Tutorial on Metric Learning

Aurélien Bellet

Department of Computer Science
Viterbi School of Engineering
University of Southern California

USC Viterbi
School of Engineering

Computational Intelligence and Learning Doctoral School
October 15, 2013

Quick advertisement

Recent survey

All the topics, methods and references covered in this tutorial (and others) are discussed at more length in my recent survey (joint work with Amaury Habrard and Marc Sebban).

Reference

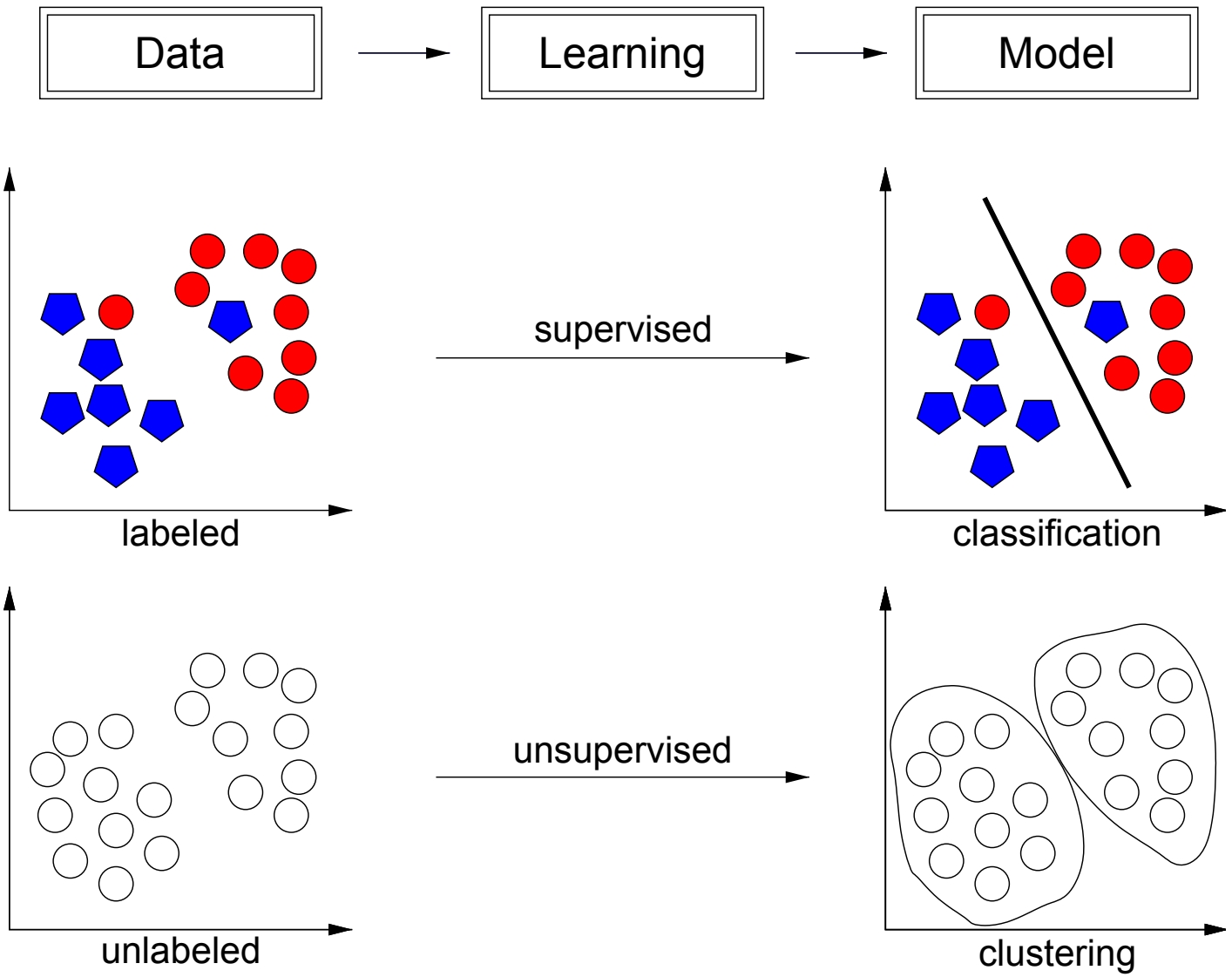
Bellet, A., Habrard, A., and Sebban, M. (2013). A Survey on Metric Learning for Feature Vectors and Structured Data. Technical report, arXiv:1306.6709

Download from arXiv

`http://arxiv.org/abs/1306.6709`

Machine learning

Learn to generalize from examples



Numerical and structured data

Numerical data

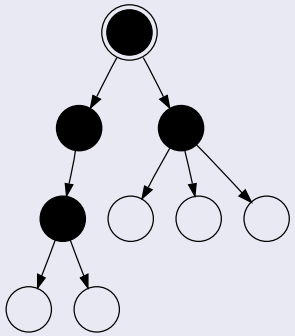
- Each data instance is a numerical feature vector.
- Example: the age, body mass index, blood pressure, ... of a patient.

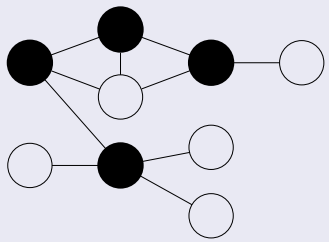
$$\mathbf{x} = \begin{pmatrix} 26 \\ 21.6 \\ 102 \\ \dots \end{pmatrix}$$

Structured data

- Each instance is a structured object: a string, a tree or a graph.
- Examples: French words, DNA sequences, XML documents, molecules, social communities...

ACGGCTT





Importance of metrics

Pairwise metric

Informally, a way of measuring the distance (or similarity) between objects.

Metrics are ubiquitous in machine learning

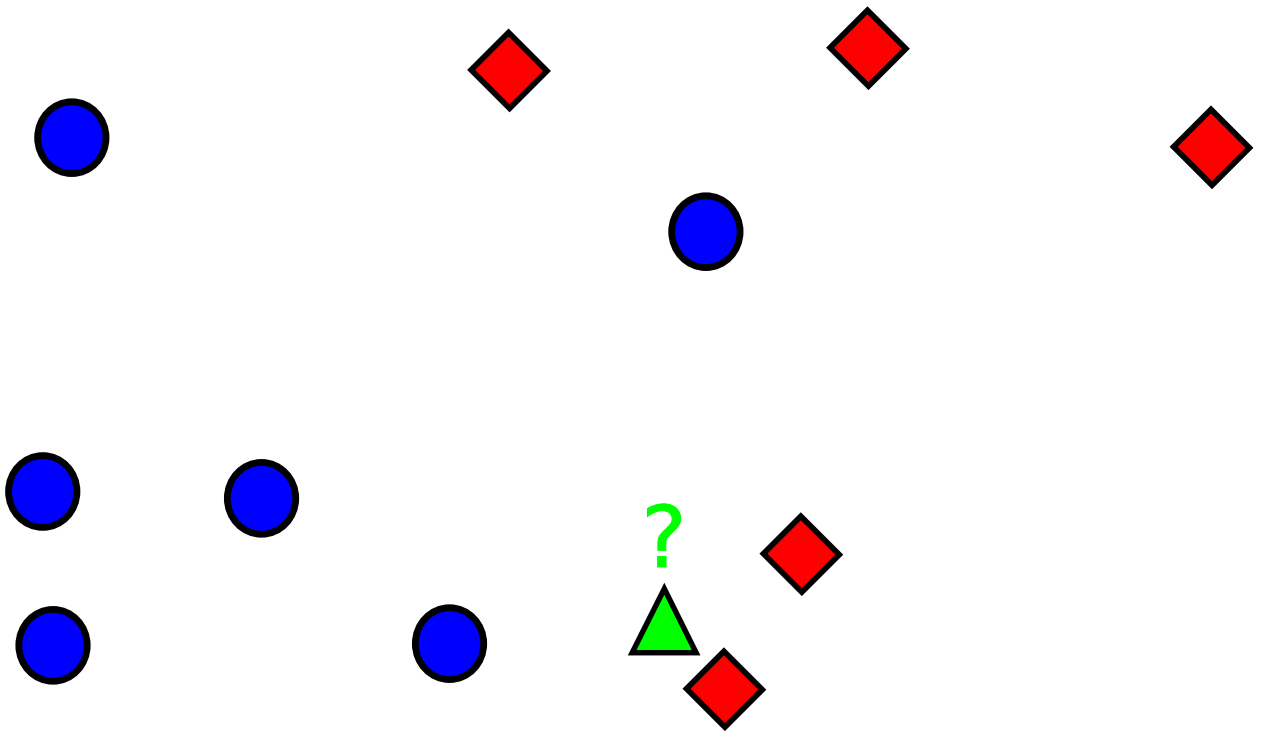
- Get yourself a good metric and you've basically solved the problem.
- Metrics are convenient proxies to manipulate complex objects.

Applications

- Classification: k-Nearest Neighbors, Support Vector Machines...
- Clustering: K-Means and its variants.
- Information Retrieval / Ranking: search by query, image retrieval...
- Data visualization in high dimensions.
- ...

Importance of metrics

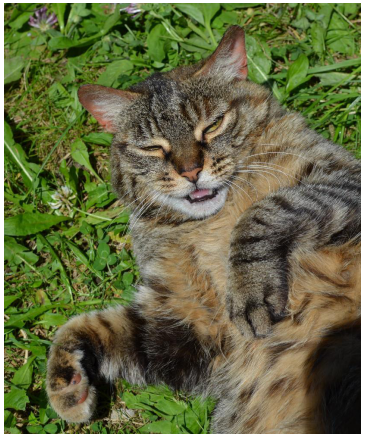
Application: classification



Importance of metrics

Application: document retrieval

Query image



Most similar images





Metric learning

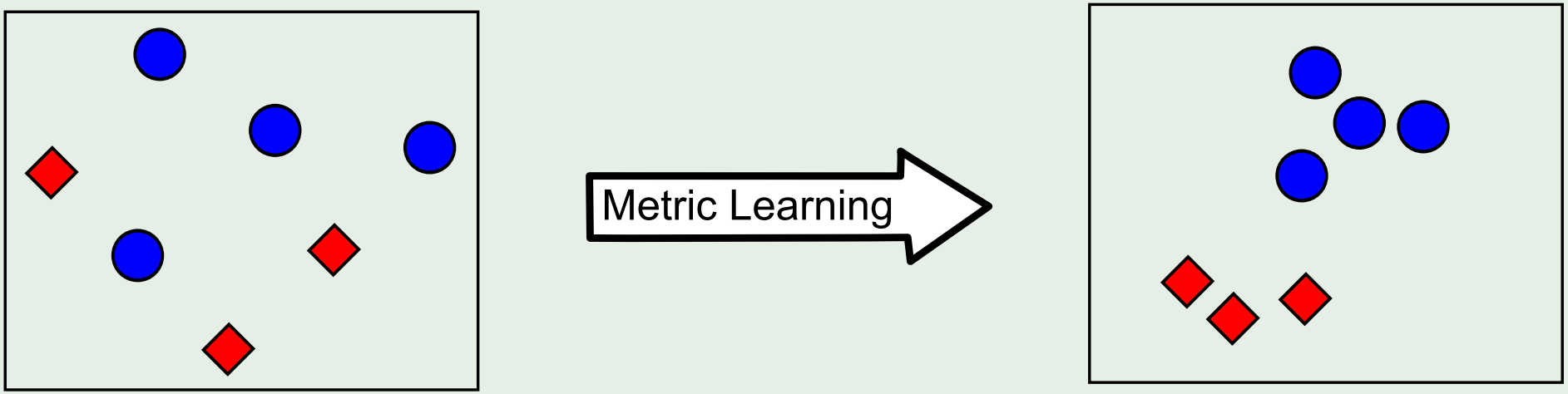
Adapt the metric to the problem of interest

The notion of good metric is problem-dependent

Each problem has its own semantic notion of similarity, which is often badly captured by standard metrics (e.g., Euclidean distance).

Solution: learn the metric from data

Basic idea: learn a metric that assigns small (resp. large) distance to pairs of examples that are semantically similar (resp. dissimilar).



Outline

1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Outline

1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Definition (Distance function)

A distance over a set $\mathcal{X}$ is a pairwise function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies the following properties $\forall x, x', x'' \in \mathcal{X}$:

1. $d(x, x') \geq 0$ (nonnegativity),
2. $d(x, x') = 0$ if and only if $x = x'$ (identity of indiscernibles),
3. $d(x, x') = d(x', x)$ (symmetry),
4. $d(x, x'') \leq d(x, x') + d(x', x'')$ (triangle inequality).

Definition (Similarity function)

A (dis)similarity function is a pairwise function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. K is symmetric if $\forall x, x' \in \mathcal{X}$, $K(x, x') = K(x', x)$.

Minkowski distances

Minkowski distances are a family of distances induced by ℓ_p norms:

$$d_p(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_p = \left(\sum_{i=1}^{d} |x_i - x_i'|^p \right)^{1/p},$$

for $p \geq 1$. We can recover three widely used distances:

- For $p = 1$, the Manhattan distance $d_{man}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} |x_i - x_i'|$.
- For $p = 2$, the "ordinary" Euclidean distance:

$$d_{euc}(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{d} |x_i - x_i'|^2 \right)^{1/2} = \sqrt{(\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}')}$$

.

- For $p \to \infty$, the Chebyshev distance $d_{che}(\mathbf{x}, \mathbf{x}') = \max_i |x_i - x_i'|$.

The Mahalanobis distance

The Mahalanobis (pseudo) distance is defined as follows:

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')},$$

The Mahalanobis distance

The Mahalanobis (pseudo) distance is defined as follows:

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')},$$

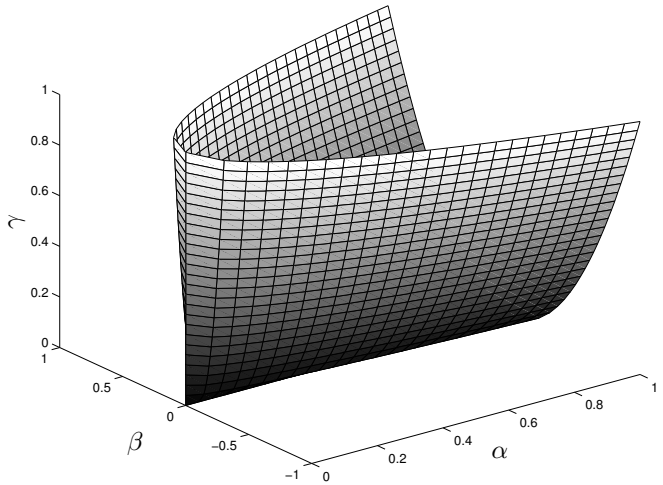
where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a symmetric PSD matrix.

The original term refers to the case where $\mathbf{x}$ and $\mathbf{x}'$ are random vectors from the same distribution with covariance matrix $\mathbf{\Sigma}$, with $\mathbf{M} = \mathbf{\Sigma}^{-1}$.

Disgression: PSD matrices

Definition (PSD matrix)

A matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ is positive semi-definite (PSD) if all its eigenvalues are nonnegative. The cone of symmetric PSD $d \times d$ real-valued matrices is denoted by $\mathbb{S}_+^d$. As a shortcut for $\mathbf{M} \in \mathbb{S}_+^d$ we use $\mathbf{M} \succeq 0$.



Useful properties

If $\mathbf{M} \succeq 0$, then

- $\mathbf{x}^T\mathbf{M}\mathbf{x} \geq 0 \ \forall \mathbf{x}$ (as a linear operator, can be seen as nonnegative scaling).
- $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ for some matrix $\mathbf{L}$.

(in fact each of these is sufficient for $\mathbf{M} \succeq 0$)

Cosine similarity

The cosine similarity measures the cosine of the angle between two instances, and can be computed as

$$K_{cos}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}.$$

It is widely used in data mining (better notion of similarity for bag-of-words + efficiently computable for sparse vectors).

Bilinear similarity

The bilinear similarity is related to the cosine but does not include normalization and is parameterized by a matrix $\mathbf{M}$:

$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}',$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is not required to be PSD nor symmetric.

String edit distance

The edit distance is the cost of the cheapest sequence of operations (*script*) turning a string into another. Allowable operations are *insertion*, *deletion* and *substitution* of symbols. Costs are gathered in a matrix $\mathbf{C}$.

Properties

- It is a proper distance if and only if $\mathbf{C}$ satisfies:

$$C_{ij} \geq 0, \quad C_{ij} = C_{ji}, \quad C_{ik} \leq C_{ij} + C_{jk} \quad \forall i, j, k.$$

- The edit distance can be computed in $O(mn)$ time by dynamic programming (m, n length of the two strings).
- Generalization to trees (quadratic or cubic complexity) and graphs (NP-complete).

Example 1: Standard (Levenshtein) distance

C	$	a	b
$	0	1	1
a	1	0	1
b	1	1	0

$\Longrightarrow$ edit distance between abb and aa is 2 (needs at least two operations)

Example 2: Specific Cost Matrix

C	$	a	b
$	0	2	10
a	2	0	4
b	10	4	0

$\Longrightarrow$ edit distance between abb and aa is 10 ($\mathtt{a} \to \$, \mathtt{b} \to \mathtt{a}, \mathtt{b} \to \mathtt{a}$)

$: empty symbol, Σ: alphabet, **C**: $(|\Sigma| + 1) \times (|\Sigma| + 1)$ matrix with positive values.

Definition (Convex set)

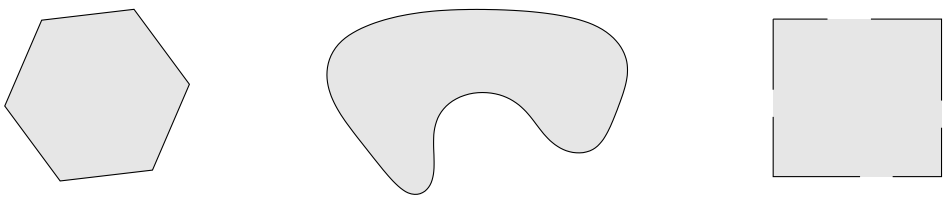
A **convex set** C contains line segment between any two points in the set.

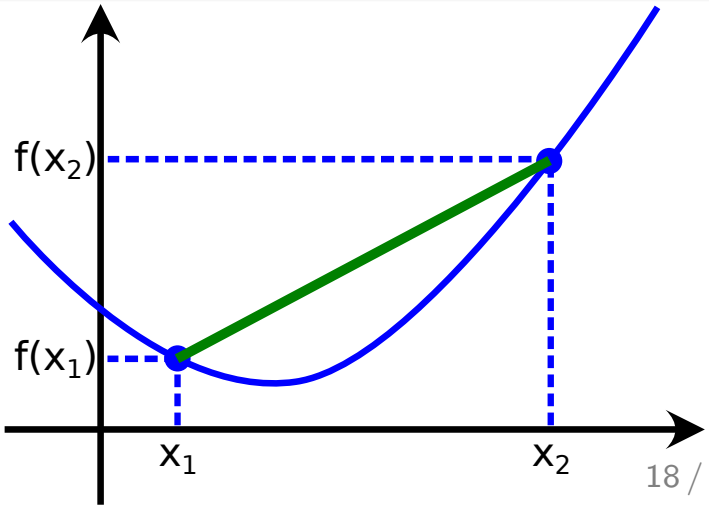
$$\mathbf{x}_1, \mathbf{x}_2 \in C, \quad 0 \leq \alpha \leq 1 \quad \Longrightarrow \quad \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in C$$

Definition (Convex function)

The function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if

$$\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, \quad 0 \leq \alpha \leq 1 \quad \Longrightarrow \quad f(\alpha\mathbf{x}_1+(1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1)+(1-\alpha)f(\mathbf{x}_2)$$





Convex optimization in a nutshell

Convex functions

Useful fact

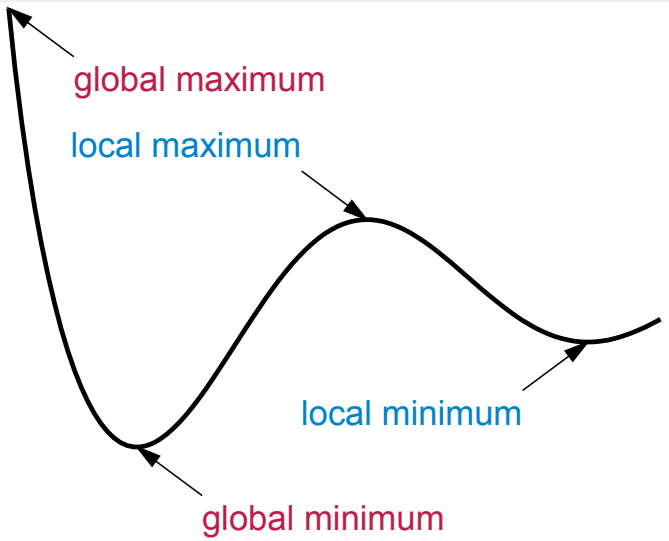
The function $f : \mathbb{R}^d \to \mathbb{R}$ is convex iff its Hessian matrix $\nabla^2 f(\mathbf{x})$ is PSD.

