggers a change in $\hat{y}$ affecting the MSE. It is more helpful to optimize A rather than L, because optimization for L requires to fulfill semi-positive constraint which is expensive to maintain. Obviously trying all possible values of A is not feasible. Some sort of search mechanism is required to find an optimal value of A. Votes V_j in equation 5 can be replaced by W_j as:

$$W_j = \exp\left(\frac{-\|A\boldsymbol{x} - A\boldsymbol{x_j}\|_2^2}{2\sigma^2}\right) \tag{6}$$

The proposed gradient based technique (MEGM) is based on a gradient descent algorithm to minimize MSE (lets denote by E_A). The gradient E_A is evaluated to find an optimal A matrix. Convergence to the global minimum is not guaranteed. The risk of local minima can be reduced by running the algorithm several times and choosing the output with minimum error E_A. The gradient of E_A with respect to matrix A is:

$$\frac{\partial E}{\partial A} = (y_i - \hat{y}_i)\frac{1}{\sum_j W_j}\sum_j (y_j - \hat{y}_j)\frac{\partial W_j}{\partial A} \tag{7}$$

The size of the Gaussian kernel centered at the query point (σ in equation 6) is set proportional to the distance of the k nearest neighbors. Generally the average distance of half of the nearest neighbors is used, as this measure is more stable under a varying distance metric and in the presence of outliers:

$$\sigma^2 = \frac{1}{2}\frac{1}{P}\sum_{p=1}^{P}\|\boldsymbol{x} - \boldsymbol{x}_p\|^2 \quad \text{where } P = k/2 \tag{8}$$

$\frac{\partial W_j}{\partial A}$ in equation 7 can be derived as:

$$\frac{\partial W_j}{\partial A} = 2W_j A(\boldsymbol{x} - \boldsymbol{x}_j)(\boldsymbol{x} - \boldsymbol{x}_j)^T \tag{9}$$

Combining equations 7 and 9 we can write the gradient of E_A with respect to matrix A as:

$$\frac{\partial E}{\partial A} = 2A(y_i - \hat{y}_i)\frac{1}{\sum_j W_j}\sum_j (y_j - \hat{y}_j)W_j(\boldsymbol{x} - \boldsymbol{x}_j)(\boldsymbol{x} - \boldsymbol{x}_j)^T \tag{10}$$

Equation 10 represents the gradient of the error function with respect to matrix A which is minimized to get an optimal A. The Polack-Ribiere flavour of conjugate gradients is used to compute search directions, and a line search using quadratic and cubic polynomial approximations and the Wolfe-Powell stopping criteria is used together with the slope ratio method for guessing initial step sizes.

4 Experimental Results

In this section we present results on various machine learning databases from UCIML repository [15]. MEGM's results are compared with other metric learning approaches like NCA, RCA and LMNN. The size of neighborhood (k) as discussed in section 3 is consistently set equal to the log2(cardinality of data set) for all databases.

To obtain the final classification results, one nearest neighbor (1-NN) classification is used. As mentioned in section 3, since both NCA and MEGM suffers from local minima problems, some care has to be taken to make sure that it does not effect results. For all databases, we run MEGM and NCA thrice with different training data samples and selected the best results. In order to make sure that our results are not biased to NCA and MEGM due to this procedure, reported results for all other techniques for example k-NN, LMNN and RCA are computed this way. That is each method is run thrice using different training samples and best results are selected in each run. Percentage error rates are reported for all data-sets.

To test the performance of MEGM with other metric learning methods, we selected major UCIML databases. The error rate of each method for different databases is shown in figure 2. The number of data, features and classes for each database is reported in the title. Error rate of each method is obtained using 40 rounds of 2 fold cross-validation. The mean and standard deviation of the results are reported in the performance graphs. Prior to training, all features were normalized to have zero mean and a unit variance.

As can be seen MEGM not only improved k-NN classification performance but in most cases resulted in better performance than other metric learning techniques like NCA, RCA and LMNN. MEGM outperforms other methods on Balance and Hayesroth databases. Also it performed marginally better than

other techniques on Credit-screeing, Dermatology, Sonar, Statlog-heart, vowel and Monks2. On Monks1 and Monks3 both MEGM and NCA performs equally well and error rate is close to zero for both these methods.

