Distance Metric Learning

2014-03-06

전상혁

Introduction

- Distance Metric learning is to learn a distance metric for the input space of data
- Data is from a given collection of pair of similar/dissimilar points that preserves the distance relation among the training data
- Learned metric can significantly improve the performance of algorithms such as classification, clustering and many other tasks

Distance Metric

- A distance metric or distance function is a function that defines a distance between elements of a set
  - Example1: Euclidean metric

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_i - q_i)^2 + \ldots + (p_n - q_n)^2}.$$

  - Example2: Discrete metric
    - $if\ x = y\ then\ d(x,y) = 0, Otherwise, d(x,y) = 1$

- A metric on a set X is a function $d: X \times X \rightarrow \mathbf{R}$ (where **R** is real number)
  - $d(x,y) \geq 0$
  - $d(x,y) = 0\ if\ and\ only\ if\ x = y$
  - $d(x,y) = d(y,x)$
  - $d(x,z) \leq \mathrm{d(x,y) + d(y,z)}$

Introduction

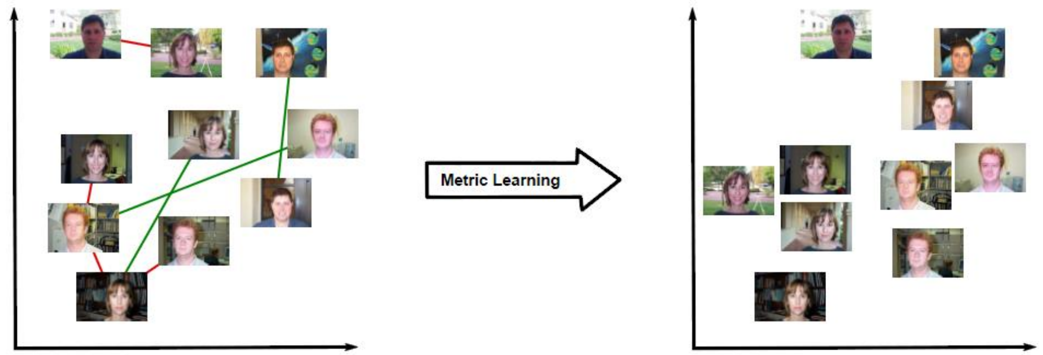


Figure 1: Illustration of metric learning applied to a face recognition task. For simplicity, images are represented as points in 2 dimensions. Pairwise constraints, shown in the left pane, are composed of images representing the same person (must-link, shown in green) or different persons (cannot-link, shown in red). We wish to adapt the metric so that there are fewer constraint violations (right pane). Images are taken from the Caltech Faces dataset.[8]

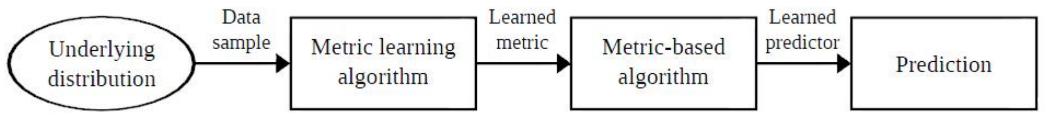


Figure 2: The common process in metric learning. A metric is learned from training data and plugged into an algorithm that outputs a predictor (e.g., a classifier, a regressor, a recommender system...) which hopefully performs better than a predictor induced by a standard (non-learned) metric.

Bellet, A., Habrard, A., and Sebban, M. A Survey on Metric Learning for Feature Vectors and Structured Data, 2013

Categories of distance metric learning

- Supervised distance metric learning
  - Constraints or labels given to the algorithm
    - Equivalence constraints where pairs of data points that belong to the same classes (semantically-similar)
    - Inequivalence constraints where pairs of data points that belong to the different classes (semantically-dissimilar)
  - local adaptive supervised distance metric learning, NCA, RCA
- Unsupervised distance metric learning
  - Dimensionality reduction techniques
  - Linear methods (PCA, MDS) and non-linear methods (ISOMAP, LLE, LE)
- Kernel methods towards learning distance metrics
  - Kernel alignment, extension work of learning idealized kernel

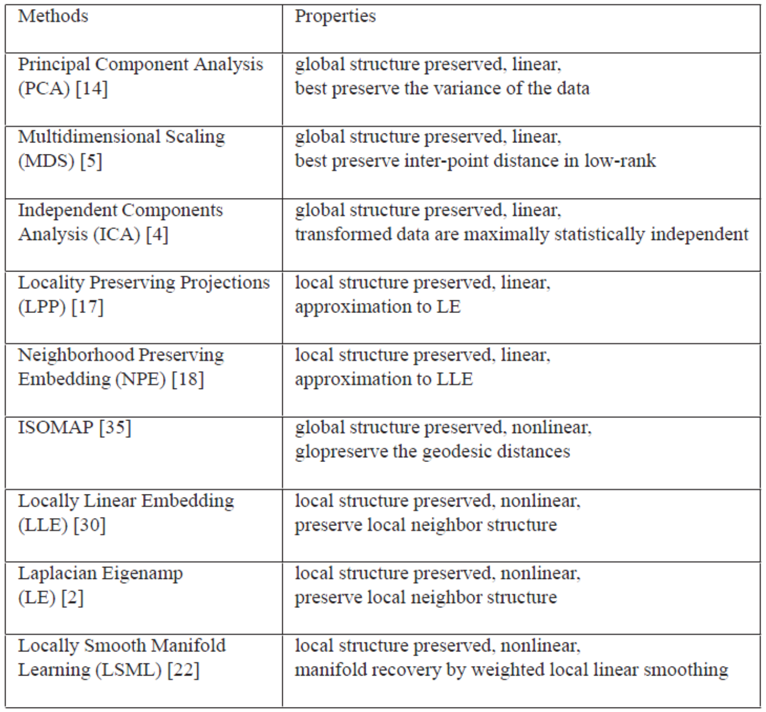
Liu Yang, Distance Metric Learning: A Comprehensive Survey, 2005
Liu Yang, An Overview of Distance Metric Learning, 2007