Example

A quadratic function $f(\mathbf{x}) = a + \mathbf{b}^T\mathbf{x} + \mathbf{x}^T\mathbf{Q}\mathbf{x}$ is convex iff $\mathbf{Q}$ is PSD.

A key property

If $f : \mathbb{R}^d \to \mathbb{R}$ is convex, then any local minimum of f is also a global minimum of f.



General formulation

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{C}, \end{aligned}$$

where f is a convex function and $\mathcal{C}$ is a convex set.

Convex optimization is (generally) tractable

A global minimum can be found (up to the desired precision) in time polynomial in the dimension and number of constraints.

What about nonconvex optimization?

In general, cannot guarantee global optimality of solution.

Convex optimization in a nutshell

Projected (sub)gradient descent

Basic algorithm when f is (sub)differentiable

1: Let $\mathbf{x}^{(0)} \in \mathcal{C}$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{x}^{(k+\frac{1}{2})} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$ for some $\alpha > 0$ *(gradient descent step)*
4: $\quad \mathbf{x}^{(k+1)} = \prod_{\mathcal{C}}(\mathbf{x}^{(k+\frac{1}{2})})$ *(projection step)*
5: **end for**

demo for unconstrained case: `http://goo.gl/7Q46EA`

Euclidean projection

The Euclidean projection of $\mathbf{x} \in \mathbb{R}^d$ onto $\mathcal{C} \subseteq \mathbb{R}^d$ is defined as follows:

$$\prod_{\mathcal{C}}(\mathbf{x}) = \arg\min_{\mathbf{x}' \in \mathcal{C}} \|\mathbf{x} - \mathbf{x}'\|.$$

Convergence

With proper step size, this procedure converges to a local optimum (thus a global optimum for convex optimization) and has a $O(1/k)$ convergence rate (or better).

Outline

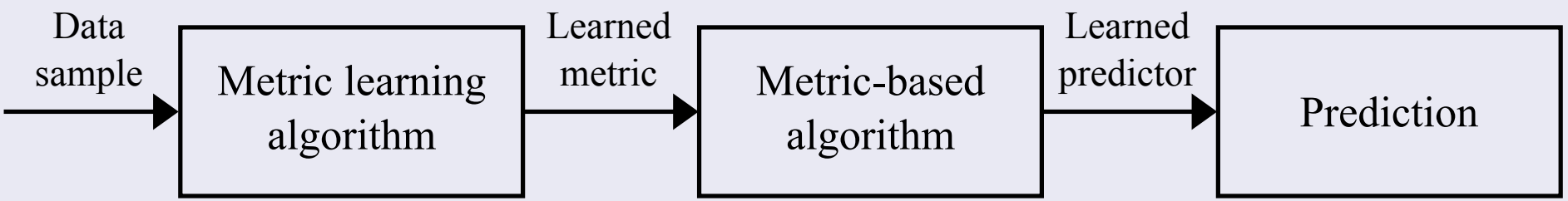
1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Metric learning in a nutshell

A hot research topic

- Really started with a paper at NIPS 2002 (cited over 1,300 times).
- Ever since, several papers each year in top conferences and journals.
- Since 2010, tutorials and workshops at major machine learning (NIPS, ICML) and computer vision (ICCV, ECCV) venues.

Common process



Learning from side information

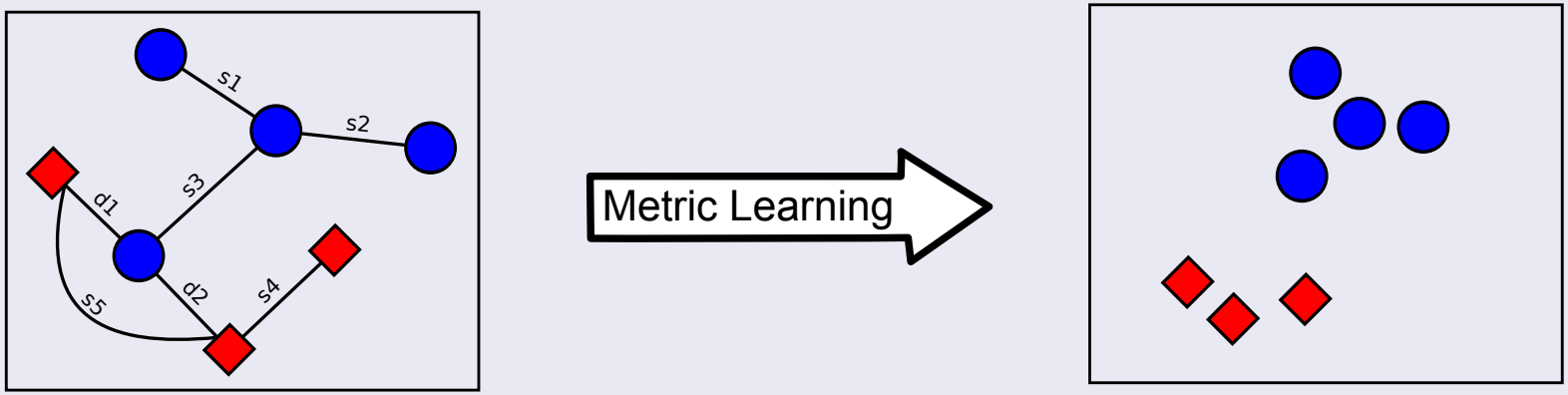
- Must-link / cannot-link constraints:

$$\begin{aligned} \mathcal{S} &= \{(x_i, x_j) : x_i \text{ and } x_j \text{ should be similar}\}, \\ \mathcal{D} &= \{(x_i, x_j) : x_i \text{ and } x_j \text{ should be dissimilar}\}. \end{aligned}$$

- Relative constraints:

$$\mathcal{R} = \{(x_i, x_j, x_k) : x_i \text{ should be more similar to } x_j \text{ than to } x_k\}.$$

Geometric intuition: learn a projection of the data



Metric learning in a nutshell

Basic setup

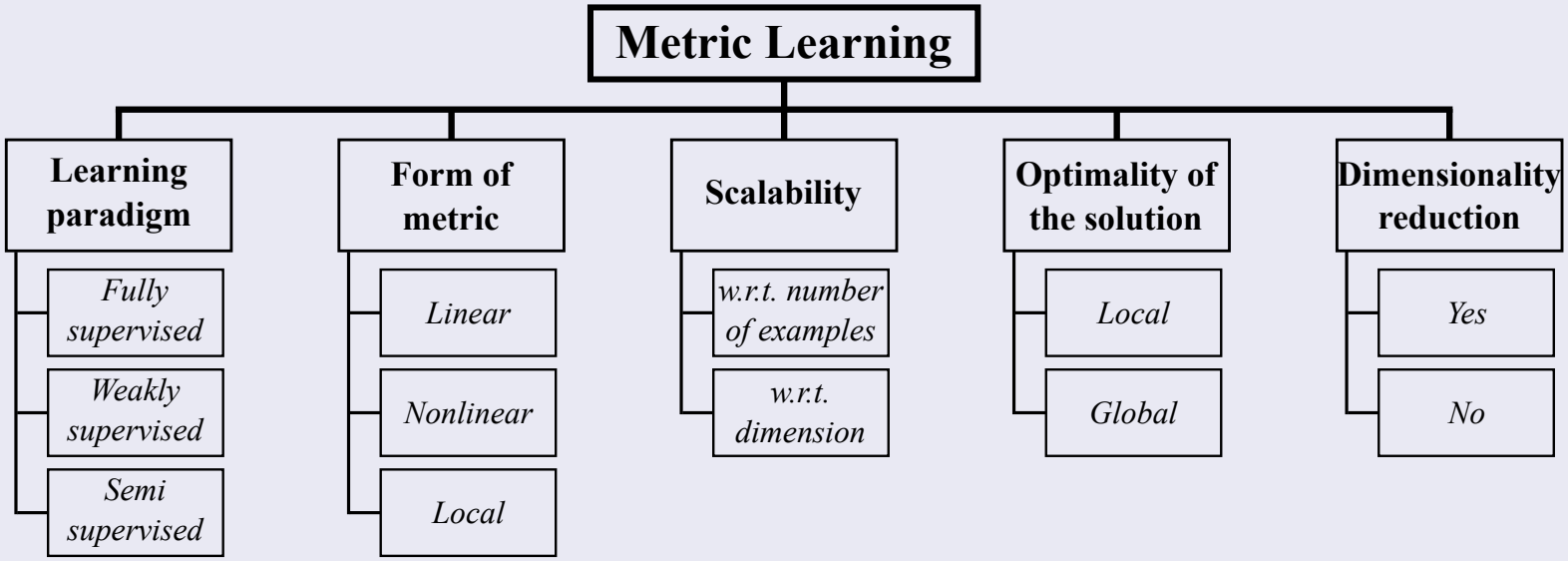
General formulation

Given a metric, find its parameters $\mathbf{M}^*$ as

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \left[\ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R}) + \lambda R(\mathbf{M})\right],$$

where $\ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R})$ is a loss function that penalizes violated constraints, $R(\mathbf{M})$ is some regularizer on $\mathbf{M}$ and $\lambda \geq 0$ is the regularization parameter.

Five key properties of algorithms



Applications in virtually any field

Whenever the notion of metric plays an important role. Fun examples of applications include music recommendation, identity verification, cartoon synthesis and assessing the efficacy of acupuncture, to name a few.

Main fields of application

- Computer Vision: compare images or videos in ad-hoc representations. Used in image classification, object/face recognition, tracking, image annotation...
- Information retrieval: rank documents by relevance.
- Bioinformatics: compare structured objects such as DNA sequences or temporal series.

This tutorial

Metric learning as optimizing a parametric notion of distance or similarity in a fully/weakly/semi supervised way. Illustrate the main techniques by going through some representative algorithms.

Related topics (not covered)

- Kernel learning: typically nonparametric (learn the Gram matrix).
- Multiple Kernel Learning: learn to combine a set of predefined kernels.
- Dimensionality reduction: often unsupervised, primary goal is really to reduce data dimension.

Outline

1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Mahalanobis distance learning

More on the Mahalanobis distance

Recap

The Mahalanobis (pseudo) distance is defined as follows:

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')},$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a symmetric PSD matrix.

Interpretation

Equivalent to a Euclidean distance after a linear projection $\mathbf{L}$:

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') &= \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')} = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{x}')} \\ &= \sqrt{(\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')^T (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')}. \end{aligned}$$

If $\mathbf{M}$ has rank k, $\mathbf{L} \in \mathbb{R}^{k \times d}$ (dimensionality reduction).

Remark

For convenience, often use the squared distance (linear in $\mathbf{M}$).

Mahalanobis distance learning

MMC [Xing et al., 2002]

Formulation

$$\max_{\mathbf{M} \in \mathbb{R}^{d \times d}} \quad \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq 1,$$

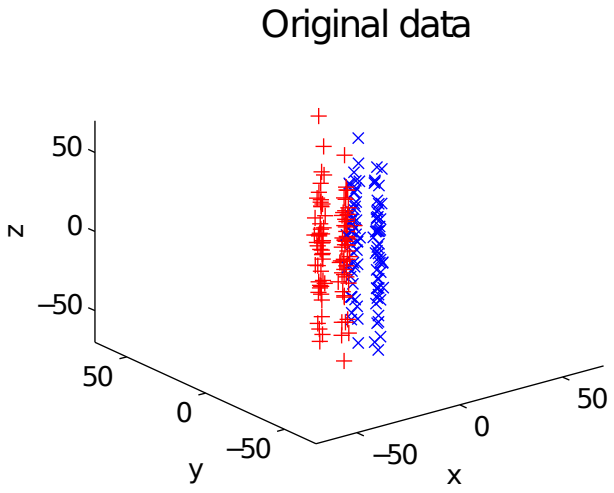
$$\mathbf{M} \succeq 0.$$

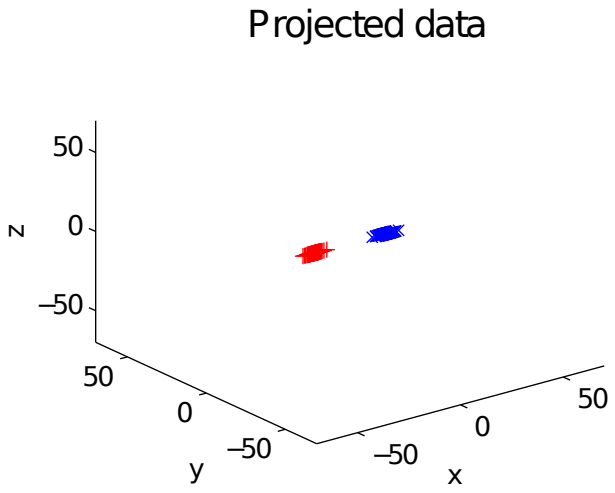
Remarks

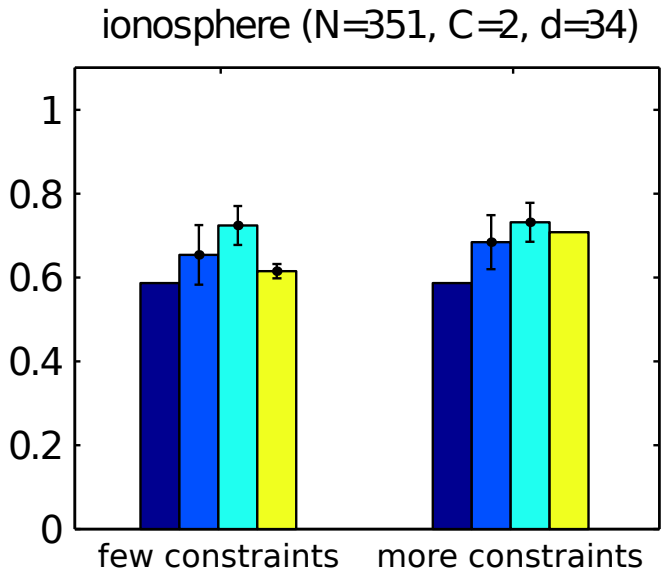
- Proposed for clustering with side information.
- The problem is convex in $\mathbf{M}$ and always feasible (with $\mathbf{M} = \mathbf{0}$).
- Solved with a projected gradient descent algorithm.
- For large d, the bottleneck is the projection on the set $\mathbf{M} \succeq 0$: requires eigenvalue decomposition which scales in $O(d^3)$.
- Only look at sum of distances.

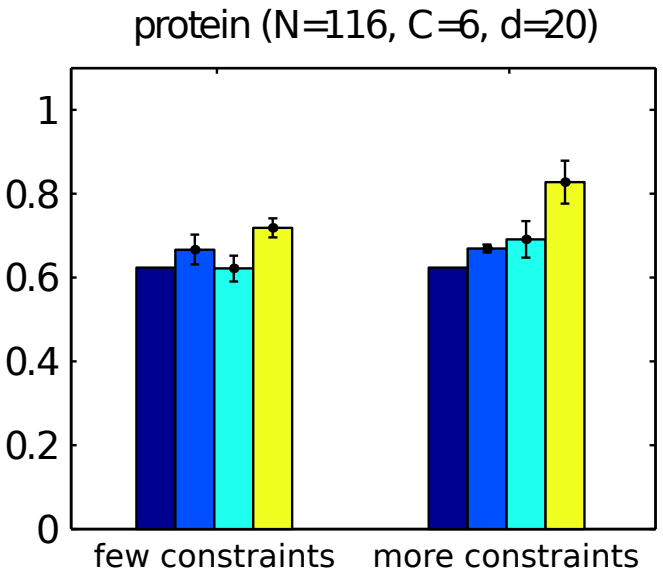
Mahalanobis distance learning

MMC [Xing et al., 2002]









left to right: K-Means, K-Means+MMC-Diag, K-Means+MMC-Full, Constrained K-Means

Mahalanobis distance learning

S&J [Schultz and Joachims, 2003]

Formulation

$$\begin{aligned} \min_{\mathbf{M}} \quad & \|\mathbf{M}\|_{\mathcal{F}}^2 \\ \text{s.t.} \quad & d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_k) - d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) \geq 1 \qquad \forall (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R} \\ & \mathbf{M} \succeq 0, \end{aligned}$$

Remarks

- Regularization on $\mathbf{M}$ (keeps the metric "simple", avoid overfitting).
- One large-margin constraint per triplet.

Mahalanobis distance learning

S&J [Schultz and Joachims, 2003]

Formulation with soft constraints

$$\begin{aligned} \min_{\mathbf{M},\xi\geq 0} \quad & \|\mathbf{M}\|_{\mathcal{F}}^2 \quad + \quad C\sum_{i,j,k}\xi_{ijk} \\ \text{s.t.} \quad & d_{\mathbf{M}}^2(\mathbf{x}_i,\mathbf{x}_k) - d_{\mathbf{M}}^2(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \; - \; \xi_{ijk} \quad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R} \\ & \mathbf{M}\succeq 0, \end{aligned}$$

Remarks

- Regularization on $\mathbf{M}$ (keeps the metric "simple", avoid overfitting).
- One large-margin constraint per triplet that may be violated.
- Advantages:
  - more flexible constraints.

Mahalanobis distance learning

S&J [Schultz and Joachims, 2003]

Formulation with soft constraints and PSD enforcement by design

$$\min_{\mathbf{W},\xi\geq 0} \quad \|\mathbf{M}\|_{\mathcal{F}}^2 \quad + \quad C\sum_{i,j,k}\xi_{ijk}$$

$$\text{s.t.} \quad d_{\mathbf{M}}^2(\mathbf{x}_i,\mathbf{x}_k) - d_{\mathbf{M}}^2(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R}$$

$$\cancel{\mathbf{M}\succeq 0,}$$

where $\mathbf{M} = \mathbf{A}^T\mathbf{W}\mathbf{A}$, where $\mathbf{A}$ is fixed and known and $\mathbf{W}$ diagonal.

Remarks

- Regularization on $\mathbf{M}$ (keeps the metric "simple", avoid overfitting).
- One large-margin constraint per triplet that may be violated.
- Advantages:
  - more flexible constraints.
  - no PSD constraint to deal with.
- Drawbacks:
  - how to choose appropriate $\mathbf{A}$?
  - only learns a weighting of the attributes defined by $\mathbf{A}$.

