Information-theoretic Metric Learning

Brian Kulis
University of Texas at Austin

24th International Conference on Machine Learning
Corvallis, OR, USA
June 21, 2007

Joint work with Jason V. Davis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon

Introduction

Metric Learning

- Important problem in machine learning
  - Metric governs success or failure of learning algorithm
  - Exploit distance information intrinsically available
  - Useful in many domains
- Several existing approaches
  - SDP approach [Xing et al.]
  - Large-margin nearest neighbor (LMNN) [Weinberger et al.]
  - Collapsing Classes (MCML) [Globerson and Roweis]
  - Online Metric Learning (POLA) [Shalev-Shwartz et al.]
  - Many others!

Our Approach

- Contributions
  - Simple and scalable
  - Incorporates a variety of constraints
  - Online learning variant with regret bounds
  - Allows kernelization for learning a kernel function
- Existing methods fail to satisfy all the above

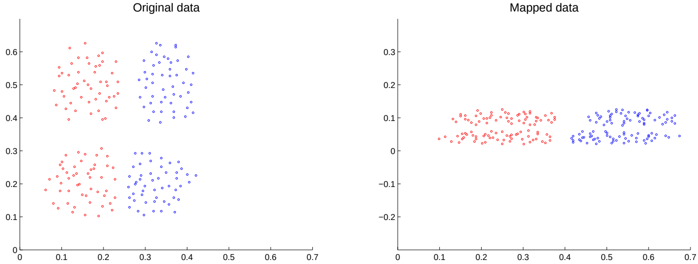
Mahalanobis Distances

Like many others, we consider learning *Mahalanobis distances*

- Distance parameterized by p.d. $d \times d$ matrix A:

$$d_A(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T A(\mathbf{x} - \mathbf{y})$$

- Often A is inverse of the covariance matrix
- Generalizes squared Euclidean distance ($A = I$)
- Rotates and scales input data
- Standard for metric learning



Problem Formulation

Introduce constraints—Examples:

- Assume pairwise similarity and dissimilarity constraints

$$\begin{aligned} d_A(\mathbf{x}_i, \mathbf{x}_j) &\leq u \quad \text{if } (i,j) \in S \text{ [similarity constraints]} \\ d_A(\mathbf{x}_i, \mathbf{x}_j) &\geq \ell \quad \text{if } (i,j) \in D \text{ [dissimilarity constraints]} \end{aligned}$$

- Other linear constraints on A possible

Goal:

- Learn a metric d_A which is "close" to some starting metric d_{A_0}
  - d_A satisfies additional prespecified constraints
  - Need notion of distance between metrics

The Gaussian Connection

- Exploit connection to Gaussian distributions:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A) = \frac{1}{Z}\exp\left(-\frac{1}{2}d_A(\mathbf{x}, \boldsymbol{\mu})\right)$$

- Bijection between set of Mahalanobis distances and set of Gaussians with fixed $\boldsymbol{\mu}$
- Compare Gaussians using relative entropy

$$\begin{array}{ccc} d_A(\mathbf{x}, \mathbf{y}) & & d_{A_0}(\mathbf{x}, \mathbf{y}) \\ \updownarrow & & \updownarrow \\ \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A) & \leftrightarrow & \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0) \\ & \uparrow & \\ & \textbf{Differential Relative Entropy} & \end{array}$$

The Optimization Problem

$$\begin{aligned} \min_A \quad & \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0) \log \left(\frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0)}{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A)} \right) d\mathbf{x} \\ \text{subject to} \quad & d_A(\mathbf{x}_i, \mathbf{x}_j), \ \leq u, \ (i,j) \in S \\ & d_A(\mathbf{x}_i, \mathbf{x}_j) \geq \ell, \ (i,j) \in D \\ & A \succeq 0 \end{aligned}$$

- Utilize connection between KL-divergence and the LogDet divergence (assume $\boldsymbol{\mu}$ fixed)

$$\int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0) \log \left(\frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0)}{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A)} \right) d\mathbf{x} = \frac{1}{2} D_{\ell d}(A, A_0)$$

$$D_{\ell d}(A, A_0) = \text{trace}(AA_0^{-1}) - \log\det(AA_0^{-1}) - d$$

The Optimization Problem

KL Formulation		**LogDet Formulation**
$\min_A \quad \int \mathcal{N}(\mathbf{x}\vert\boldsymbol{\mu}, A_0) \log \frac{\mathcal{N}(\mathbf{x}\vert\boldsymbol{\mu}, A_0)}{\mathcal{N}(\mathbf{x}\vert\boldsymbol{\mu}, A)} d\mathbf{x}$		$D_{\ell d}(A, A_0)$
$\text{s.t.} \quad d_A(\mathbf{x}_i, \mathbf{x}_j) \leq u$	$\Leftrightarrow$	$\mathrm{tr}(A(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u$
$d_A(\mathbf{x}_i, \mathbf{x}_j) \geq \ell$		$\mathrm{tr}(A(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell$
$A \succeq 0$		$A \succeq 0$

- Utilize connection between KL-divergence and the LogDet divergence (assume $\boldsymbol{\mu}$ fixed)

