

Metric and Kernel Learning

Inderjit S. Dhillon
University of Texas at Austin

Cornell University
Oct 30, 2007

Joint work with Jason Davis, Prateek Jain, Brian Kulis and Suvrit Sra

Metric Learning

- Goal: “Learn” Distance Metric between Data
- Important problem in Data Mining & Machine Learning
- Can govern success or failure of data mining algorithm

Metric Learning: Example I



Similarity by Person(identity) or by Pose

Metric Learning: Example II

- Consider a set of text documents
- Each document is a review of a classical music piece
- Might be clusterable in one of two ways
 - By Composer (Beethoven, Mozart, Mendelssohn)
 - By Form (Symphony, Sonata, Concerto)
- Similarity by Composer or by Form

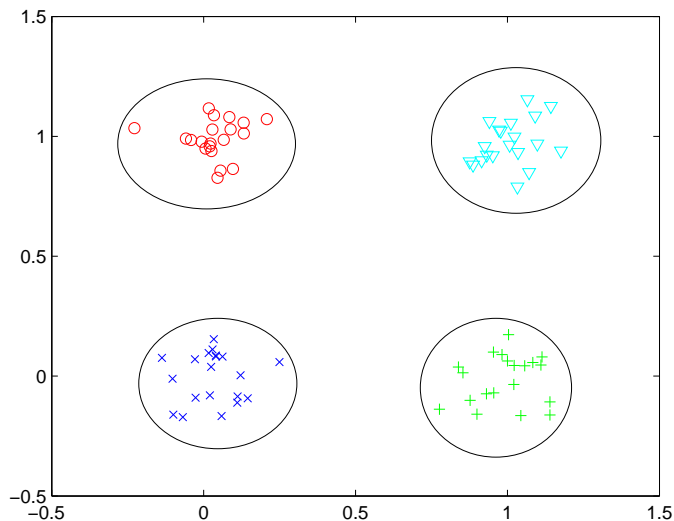
Mahalanobis Distances

- We restrict ourselves to learning *Mahalanobis distances*:
 - Distance parameterized by positive definite matrix Σ :

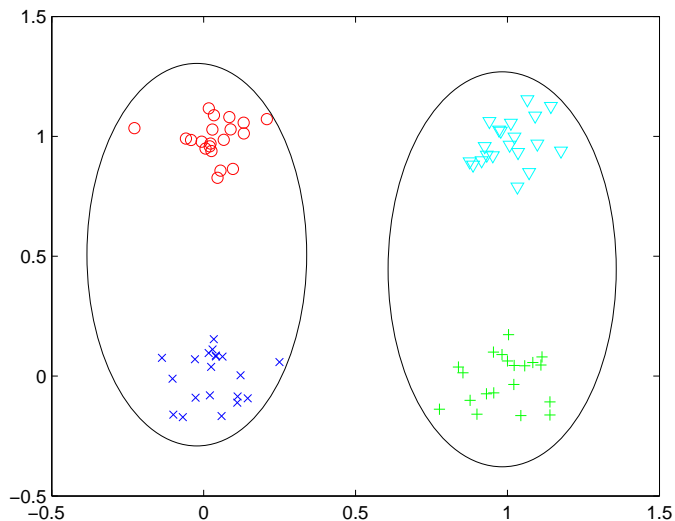
$$d_{\Sigma}(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma (\mathbf{x}_1 - \mathbf{x}_2)$$

- Often Σ is the inverse of the covariance matrix
- Generalizes squared Euclidean distance ($\Sigma = I$)
- Rotates and scales input data

