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

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

[1] Clayton School of IT, Monash University, Melbourne VIC 3800, Australia

[2] School of Computer Science, University of Adelaide, North Terrace SA 5005, Australia

December 9, 2010

Our Research Goals

- Machine learning often deals with the problem of learning in high dimensions
- Euclidean distance in an *n* dimensional space is defined as:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2} \quad (1)$$

$$d(x,y) = \sqrt{(\vec{x} - \vec{y})^T I (\vec{x} - \vec{y})} \quad (2)$$

  where I is a diagonal matrix
- In machine learning problems, however, features are often not commensurate; nor are they equally informative
  - this means that one usually should not simply rely on Euclidean distance
- Finding appropriate distance measures for use in machine learning algorithms is a key goal of machine learning research

Our Research Goals (Contd)

The performance of most machine learning algorithms depends critically on the distance measure used. For example

- A Nearest Neighbor (NN) classifier with a suitably defined distance metric has been shown to perform as well as Support Vector Machine (SVM) and Neural Network classifier
- A kernel is a measure of similarity between data objects. SVM and Gaussian Process (GP) performance depends on the choice of the kernel and its parameters. Consider the case of the well-known Gaussian kernel:

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}\right) \tag{3}$$

- For simplicity, the kernel is usually assumed to be isotropic and σ is tuned via cross-validation. Clearly there is scope to learn a metric to optimize the kernel, i.e. to use $\|x_i - x_j\|_A^2$, where A is the learnt metric

Metric Learning

- Metric learning techniques have emerged as a promising area of machine learning in recent years
- Metric learning algorithms aim to find a linear transformation of the data to a space where isotropic treatment of dimensions is appropriate

$$\begin{aligned} ||x - y||_A^2 &= (\vec{x} - \vec{y})^T A (\vec{x} - \vec{y}) \\ &= ||L(\vec{x} - \vec{y})||^2 \quad \text{where } A = L^T L \end{aligned} \tag{4}$$

The transformed data $L\vec{x}$ lie in a space where it is appropriate to measure distances $d_A^2(\vec{x}, \vec{y})$ isotropically
By constraining the matrix A in equation 4 to be diagonal with zeros on some diagonal elements, feature selection can be achieved. Typically in machine learning, feature selection and metric learning methods are introduced in different scenarios, but they share related goals

Metric Learning (Contd)

Why metric learning?

- The performance of a Nearest Neighbor classifier depends on the size of neighborhood (K), as it controls its bias and variance. Due to the Curse of Dimensionality, even for moderate numbers of dimensions, a very large amount of training data is required to make even a $K = 1$ neighborhood relatively small. Bias can thus be large even for the smallest possible K
- Bias can be reduced by learning a metric that gives no influence to irrelevant features, thereby reducing the dimensionality. This in turn reduces the required diameter of the K-NN neighborhood, thus reducing the bias

Mean Error Square based Metric Learning (MEGM) Algorithm

- Motivated from bias reduction in high dimensions, as opposed to prevalent methods which maximizes margin.
- Algorithm can be easily modified for feature selection and feature weighting.
- It minimizes the following mean square error (using gradient descent):

$$\mathrm{MSE}(\hat{y}) = \sum_{t=1}^{C} \sum_{i=1}^{N} (y_{ti} - \hat{y}_{ti})^2$$

where

$$\hat{y}(\vec{x}_i) = \frac{\sum_j y_j V_j}{\sum_j V_j}$$

MEGM (Contd)

- The vote casted by each neighbor is:

$$V_j = \exp\left(\frac{-d^2(\vec{x}_i, \vec{x}_j)}{2\sigma^2}\right)$$

which is in fact:

$$W_j = \exp\left(-\|L\vec{x}_i - L\vec{x}_j\|_2^2\right)$$

- The gradient can be written as:

$$\frac{\partial \mathcal{E}}{\partial L} = 2L(y_i - \hat{y}_i)\frac{1}{\sum_j W_j}\sum_j (y_j - \hat{y}_j) W_j \left((\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^T\right)$$

MEGM pros and cons

Cons:

- Suffers from local minima problems (can be avoided).
- A first look at the objective function will lead you to believe that the classifier is over-fitted.
- You may doubt working on small data sets or with high dimensions.

Pros:

- Performance-wise better than other metric learning algorithms.
- Extremely simple. Handles multi-class classification scenarios almost effortlessly.
- Performance is better in most cases than standard SVM (which are computational expensive as you need to train one versus all or one versus one classifiers).

Experimental Results

The performance of proposed MEGM algorithm is compared with other methods on following databases:

- yalefaces, yalefacesB, AT&Tfaces
- caltechfaces, caltechfacesB
- isolet, USPS, Coil100
- UCIML databases

Experimental Results

The performance is compared with the following methods:

- **KNN:** 1 Nearest neighbor classification algorithm.
- **RCA:** Relevant Component Analysis algorithm.
- **LMNN:** Large Margin Nearest Neighbor metric learning algorithm.
- **NCA:** Neighborhood Component Analysis.