Categories of distance metric learning

- Maximum margin based distance metric learning
  - Formulate distance metric learning as a constrained convex programming problem, and attempt to learn complete distance metric from training data
  - Expensive computation, overfitting problem when limited training samples and large feature dimension
  - Large margin nearest neighbor classification (LMNN), SDP methods to solve the kernelized margin maximization

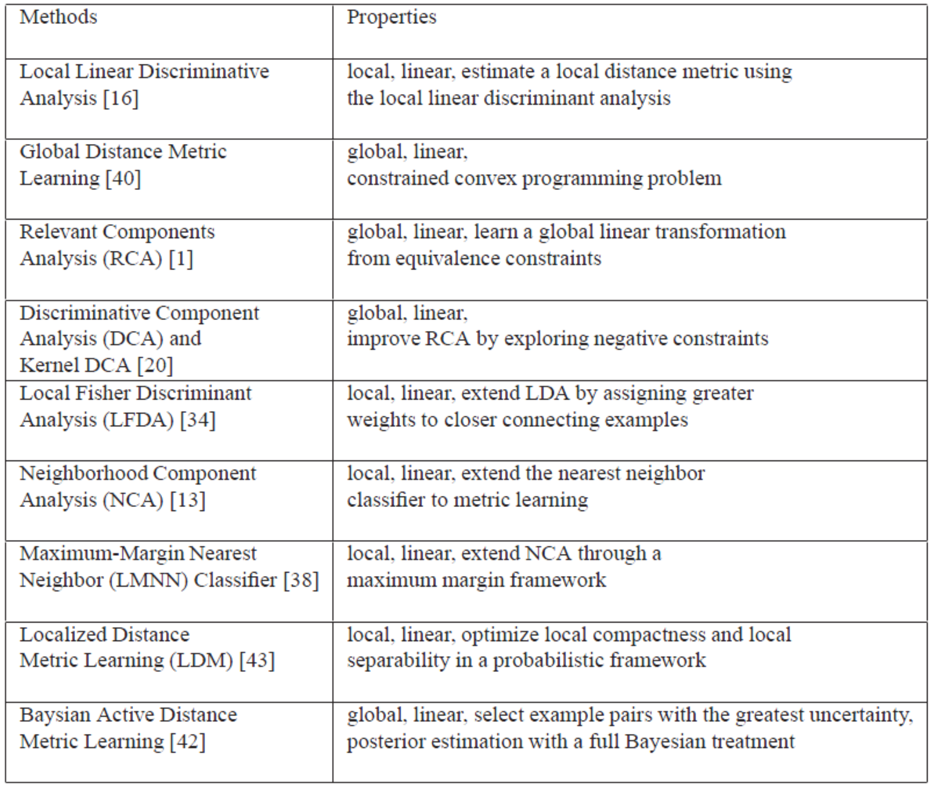
Liu Yang, Distance Metric Learning: A Comprehensive Survey, 2005
Liu Yang, An Overview of Distance Metric Learning, 2007

Categories of distance metric learning



Methods	Properties
Principal Component Analysis (PCA) [14]	global structure preserved, linear, best preserve the variance of the data
Multidimensional Scaling (MDS) [5]	global structure preserved, linear, best preserve inter-point distance in low-rank
Independent Components Analysis (ICA) [4]	global structure preserved, linear, transformed data are maximally statistically independent
Locality Preserving Projections (LPP) [17]	local structure preserved, linear, approximation to LE
Neighborhood Preserving Embedding (NPE) [18]	local structure preserved, linear, approximation to LLE
ISOMAP [35]	global structure preserved, nonlinear, glopreserve the geodesic distances
Locally Linear Embedding (LLE) [30]	local structure preserved, nonlinear, preserve local neighbor structure
Laplacian Eigenamp (LE) [2]	local structure preserved, nonlinear, preserve local neighbor structure
Locally Smooth Manifold Learning (LSML) [22]	local structure preserved, nonlinear, manifold recovery by weighted local linear smoothing

Table 1: Unsupervised distance metric learning methods. This group of methods essentially learn a low-dimensional embedding of the original feature space; and can be categorized along two dimensions: preserving glocal structure vs. preserving local structure; and linear vs. nonlinear



Methods	Properties
Local Linear Discriminative Analysis [16]	local, linear, estimate a local distance metric using the local linear discriminant analysis
Global Distance Metric Learning [40]	global, linear, constrained convex programming problem
Relevant Components Analysis (RCA) [1]	global, linear, learn a global linear transformation from equivalence constraints
Discriminative Component Analysis (DCA) and Kernel DCA [20]	global, linear, improve RCA by exploring negative constraints
Local Fisher Discriminant Analysis (LFDA) [34]	local, linear, extend LDA by assigning greater weights to closer connecting examples
Neighborhood Component Analysis (NCA) [13]	local, linear, extend the nearest neighbor classifier to metric learning
Maximum-Margin Nearest Neighbor (LMNN) Classifier [38]	local, linear, extend NCA through a maximum margin framework
Localized Distance Metric Learning (LDM) [43]	local, linear, optimize local compactness and local separability in a probabilistic framework
Baysian Active Distance Metric Learning [42]	global, linear, select example pairs with the greatest uncertainty, posterior estimation with a full Bayesian treatment