Mahalanobis distance learning

NCA [Goldberger et al., 2004]

Main idea

Optimize leave-one-out error of a stochastic nearest neighbor classifier in the projection space induced by $d_{\mathbf{M}}$. Use $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ and define the probability of $\mathbf{x}_j$ being the neighbor of $\mathbf{x}_i$ as

$$p_{ij} = \frac{\exp(-\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|_2^2)}{\sum_{l \neq i} \exp(-\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_l\|_2^2)}, \quad p_{ii} = 0.$$

Formulation

$$\max_{\mathbf{L}} \quad \sum_i p_i,$$

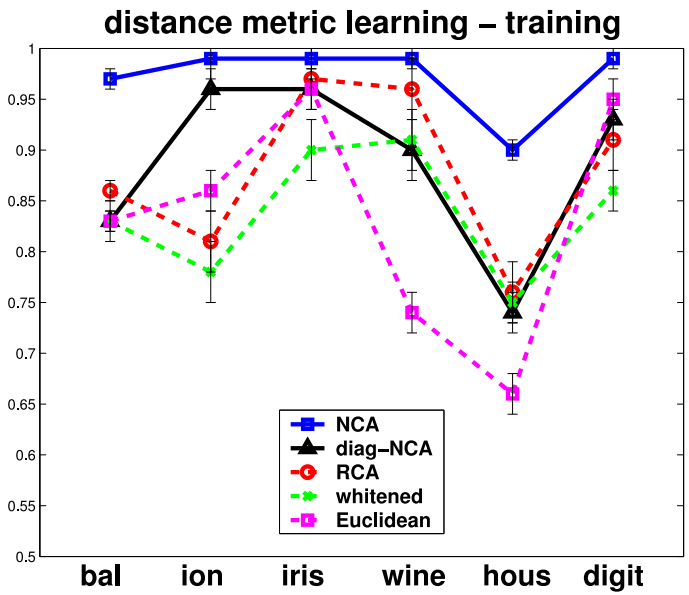
where $p_i = \sum_{j:y_j=y_i} p_{ij}$ is the probability that $\mathbf{x}_i$ is correctly classified.

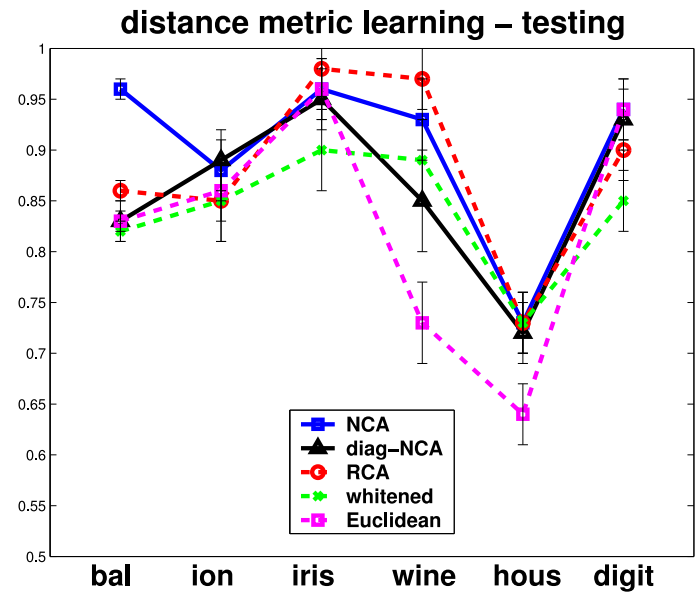
Remarks

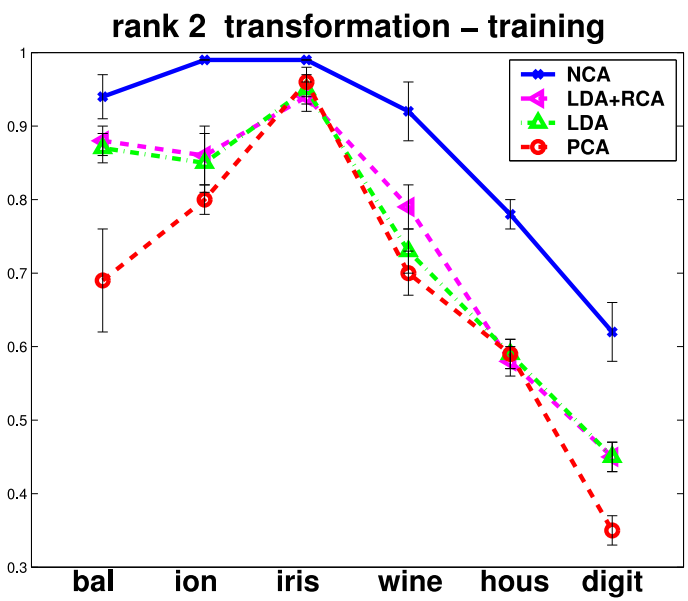
- Fully supervised, tailored to 1NN classifier.
- Can pick $\mathbf{L}$ rectangular for dimensionality reduction.
- Nonconvex (thus subject to local optima).

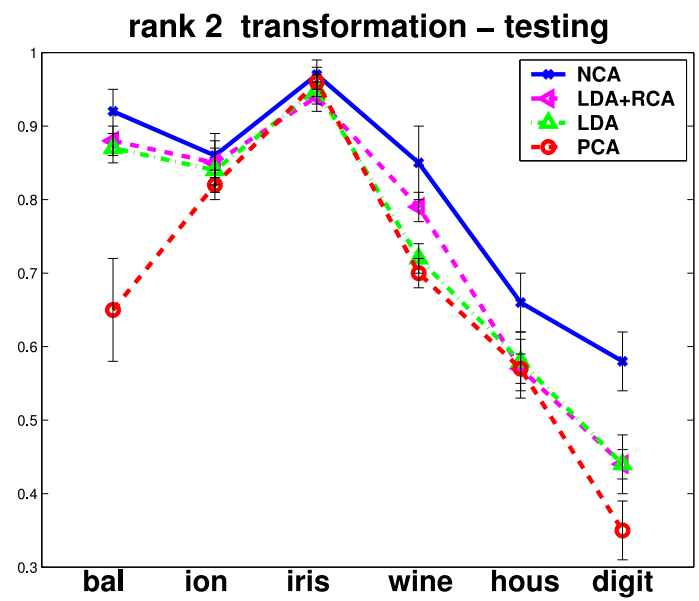
Mahalanobis distance learning

NCA [Goldberger et al., 2004]









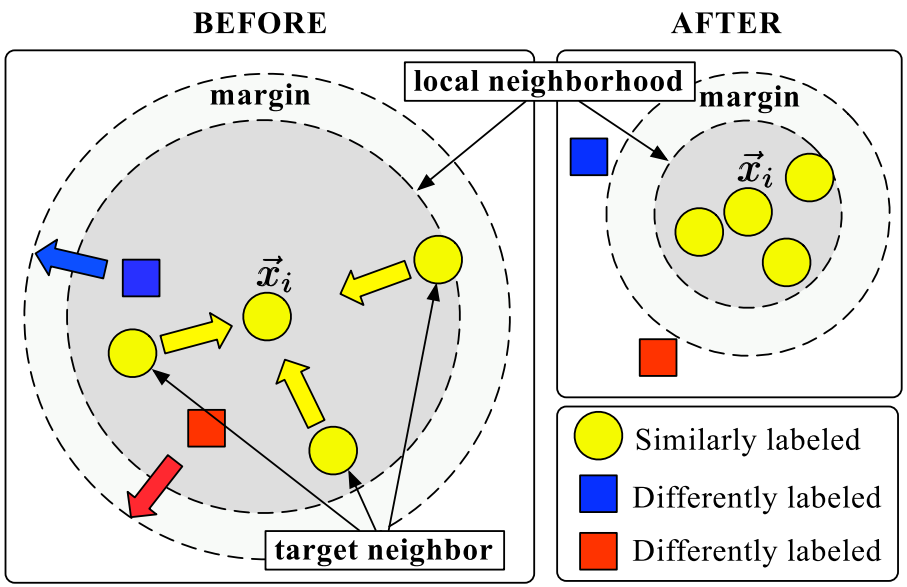
Mahalanobis distance learning

LMNN [Weinberger et al., 2005]

Main Idea

Define constraints tailored to k-NN in a local way: the k nearest neighbors should be of same class ("target neighbors"), while examples of different classes should be kept away ("impostors"):

$$\begin{aligned} \mathcal{S} &= \{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j \text{ and } \mathbf{x}_j \text{ belongs to the } k\text{-neighborhood of } \mathbf{x}_i\}, \\ \mathcal{R} &= \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) : (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, y_i \neq y_k\}. \end{aligned}$$



Mahalanobis distance learning

LMNN [Weinberger et al., 2005]

Formulation

$$\min_{\mathbf{M}\in\mathbb{S}_+^d,\xi\geq 0} \quad (1-\mu)\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \quad + \quad \mu\sum_{i,j,k}\xi_{ijk}$$

$$\text{s.t.} \quad d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1-\xi_{ijk} \qquad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R},$$

where $\mu\in[0,1]$ is a trade-off parameter.

Remarks

- Minimizing the distances between target neighbors as a regularizer.
- Advantages:
  - More flexible constraints (but require full supervision).
  - Convex, with a solver based on working set and subgradient descent. Can deal with millions of constraints and very popular in practice.
- Drawbacks:
  - Subject to overfitting in high dimension.
  - Sensitive to Euclidean distance performance.

Mahalanobis distance learning

ITML [Davis et al., 2007]

Main idea

Use a regularizer that automatically enforces PSD, the LogDet divergence:

$$\begin{aligned} D_{ld}(\mathbf{M}, \mathbf{M}_0) &= \operatorname{tr}(\mathbf{M}\mathbf{M}_0^{-1}) - \log\det(\mathbf{M}\mathbf{M}_0^{-1}) - d \\ &= \sum_{i,j} \frac{\lambda_i}{\theta_j}(\mathbf{v_i}^T\mathbf{u}_i)^2 - \sum_i \log\left(\frac{\lambda_i}{\theta_i}\right) - d, \end{aligned}$$

where $\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ and $\mathbf{M}_0 = \mathbf{U}\mathbf{\Theta}\mathbf{U}^T$ is some PSD matrix (typically $\mathbf{I}$ or $\mathbf{\Sigma}^{-1}$). Properties:

- Convex (because determinant of PSD matrix is log-concave).
- Finite if and only if $\text{range}(\mathbf{M}) = \text{range}(\mathbf{M}_0)$.
- Implicitly makes $\mathbf{M}$ PSD and same rank as $\mathbf{M}_0$.
- Information-theoretic interpretation: minimize KL divergence between two Gaussians parameterized by $\mathbf{M}$ and $\mathbf{M}_0$.

Mahalanobis distance learning

ITML [Davis et al., 2007]

Formulation

$$\begin{aligned} \min_{\mathbf{M}\in\mathbb{S}_+^d,\xi\geq 0} \quad & D_{ld}(\mathbf{M},\mathbf{M}_0) \quad + \quad \lambda\sum_{i,j}\xi_{ij} \\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \leq u+\xi_{ij} && \forall(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S} \\ & d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq v-\xi_{ij} && \forall(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}, \end{aligned}$$

where $u, v, \lambda \geq 0$ are threshold and trade-off parameters.

Remarks

- Soft must-link / cannot-link constraints.
- Simple algorithm based on Bregman projections. Each iteration is quadratic in d instead of cubic with projections on PSD cone.
- Drawback: the choice of $\mathbf{M}_0$ can have an important influence on the quality of the learned metric.

Large-scale metric learning

- Consider the situation where the number of training constraints is very large, say $> 10^6$ (this can happen even with a moderate number of training examples, e.g. in LMNN).
- Previous algorithms become huge, possibly intractable optimization problems (gradient computation and/or projections become very expensive).

One solution: online learning

- In online learning, the algorithm receives training instances one at a time and updates the current hypothesis at each step.
- Performance typically inferior to that of batch algorithms, but allows to tackle large-scale problems.

Setup, notations, definitions

- Start with a hypothesis h_1.
- At each step $t = 1, \ldots, T$, the learner receives a training example $z_t = (x_t, y_t)$ and suffers a loss $\ell(h_t, z_t)$. Depending on the loss, it generates a new hypothesis h_{t+1}.
- The goal is to find a low-regret hypothesis (how much worse we do compared to the best hypothesis in hindsight):

$$R = \sum_{t=1}^{T} \ell(h_t, z_t) - \sum_{t=1}^{T} \ell(h^*, z_t),$$

  where h^* is the best batch hypothesis on the T examples.
- Ideally, have a regret bound of the form $R \leq O(T)$.

Mahalanobis distance learning

LEGO [Jain et al., 2008]

Formulation

At each step, receive $(\mathbf{x}_t, \mathbf{x}'_t, y_t)$ where y_t is the target distance between $\mathbf{x}_t$ and $\mathbf{x}'_t$, and update as follows:

$$\mathbf{M}^{t+1} = \arg\min_{\mathbf{M} \succeq 0} D_{ld}(\mathbf{M}, \mathbf{M}^t) + \lambda \ell(\mathbf{M}, \mathbf{x}_t, \mathbf{x}'_t, y_t),$$

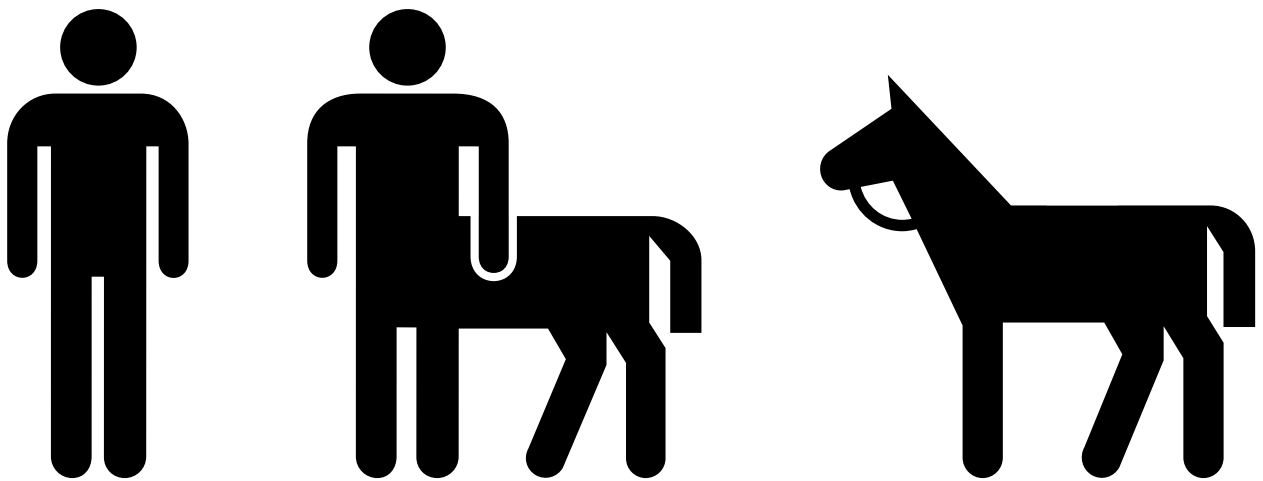
where ℓ is a loss function (square loss, hinge loss...).

Remarks

- It turns out that the above update has a closed-form solution which maintains $\mathbf{M} \succeq 0$ automatically.
- Can derive a regret bound.

Drawbacks of Mahalanobis distance learning

- Maintaining $\mathbf{M} \succeq 0$ is often costly, especially in high dimensions. If the decomposition $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ is used, formulations become nonconvex.
- Objects must have same dimension.
- Distance properties can be useful (e.g., for fast neighbor search), but restrictive. Evidence that our notion of (visual) similarity violates the triangle inequality (example below).



Linear similarity learning

OASIS [Chechik et al., 2009]

Recap

The bilinear similarity is defined as follows:

$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}',$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is not required to be PSD nor symmetric. In general, it satisfies none of the distance properties.

Formulation

At each step, receive triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R}$ and update as follows:

$$\begin{aligned} \mathbf{M}^t = \quad & \underset{\mathbf{M}, \xi \geq 0}{\arg\min} \quad \frac{1}{2} \|\mathbf{M} - \mathbf{M}^{t-1}\|_{\mathcal{F}}^2 + C\xi \\ & \text{s.t.} \quad 1 - K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) \leq \xi. \end{aligned}$$

Linear similarity learning

OASIS [Chechik et al., 2009]

Formulation

At each step, receive triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R}$ and update as follows:

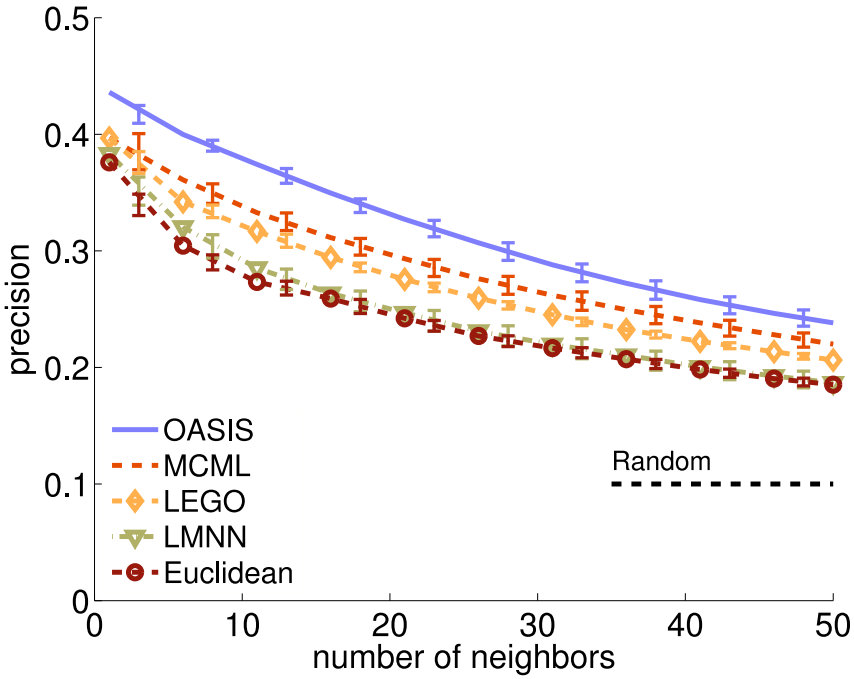
$$\begin{aligned} \mathbf{M}^t = \quad & \underset{\mathbf{M}, \xi \geq 0}{\arg\min} \quad \frac{1}{2}\|\mathbf{M} - \mathbf{M}^{t-1}\|_{\mathcal{F}}^2 + C\xi \\ & \text{s.t.} \quad 1 - K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) \leq \xi. \end{aligned}$$

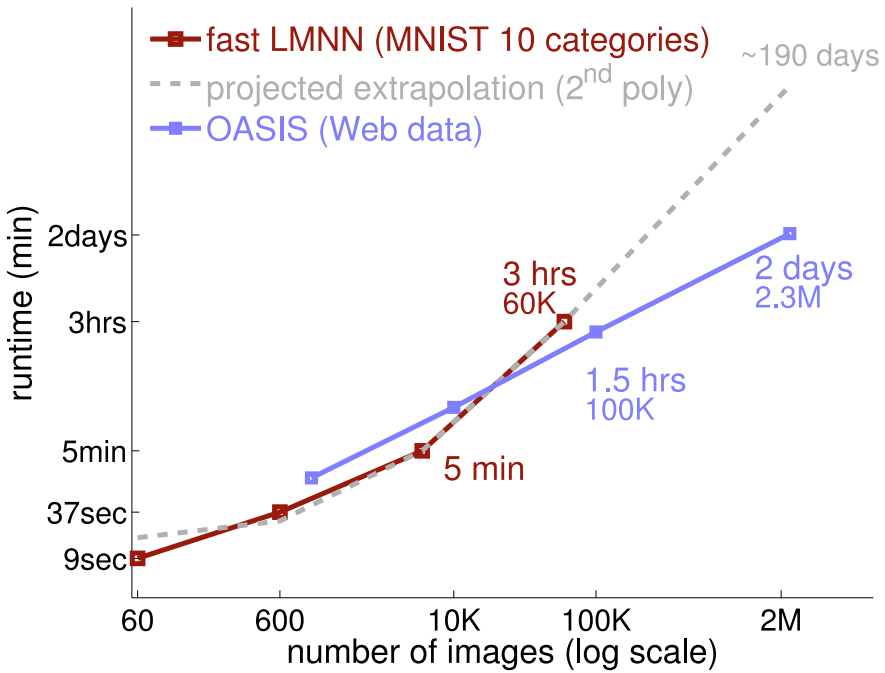
Remarks

- Passive-aggressive algorithm: no update if the triplet satisfies the margin. Otherwise, simple closed-form update.
- Scale to very large datasets with good generalization performance. Evaluated on 2.3M training instances (image retrieval task).
- Limitation: restricted to Frobenius norm (no fancy regularization).