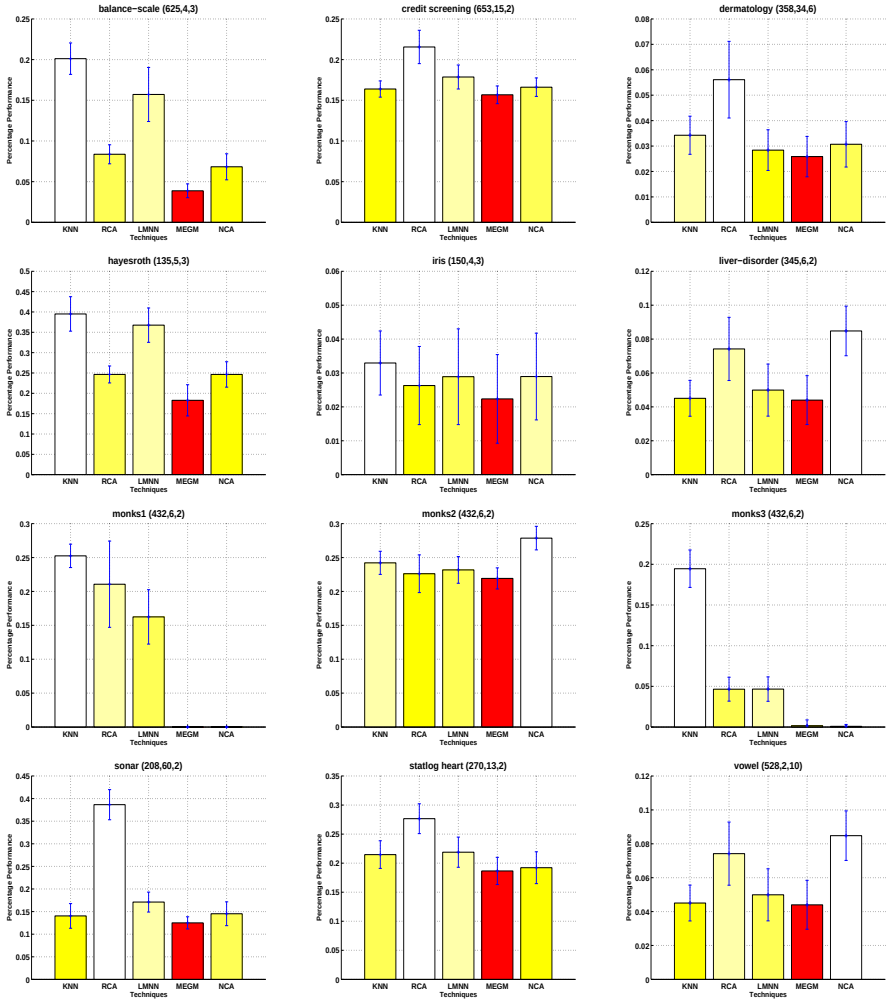

Fig. 2. Error rate comparison of various techniques on UCIML databases.

Though MEGM performed better than other approaches on most databases as shown in figure 2, NCA performance is also noteworthy especially on balance, monks1 and statlog-heart. It performed marginally better than other techniques on Ionosphere, Housevote and Hepatitis as shown in figure 3.

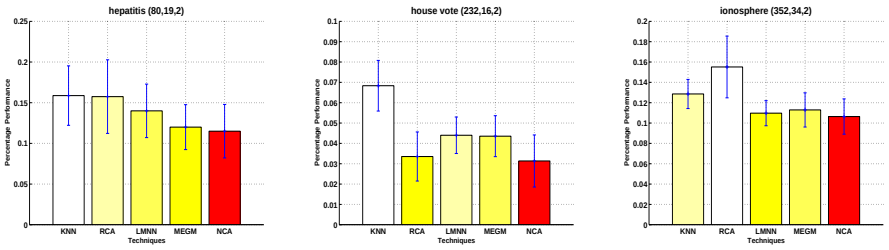

Fig. 3. Error rate comparison of various techniques on UCIML databases, NCA performs best on these data-sets.

To compare the robustness of our algorithm with other algorithms we used the technique described in [11]. 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:

$$b_m = \frac{e_m}{\min_{1 \leq k \leq 5} e_k} \tag{11}$$

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 4 shows the distribution of b_m for each method over all 15 UCIML data-sets considered. As can be seen, MEGM turned out to be the most robust of all with NCA coming second. LMNN also performs good except for the presence of outliers.

5 Conclusion

The main pro of our proposed MEGM algorithm is its simplicity. As discussed, MEGM minimizes MSE's gradient using a simple gradient descent algorithm. MEGM improves k-NN classification by learning a data dependent distance metric and performs well on most if not all databases. Also, it deals with multi-class problems effortlessly as opposed to binary classifiers like SVM where a one-versus-one and one-versus-all strategy is used. On the other hand, once a metric is learnt using MEGM, a simple nearest neighbor classification is required. In data-sets where number of classes are very large, nearest neighbor methods should be preferable for their computational efficiency. Therefore k-NN methods equipped with a proper distance metric (for example, one trained with MEGM) can be extremely useful.