Table 2: Supervised distance metric learning methods

Liu Yang, An Overview of Distance Metric Learning, 2007

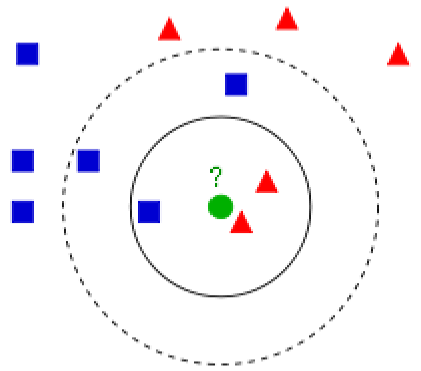
Network Intelligence and Analytics Lab. KAIST Korea Advanced Institute of Science and Technology KAIST EE ELECTRICAL ENGINEERING

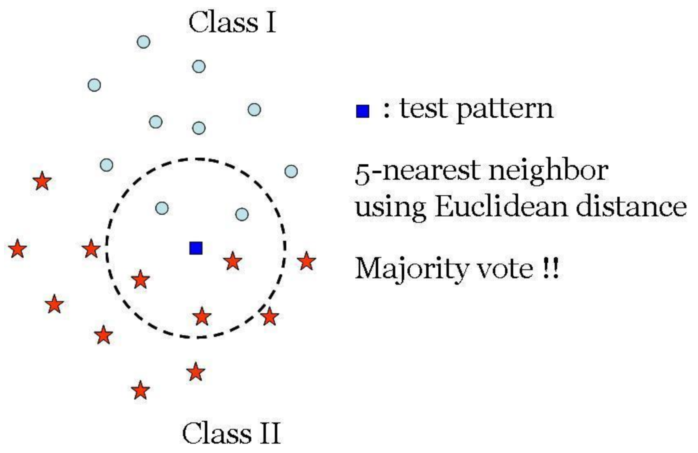
Large Margin Nearest Neighbor Classification (LMNN)

Kilian Weinberger, et al. Distance metric learning for large margin nearest neighbor classification. NIPS, 2006.

Motivation – better KNN

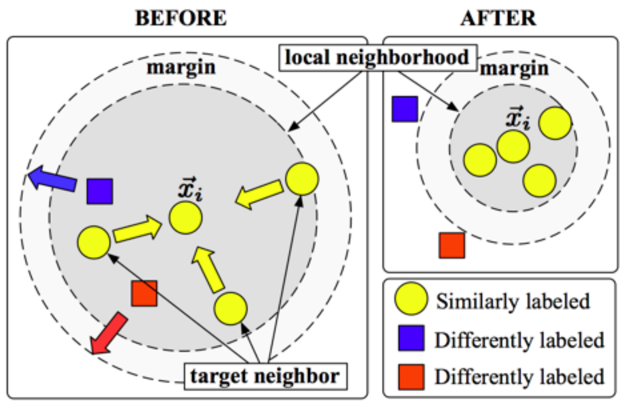
- KNN: simple non-parametric classification method (Also can be used for clustering, regression, anomaly detection, and etc.)
- k is user-defined constant, and an unlabeled vector is classified by assigning the label which is most frequent among the k training samples nearest to that unlabeled vector
- Drawback: when the class distribution is skewed, basic "majority voting" classification may not works well.





Introduction

- Large margin nearest neighbor (LMNN) method
  - Neighbors are selected by using the learned Mahanalobis distance metric
  - Learn a Mahanalobis distance metric for kNN classification by semidefinite programming
  - Find $L: R^d \to R^d$, which will be used to compute squared distance as
$$D(\vec{x_i}, \vec{x_j}) = \left\| L(\vec{x_i} - \vec{x_j}) \right\|^2$$
by minimizing some cost function $\epsilon(L)$



Mahalanobis Distance Metric

- Relative measure of a data point's distance from a common point
- It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant
- $d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^\top \mathbf{\Omega}(\vec{x}-\vec{y})}$ where $\mathbf{\Omega}$ is positive semidefinte matrix
- Example: the distance between two random variable $\vec{x},\vec{y}$ of the same distribution with covariance matrix S is defined as
  - $d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^\top S^{-1}(\vec{x}-\vec{y})}$
  - Euclidean distance is special case when S is identity matrix
- Recall: distance $D(\vec{x_i},\vec{x_j}) = \left\|L(\vec{x_i}-\vec{x_j})\right\|^2$
- We can rewrite eq. as $D(\vec{x_i},\vec{x_j}) = (\vec{x_i}-\vec{x_j})\mathbf{M}^\top(\vec{x_i}-\vec{x_j})$ where the matrix $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$, parameterizes the Mahalanobis distance metric induced by the linear transform $\mathbf{L}$

Semidefinite Programming

- A kind of convex optimization problem
- x is a vector of n variables, c is a constant n dimensional vector
- Linear programming (LP)
  - $minimize\ c \cdot x$

$$\begin{aligned} &s.t.\ \ a_i \cdot x = b_i, i = 1, \dots, m \\ &x \in R_+^n \end{aligned}$$