Linear similarity learning

OASIS [Chechik et al., 2009]



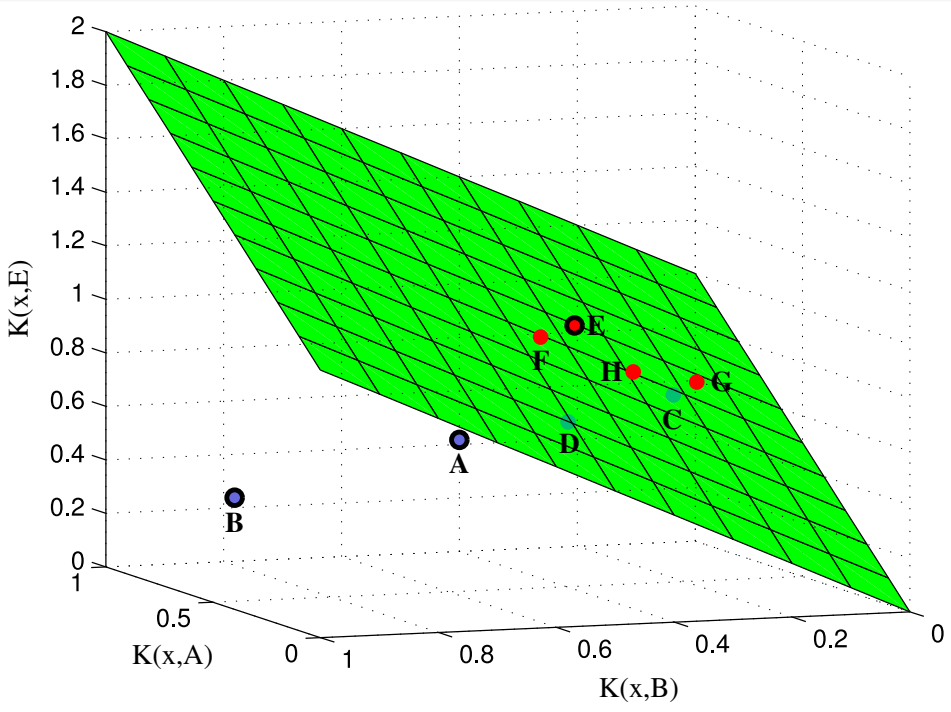


Linear similarity learning

SLLC [Bellet et al., 2012b]

Main idea

- Similarity learning to improve linear classification (instead of k-NN).
- Based on the theory of learning with (ϵ, γ, τ)-good similarity functions [Balcan et al., 2008]. Basic idea: use similarity as features (similarity map).
- If the similarity satisfies some property, then there exists a sparse linear classifier with small error in that space (more on this later).



Linear similarity learning

SLLC [Bellet et al., 2012b]

Formulation

$$\min_{\mathbf{M}\in\mathbb{R}^{d\times d}} \quad \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i \frac{1}{\gamma|\mathcal{R}|}\sum_{\mathbf{x}_j\in\mathcal{R}} y_j K_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)\right]_+ \quad + \quad \beta\|\mathbf{M}\|_{\mathcal{F}}^2,$$

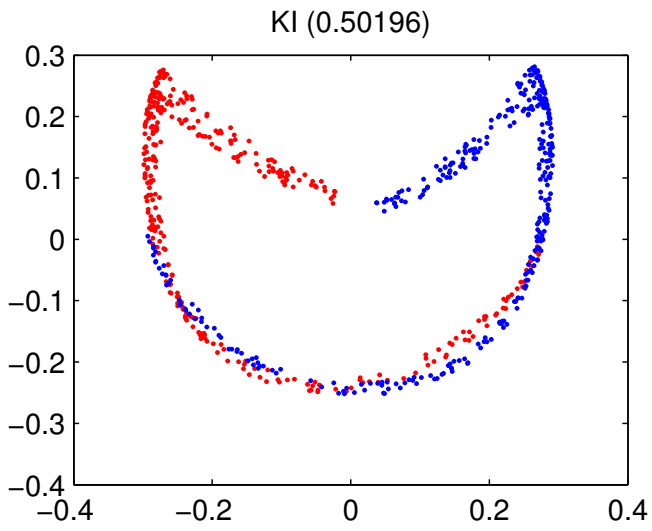
where $\mathcal{R}$ is a set of reference points randomly selected from the training sample, γ is the margin parameter, $[\cdot]_+$ is the hinge loss and β the regularization parameter.

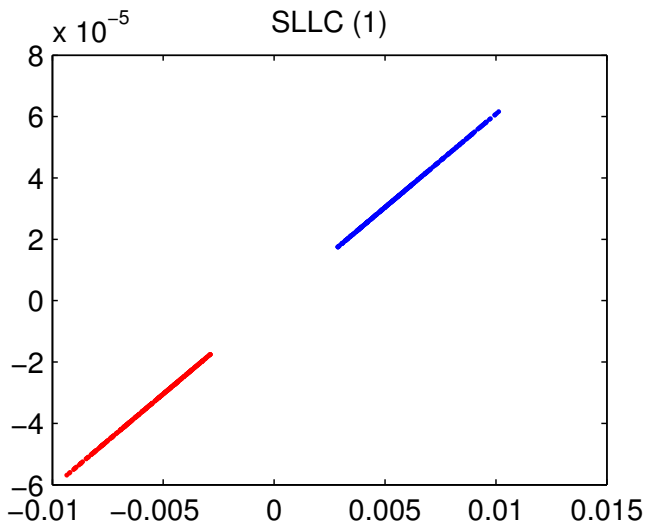
Remarks

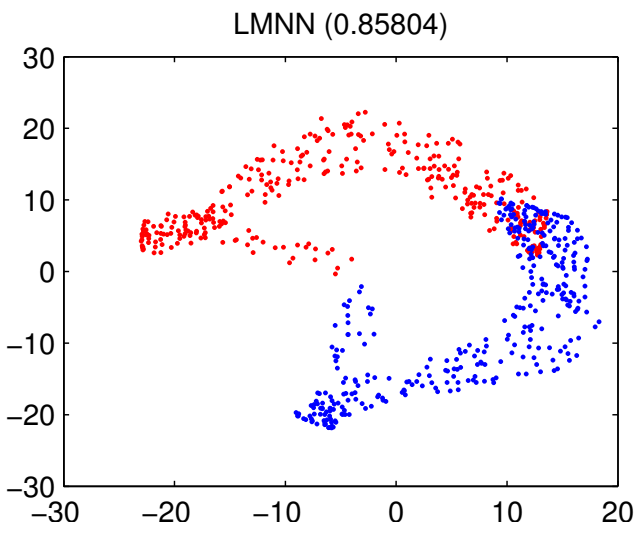
- Basically learn $K_{\mathbf{M}}$ such that training examples are more similar on average to reference points of same class than to those of opposite class by a margin γ. Thus, more flexible constraints that are sufficient for linear classification.
- Leads to very sparse classifiers with error bounds (more on this later).
- Drawback: multi-class setting requires several classifiers.

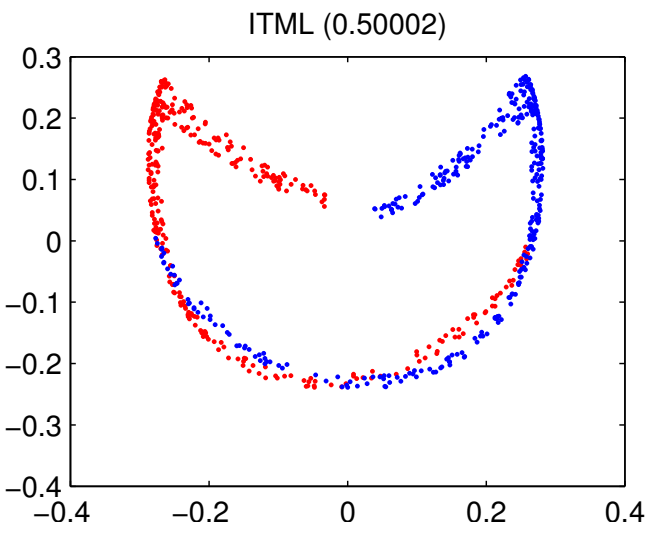
Linear similarity learning

SLLC [Bellet et al., 2012b]



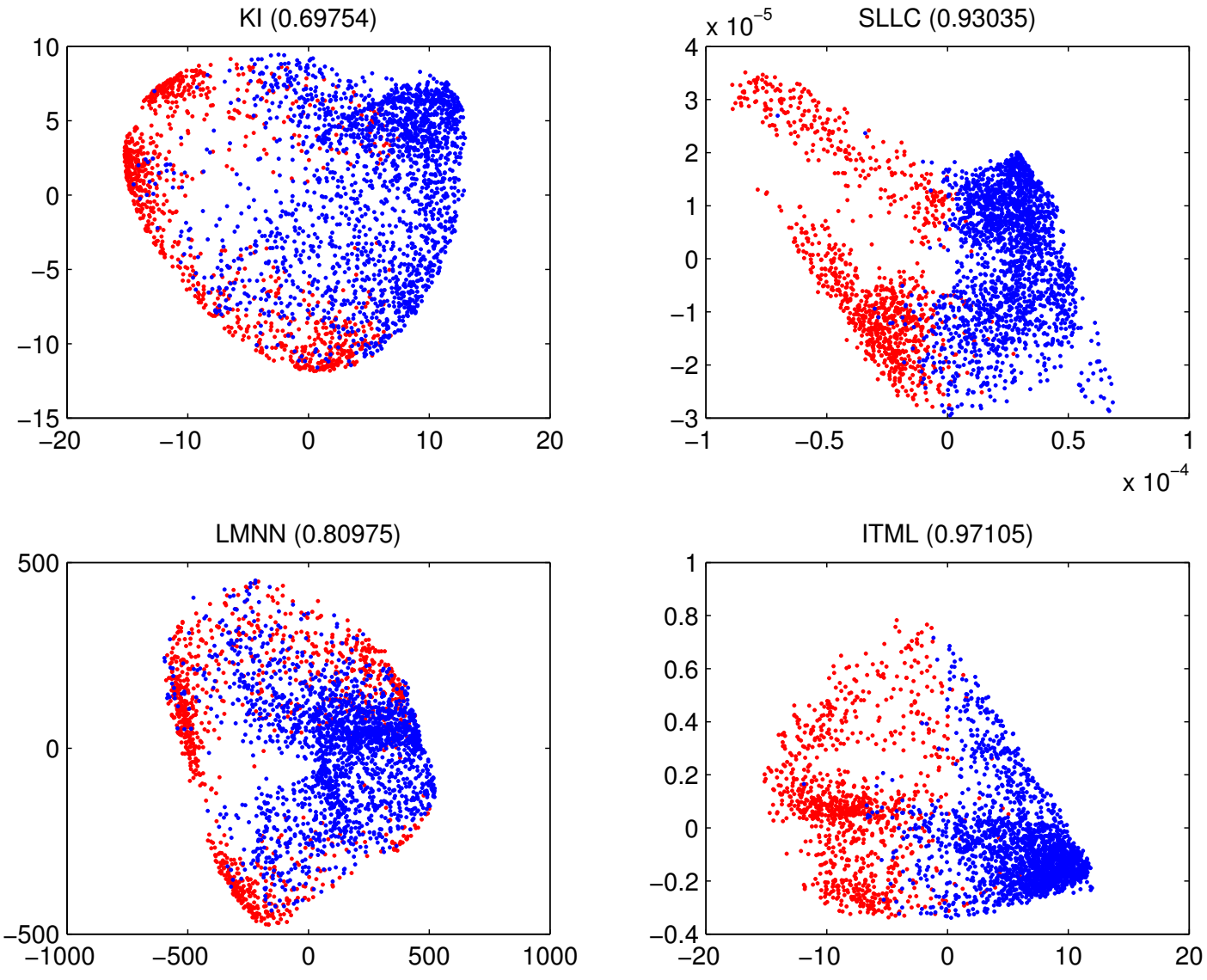






Linear similarity learning

SLLC [Bellet et al., 2012b]



Summary

- Well-studied framework, often with convex formulations.
- Can learn online (or using stochastic gradient) to scale to large datasets.
- Mahalanobis if distance properties are needed, otherwise more flexibility with a similarity function.

Limitation

A linear metric is often unable to accurately capture the complexity of the task (multimodal data, nonlinear class boundaries).

Outline

1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Nonlinear metric learning

The big picture

Three approaches

1. Kernelization of linear methods.
2. Learning a nonlinear metric.
3. Learning several local linear metrics.

Definition (Kernel function)

A symmetric similarity function K is a kernel if there exists a (possibly implicit) mapping function $\phi : \mathcal{X} \to \mathbb{H}$ from the instance space $\mathcal{X}$ to a Hilbert space $\mathbb{H}$ such that K can be written as an inner product in $\mathbb{H}$:

$$K(x, x') = \left\langle \phi(x), \phi(x') \right\rangle .$$

Equivalently, K is a kernel if it is positive semi-definite (PSD), i.e.,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

for all finite sequences of $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$.

Kernel trick

Let $K(x, x') = \langle \phi(x), \phi(x') \rangle$ a kernel. We have training data $\{\mathbf{x}_i\}_{i=1}^n$ and we use $\phi_i \stackrel{def}{=} \phi(\mathbf{x}_i)$. We want to compute a (squared) Mahalanobis distance in kernel space:

$$d^2_{\mathbf{M}}(\phi_i, \phi_j) = (\phi_i - \phi_j)^T \mathbf{M} (\phi_i - \phi_j) = (\phi_i - \phi_j)^T \mathbf{L}^T \mathbf{L} (\phi_i - \phi_j).$$

Let $\mathbf{\Phi} = [\phi_1, \ldots, \phi_n]$ and use the parameterization $\mathbf{L}^T = \mathbf{\Phi U}^T$. Now:

$$d^2_{\mathbf{M}}(\phi_i, \phi_j) = (\mathbf{k}_i - \mathbf{k}_j)^T \mathbf{M} (\mathbf{k}_i - \mathbf{k}_j),$$

where $\mathbf{k}_i = \mathbf{\Phi}^T \phi_i = [K(x_1, x_i), \ldots, K(x_n, x_i)]^T$ and $\mathbf{M}$ is $n \times n$.

Now, why is this interesting?

- Similar trick as in kernel SVM: take K to be some nonlinear kernel. Say, the RBF kernel (inducing an *infinite* dimensional space):

$$K_{rbf}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right).$$

- We get a distance computed after a nonlinear, high-dimensional projection of the data, but distance computations are done inexpensively through the kernel.
- To learn our distance, need to estimate n^2 parameters: independent of original and projection space dimensions!
- Justified theoretically through a representer theorem [Chatpatanasiri et al., 2010].

Limitations

- Some algorithms have been shown to be kernelizable, but in general this is not trivial: a new formulation of the problem has to be derived, where interface to the data is limited to inner products, and sometimes a different implementation is necessary.
- When the number of training examples n is large, learning n^2 parameters may be intractable.

A solution: KPCA trick

- Use KPCA (PCA in kernel space) to get a nonlinear but low-dimensional projection of the data.
- Then use unchanged algorithm!
- Again, theoretically justified [Chatpatanasiri et al., 2010].

Nonlinear metric learning

Learning a nonlinear metric: GB-LMNN [Kedem et al., 2012]

Main idea

- Learn a nonlinear mapping ϕ to optimize the Euclidean distance $d_\phi(\mathbf{x}, \mathbf{x}') = \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|_2$ in the transformed space.
- Same objective as LMNN.
- They use $\phi = \phi_0 + \alpha \sum_{t=1}^{T} h_t$, where ϕ_0 is the mapping learned by linear LMNN, and $h_1, \ldots, h_T$ are gradient boosted regression trees of limited depth p.
- At iteration t, the tree that best approximate the negative gradient $-g_t$ of the objective with respect to the current transformation ϕ_{t-1}:

$$h_t(\cdot) = \arg\min_h \sum_{i=1}^{n} \left(g_t(\mathbf{x}_i) + h(\mathbf{x}_i)\right)^2 .$$

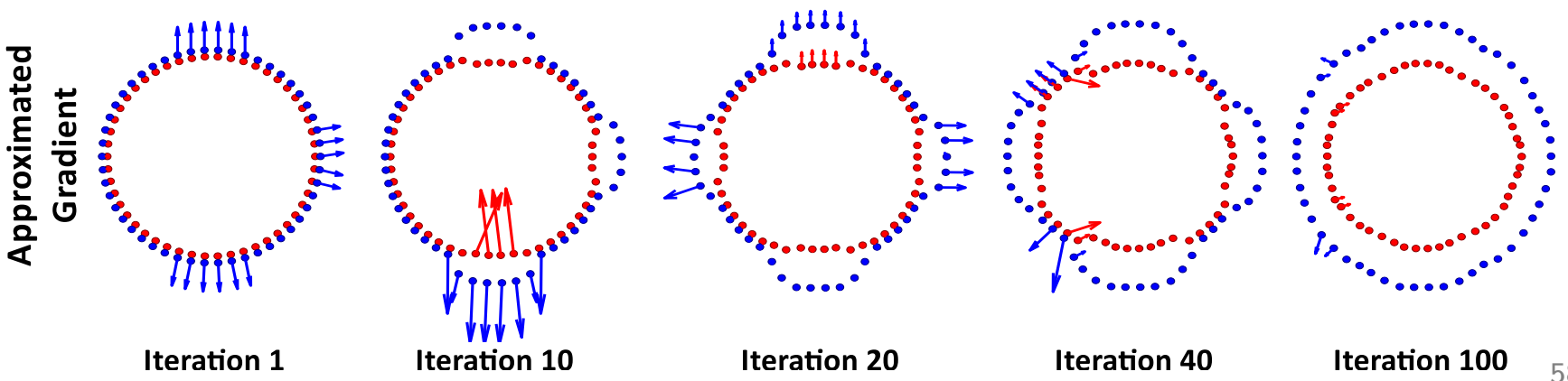
- Intuitively, each tree divides the space into 2^p regions, and instances falling in the same region are translated by the same vector (thus examples in different regions are translated in different directions).

Nonlinear metric learning

Learning a nonlinear metric: GB-LMNN [Kedem et al., 2012]

Remarks

- Dimensionality reduction can be achieved by learning trees with low-dimensional output.
- The objective function is nonconvex, so bad local minima are an issue, although this is attenuated by initializing with linear LMNN.
- In practice, GB-LMNN performs well and seems quite robust to overfitting.
- Drawback: distance evaluation may become expensive when a large number of trees is needed.



Motivation

- Simple linear metrics perform well locally.
- Since everything is linear, can keep formulation convex.