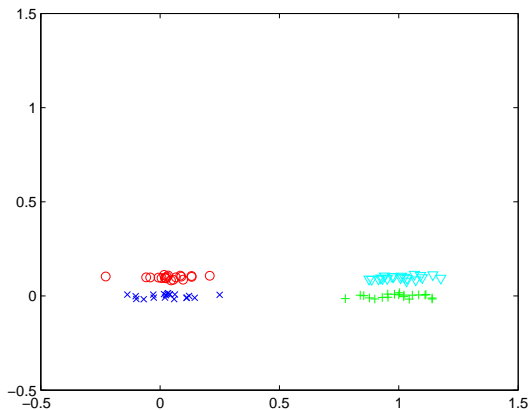
Example: Four Blobs



Example: Four Blobs



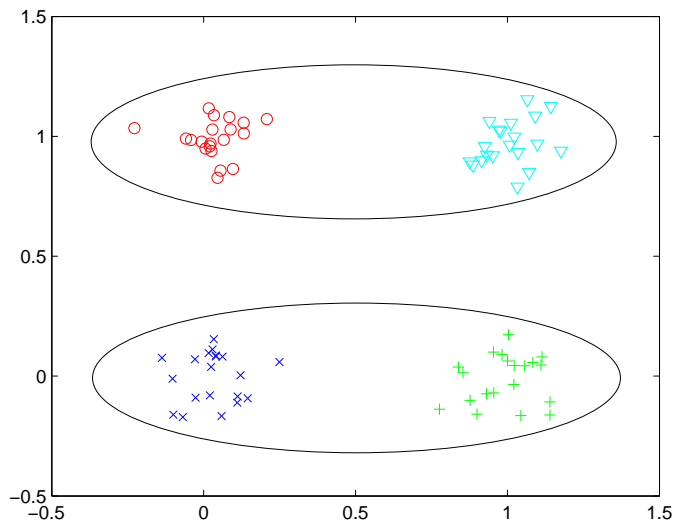
Example: Four Blobs



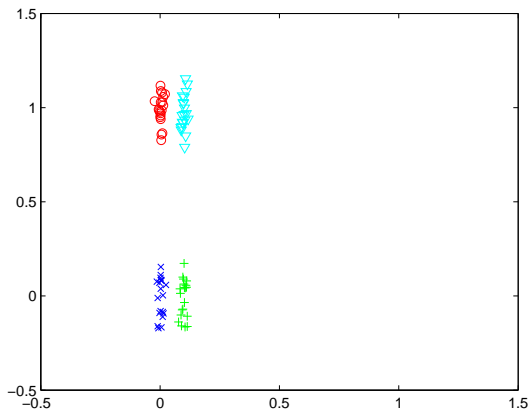
- Want to learn:

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}$$

Example: Four Blobs



Example: Four Blobs



- Want to learn:

$$\Sigma = \begin{pmatrix} \epsilon & 0 \\ 0 & 1 \end{pmatrix}$$

Problem Formulation

- **Metric Learning Goal:**

$$\min_{\Sigma} \text{loss}(\Sigma, \Sigma_0)$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \leq u \quad \text{if } (i, j) \in S \text{ [similarity constraints]}$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \geq \ell \quad \text{if } (i, j) \in D \text{ [dissimilarity constraints]}$$

- Learn spd matrix Σ that is “close” to the baseline spd matrix Σ_0
- Other linear constraints on Σ are possible
- Constraints can arise from various scenarios
 - Unsupervised: Click-through feedback
 - Semi-supervised: must-link and cannot-link constraints
 - Supervised: points in the same class have “small” distance, etc.

Problem Formulation

- **Metric Learning Goal:**

$$\min_{\Sigma} \text{loss}(\Sigma, \Sigma_0)$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \leq u \quad \text{if } (i, j) \in S \text{ [similarity constraints]}$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \geq \ell \quad \text{if } (i, j) \in D \text{ [dissimilarity constraints]}$$

- Learn spd matrix Σ that is “close” to the baseline spd matrix Σ_0
- Other linear constraints on Σ are possible
- Constraints can arise from various scenarios
 - Unsupervised: Click-through feedback
 - Semi-supervised: must-link and cannot-link constraints
 - Supervised: points in the same class have “small” distance, etc.
- QUESTION: What should “loss” be?