- X is positive semidefinite matrix if $v^\top X v \geq 0\ for\ any\ v \in R^n$ $(X \succcurlyeq 0)$
- If $C(X)$ is a linear function of X, then $C(X)$ can be written as $C \cdot X$
- Semidefinite program (SDP)
  - $minimize\ C \cdot X$

$$\begin{aligned} &s.t.\ \ A_i \cdot X = b_i, i = 1, \dots, m \\ &X \succcurlyeq 0 \end{aligned}$$

Cost Function

- $\epsilon(L) = \sum_{ij} \eta_{ij} \left\| \mathbf{L}(\vec{x_i} - \vec{x_j}) \right\|^2 + c \sum_{ijl} \eta_{ij} \, (1 - y_{il}) \left[1 + \left\| \mathbf{L}(\vec{x_i} - \vec{x_j}) \right\|^2 - \| \mathbf{L}(\vec{x_i} - \vec{x_l}) \|^2 \right]_+$
- Let $\{(\vec{x_i}, \mathrm{y_i})\}_{\mathrm{i}=1}^{\mathrm{n}}$ denote a training set of n labeled examples with inputs $\vec{x_i} \in R^d$ and discrete (but not necessarily binary) class labels y_i
- $y_{ij} \in \{0,1\}$ is binary matrix to indicate whether or not the labels y_i and y_j match
- $\eta_{ij} \in \{0,1\}$ to indicate whether input $\vec{x_j}$ is a target neighbor of input $\vec{x_i}$, this value is fixed and does not change during learning
  - Target neighbor: each instance $\vec{x_i}$ has exactly k different target neighbors within data set, which all share the same class label $\vec{y_i}$
  - Target neighbors are the data points that should become nearest neighbors under the learned metric
  - C.f. Impostors: same as target neighbor except it has different class label
- Term $[z]_+ = \max(z, 0)$
- $c > 0$ some positive constant (typically set by cross validation)

Cost Function

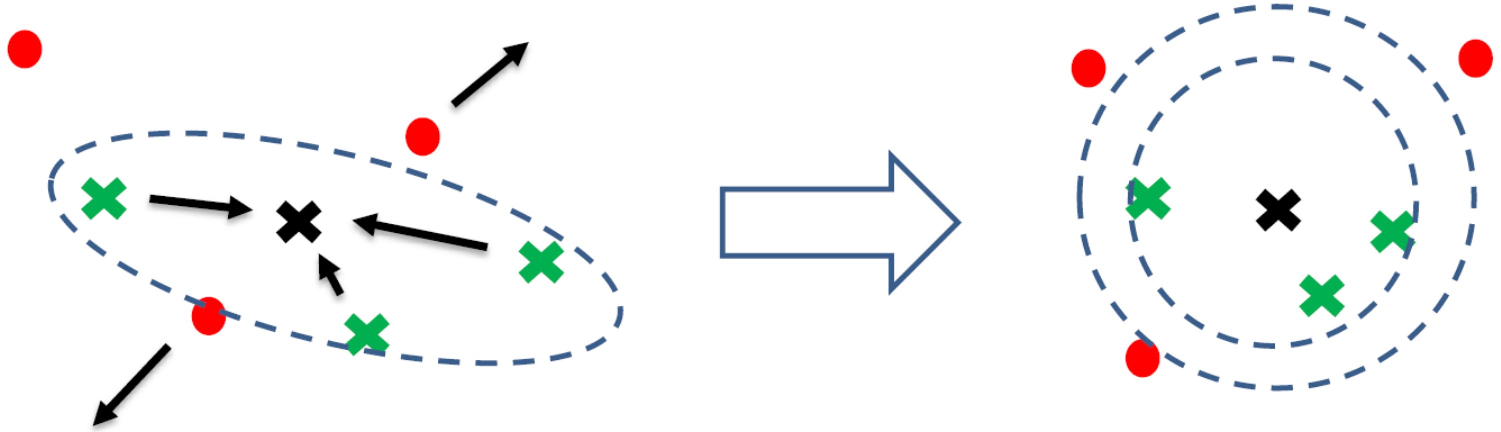
- $\epsilon(L) = \sum_{ij} \eta_{ij} \left\| \mathbf{L}(\vec{x_i} - \vec{x_j}) \right\|^2 + c \sum_{ijl} \eta_{ij} (1 - y_{il}) \left[1 + \left\| \mathbf{L}(\vec{x_i} - \vec{x_j}) \right\|^2 - \| \mathbf{L}(\vec{x_i} - \vec{x_l}) \|^2 \right]_+$
- The first optimization goal is achieved by minimizing the average distance between instance and their target neighbors
  - $\sum_{i,j \in N_i} d(\vec{x}_i, \vec{x}_j)$
- The second goal is achieved by constraining impostors $\vec{x_l}$ to be one unit further away than target neighbors $\vec{x_j}$
  - $\forall_{i,j \in N_i, l, y_l \neq y_i} d(\vec{x}_i, \vec{x}_j) + 1 \leq d(\vec{x}_i, \vec{x}_l)$
  - N_i denote the set of target neighbors for a data point $\vec{x_i}$

Another Interesting Formulation… LMNN

$$\Psi_{\text{pull}}(M) = \sum_{i,j(i)} \rho_M^2(x_i, x_j)$$

$$\Psi_{\text{push}}(M) = \sum_{i,j(i),l(i,j)} 1 + \rho_M^2(x_i, x_j) - \rho_M^2(x_i, x_l)$$

point	i
true neighbor	$j(i)$
imposter	$l(i,j)$



$$\Psi(M) = \lambda\ \Psi_{\text{pull}}(M)\ + (1-\lambda)\ \Psi_{\text{push}}(M)$$