A drawback of MEGM includes local minima problem. Standard approaches to avoid local minima are to be used. Also one is tempted to think of over-fitting

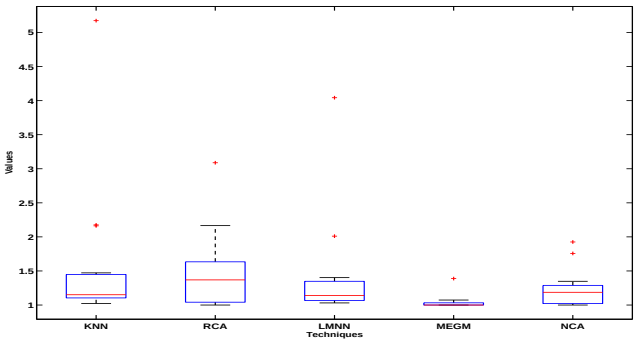

Fig. 4. Box plots depicting the comparison of robustness of different techniques on various UCIML data-sets.

if the objective function is only MSE. In this work, we did not encounter any over-fitting. As a future work, we are investigating to modify our objective function to include a generalization term, that is penalize large changes in A matrix to avoid over-fitting. We are currently investigating to combine MEGM's and NCA's objective function to improve our results. As in this study, MEGM which is based on the minimization of MSE resulted in better performance than NCA and other metric learning algorithms which maximizes margin explicitly, a natural extension to the proposed method is to combine the two approaches. That is learn a metric by simultaneously maximizing the margin and minimizing the MSE. The objective functions of MEGM and NCA is combined in the following equation:

$$E_A = \sum_{i=1}^{N} \left(y_i - \exp\left(\frac{-\|A\boldsymbol{x} - A\boldsymbol{x_j}\|_2^2}{2\sigma^2} \right) \right) + \left(\frac{\exp(\|Ax_i - Ax_j\|^2)}{\sum_{k \neq i} \exp(-\|Ax_i - Ax_k\|^2)} \right) \quad (12)$$

We are investigating gradient based methods to optimize for A in equation 12. Considering the MEGM results, the combination with NCA can lead to good results.

There has been a lot of work done in adaptive distance metric [16, 17]. In adaptive metric learning a separate metric is learnt for each query point. We are currently modifying MEGM to work in such local settings. Training a separate metric for each query point can become computationally expensive. We are investigating clustering techniques to cluster data first and than train a separate metric for each cluster.

In summary, we proposed a simple mean square error's gradient based metric learning algorithm (MEGM) in this paper and showed that MEGM not only results in classification improvement of k-NN classifier but also performs bet-

ter than other metric learning algorithms. Results are shown on major UCIML databases. Our results are encouraging and requires additional investigation to further improve MEGM performance as described.

References

1. T. Cover, "Rates of convergence for nearest neighbor procedures," in *Proceedings of the International Conference on Systems Sciences*, 1968.
2. 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.
3. R. Snapp and S. Venkatesh, "Asymptotic expansions of the k-nearest neighbor risk," *The Annals of Statistics*, 1998.
4. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.
5. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood component analysis," in *Proceedings of Neural Inforamtion and Processing Systems*, 2005.
6. J. Davis and I. Dhillon, "Structured metric learning for high dimensional problems," in *ACM SIGKDD conference on Knowledge Discovery and Data Mining*, 2008.
7. K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proceedings of Neural Inforamtion and Processing Systems*, 2005.
8. B. Sriperumbudar, O. Lang, and G. Lanckriet, "Metric embedding for kernel classification rules," in *Proceedings of the International Conference on Machine Learning*, 2008.
9. A. Globerson and S. Roweis, "Metric learning by collapsing classes," in *Proceedings of Neural Inforamtion and Processing Systems*, 2005.
10. E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Proceedings of Neural Inforamtion and Processing Systems*, 2002.
11. J. Friedman, "Flexible metric nearest neighbor classification," Tech Report, Dept. of Statistics, Stanford University, Tech. Rep., 1994.
12. D. Lowe, "Similarity metric learning for a variable-kernel classifier," in *Proceedings of Neural Inforamtion and Processing Systems*, 1996.
13. K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proceedings of Neural Inforamtion and Processing Systems*, 2006.
14. A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relation," in *Proceedings of the International Conference on Machine Learning*, 2003.
15. C. Mertz and P. Murphy, "Machine learning repository," 2005. [Online]. Available: http://archive.ics.uci.edu/ml/
16. T. Hastie and R. Tibshirani, "Discriminative adaptive nearest neighbor classification," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 1996.
17. N. Zaidi, D. M. Squire, and D. Suter, "BoostML: An adaptive metric learning for nearest neighbor classification," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2010.