LogDet Divergence

- We use $\text{loss}(\Sigma, \Sigma_0)$ to be the Log-Determinant Divergence:

$$D_{\ell d}(\Sigma, \Sigma_0) = \text{trace}(\Sigma \Sigma_0^{-1}) - \log \det(\Sigma \Sigma_0^{-1}) - d$$

- **Our Goal:**

$$\min_{\Sigma} D_{\ell d}(\Sigma, \Sigma_0)$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \leq u \quad \text{if } (i, j) \in S \text{ [similarity constraints]}$$

$$(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \geq \ell \quad \text{if } (i, j) \in D \text{ [dissimilarity constraints]}$$

Preview

- Salient points of our Approach:
 - Metric Learning is equivalent to “Kernel Learning”
 - Generalizes to Unseen Data Points
 - Can improve upon an input metric or kernel
 - No expensive eigenvector computation or semi-definite programming
- Most existing methods fail to satisfy one or more of the above

Brief Digression

Bregman Divergences

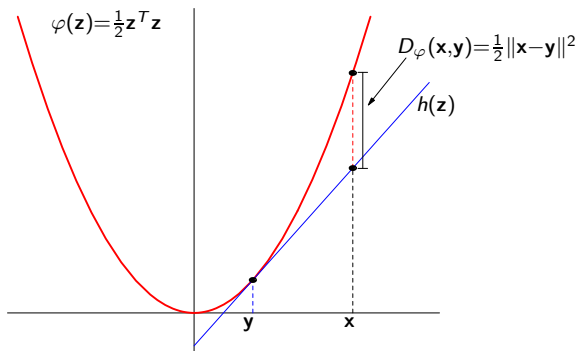
- Let $\varphi : S \rightarrow \mathbb{R}$ be a differentiable, strictly convex function of “Legendre type” ($S \subseteq \mathbb{R}^d$)
- The Bregman Divergence $D_\varphi : S \times \text{relint}(S) \rightarrow \mathbb{R}$ is defined as

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla \varphi(\mathbf{y})$$

Bregman Divergences

- Let $\varphi : S \rightarrow \mathbb{R}$ be a differentiable, strictly convex function of “Legendre type” ($S \subseteq \mathbb{R}^d$)
- The Bregman Divergence $D_\varphi : S \times \text{relint}(S) \rightarrow \mathbb{R}$ is defined as

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla \varphi(\mathbf{y})$$

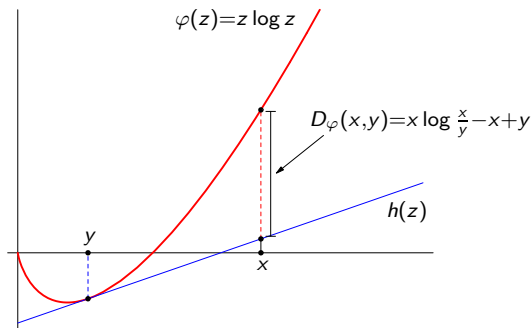


Squared Euclidean distance is a Bregman divergence

Bregman Divergences

- Let $\varphi : S \rightarrow \mathbb{R}$ be a differentiable, strictly convex function of “Legendre type” ($S \subseteq \mathbb{R}^d$)
- The Bregman Divergence $D_\varphi : S \times \text{relint}(S) \rightarrow \mathbb{R}$ is defined as