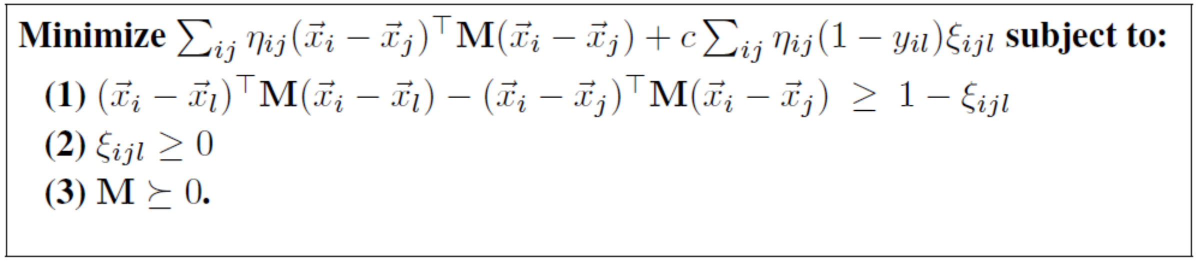
Advantages:

- Local constraints, so directly improves nearest neighbor quality!

Weinberger, Saul, *JMLR 2009.*

Convex Optimization

- Recall: distance $D(\vec{x_i}, \vec{x_j}) = \|L(\vec{x_i} - \vec{x_j})\|^2$
- Recall: We can rewrite eq. as $D(\vec{x_i}, \vec{x_j}) = (\vec{x_i} - \vec{x_j})\mathbf{M}^\top(\vec{x_i} - \vec{x_j})$ where the matrix $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$, parameterizes the Mahalanobis distance metric induced by the linear transform $\mathbf{L}$
- The hinge loss can be "mimicked" by introducing slack variable ζ_{ij} for all pairs of differently labeled inputs (i.e., for all $< i, j >$ such that $y_{ij} = 0$)



Minimize $\sum_{ij} \eta_{ij}(\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) + c\sum_{ij} \eta_{ij}(1 - y_{il})\xi_{ijl}$ **subject to:**

(1) $(\vec{x}_i - \vec{x}_l)^\top \mathbf{M}(\vec{x}_i - \vec{x}_l) - (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \ \geq \ 1 - \xi_{ijl}$

(2) $\xi_{ijl} \geq 0$

(3) $\mathbf{M} \succeq 0$.

Results

- PCA was used to reduce the dimensionality of image, speech, and text data, both to speed up training and avoid overfitting
- Both the number of target neighbors (k) and the weighting parameter (c) in optimization problem were set by cross validation
- KNN Euclidean distance
- KNN Mahalanobis distance
- Energy based classification
  - $$y_t = \text{argmin}_{y_t} \sum_j \eta_{tj} \|\mathbf{L}(\vec{x}_t - \vec{x}_j)\|^2 + c \sum_{j,i=t \vee l=t} \eta_{ij}(1 - y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$
- Multiclass SVM

Results

	Iris	Wine	Faces	Bal	Isolet	News	MNIST
examples (train)	106	126	280	445	6238	16000	60000
examples (test)	44	52	120	90	1559	2828	10000
classes	3	3	40	3	26	20	10
input dimensions	4	13	1178	4	617	30000	784
features after PCA	4	13	30	4	172	200	164
constraints	5278	7266	78828	76440	37 Mil	164 Mil	3.3 Bil
active constraints	113	1396	7665	3099	45747	732359	243596
CPU time (per run)	2s	8s	7s	13s	11m	1.5h	4h
runs	100	100	100	100	1	10	1

Table 1: Properties of data sets and experimental parameters for LMNN classification.

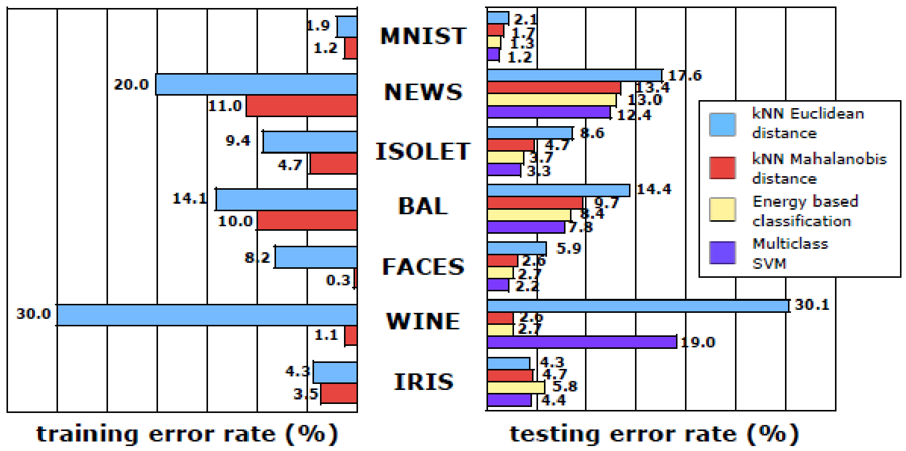
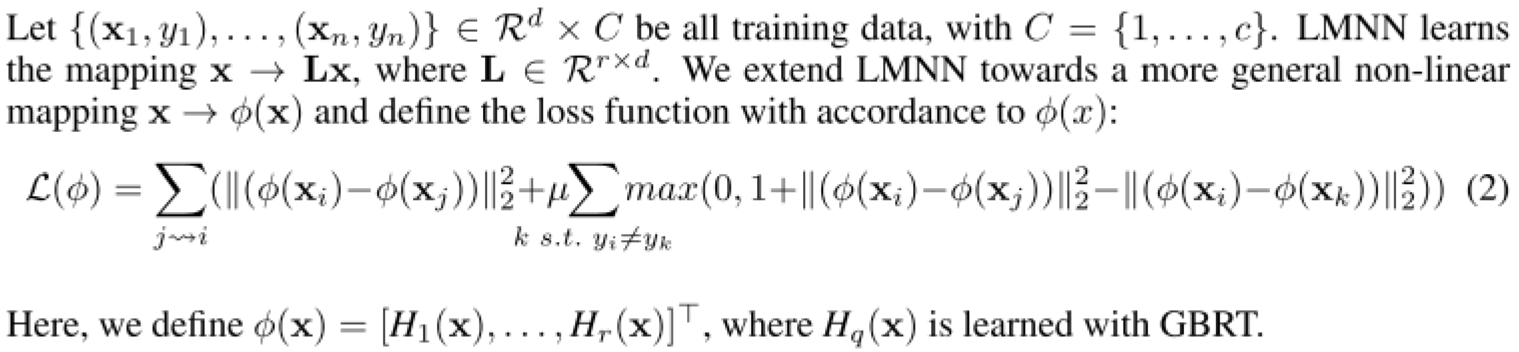