Pitfalls

- How to split the space?
- How to avoid a blow-up in number of parameters to learn, and avoid overfitting?
- How to make local metrics comparable?
- How to generalize local metrics to new regions?
- How to obtain a proper global metric?
- ...

Nonlinear metric learning

Local metric learning: MM-LMNN [Weinberger and Saul, 2009]

Main idea

- Extension of LMNN where data is first divided into C clusters and a metric is learned for each cluster.
- Learn the metrics in a coupled fashion, where the distance to a target neighbor or an impostor $\mathbf{x}$ is measured under the local metric associated with the cluster to which $\mathbf{x}$ belongs.

Formulation (one metric per class)

$$\min_{\mathbf{M}_1,\ldots,\mathbf{M}_C \in \mathbb{S}_+^d, \xi \geq 0} \quad (1-\mu) \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}_{y_j}}(\mathbf{x}_i,\mathbf{x}_j) \quad + \quad \mu \sum_{i,j,k} \xi_{ijk}$$

$$\text{s.t.} \quad d^2_{\mathbf{M}_{y_k}}(\mathbf{x}_i,\mathbf{x}_k) - d^2_{\mathbf{M}_{y_j}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1-\xi_{ijk} \quad \forall(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in \mathcal{R}.$$

Nonlinear metric learning

Local metric learning: MM-LMNN [Weinberger and Saul, 2009]

Formulation (one metric per class)

$$\min_{\mathbf{M}_1,\ldots,\mathbf{M}_C \in \mathbb{S}_+^d, \xi \geq 0} \quad (1-\mu) \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}_{y_j}}(\mathbf{x}_i,\mathbf{x}_j) \quad + \quad \mu \sum_{i,j,k} \xi_{ijk}$$

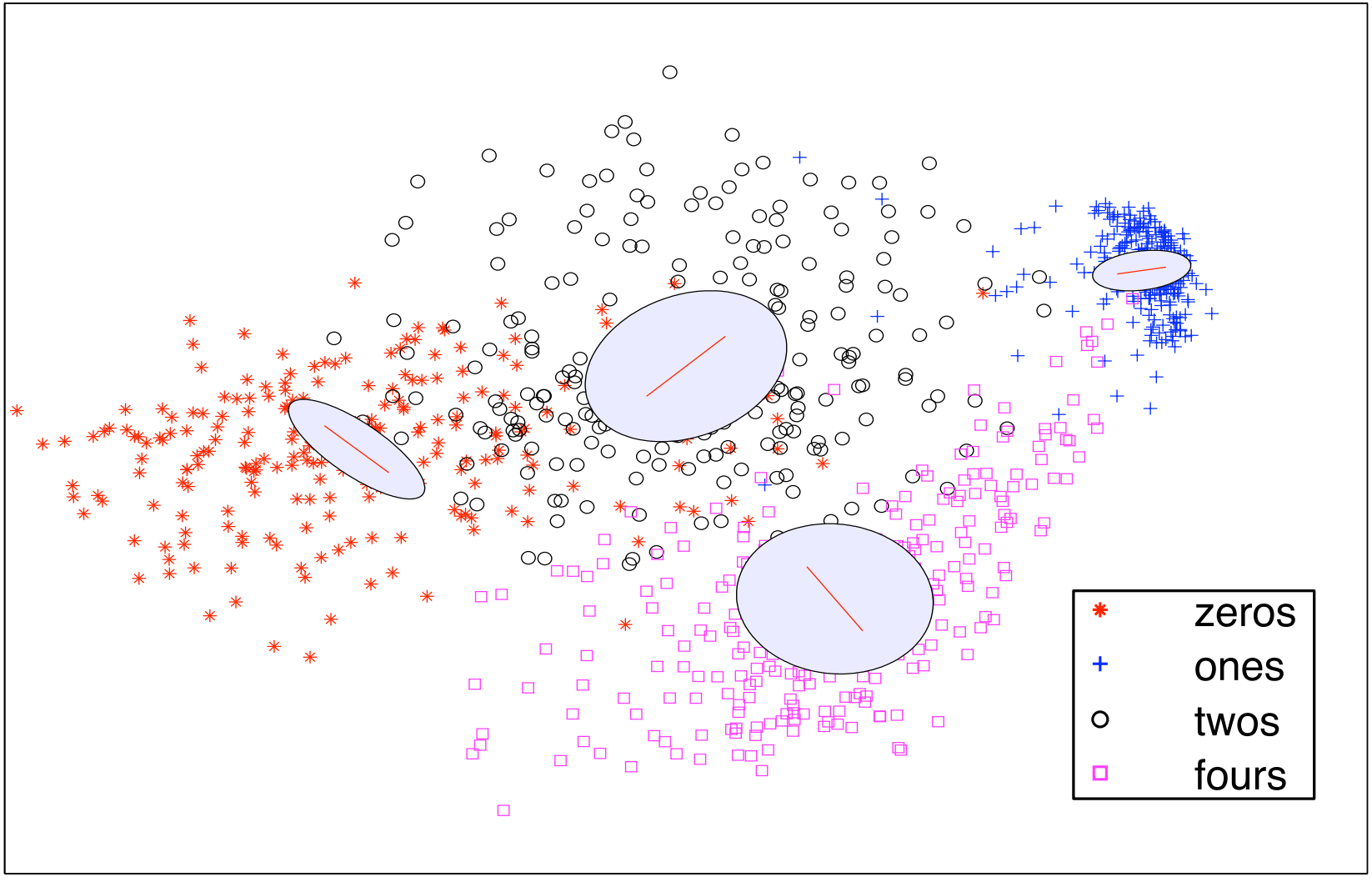
$$\text{s.t.} \quad d^2_{\mathbf{M}_{y_k}}(\mathbf{x}_i,\mathbf{x}_k) - d^2_{\mathbf{M}_{y_j}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall (\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in \mathcal{R}.$$

Remarks

- Advantages:
  - The problem remains convex.
  - Can lead to significant improvement over LMNN.
- Drawbacks:
  - Subject to important overfitting.
  - Computationally expensive as the number of metrics grows.
  - Change in metric between two neighboring regions can be very sharp.

Nonlinear metric learning

Local metric learning: MM-LMNN [Weinberger and Saul, 2009]



Nonlinear metric learning

Local metric learning: PLML [Wang et al., 2012]

Main idea

- Learn a metric $d^2_{\mathbf{M}_i}$ for each training instance $\mathbf{x}_i$ as a weighted combination of metrics defined at anchor points $\mathbf{u}_1, \ldots, \mathbf{u}_m$ throughout the space.
- First, learn the weights based on manifold (smoothness) assumption.
- Then, learn the anchor metrics.

Weight learning

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{X} - \mathbf{W}\mathbf{U}\|_{\mathcal{F}}^2 + \lambda_1 \operatorname{tr}(\mathbf{W}\mathbf{G}) + \lambda_2 \operatorname{tr}(\mathbf{W}^T\mathbf{L}\mathbf{W}) \\ \text{s.t.} \quad & W_{ib_j} \geq 0, \quad \forall i, b_j \\ & \sum_{j=1}^{m} W_{ib_j} = 1, \quad \forall i, \end{aligned}$$

where $\mathbf{G}$ holds distances between each training point and each anchor point, and $\mathbf{L}$ is the Laplacian matrix constructed on the training instances.

Nonlinear metric learning

Local metric learning: PLML [Wang et al., 2012]

Anchor metric learning

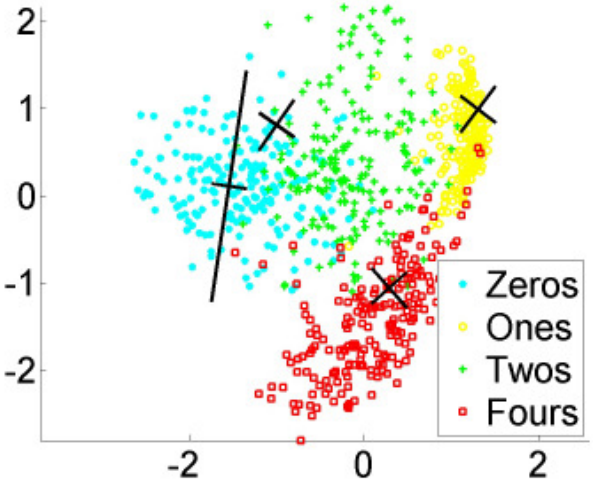
Similar to (MM)-LMNN.

Remarks

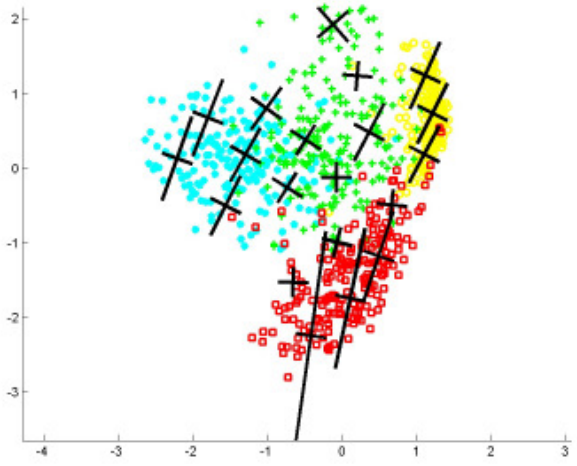
- Advantages:
  - Both subproblems are convex.
  - A specific metric for each training point, while only learning a small number of anchor metrics.
  - Smoothly varying metrics reduce overfitting.
- Drawbacks:
  - Learning anchor metrics can still be expensive.
  - No discriminative information when learning the weights.
  - No principled way to get metric for a test point (in practice, use weights of nearest neighbor).
  - Relies a lot on Euclidean distance.
  - Large number of hyperparameters.

Nonlinear metric learning

Local metric learning: PLML [Wang et al., 2012]



MM-LMNN



PLML

Three ways of tackling the problem

1. Learn a nonlinear metric implicitly through kernelization. Elegant, but choosing a relevant kernel might be difficult.
2. Learn a nonlinear form of metric. Powerful although it leads to nonconvex formulations.
3. Learn multiple local metrics. Very expressive and retains convexity, but must watch out for overfitting and high complexity.

Outline

1. Preliminaries
   Metrics, basic notions of convex optimization
2. Metric Learning in a Nutshell
   Basic formulation, type of constraints, key properties
3. Linear Metric Learning
   Mahalanobis distance learning, similarity learning
4. Nonlinear Metric Learning
   Kernelization of linear methods, nonlinear and local metric learning
5. Metric Learning for Other Settings
   Multi-task, ranking, histogram data, semi-supervised, domain adaptation
6. Metric Learning for Structured Data
   String and tree edit distance learning
7. Deriving Generalization Guarantees
   Basic notions of statistical learning theory, the specifics of metric learning
8. Summary and Outlook

Metric learning for other settings

Multi-task learning: mt-LMNN [Parameswaran and Weinberger, 2010]

Multi-task learning

Given T (somehow related) tasks, learn in a coupled fashion to leverage commonalities between tasks. When data is scarce, this can outperform separate learning of each task.

Multi-task metric learning

Learn a metric for each task while sharing information between metrics.

Formulation

Learn a shared Mahalanobis metric $d_{\mathbf{M}_0}$ as well as task-specific metrics $d_{\mathbf{M}_1}, \ldots, d_{\mathbf{M}_t}$ and define the metric for task t as

$$d_t(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T (\mathbf{M}_0 + \mathbf{M}_t)(\mathbf{x} - \mathbf{x}'),$$

and learn as in LMNN but with the regularizer

$$\gamma_0 \|\mathbf{M}_0 - \mathbf{I}\|_{\mathcal{F}}^2 + \sum_{t=1}^{T} \gamma_t \|\mathbf{M}_t\|_{\mathcal{F}}^2.$$

Metric learning for other settings

Multi-task learning: mt-LMNN [Parameswaran and Weinberger, 2010]

Formulation

Learn a shared Mahalanobis metric $d_{\mathbf{M}_0}$ as well as task-specific metrics $d_{\mathbf{M}_1}, \ldots, d_{\mathbf{M}_t}$ and define the metric for task t as

$$d_t(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T (\mathbf{M}_0 + \mathbf{M}_t)(\mathbf{x} - \mathbf{x}'),$$

and learn as in LMNN but with the regularizer

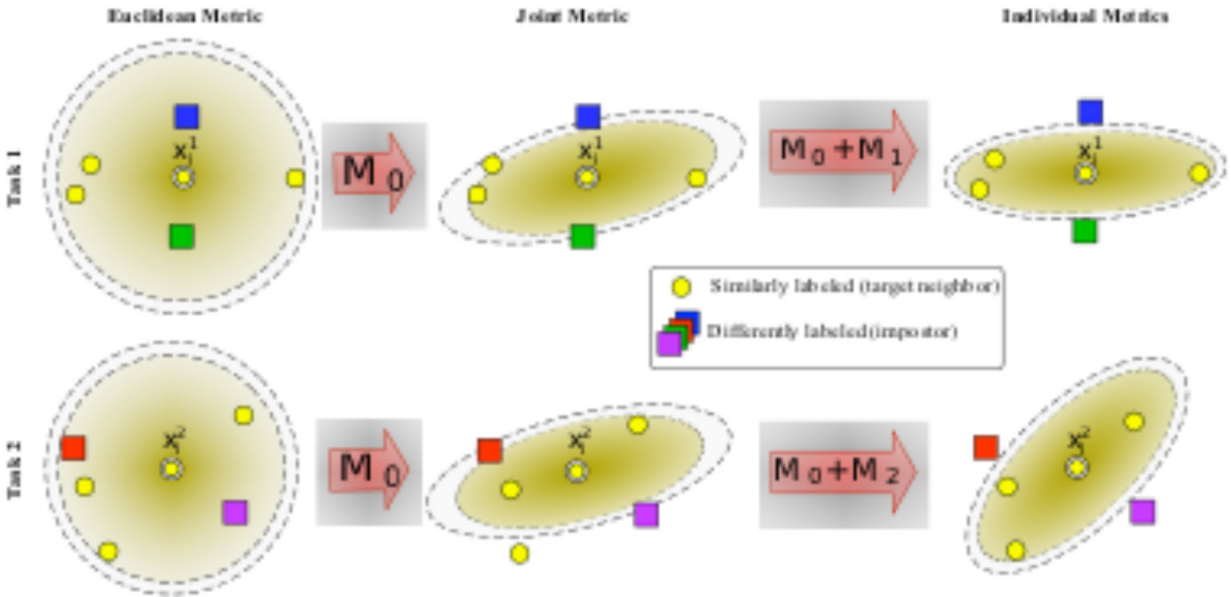
$$\gamma_0 \|\mathbf{M}_0 - \mathbf{I}\|_{\mathcal{F}}^2 + \sum_{t=1}^{T} \gamma_t \|\mathbf{M}_t\|_{\mathcal{F}}^2.$$

Remarks

- When $\gamma_0 \to \infty$, reduces to T independent LMNN formulations.
- When $\gamma_{t>0} \to \infty$, reduces to learning one metric on union of data.
- In-between: trade-off between shared and task-specific component.
- Remains convex. Works well when the strong assumption that all tasks share the same common part is reasonable.

Metric learning for other settings

Multi-task learning: mt-LMNN [Parameswaran and Weinberger, 2010]



Ranking

Produce a ranked list of examples where relevant ones are ranked higher than irrelevant ones. Typically want to optimize a measure of ranking quality, such as the Area Under the ROC Curve (AUC), Precision-at-k, or Mean Average Precision (MAP).

Ranking with a metric

Let $\mathcal{P}$ the set of all permutations (i.e., possible rankings) over the training set. Given a Mahalanobis distance $d^2_{\mathbf{M}}$ and a query $\mathbf{x}$, the predicted ranking $p \in \mathcal{P}$ consists in sorting the instances by ascending $d^2_{\mathbf{M}}(\mathbf{x}, \cdot)$.

Metric learning for other settings

Metric learning to rank: MLR [McFee and Lanckriet, 2010]

Formulation based on Structural SVM

$$\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad \|\mathbf{M}\|_* \quad + \quad C \sum_i \xi_i$$

$$\text{s.t.} \quad \langle \mathbf{M}, \psi(\mathbf{x}_i, p_i) - \psi(\mathbf{x}_i, p) \rangle_{\mathcal{F}} \geq \Delta(p_i, p) - \xi_i \quad \forall i \in \{1, \ldots, n\}, p \in \mathcal{P},$$

where

- $\|\mathbf{M}\|_* = \text{tr}(\mathbf{M})$ is the nuclear norm (induces sparsity in the spectrum),
- $\langle \mathbf{A}, \mathbf{B} \rangle_{\mathcal{F}} = \sum_{i,j} A_{ij} B_{ij}$ the Frobenius inner product,
- $\psi : \mathbb{R} \times \mathcal{P} \to \mathbb{S}^d$ the feature encoding of an input-output pair $(\mathbf{x}, p)$, designed such that the ranking p which maximizes $\langle \mathbf{M}, \psi(\mathbf{x}, p) \rangle_{\mathcal{F}}$ is the one given by ascending $d^2_{\mathbf{M}}(\mathbf{x}, \cdot)$,
- $\Delta(p_i, p) \in [0, 1]$ the "margin" representing the loss of predicting ranking p instead of the true ranking p_i.

Metric learning for other settings

Metric learning to rank: MLR [McFee and Lanckriet, 2010]

Big issue

The number of constraints is super-exponential in the number of training instances!

Solution: use a cutting-plane algorithm

Basically, iteratively optimize over a small set of active constraints (adding the most violated ones at each step) using subgradient descent. As it turns out, the most violated constraints can be identified efficiently.

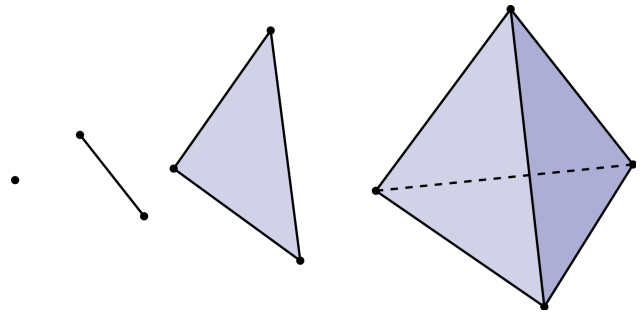
Remarks

- Convex and can induce low-rank solutions.
- However, requires projections onto the PSD cone.
- Performs well in ranking tasks, in terms of the optimized ranking measure.

Histogram data

Histograms are feature vectors that lie on the probability simplex $\mathcal{S}^d$:

$$\mathcal{S}^d = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x} \geq 0, \sum_i x_i = 1\}.$$



Example

In areas dealing with complex objects, such as natural language processing, computer vision or bioinformatics: an instance is often represented as a bag of features, i.e., a vector containing the frequency of each feature in the object.

Metrics for histogram data

Some specific metrics are more appropriate than say, the Euclidean distance:

- The χ^2 distance $\chi^2(\mathbf{x}, \mathbf{x}') = \frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{x_i + x_i'}$ (introduces a normalization factor). It is a nonlinear proper distance.
- The Earth Mover's Distance (EMD), a more powerful *cross-bin* measure (which is also a proper distance under some conditions).

Metric learning for other settings

Metric learning for histogram data: χ^2-LMNN [Kedem et al., 2012]

Formulation

Generalize the χ^2 distance with a linear transformation:

$$\chi^2_{\mathbf{L}}(\mathbf{x}, \mathbf{x}') = \chi^2(\mathbf{L}\mathbf{x}, \mathbf{L}\mathbf{x}'),$$

where $\mathbf{L} \in \mathbb{R}^{r \times d}$, with the constraint that $\mathbf{L}$ maps any $\mathbf{x} \in \mathcal{S}^d$ onto $\mathcal{S}^r$.
Can even get an unconstrained formulation by using a change of variable.