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla \varphi(\mathbf{y})$$

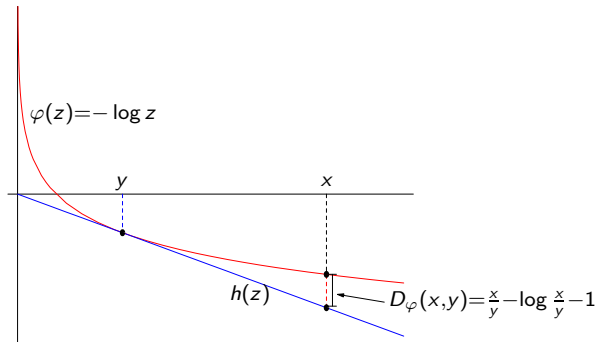


Relative Entropy (or KL-divergence) is another Bregman divergence

Bregman Divergences

- Let $\varphi : S \rightarrow \mathbb{R}$ be a differentiable, strictly convex function of “Legendre type” ($S \subseteq \mathbb{R}^d$)
- The Bregman Divergence $D_\varphi : S \times \text{relint}(S) \rightarrow \mathbb{R}$ is defined as

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla \varphi(\mathbf{y})$$



Itakura-Saito Dist.(used in signal processing) is also a Bregman divergence

Examples of Bregman Divergences

Function Name	$\varphi(x)$	$D_{\varphi}(x,y)$
Squared norm	$\frac{1}{2}x^2$	$\frac{1}{2}(x-y)^2$
Shannon entropy	$x \log x - x$	$x \log \frac{x}{y} - x + y$
Bit entropy	$x \log x + (1-x) \log(1-x)$	$x \log \frac{x}{y} + (1-x) \log \frac{1-x}{1-y}$
Burg entropy	$-\log x$	$\frac{x}{y} - \log \frac{x}{y} - 1$
Hellinger	$-\sqrt{1-x^2}$	$(1-xy)(1-y^2)^{-1/2} - (1-x^2)^{1/2}$
ℓ_p quasi-norm	$-x^p \quad (0 < p < 1)$	$-x^p + pxy^{p-1} - (p-1)y^p$
ℓ_p norm	$ x ^p \quad (1 < p < \infty)$	$ x ^p - p x \operatorname{sgn} y y ^{p-1} + (p-1) y ^p$
Exponential	e^x	$e^x - (x-y+1)e^y$
Inverse	$1/x$	$1/x + x/y^2 - 2/y$

Bregman Matrix Divergences

- Define

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \text{trace}((\nabla\varphi(Y))^T(X - Y))$$

Bregman Matrix Divergences

- Define

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \text{trace}((\nabla\varphi(Y))^T(X - Y))$$

- Squared Frobenius norm: $\varphi(X) = \|X\|_F^2$. Then

$$D_\varphi(X, Y) = \frac{1}{2}\|X - Y\|_F^2$$

Bregman Matrix Divergences

- Define

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \text{trace}((\nabla\varphi(Y))^T(X - Y))$$

- Squared Frobenius norm: $\varphi(X) = \|X\|_F^2$. Then

$$D_\varphi(X, Y) = \frac{1}{2}\|X - Y\|_F^2$$

- von Neumann Divergence: For $X \succeq 0$, $\varphi(X) = \text{trace}(X \log X)$. Then

$$D_\varphi(X, Y) = \text{trace}(X \log X - X \log Y - X + Y)$$

- also called quantum relative entropy

Bregman Matrix Divergences

- Define

$$D_\varphi(X, Y) = \varphi(X) - \varphi(Y) - \text{trace}((\nabla\varphi(Y))^T(X - Y))$$

- Squared Frobenius norm: $\varphi(X) = \|X\|_F^2$. Then

$$D_\varphi(X, Y) = \frac{1}{2}\|X - Y\|_F^2$$

- von Neumann Divergence: For $X \succeq 0$, $\varphi(X) = \text{trace}(X \log X)$. Then

$$D_\varphi(X, Y) = \text{trace}(X \log X - X \log Y - X + Y)$$

- also called quantum relative entropy

- LogDet divergence: For $X \succ 0$, $\varphi(X) = -\log \det X$. Then

$$D_\varphi(X, Y) = \text{trace}(XY^{-1}) - \log \det(XY^{-1}) - d$$

LogDet Divergence: Properties I

$$D_{\ell d}(X, Y) = \text{trace}(XY^{-1}) - \log \det(XY^{-1}) - d$$

- Properties:

- Not symmetric
- Triangle inequality does not hold
- Can be unbounded
- Convex in first argument (not in second)
- Pythagorean Property holds:

$$D_{\ell d}(X, Y) \geq D_{\ell d}(X, P_{\Omega}(Y)) + D_{\ell d}(P_{\Omega}(Y), Y)$$

- Divergence between inverses:

$$D_{\ell d}(X, Y) = D_{\ell d}(Y^{-1}, X^{-1})$$

LogDet Divergence: Properties II

$$\begin{aligned} D_{\ell d}(X, Y) &= \text{trace}(XY^{-1}) - \log \det(XY^{-1}) - d, \\ &= \sum_{i=1}^d \sum_{j=1}^d (\mathbf{v}_i^T \mathbf{u}_j)^2 \left(\frac{\lambda_i}{\theta_j} - \log \frac{\lambda_i}{\theta_j} - 1 \right) \end{aligned}$$

- Properties:

- Scale-invariance

$$D_{\ell d}(X, Y) = D_{\ell d}(\alpha X, \alpha Y), \quad \alpha \geq 0$$

- In fact, for any invertible M

$$D_{\ell d}(X, Y) = D_{\ell d}(M^T X M, M^T Y M)$$

LogDet Divergence: Properties II

$$\begin{aligned}D_{\ell d}(X, Y) &= \text{trace}(XY^{-1}) - \log \det(XY^{-1}) - d, \\&= \sum_{i=1}^r \sum_{j=1}^r (\mathbf{v}_i^T \mathbf{u}_j)^2 \left(\frac{\lambda_i}{\theta_j} - \log \frac{\lambda_i}{\theta_j} - 1 \right)\end{aligned}$$

- Properties:

- Scale-invariance

$$D_{\ell d}(X, Y) = D_{\ell d}(\alpha X, \alpha Y), \quad \alpha \geq 0$$

- In fact, for any invertible M

$$D_{\ell d}(X, Y) = D_{\ell d}(M^T X M, M^T Y M)$$

- Definition can be extended to rank-deficient matrices
- Finiteness:

$D_{\ell d}(X, Y)$ is finite iff X and Y have the same range space

Information-Theoretic Interpretation

- Differential Relative Entropy between two Multivariate Gaussians:

$$\int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_0) \log \left(\frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_0)}{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})} \right) d\mathbf{x} = \frac{1}{2} D_{\ell d}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}_0)$$

- Thus, the following two problems are equivalent

Relative Entropy Formulation

$$\min_{\boldsymbol{\Sigma}} \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_0) \log \frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_0)}{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})} d\mathbf{x}$$

$$\text{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u$$

$$\text{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell$$

$$\boldsymbol{\Sigma} \succeq 0$$

LogDet Formulation

$$\min_{\boldsymbol{\Sigma}} D_{\ell d}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}_0)$$

$$\Leftrightarrow \text{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u$$

$$\text{tr}(\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell$$

$$\boldsymbol{\Sigma} \succeq 0$$

Stein's Loss

- LogDet divergence is known as Stein's loss in the statistics community
- Stein's loss is the unique *scale invariant* loss-function for which the uniform minimum variance unbiased estimator is also a minimum risk equivariant estimator

Quasi-Newton Optimization

- LogDet Divergence arises in the BFGS and DFP updates
 - Quasi-Newton methods
 - Approximate Hessian of the function to be minimized
- [Fletcher, 1991] BFGS update can be written as:

$$\begin{aligned} \min_B \quad & D_{\ell d}(B, B_t) \\ \text{subject to} \quad & B s_t = y_t \quad (\text{"Secant Equation"}) \end{aligned}$$

- $s_t = x_{t+1} - x_t$, $y_t = \nabla f_{t+1} - \nabla f_t$
- Closed-form solution:

$$B_{t+1} = B_t - \frac{B_t s_t s_t^T B_t}{s_t^T B_t s_t} + \frac{y_t y_t^T}{s_t^T y_t}$$