Figure 2: Training and test error rates for kNN classification using Euclidean versus Mahalanobis distances. The latter yields lower test error rates on all but the smallest data set (presumably due to over-training). Energy-based classification (see text) generally leads to further improvement. The results approach those of state-of-the-art multiclass SVMs.

GBLMNN: Non-linear method for LMNN

- Building non-linear mapping for metric learning

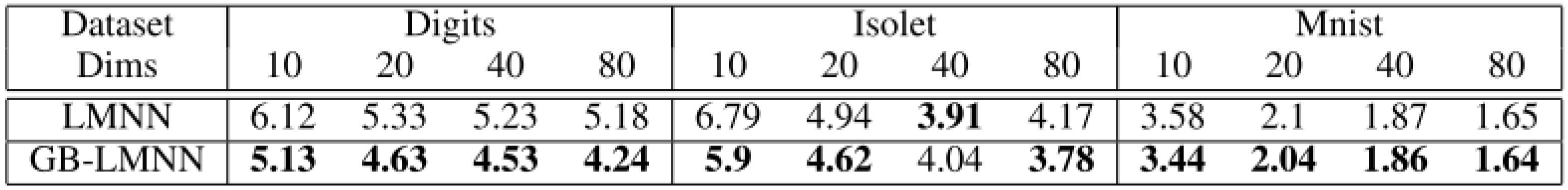


Let $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \in \mathcal{R}^d \times C$ be all training data, with $C = \{1, \dots, c\}$. LMNN learns the mapping $\mathbf{x} \rightarrow \mathbf{L}\mathbf{x}$, where $\mathbf{L} \in \mathcal{R}^{r \times d}$. We extend LMNN towards a more general non-linear mapping $\mathbf{x} \rightarrow \phi(\mathbf{x})$ and define the loss function with accordance to $\phi(x)$:

$$\mathcal{L}(\phi) = \sum_{j \rightsquigarrow i}(\|(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))\|_2^2 + \mu \sum_{k\ s.t.\ y_i \neq y_k} max(0, 1 + \|(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))\|_2^2 - \|(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_k))\|_2^2)) \quad (2)$$

Here, we define $\phi(\mathbf{x}) = [H_1(\mathbf{x}), \dots, H_r(\mathbf{x})]^\top$, where $H_q(\mathbf{x})$ is learned with GBRT.

- GBRT: Gradient Boosting Regression Tree



Dataset	Digits				Isolet				Mnist			
Dims	10	20	40	80	10	20	40	80	10	20	40	80
LMNN	6.12	5.33	5.23	5.18	6.79	4.94	**3.91**	4.17	3.58	2.1	1.87	1.65
GB-LMNN	**5.13**	**4.63**	**4.53**	**4.24**	**5.9**	**4.62**	4.04	**3.78**	**3.44**	**2.04**	**1.86**	**1.64**

- Code: http://www.cse.wustl.edu/~kilian/code/code.html

Large Margin Component Analysis (LMCA)

Torresani, L., & Lee, K. (2007). Large margin component analysis. Advances in Neural Information Processing

Problem: large number of features

- In problems involving thousands of features, distance metric learning cannot be used
  - High computation complexity
  - Overffiting problem
- Previous works have relied on a two-step solution
  - Apply dimension reduction algorithm (e.g. PCA)
  - Learn a metric in the resulting low-dimensional subspace
- LMCA: provide better solution for high dimensional problem

Linear Dimensionality Reduction for KNN (1/3)

- Recall: cost function of metric function

$$\epsilon(\mathbf{L}) = \sum_{ij} \eta_{ij} ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)||^2 + c \sum_{ijl} \eta_{ij} (1 - y_{il}) h(||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)||^2 - ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_l)||^2 + 1),$$

- In previous work, L was $D \times D$ matrix
- Idea: learn non square matrix size $d \times D, with\ d \ll D$
- Problem: cost function $\epsilon(\mathbf{M}), where\ \mathbf{M} = \mathbf{L}^{\mathsf{T}}\mathbf{L}$ is no longer convex
  - M has low rank d, not D

Linear Dimensionality Reduction for KNN (2/3)