Remarks

- Same objective function as LMNN, solved using a standard subgradient descent procedure.
- Nonconvex.
- Although subject to local minima, significant improvement on histogram data compared to standard histogram metrics and learned Mahalanobis distance.
- Exhibits promising results for dimensionality reduction (when $r < d$).

Semi-supervised learning

When labeled data is scarce, it is useful to leverage the information brought by (potentially inexpensive to obtain) unlabeled data (for instance to help regularize the model).

Semi-supervised metric learning

In the context of metric learning, by unlabeled data we refer to training examples for which we have no labels and that are not associated with any constraints. In particular, we want to use the "unlabeled pairs":

$$\mathcal{U} = \{(\mathbf{x}_i, \mathbf{x}_j) : i \neq j, (\mathbf{x}_i, \mathbf{x}_j) \notin \mathcal{S} \cup \mathcal{D}\}.$$

Metric learning for other settings

Semi-supervised metric learning: LRML [Hoi et al., 2008]

Main idea

- Follow the principle of manifold regularization for semi-supervised learning by resorting to a weight matrix $\mathbf{W}$ that encodes the similarity between pairs of points:

$$W_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \mathcal{N}(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

  where $\mathcal{N}(\mathbf{x}_j)$ denotes the nearest neighbor list of $\mathbf{x}_j$ based on the Euclidean distance.

- Based on $\mathbf{W}$, they use Laplacian regularization:

$$\frac{1}{2}\sum_{i,j=1}^{n} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) W_{ij} = \mathrm{tr}(\mathbf{X}\mathbf{L}\mathbf{X}^{\mathbf{T}}\mathbf{M}),$$

  where $\mathbf{X}$ is the data matrix and $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix with $\mathbf{D}$ a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$.

Metric learning for other settings

Semi-supervised metric learning: LRML [Hoi et al., 2008]

Remarks

$$\frac{1}{2}\sum_{i,j=1}^{n} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) W_{ij} = \mathrm{tr}(\mathbf{X}\mathbf{L}\mathbf{X}^\mathsf{T}\mathbf{M})$$

- Intuition: favor a metric that is smooth over the adjacency graph defined by $\mathbf{W}$.
- This regularizer is obviously convex and can be plugged in existing metric learning algorithms.
- Significantly outperforms supervised approaches when side information is scarce.
- Drawback: computing $\mathbf{W}$ can be expensive for large datasets.

Metric learning for other settings

Metric learning for domain adaptation

Domain adaptation

Domain adaptation (DA) tries to tackle the case where the distribution of the test data is different from that of the training data. Can work when the two distributions are "not too different".

Example: ham or spam?

Spam filters are learned on a set of ham/spam emails, but once deployed on an actual user mailbox, the distribution of emails may be quite different.

Example: object recognition

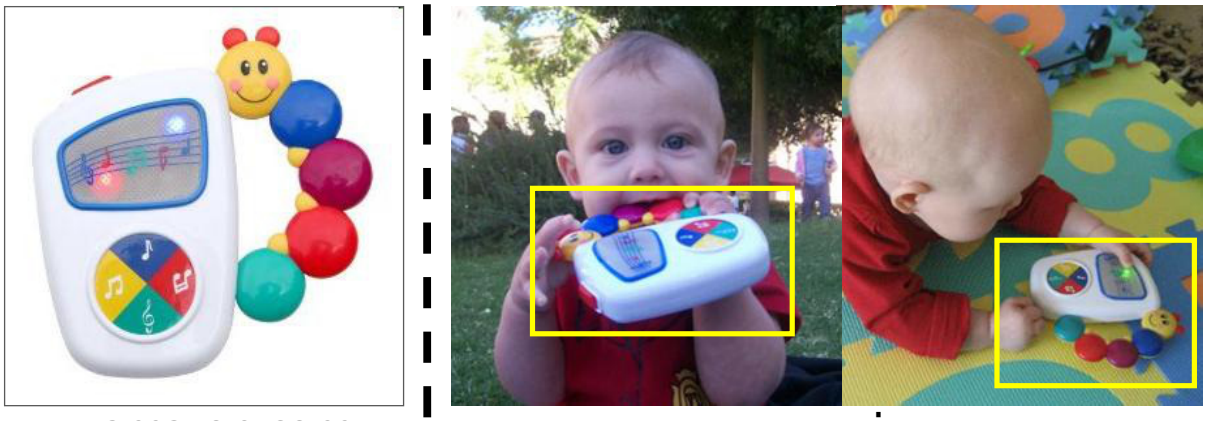
For instance, systems learned on high-resolution images but applied to webcam images.

Example: speech recognition

A speech recognition system must adapt to the user's specific voice, accent, local dialect, etc.

Metric learning for other settings

Metric learning for domain adaptation



Taken from [Kulis et al., 2011]

Metric learning for other settings

Metric learning for domain adaptation: DAML [Geng et al., 2011]

"Unsupervised" DA

The learner has access to labeled data of the source distribution but only unlabeled data from the target distribution.

Classic strategy

Bring the source and target distribution closer!

Application to metric learning

Use the empirical Maximum Mean Discrepancy (MMD), a nonparametric way of measuring the distribution difference between the source sample S and the target sample T:

$$MMD(S,T) = \left\| \frac{1}{|S|} \sum_{i=1}^{|S|} \phi(\mathbf{x}_i) - \frac{1}{|T|} \sum_{i=1}^{|T|} \phi(\mathbf{x}'_i) \right\|_{\mathcal{H}}^2,$$

where $\phi(\cdot)$ is a nonlinear feature mapping function to $\mathcal{H}$.

Metric learning for other settings

Metric learning for domain adaptation: DAML [Geng et al., 2011]

Remarks

$$MMD(S,T) = \left\| \frac{1}{|S|} \sum_{i=1}^{|S|} \phi(\mathbf{x}_i) - \frac{1}{|T|} \sum_{i=1}^{|T|} \phi(\mathbf{x}'_i) \right\|_{\mathcal{H}}^2$$

- The MMD can be computed efficiently using the kernel trick.
- Thus this convex term can be used as regularization in kernelized metric learning algorithms!
- Intuitively, learn a transformation in kernel space s.t. the distance performs well on source data while bringing distributions closer.
- Performs well in practice.
- Metric learning for DA has broader applicability than adapting a specific classifier.

Summary of the reviewed methods

Standard setting

	MMC	S&J	NCA	LMNN	ITML	LEGO
Year	2002	2003	2004	2005	2007	2008
Supervision	Weak	Weak	Full	Full	Weak	Weak
Form of metric	Linear	Linear	Linear	Linear	Linear	Linear
Scalability in n	★☆☆	★☆☆	★☆☆	★★☆	★☆☆	★★★
Scalability in d	☆☆☆	★★☆	★★☆	★☆☆	★★☆	★★☆
Optimum	Global	Global	Local	Global	Global	Global
Dim. reduc	No	No	Yes	No	Yes	Yes
Regularizer	None	Frobenius	None	None	LogDet	LogDet
Source code	Yes	No	Yes	Yes	Yes	No
Other info	—	—	For k-NN	For k-NN	—	Online

	OASIS	SLLC	GB-LMNN	MM-LMNN	PLML
Year	2009	2012	2012	2008	2012
Supervision	Weak	Full	Full	Full	Weak
Form of metric	Linear	Linear	Nonlinear	Local	Local
Scalability in n	★★★	★★☆	★★☆	★★☆	★★☆
Scalability in d	★★☆	★★☆	★★☆	★☆☆	☆☆☆
Optimum	Global	Global	Local	Global	Global
Dim. reduc	No	No	Yes	No	No
Regularizer	Frobenius	Frobenius	None	None	Manifold+Frob
Source code	Yes	No	Yes	Yes	Yes
Other info	Online	Linear classif.	—	—	—

Summary of the reviewed methods

Other settings

	mt-LMNN	**MLR**	**χ^2-LMNN**	**LRML**	**DAML**
Year	2010	2010	2012	2008	2011
Supervision	Full	Full	Full	Semi	Semi
Form of metric	Linear	Linear	Nonlinear	Linear	Nonlinear
Scalability in n	★★☆	★★☆	★★☆	★☆☆	★☆☆
Scalability in d	☆☆☆	☆☆☆	★★☆	☆☆☆	☆☆☆
Optimum	Global	Global	Local	Global	Global
Dim. reduc	No	Yes	Yes	No	No
Regularizer	Frobenius	Nuclear norm	None	Laplacian	MMD
Source code	Yes	Yes	No	Yes	No
Setting	Multi-task	Ranking	Histogram	Semi-sup.	Domain adapt.

Outline

Metric learning for structured data

An important but difficult problem

Structured metrics are convenient...

Metrics for structured data (strings, trees, graphs) act like a proxy to manipulate these complex objects: one can then use any metric-based algorithm!

...but state-of-the-art is limited

- Typically involves more complex procedures (structured metrics are combinatorial by nature).
- Often enough, structured objects are simply represented by a feature vector (e.g., bag-of-words), even though this implies a loss of structural information.
- Most of the work on learning metrics that operate directly on structured objects has focused on edit distance-based metrics.

Definition (Alphabet and string)

- An **alphabet** Σ is a finite nonempty set of symbols.
- A **string** x is a finite sequence of symbols from Σ.
- The **empty string**/symbol is denoted by \$.
- Σ^* is the set of all finite strings that can be generated from Σ.
- The **length** of a string x is denoted by $|x|$.

Definition (More formal definition of string edit distance)

Let **C** be a nonnegative $(|\Sigma|+1) \times (|\Sigma|+1)$ matrix giving the cost of the following elementary edit operations: insertion, deletion and substitution of a symbol, where symbols are taken from $\Sigma \cup \{\$\}$.
Given two strings $x, x' \in \Sigma^*$, an **edit script** is a sequence of operations that turns x into x'. The **string edit distance** between x and x' is defined as the cost of the cheapest edit script and can be computed in $O(|x| \cdot |x'|)$ time by dynamic programming.

Example 1: Standard (Levenshtein) distance

C	$	a	b
$	0	1	1
a	1	0	1
b	1	1	0

$\Longrightarrow$ edit distance between abb and aa is 2 (needs at least two operations)

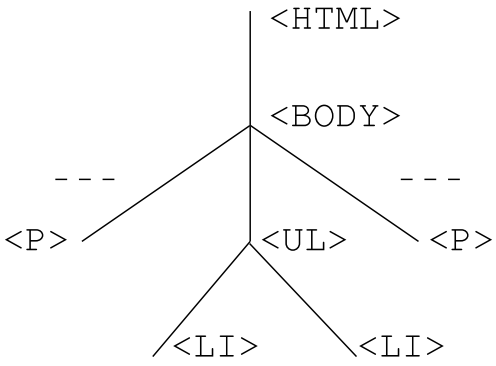
Example 2: Specific Cost Matrix

C	$	a	b
$	0	2	10
a	2	0	4
b	10	4	0

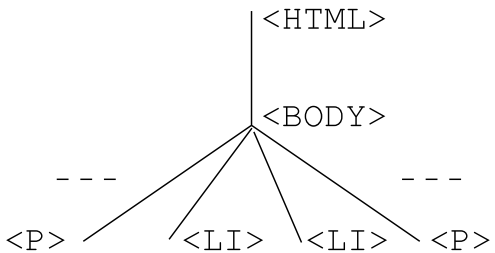
$\Longrightarrow$ edit distance between abb and aa is 10 ($\mathtt{a} \to \$, \mathtt{b} \to \mathtt{a}, \mathtt{b} \to \mathtt{a}$)

Tree edit distance: two variants

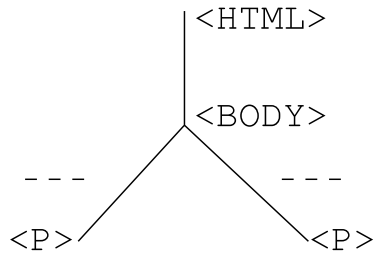
1. when a node is deleted all its children are connected to its father. Worst-case complexity: n^3, where n is the number of nodes in the largest tree.
2. insertions and deletions are restricted to the leaves of the tree. This version can be computed in quadratic time.



Original tree



after deletion of the <UL> node in the first variant



after deletion of the <UL> node in the second variant

Graph edit distance

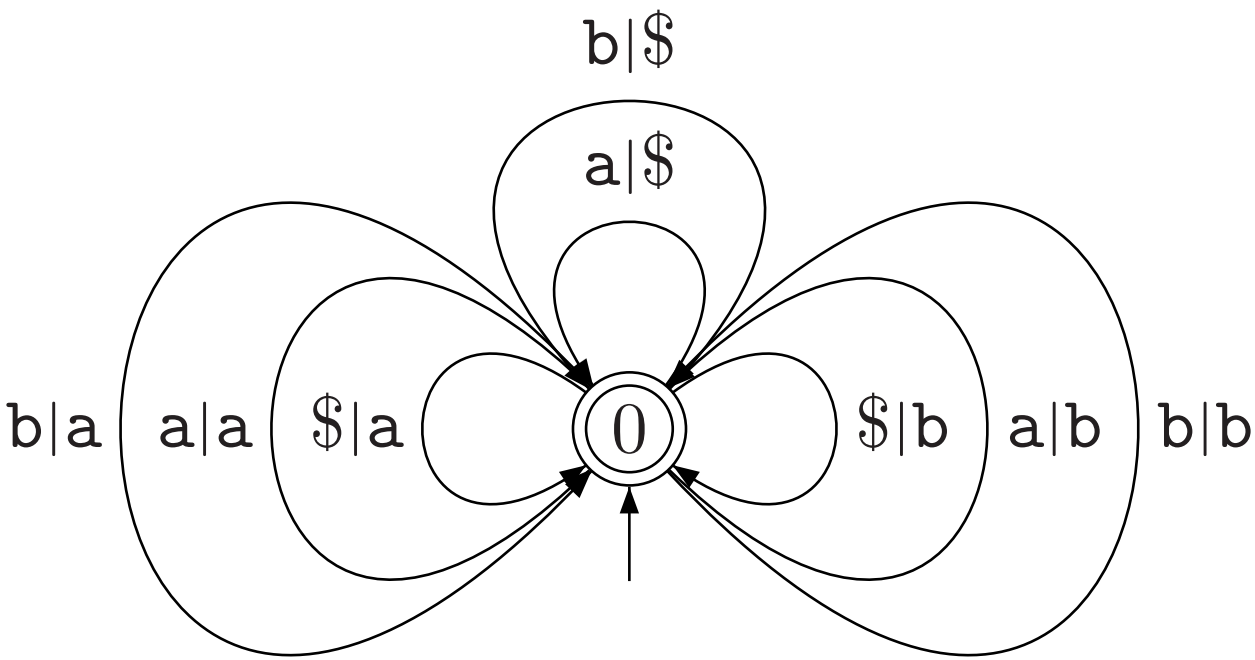
There an extension to general graphs, but it is NP-hard to compute.

Stochastic edit distance

- Edit distance learning is challenging because the optimal edit script depends on the costs themselves.
- Therefore, updating the costs may change the optimal edit script.
- Instead consider **stochastic version of ED**:
  - The cost matrix defines a probability distribution over the edit operations.
  - Define an edit similarity as the posterior probability $p_e(x'|x)$ that an input string x is turned into an output string x'.
  - Corresponds to summing over all possible scripts.
  - Represent this stochastic process by a probabilistic model and do parameter inference.
  - Maximize the expected log-likelihood of must-link pairs via an EM iterative procedure.

Metric learning for structured data

A stochastic memoryless transducer



Metric learning for structured data

Stochastic string edit distance learning: [Oncina and Sebban, 2006]

Expectation Step

Given current edit probabilities for each operation, compute the frequency of each operation. This takes the form of a probabilistic version of the dynamic programing algorithm for the standard edit distance.

Maximization Step

Given current frequencies, compute the updated probabilities for each operation by maximizing the likelihood of the training pairs under the constraints:

$$\forall u \in \Sigma, \sum_{v \in \Sigma \cup \{\$\}} \mathbf{C}_{v|u} + \sum_{v \in \Sigma} \mathbf{C}_{v|\$} = 1, \quad \text{with} \sum_{v \in \Sigma} \mathbf{C}_{v|\$} + c(\#) = 1,$$

where $\#$ is a termination symbol and $c(\#)$ the associated probability.

Metric learning for structured data

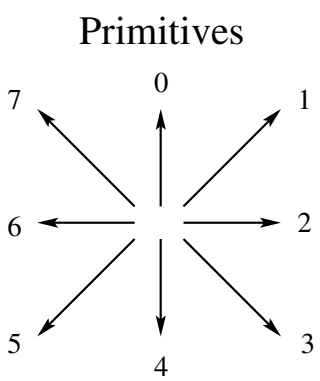
Stochastic string edit distance learning: O&S [Oncina and Sebban, 2006]

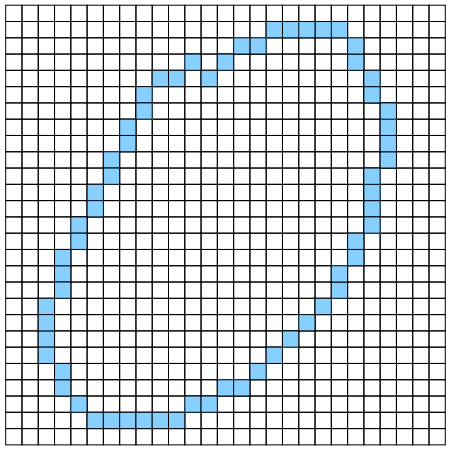
Remarks

- Advantages:
  - Elegant probabilistic framework.
  - Works reasonably well in practice in classification tasks, with the advantages of a probabilistic models (e.g., data generation).
  - The same idea can be used to learn a stochastic tree edit distance (but requires more complex steps).
- Drawbacks:
  - Converges only to a local minimum in practice.
  - Costly as it requires to run the DP algorithm on each pair at each iteration.
  - Not flexible: impossible to use cannot-link constraints or incorporate background knowledge.