Performance on UCIML databases

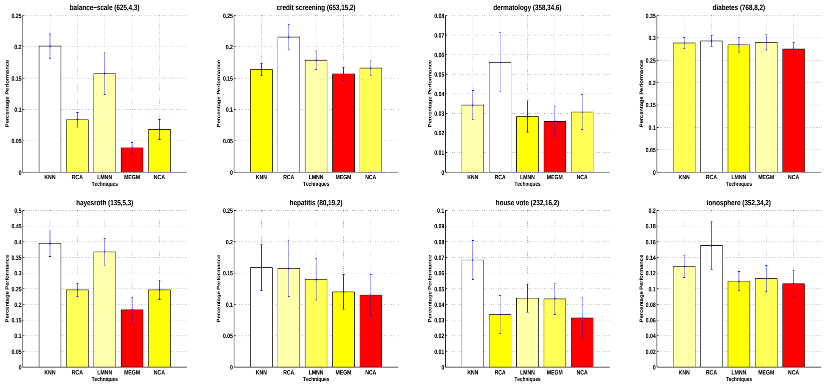

Figure: Error rate comparison of various techniques on UCIML databases.

Performance on UCIML databases

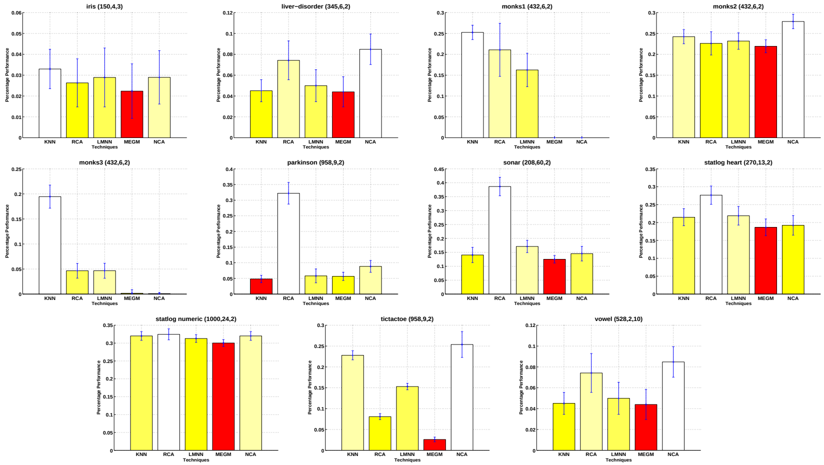

Figure: Error rate comparison of various techniques on UCIML databases.

Performance on UCIML databases

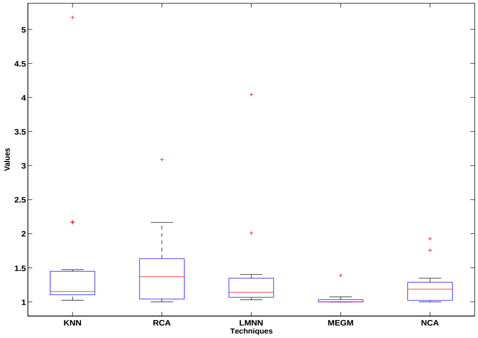

Figure: Boxplots depicting the robustness of KNN, RCA, LMNN, MEGM and NCA metric learning algorithms on the classification results of all 19 UCIML databases.

Performance on other databases

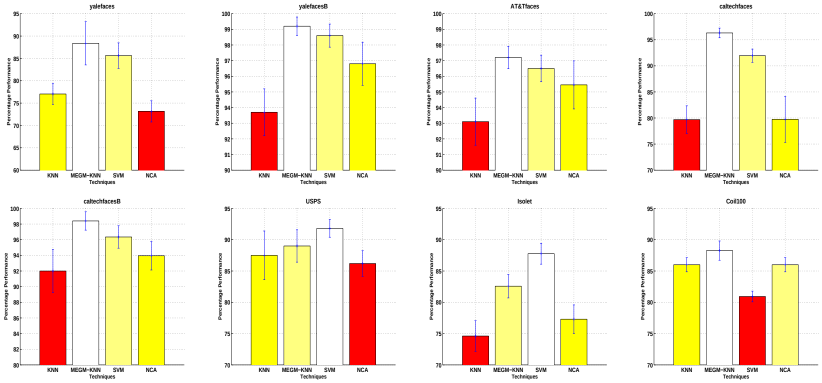

Figure: Percentage performance (Correctness rate) of KNN, MEGM-KNN, SVM and NCA on various object databases.

Metric Learning for Support Vector Machine and Gaussian Process classification

- Same proposed metric learning algorithm has been used to learn kernel for Support Vector Machines. The results are published in IVCVZ and DICTA 2010. We found that tuning a kernel using the proposed metric learning algorithms results in either better or similar performance to cross-validation methods for tuning kernels.
- We are investigating results for Gaussian process classification.

Conclusion (contd)

- A simple Mean Square Error's gradient based metric learning algorithm is proposed in the paper. The performance of the algorithm is compared with other state of the art metric learning algorithms on various UCIML and object database
- Future work includes combining MEGM and margin maximization based (NCA) algorithm
- Modifying MEGM algorithm for local adaptive metric learning.