- Similar form for DFP update

Algorithm: Bregman Projections for LogDet

- Algorithm: Cyclic Bregman Projections (successively onto each linear constraint) — converges to globally optimal solution
- Use Bregman projections to update the Mahalanobis matrix:

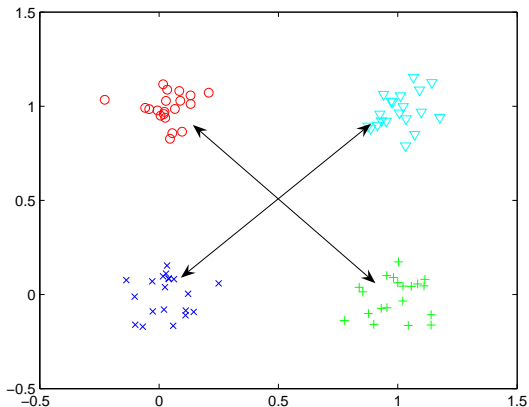
$$\begin{aligned} \min_{\Sigma} \quad & D_{ld}(\Sigma, \Sigma_t) \\ \text{s.t.} \quad & (\mathbf{x}_i - \mathbf{x}_j)^T \Sigma (\mathbf{x}_i - \mathbf{x}_j) \leq u \end{aligned}$$

- Can be solved by rank-one update:

$$\Sigma_{t+1} = \Sigma_t + \beta_t \Sigma_t (\mathbf{x}_i - \mathbf{x}_j) (\mathbf{x}_i - \mathbf{x}_j)^T \Sigma_t$$

- Advantages:
 - Automatic enforcement of positive semidefiniteness
 - Simple, closed-form projections
 - No eigenvector calculation
 - Easy to incorporate slack for each constraint

Example: Noisy XOR



- No linear transformation for XOR grouping

Kernel Methods

- Map input data to higher-dimensional “feature” space:

$$\mathbf{x} \rightarrow \varphi(\mathbf{x})$$

- Idea: Run machine learning algorithm in feature space
- Noisy XOR Example:

$$\mathbf{x} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

Kernel Methods

- Map input data to higher-dimensional “feature” space:

$$\mathbf{x} \rightarrow \varphi(\mathbf{x})$$

- Idea: Run machine learning algorithm in feature space
- Noisy XOR Example:

$$\mathbf{x} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Kernel function: $K(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle$
- “Kernel trick” — no need to explicitly form high-dimensional features
- Noisy XOR Example: $\langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle = (\mathbf{x}^T \mathbf{y})^2$

Connection to Kernel Learning

LogDet Formulation (1)

$$\begin{aligned} \min_{\Sigma} \quad & D_{\ell d}(\Sigma, \Sigma_0) \\ \text{s.t.} \quad & \text{tr}(\Sigma(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u \\ & \text{tr}(\Sigma(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell \\ & \Sigma \succeq 0 \end{aligned}$$

Kernel Formulation (2)

$$\begin{aligned} \min_K \quad & D_{\ell d}(K, K_0) \\ \text{s.t.} \quad & \text{tr}(K(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T) \leq u \\ & \text{tr}(K(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T) \geq \ell \\ & K \succeq 0 \end{aligned}$$

- (1) optimizes w.r.t. the $d \times d$ Mahalanobis matrix Σ
- (2) optimizes w.r.t. the $N \times N$ kernel matrix K
- Let $K_0 = X^T \Sigma_0 X$, where X is the input data
- Let Σ^* be optimal solution to (1) and K^* be optimal solution to (2)
- **Theorem:** $K^* = X^T \Sigma^* X$