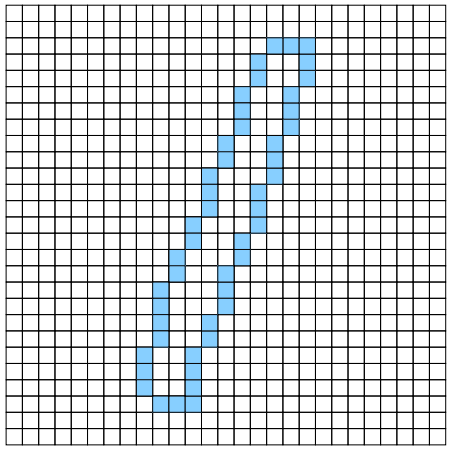
Metric learning for structured data

Stochastic string edit distance learning: O&S [Oncina and Sebban, 2006]

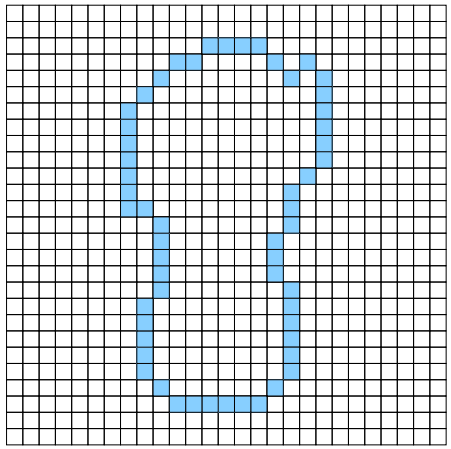




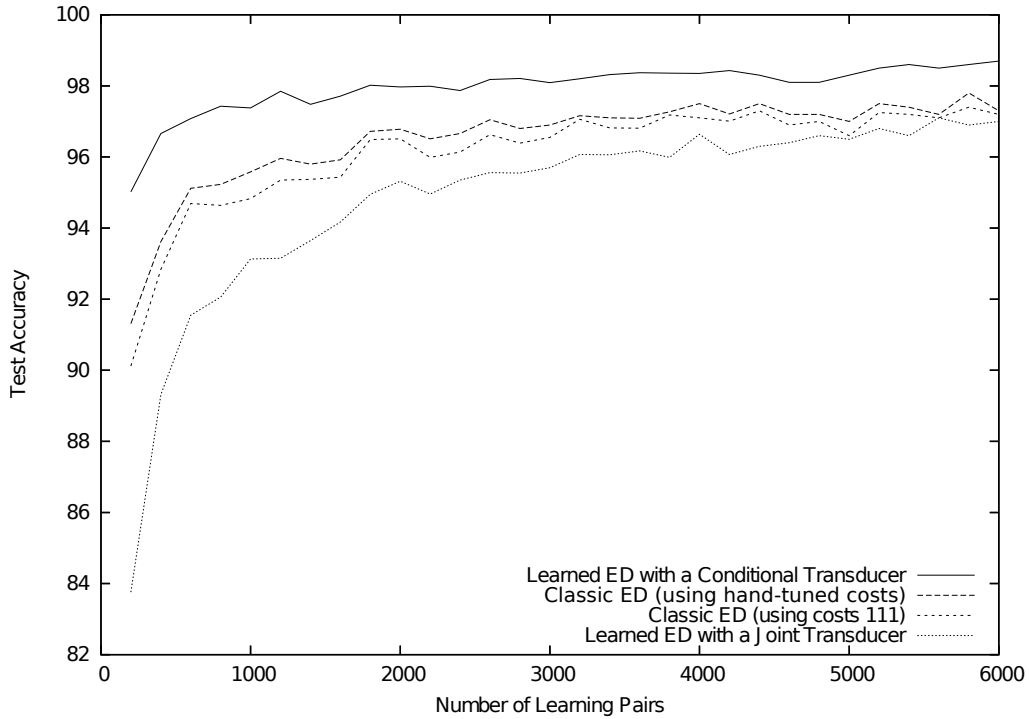
22223434344454445454555555656566
66677070001001010101012131121



22445445445445445445446670010
00101010010100101



22223313444445544544344444556666
66770000100007600000011121



Main idea

- Draw inspiration from the success stories of metric learning for feature vectors, based on large-margin constraints and convex optimization.
- Requires a key simplification of the problem: **fix the edit script**. New notion of edit distance:

$$e_{\mathbf{C}}(x, x') = \sum_{0 \leq i,j \leq |\Sigma|} C_{i,j} \times \#_{i,j}(x, x'),$$

where $\#_{i,j}(x, x')$ is the number of times the operation $i \to j$ appears in the Levenshtein script.

- $e_{\mathbf{C}}$ is simply a linear function of the edit cost matrix!

Formulation

$$\min_{\mathbf{C}\geq 0,\xi\geq 0,B_1\geq 0,B_2\geq 0} \quad \sum_{i,j}\xi_{ij} \quad + \quad \beta\|\mathbf{C}\|_{\mathcal{F}}^2$$

$$\begin{aligned} \text{s.t.} \quad & e_{\mathbf{C}}(\mathrm{x},\mathrm{x}') \geq B_1 - \xi_{ij} && \forall(\mathrm{x}_i,\mathrm{x}_j)\in\mathcal{D} \\ & e_{\mathbf{C}}(\mathrm{x},\mathrm{x}') \leq B_2 + \xi_{ij} && \forall(\mathrm{x}_i,\mathrm{x}_j)\in\mathcal{S}, \\ & B_1 - B_2 = \gamma, \end{aligned}$$

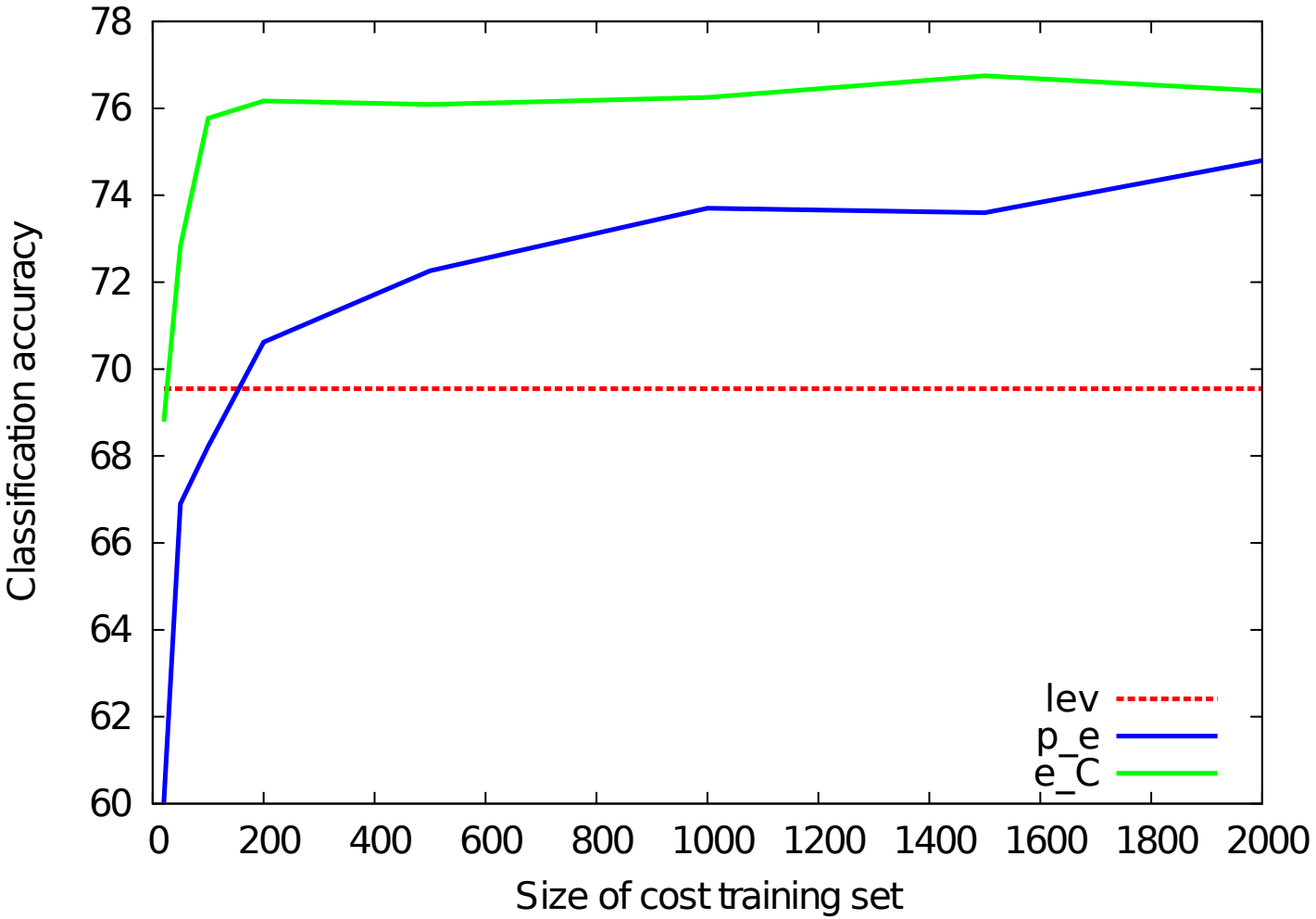
where γ is a margin parameter.

Remarks

- Advantages:
  - Convex, easier to solve than for Mahalanobis (no PSD constraint).
  - Does not suffer from limitations of stochastic edit distance learning.
  - Shown to outperform stochastic approach for k-NN and linear classification.
  - Straightforward adaptation to trees! (use Levenshtein tree edit script)
  - Generalization guarantees (more on this later).
- Drawbacks:
  - Less general than the proper edit distance.
  - Depends on the relevance of the Levenshtein script (although other scripts could be used).

Metric learning for structured data

Large-margin edit similarity learning: GESL [Bellet et al., 2012a]



Outline

Input

A sample of N_T labeled examples $T = \{z_i = (x_i, y_i)\}_{i=1}^{N_T}$ independently and identically distributed (i.i.d.) according to an unknown distribution P over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Simple case: **binary classification**, where $\mathcal{Y} = \{-1, 1\}$.

Output

A hypothesis (model) h that is able to **accurately predict** the labels of (unseen) examples drawn from P.

Definition (True risk)

Given a loss function ℓ measuring the agreement between the prediction $h(x)$ and the true label y, we define the **true risk** by:

$$R^{\ell}(h) \quad = \quad \mathbb{E}_{z \sim P}\left[\ell(h, z)\right].$$

Definition (Empirical risk)

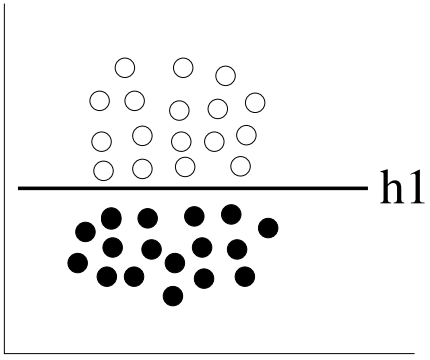
The **empirical risk** of an hypothesis h is the average loss suffered on the training sample T:

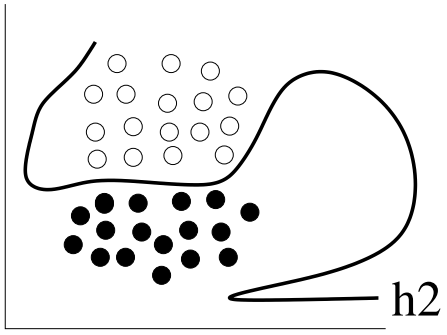
$$R_T^\ell(h) \quad = \quad \frac{1}{N_T} \sum_{i=1}^{N_T} \ell(h, z_i).$$

Generalization guarantees

Under some conditions, we may be able to **bound the deviation between the true risk and the empirical risk** of an hypothesis, i.e., how much we "trust" $R_T^\ell(h)$:

$$\Pr[|R^\ell(h) - R_T^\ell(h)| > \mu] \leq \delta. \qquad \text{(PAC bounds)}$$





Occam's razor principle

- "Pick simplest explanation consistent with past data".
- In practice: a trade-off between the complexity of h and the confidence we have in its generalization performance.

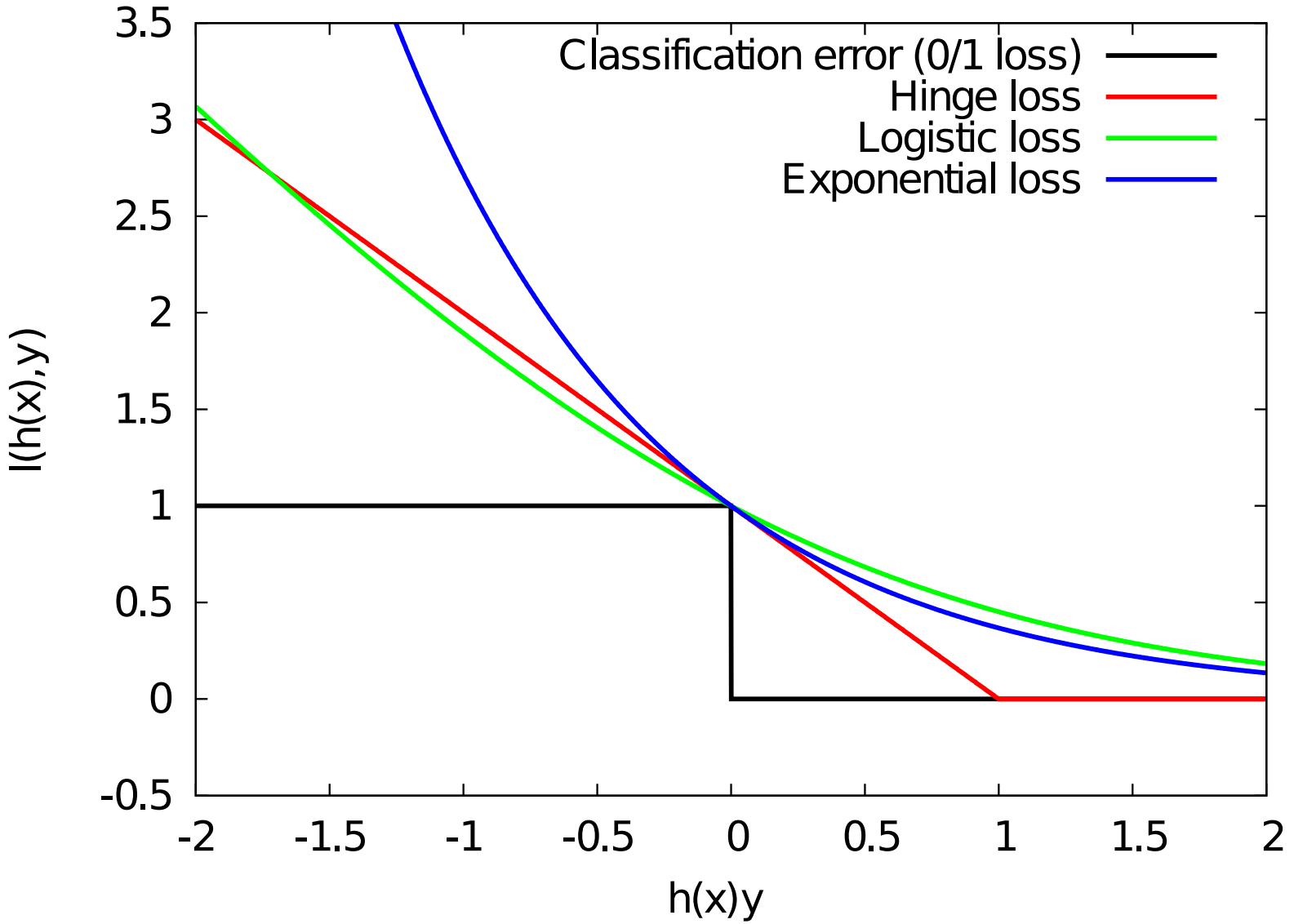
Regularized risk minimization

$$h_T = \arg\min_{h \in \mathcal{H}} \; R^{\ell}_T(h) + \lambda r(h),$$

where $\mathcal{H}$ is some hypothesis class (e.g., linear separators), $r(h)$ is a regularizer (usually some norm $\|h\|$) that penalizes "complex" hypotheses, and λ is the trade-off parameter.

Deriving generalization guarantees

Loss functions for binary classification



Metric learning as regularized risk minimization

Most metric learning formulations can be seen as regularized risk minimization.

Example: S&J formulation

$$\min_{\mathbf{M}\succeq 0, \xi \geq 0} \quad \|\mathbf{M}\|_{\mathcal{F}}^2 \quad + \quad C \sum_{i,j,k} \xi_{ijk}$$
$$\text{s.t.} \quad d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R}$$

Equivalent formulation

$$\min_{\mathbf{M}\succeq 0} \quad \|\mathbf{M}\|_{\mathcal{F}}^2 \quad + \quad C \sum_{i,j,k} [1 - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)]_+,$$

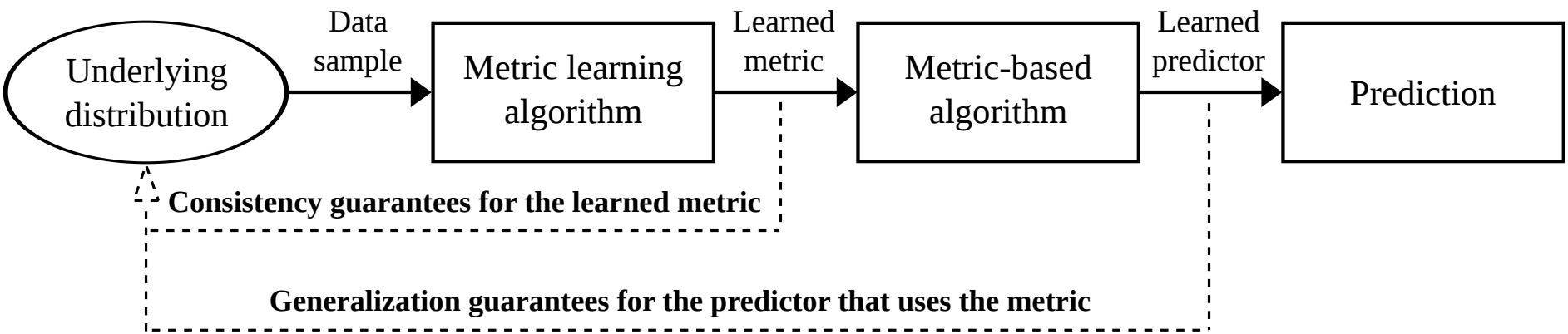
where $[\cdot]_+ = \max(0, \cdot)$ is the hinge loss.

Metric learning is a special, challenging problem

1. Work with pairs or triplets built from individual i.i.d. examples. Violates the i.i.d. assumption at the pair/triplet level.
2. In fact, can be seen as some sort of binary classification problem on pairs of examples. The label of individual examples does not matter: we are mostly interested in the "pair label" (similar/dissimilar).
3. The question of generalization is two-fold!

Deriving generalization guarantees

Back to metric learning



Deriving generalization guarantees

Consistency guarantees for the learned metric: uniform stability [Jin et al., 2009]

Definition (Uniform stability for metric learning)

A learning algorithm has a uniform stability in κ/N_T, where κ is a positive constant, if

$$\forall (T, z), \forall i, \sup_{z_1, z_2} |\ell(\mathcal{A}_T, z_1, z_2) - \ell(\mathcal{A}_{T^{i,z}}, z_1, z_2)| \leq \frac{\kappa}{N_T},$$

where $\mathcal{A}_T$ is the metric learned by $\mathcal{A}$ from T, and $T^{i,z}$ is the set obtained by replacing $z_i \in T$ by a new example z.

Theorem (Uniform stability bound)

For any algorithm $\mathcal{A}$ with uniform stability κ/N_T, with probability $1 - \delta$ over the random sample T, we have:

$$R^\ell(\mathcal{A}_T) \leq R^\ell_T(\mathcal{A}_T) + \frac{2\kappa}{N_T} + (2\kappa + B)\sqrt{\frac{\ln(2/\delta)}{2N_T}},$$

where B is a problem-dependent constant.

Deriving generalization guarantees

Consistency guarantees for the learned metric: uniform stability [Jin et al., 2009]

Applicability

Can essentially be applied to regularized risk minimization procedures that are convex (need optimality of the solution) with smooth regularization (typically the Frobenius norm).

Stability of SLLC

$$\kappa \approx \frac{1}{\gamma}(\frac{1}{\beta\gamma} + \frac{2}{\alpha}),$$

where α the proportion of reference points in the training sample.

Stability of GESL

$$\kappa \approx \frac{2(2+\alpha)W^2}{\beta},$$

where W is a bound on the string sizes.

Deriving generalization guarantees

Consistency guarantees for the learned metric: robustness [Bellet and Habrard, 2012]

Sparse algorithms are not stable [Xu et al., 2012]

Rules out formulations that use the 1-norm (sparse entries) or the nuclear norm (sparse spectrum). Need other framework.