- Idea: optimize the objective function directly with respect to the non square matrix L
- Although, equation is non-convex, it is better than minimizing the objective function with respect to L rather than with respect to the rank-deficient D×D matrix M
  - Since this optimization involves only $d \cdot D$ rather than D^2 unknowns, it can reduce the risk of overffiting
  - "the optimal rectangular matrix L computed with our method automatically satisfies the rank constraints on M without requiring the solution of difficult constrained minimization problems"

Linear Dimensionality Reduction for KNN (3/3)

- Minimize cost function using gradient-based optimizers

$$\begin{aligned}\frac{\partial \epsilon(\mathbf{L})}{\partial \mathbf{L}} &= 2\mathbf{L}\sum_{ij}\eta_{ij}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T + \\ &\quad 2c\mathbf{L}\sum_{ijl}\eta_{ij}(1-y_{il})\left[(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T-(\mathbf{x}_i-\mathbf{x}_l)(\mathbf{x}_i-\mathbf{x}_l)^T\right] \\ &\qquad h'(||\mathbf{L}(\mathbf{x}_i-\mathbf{x}_j)||^2-||\mathbf{L}(\mathbf{x}_i-\mathbf{x}_l)||^2+1)\end{aligned}$$

- "We handle the non-differentiability of h(s) at s = 0, by adopting a smooth hinge function as in [8]"

[8] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the 22nd International Conference on Machine Learning (ICML), 2005.

Nonlinear Feature Extraction (1/2)

- "Our approach learns a low-rank Mahalanobis distance metric in a high dimensional feature space F, related to the inputs by a nonlinear map $\phi: R^D \rightarrow F$"
- k is kernel function computed by the feature input products

  $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \phi_i^T \phi_j$, where for brevity we denoted $\phi_i = \phi(\mathbf{x}_i)$.
- L is now a transformation from the space F into a R^d $(d \ll D)$
- $$\begin{aligned} \frac{\partial \epsilon(\mathbf{L})}{\partial \mathbf{L}} &= 2\sum_{ij} \eta_{ij} \mathbf{L}(\phi_i - \phi_j)(\phi_i - \phi_j)^T + \\ & 2c \sum_{ijl} \eta_{ij}(1 - y_{il}) h'(s_{ijl}) \mathbf{L} \left[(\phi_i - \phi_j)(\phi_i - \phi_j)^T - (\phi_i - \phi_l)(\phi_i - \phi_l)^T \right] \end{aligned}$$

  where $s_{ijl} = (||\mathbf{L}(\phi_i - \phi_j)||^2 - ||\mathbf{L}(\phi_i - \phi_l)||^2 + 1)$.

Nonlinear Feature Extraction (2/2)

- Let $\Phi = [\phi_1, ..., \phi_n]^T$. We consider parameterizations of $\mathbf{L}$ of the form $\mathbf{L} = \Omega\Phi$, where Ω is some matrix allowing us to write $\mathbf{L}$ as a linear combination of the feature points.
- **Lemma 3.1** *The gradient in feature space can be computed as* $\frac{\partial\epsilon(\mathbf{L})}{\partial\mathbf{L}} = \Gamma\Phi$, *where* Γ *depends on features* ϕ_i *solely in terms of dot products* $(\phi_i^T\phi_j)$.
- $\mathbf{L}_{new} = \mathbf{L}_{old} - \lambda \left.\frac{\partial\epsilon(\mathbf{L})}{\partial\mathbf{L}}\right|_{\mathbf{L}=\mathbf{L}_{old}} = [\Omega_{old} - \lambda\Gamma_{old}]\,\Phi = \Omega_{new}\Phi$
- Update $\Omega \leftarrow (\Omega - \lambda\Gamma)$ until converge
- For classification, we project points onto the learned low-dim space by exploiting the kernel trick: $L\boldsymbol{\phi} = \Omega\mathbf{k}$