Kernelization

- Metric learning in kernel space
 - Assume input kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y})$
 - Want to learn

$$d_{\Sigma}(\varphi(\mathbf{x}), \varphi(\mathbf{y})) = (\varphi(\mathbf{x}) - \varphi(\mathbf{y}))^T \Sigma (\varphi(\mathbf{x}) - \varphi(\mathbf{y}))$$

- Equivalently: learn a new kernel function of the form

$$\tilde{\kappa}(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \Sigma \varphi(\mathbf{y})$$

- How to learn this only using $\kappa(\mathbf{x}, \mathbf{y})$?
- Learned kernel can be shown to be of the form

$$\tilde{\kappa}(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y}) + \sum_i \sum_j \sigma_{ij} \kappa(\mathbf{x}, \mathbf{x}_i) \kappa(\mathbf{y}, \mathbf{x}_j)$$

- Can update σ_{ij} parameters while optimizing the kernel formulation

Related Work

- Distance Metric Learning [Xing, Ng, Jordan & Russell, 2002]
- Large margin nearest neighbor(LMNN) [Weinberger, Blitzer & Saul, 2005]
- Collapsing Classes (MCML) [Globerson & Roweis, 2005]
- Online Metric Learning (POLA) [Shalev-Shwartz, Singer & Ng, 2004]
- Many others!

Experimental Results

Framework

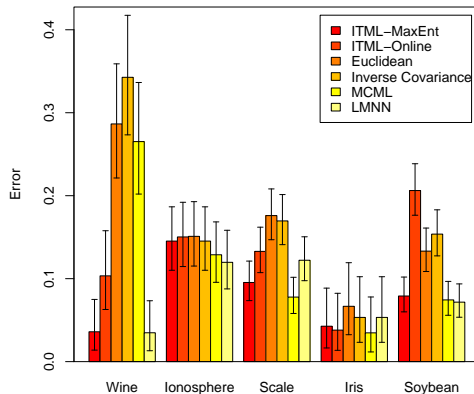
- k -nearest neighbor ($k = 4$)
- ℓ and u determined by 5th and 95th percentile of distribution
- $20c^2$ constraints, chosen randomly
- 2-fold cross validation

Algorithms

- Information-theoretic Metric Learning (offline and online)
- Large-Margin Nearest Neighbors (LMNN) [Weinberger et al.]
- Metric Learning by Collapsing Classes (MCML) [Globerson and Roweis]
- Baseline Metrics: Euclidean and Inverse Covariance

Results: UCI Data Sets

- Ran ITML with $\Sigma_0 = I$ (ITML-MaxEnt) and the inverse covariance (InverseCovariance)
- Ran online algorithm for 10^5 iterations



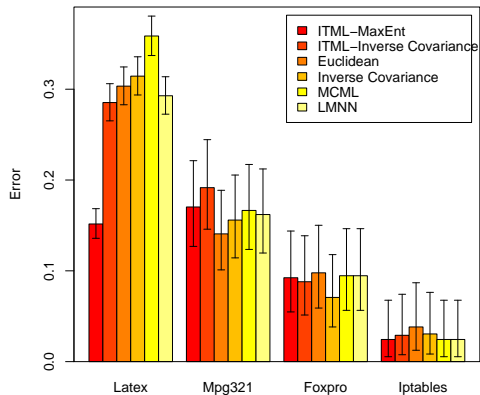
Application 1: “Clarify”

“Clarify” improves error reporting for software that uses black box components

- Motivation: Black box components complicate error messaging
- Solution: Error diagnosis via machine learning
- Representation: System collects program features during run-time
 - Function counts
 - Call-site counts
 - Counts of program paths
 - Program execution represented as a vector of counts
- Class labels: Program execution errors
- Nearest neighbor software support
 - Match program executions with others
 - Underlying distance measure should reflect this similarity

Results: Clarify

- Very high dimensionality
- Feature selection reduces the number of features to 20



Application 2: Learning Image Similarity

Goal: Learn a metric to compare images

- Start with a baseline measure
 - Use the **pyramid match kernel** [Grauman and Darrell]
 - Compares sets of image features
 - Efficient and effective measure of similarity between images
- Application of metric learning in kernel space
 - Other metric learning methods (LMNN, etc) cannot be applied
 - Does metric learning work in kernel space?