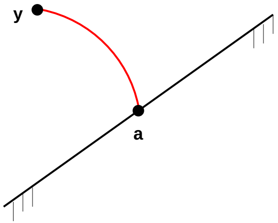
$$\int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0) \log \left(\frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A_0)}{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A)} \right) d\mathbf{x} = \frac{1}{2} D_{\ell d}(A, A_0)$$

$$D_{\ell d}(A, A_0) = \mathrm{trace}(AA_0^{-1}) - \log\det(AA_0^{-1}) - d$$

Bregman's Method

- Use the "Bregman" projection of **y** onto affine set $\mathcal{H}$,

$$P_{\mathcal{H}}(\mathbf{y}) = \underset{\mathbf{a}\in\mathcal{H}}{\operatorname{argmin}}\, D_{\varphi}(\mathbf{a}, \mathbf{y})$$



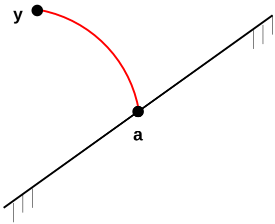
- Method projects onto each constrain and applies correction
- Dual Coordinate Ascent
- A has rank-one update (no eigenvector calculation):

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

Bregman's Method

- Use the "Bregman" projection of **y** onto affine set $\mathcal{H}$,

$$P_{\mathcal{H}}(\mathbf{y}) = \underset{\mathbf{a}\in\mathcal{H}}{\operatorname{argmin}}\, D_{\varphi}(\mathbf{a}, \mathbf{y})$$



- Advantages:
  - Scalable
  - Automatic enforcement of positive semidefiniteness
  - Simple, closed-form projections
  - No eigenvector calculation

Connection to Kernel Learning

LogDet Formulation (1)

$$\begin{aligned} \min_A \quad & D_{\ell d}(A, A_0) \\ \text{s.t.} \quad & \mathrm{tr}(A(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \leq u \\ & \mathrm{tr}(A(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T) \geq \ell \\ & A \succeq 0 \end{aligned}$$

Kernel Formulation (2)

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

- (1) opt. w.r.t. the Mahalanobis matrix, (2) w.r.t. the kernel matrix
- Let $K_0 = X^T A_0 X$, where X is the input data
- Let A^* be the opt. solution to (1) and K^* be the opt. solution to (2)
- **Theorem:** $K^* = X^T A^* X$

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

$$d_A(\varphi(\mathbf{x}), \varphi(\mathbf{y})) = (\varphi(\mathbf{x}) - \varphi(\mathbf{y}))^T A(\varphi(\mathbf{x}) - \varphi(\mathbf{y}))$$

  - Equivalently: learn a new kernel function of the form

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

  - 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

Online Metric Learning

Setup

- Want to learn metric in online setting
- Every timestep t, receive pair of points $(\mathbf{x}_t, \mathbf{y}_t)$
- Predict distance between $\mathbf{x}_t$ and $\mathbf{y}_t$, then receive "true" distance
- Record loss at step t, update A_t to A_{t+1}
- **Goal:** minimize total loss

Regret Bounds

- L_{OML}: total loss of online metric learning algorithm
- L_{A^*}: total loss of best offline algorithm

$$L_{OML} \leq r_1 L_{A^*} + r_2 D_{\ell d}(A^*, I)$$

- r_1, r_2 functions of the learning rate of the algorithm

Experimental Results

Framework

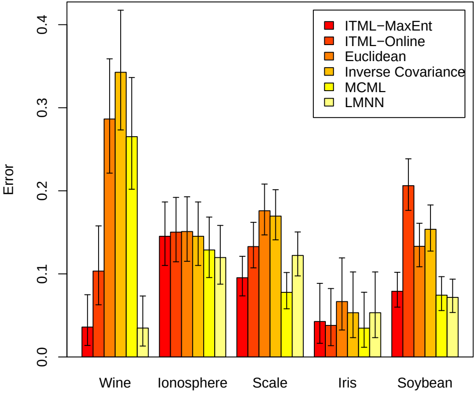
- k-nearest neighbor ($k = 4$)
- ℓ and u determined by 5th and 95th percentile of distribution
- $20c^2$ constraints, chosen randomly
- 2-fold cross validation
- Binomial confidence intervals at the 95% level

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

UCI Data Sets

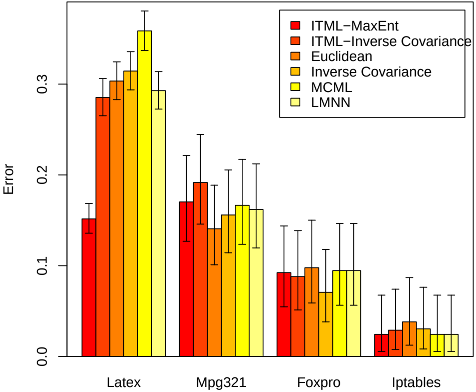
- Ran ITML with $A_0 = I$ (ITML-MaxEnt) and the inverse covariance (InverseCovariance)
- Ran online algorithm for 10^5 iterations



- ITML-MaxEnt is the best performing algorithm (within the 95% confidence intervals) across all data sets

Clarify Data Sets

- Classification for nearest-neighbor software support
- Clarify monitors predefined program features
- Each program run is one data point
- Very high dimensionality
- Feature selection reduces the number of features to 20



- ITML-MaxEnt is the best performing algorithm (within the 95% confidence intervals) across all data sets

Conclusions

Formulation

- Minimizes the relative entropy between 2 Gaussians
- Connection to LogDet divergence
- Many different constraints may be enforced

Algorithm

- Applies Bregman projections—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
- Application to nearest-neighbor software support