Results – High dimensional dataset

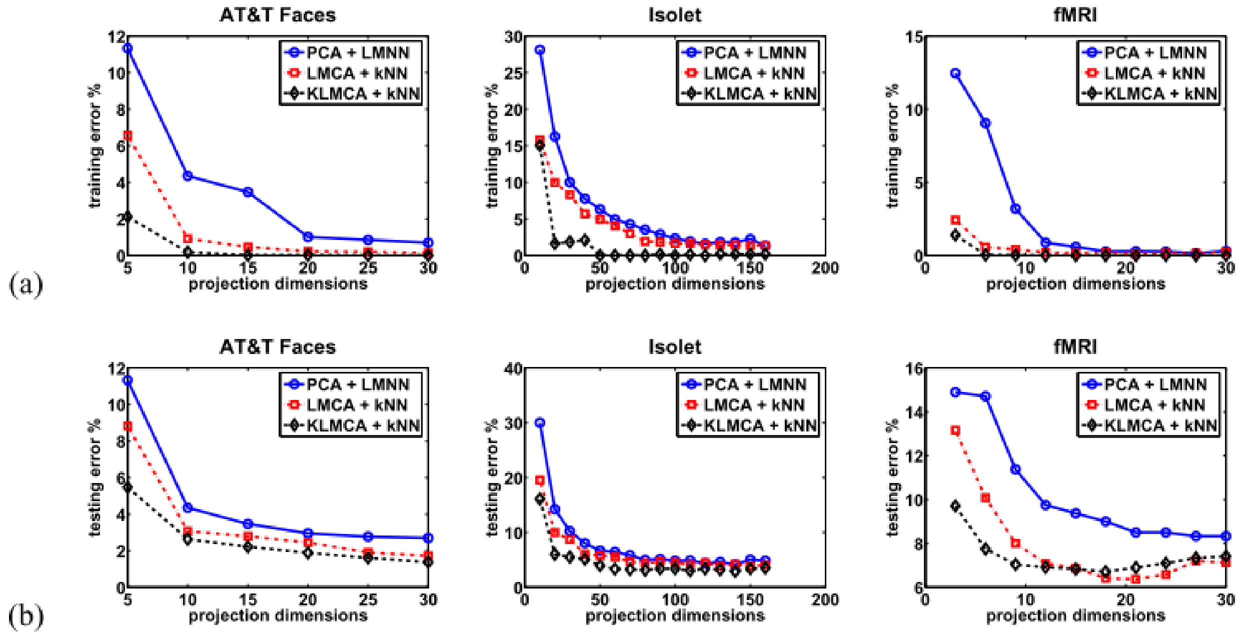


Figure 1: Classification error rates on the high-dimensional datasets Isolet, AT&T Faces and StarPlus fMRI for different projection dimensions. (a) Training error. (b) Testing error.

Results – low dimensional dataset

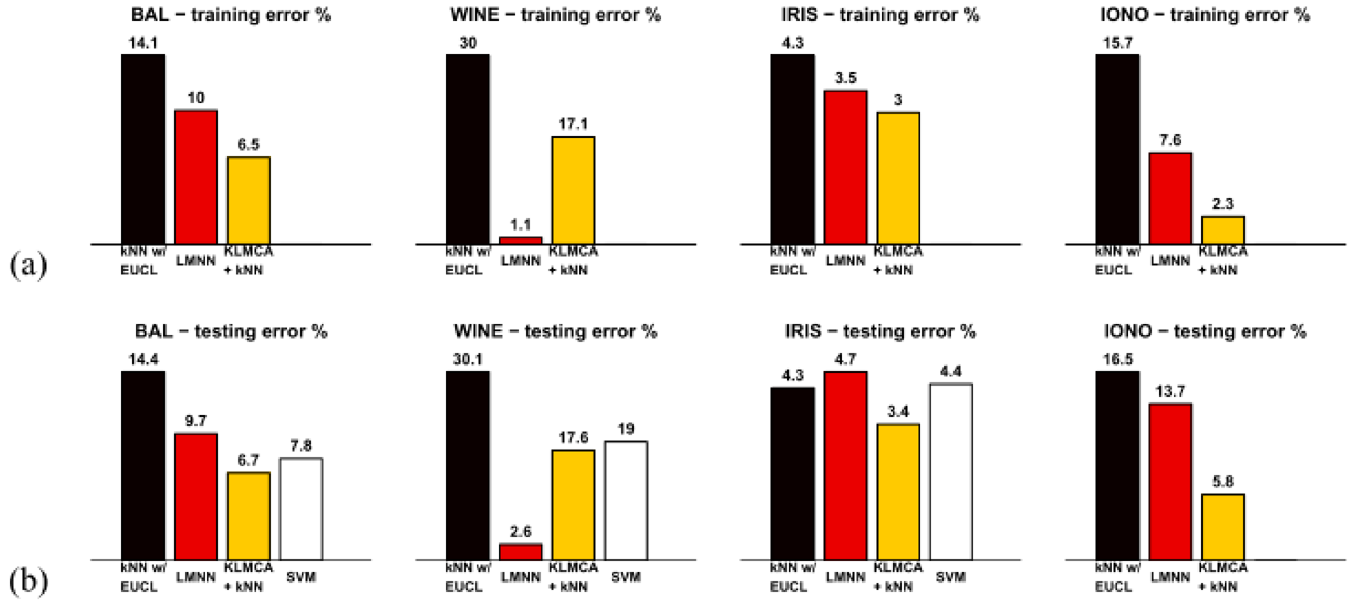


Figure 3: kNN classification accuracy on low-dimensional datasets: Bal, Wine, Iris, and Ionosphere. (a) Training error. (b) Testing error. Algorithms are kNN using Euclidean distance, LMNN [9], kNN in the nonlinear feature space computed by our KLMCA algorithm, and multiclass SVM.

Reference

- Bellet, A., Habrard, A., and Sebban, M. A Survey on Metric Learning for Feature Vectors and Structured Data, 2013
- Liu Yang, Distance Metric Learning: A Comprehensive Survey, 2005
- Liu Yang, An Overview of Distance Metric Learning, 2007
- Marco Cuturi. KAIST Machine Learning Tutorial Metrics and Kernels A few recent topics, 2013
- Nakul Verma, A tutorial on Metric Learning with some recent advances
- Aurelien Bellet, Tutorial on Metric Learning, 2013
- Brian Kulis. Tutorial on Metric Learning. International Conference on Machine Learning (ICML) 2010

Reference

- K. Q. Weinberger, J. Blitzer, and L. K. Saul (2006). In Y. Weiss, B. Schoelkopf, and J. Platt (eds.), Distance Metric Learning for Large Margin Nearest Neighbor Classification, Advances in Neural Information Processing Systems 18 (NIPS-18). MIT Press: Cambridge, MA.
- K. Q. Weinberger, L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. Journal of Machine Learning Research (JMLR) 2009, 10:207-244
- Torresani, L., & Lee, K. (2007). Large margin component analysis. Advances in Neural Information Processing
- Kedem, D., Xu, Z., & Weinberger, K. (n.d.). Gradient Boosted Large Margin Nearest Neighbors, (1), 10–12.