Caltech 101 Results

Data Set: Caltech 101

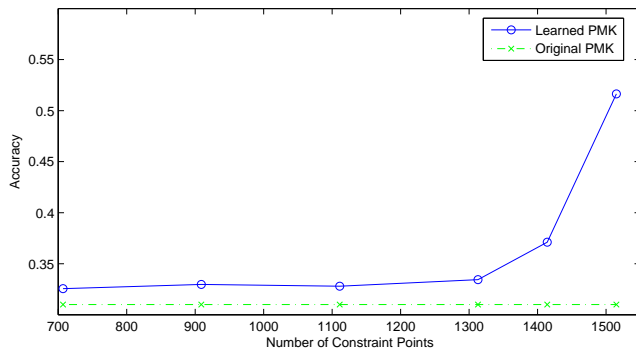
- Standard benchmark for multi-category image recognition
- 101 classes of images
- Wide variance in pose etc.
- Challenging data set

Experimental Setup

- 15 images per class in training set; rest in test set (2454 images)
- Performed 1-NN using original PMK and learned PMK



Caltech 101 Results



- When constraints are drawn from all training data, kNN accuracy is 52%, versus 32% for original PMK ([Jain, Kulis & Grauman, 2007])
- Data set is well-studied—best performance with 15 training images per class is 60%
 - Uses different features (geometric-blur)

Metric Learning in High Dimensions

Text analysis & Software analysis: Feature sets larger than 1,000

- Learning full distance matrix requires over 1 million parameters!
- Overfitting problems, intractable

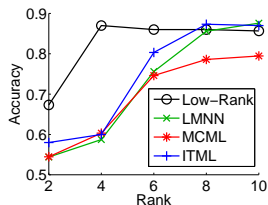
Solution: Learning low-rank Mahalanobis matrices

- LogDet divergence can be generalized to low-rank matrices
- $D_{\ell d}(X, Y)$ is finite $\leftrightarrow X$ and Y have the same range space
- Extending ITML to the low-rank case
 - If Σ_0 is low-rank $\rightarrow \Sigma$ is low-rank
 - Clustered Mahalanobis matrices
 - If Σ_0 is a block matrix, then Σ will also be a block matrix

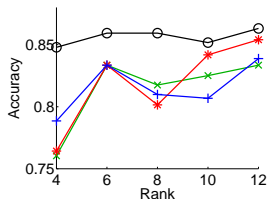
Low-rank Metric Learning: Preliminary Results

Classification accuracy for Low-Rank ITML

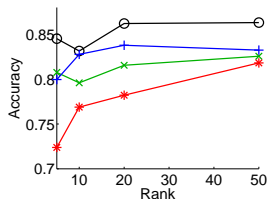
- Classic3: Text dataset, PCA basis
- Mpg321, Gcc: Software analysis, Clustered basis



Classic3



Mpg321



Gcc

Conclusions

Metric Learning Formulation

- Uses LogDet divergence
- Information-theoretic interpretation
- Equivalent to kernel learning problem
 - Can improve upon input metric or kernel
 - Generalizes to unseen data points

Algorithm

- Bregman projections result in simple rank-one updates
- Can be kernelized
- Online variant has provable regret bounds

Empirical Evaluation

- Method is competitive with existing techniques
- Scalable to large data sets
- Applied to nearest-neighbor software support & image recognition

References

- J. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, “Information-Theoretic Metric Learning”, *International Conference on Machine Learning(ICML)*, pages 209–216, June 2007.
- B. Kulis, M. Sustik, and I. S. Dhillon, “Learning Low-Rank Kernel Matrices”, *International Conference on Machine Learning(ICML)*, pages 505–512, July 2006 (longer version submitted to JMLR).