Definition (Robustness for metric learning)

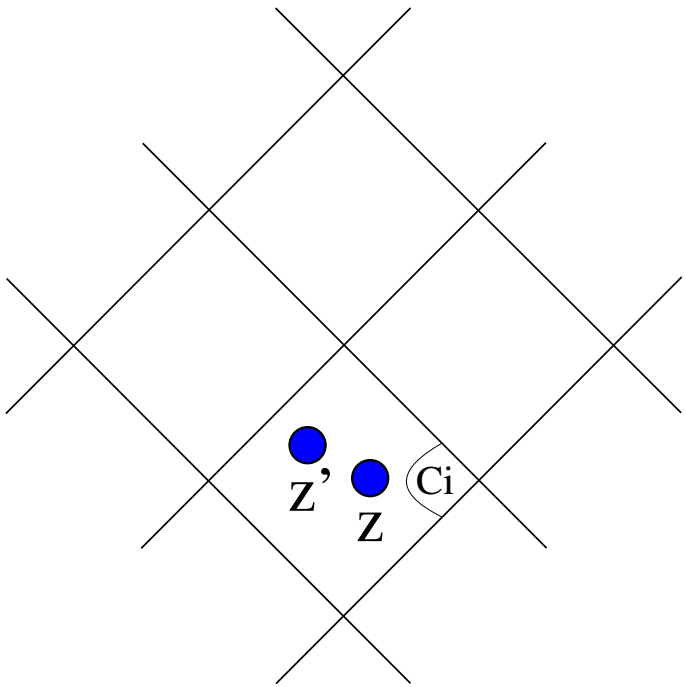
An algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$ robust for $K \in \mathbb{N}$ and $\epsilon(\cdot) : (\mathcal{Z} \times \mathcal{Z})^{N_T} \to \mathbb{R}$ if $\mathcal{Z}$ can be partitioned into K disjoints sets, denoted by $\{C_i\}_{i=1}^{K}$, such that the following holds for all $T \in \mathcal{Z}^{N_T}$:
$\forall (z_1, z_2) \in \mathcal{P}_T, \forall z, z' \in \mathcal{Z}, \forall i, j \in [K]$: if $z_1, z \in C_i$ and $z_2, z' \in C_j$ then

$$|\ell(\mathcal{A}_{\mathcal{P}_T}, z_1, z_2) - \ell(\mathcal{A}_{\mathcal{P}_T}, z, z')| \leq \epsilon(\mathcal{P}_T),$$

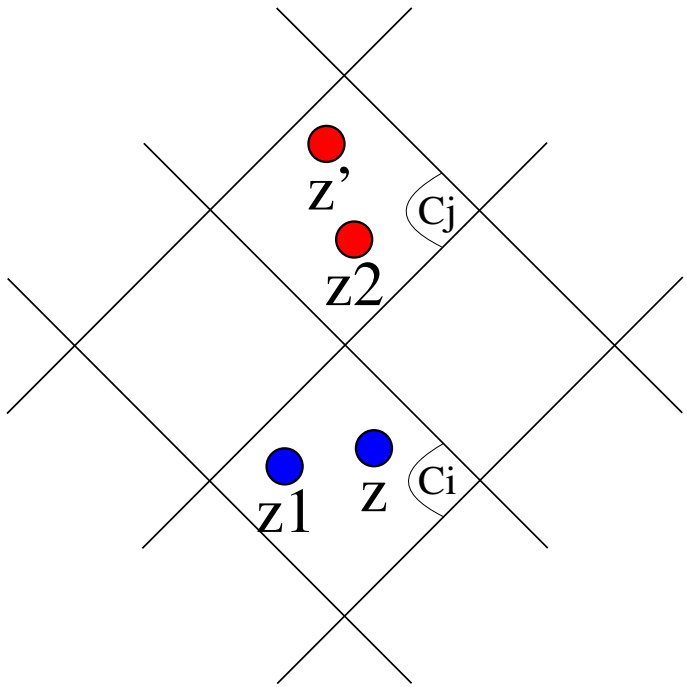
where $\mathcal{P}_T$ are the pairs built from T.

Deriving generalization guarantees

Consistency guarantees for the learned metric: robustness [Bellet and Habrard, 2012]



Classic robustness



Robustness for metric learning

Deriving generalization guarantees

Consistency guarantees for the learned metric: robustness [Bellet and Habrard, 2012]

Theorem (Robustness bound)

If a learning algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$-robust, then for any $\delta > 0$, with probability at least $1 - \delta$ we have:

$$|R^{\ell}(\mathcal{A}_{\mathcal{P}_T}) - R^{\ell}_{\mathcal{P}_T}(\mathcal{A}_{\mathcal{P}_T})| \leq \epsilon(\mathcal{P}_T) + 2B\sqrt{\frac{2K \ln 2 + 2\ln(1/\delta)}{N_T}}.$$

Applicability

Can essentially be applied to regularized risk minimization procedures that are convex with various regularizers (typically bounded below by some p-norm). Can make use of equivalence of norms. Less specific arguments.

Results for some general form

$$\min_{\mathbf{M} \succeq 0} \quad \sum_{(z_i, z_j) \in \mathcal{P}_{\mathcal{T}}} \ell(d^2_{\mathbf{M}}, z_i, z_j) \quad + \quad C\|\mathbf{M}\|$$

Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: (ϵ, γ, τ)-goodness

Definition (Balcan et al., 2008)

A similarity function $K \in [-1, 1]$ is an (ϵ, γ, τ)-**good similarity function** if there exists an indicator function $R(x)$ defining a set of "reasonable points" such that the following conditions hold:

1. A $1 - \epsilon$ probability mass of examples (x, y) satisfy:

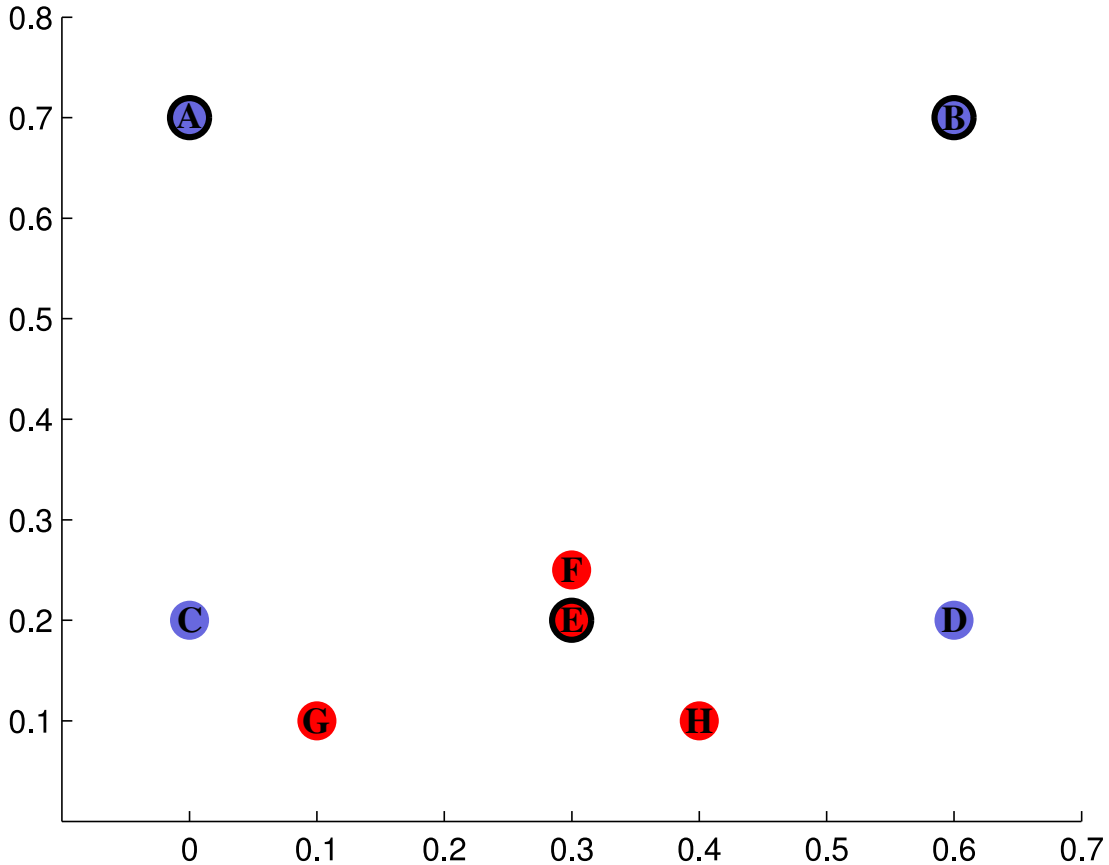
$$\mathbb{E}_{(x',y')\sim P}\left[yy'K(x,x')|R(x')\right] \geq \gamma.$$

2. $\Pr_{x'}[R(x')] \geq \tau.$

$\epsilon, \gamma, \tau \in [0, 1]$

Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: (ϵ, γ, τ)-goodness



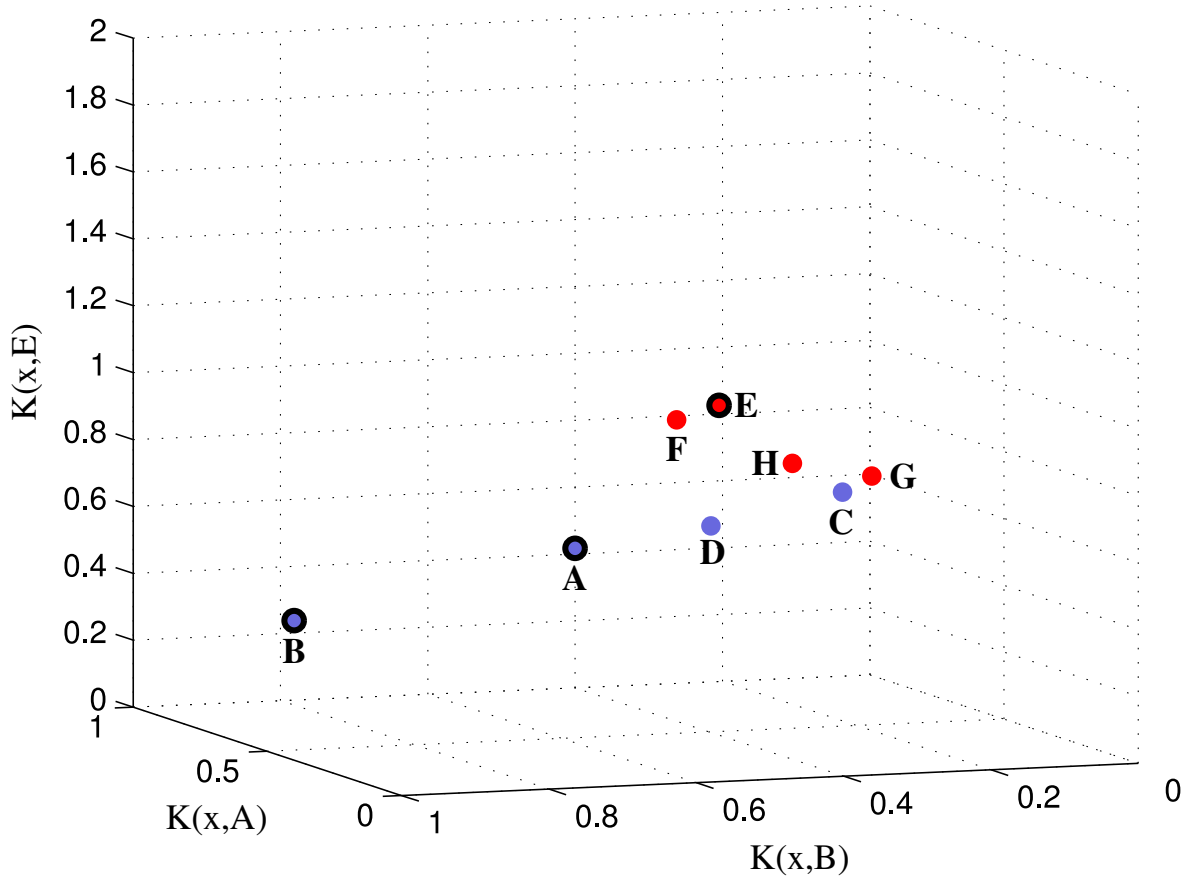
	A	B	C	D	E	F	G	H
A	1	0.40	0.50	0.22	0.42	0.46	0.39	0.28
B	0.40	1	0.22	0.50	0.42	0.46	0.22	0.37
E	0.42	0.42	0.70	0.70	1	0.95	0.78	0.86
Margin γ	0.3277	0.3277	0.0063	0.0063	0.0554	0.0106	0.0552	0.0707

Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: (ϵ, γ, τ)-goodness

Strategy

If R is known, use K to map the examples to the space ϕ of "the similarity scores with the reasonable points" (**similarity map**).

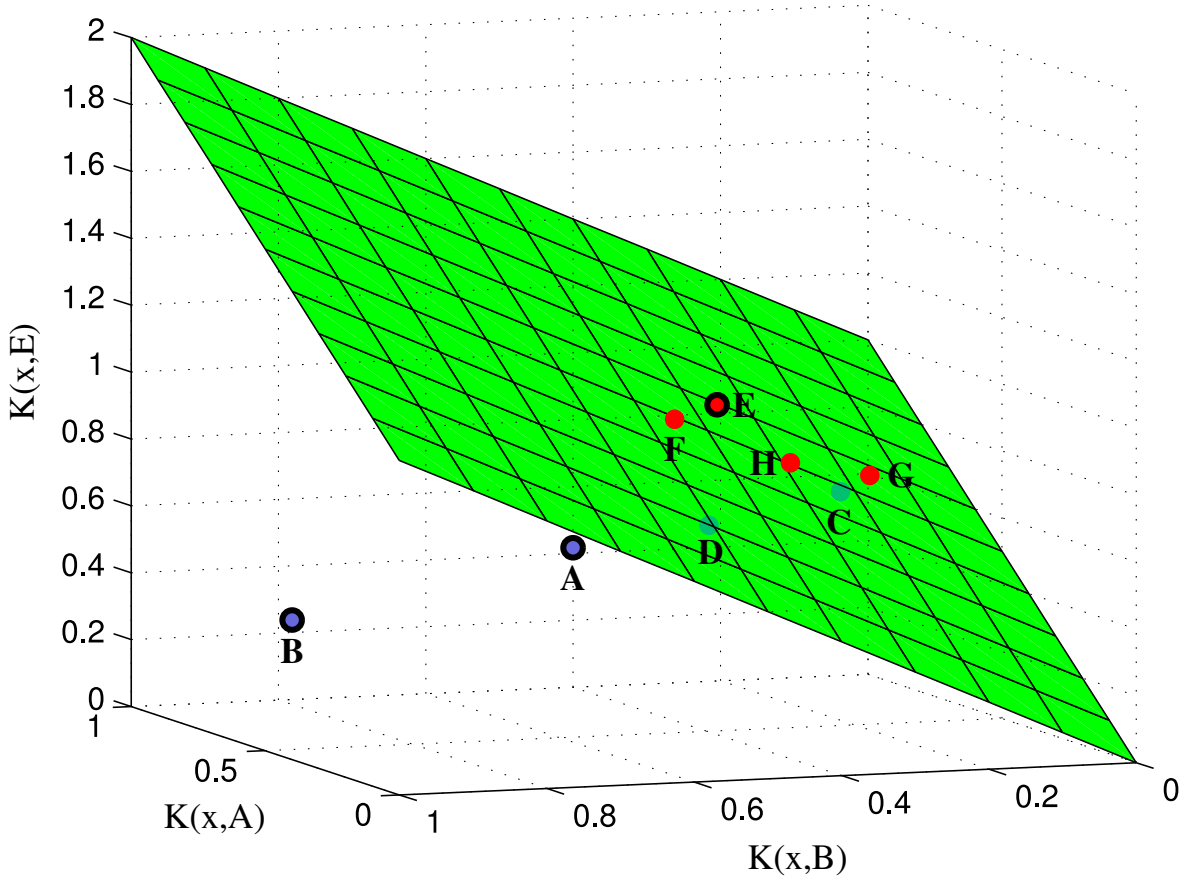


Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: (ϵ, γ, τ)-goodness

A trivial linear classifier

By definition of (ϵ, γ, τ)-goodness, we have a linear classifier in ϕ that achieves true risk ϵ at margin γ.



Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: (ϵ, γ, τ)-goodness

Theorem (Balcan et al., 2008)

If R is unknown, given K is (ϵ, γ, τ)-good and enough points to create a similarity map, with high probability there exists a linear separator α that has true risk ϵ at margin γ.

Question

Can we find this linear classifier in an efficient way?

Answer

Basically, yes: solve a Linear Program with 1-norm regularization. We get a sparse linear classifier.

Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: SLLC and GESL

GESL and SLLC have guarantees on the linear classifier

- SLLC and GESL optimize (a stricter version of) the (ϵ, γ, τ)-goodness. Their empirical/true risk are related to the empirical/true goodness.
- We have shown risk bounds for the metric learned by these algorithms.
- By the theory of (ϵ, γ, τ)-goodness, this implies guarantees for the linear classifier using the metric!

Example: uniform stability bound for SLLC

With probability $1 - \delta$, we have:

$$\epsilon \leq \epsilon_T + \frac{\kappa}{N_T} + (2\kappa + 1)\sqrt{\frac{\ln 1/\delta}{2N_T}},$$

where ϵ_T is the empirical goodness on T.

Outline

Metric learning for numerical data

- Has now reached a good level of maturity.
- Can deal with large spectrum of settings in a scalable way:
  - online methods for large-scale metric learning.
  - tackle complex tasks with nonlinear metric learning.
  - difficult settings such as ranking, multi-task, domain adaptation.

Metric learning for structured data

- Much less work.
- Advances in metric learning for numerical data have not yet propagated to structured data.
- Recent methods take inspiration from these. Promising direction.

Summary and outlook

Outlook

Interesting open questions

- Scalability in both n and d.
- More theoretical understanding. For instance, no link between quality of the learned metric and generalization performance of k-NN.
- What is unsupervised metric learning? In particular, what does it mean for a metric to be good for clustering?
- Take the structure of data into account (promising example of histogram data).
- Adapt metrics to changing data (lifelong learning, detect concept drifts).
- Learn richer metrics: notion of similarity is often multimodal. In particular, there are several ways in which two objects can be seen as similar, and there are several degrees of similarity (as opposed to the binary view similar vs. dissimilar).

Thanks a lot for your attention!

References I

[Balcan et al., 2008] Balcan, M.-F., Blum, A., and Srebro, N. (2008).
Improved Guarantees for Learning via Similarity Functions.
In *COLT*, pages 287–298.

[Bellet and Habrard, 2012] Bellet, A. and Habrard, A. (2012).
Robustness and Generalization for Metric Learning.
Technical report, University of Saint-Etienne.
arXiv:1209.1086.

[Bellet et al., 2012a] Bellet, A., Habrard, A., and Sebban, M. (2012a).
Good edit similarity learning by loss minimization.
Machine Learning Journal, 89(1):5–35.

[Bellet et al., 2012b] Bellet, A., Habrard, A., and Sebban, M. (2012b).
Similarity Learning for Provably Accurate Sparse Linear Classification.
In *ICML*, pages 1871–1878.

References II

[Bellet et al., 2013] Bellet, A., Habrard, A., and Sebban, M. (2013).
A Survey on Metric Learning for Feature Vectors and Structured Data.
Technical report, arXiv:1306.6709.

[Chatpatanasiri et al., 2010] Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., and Kijsirikul, B. (2010).
A new kernelization framework for Mahalanobis distance learning algorithms.
Neurocomputing, 73:1570–1579.

[Chechik et al., 2009] Chechik, G., Shalit, U., Sharma, V., and Bengio, S. (2009).
An Online Algorithm for Large Scale Image Similarity Learning.
In *NIPS*, pages 306–314.

References III

[Davis et al., 2007] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007).
Information-theoretic metric learning.
In *ICML*, pages 209–216.

[Geng et al., 2011] Geng, B., Tao, D., and Xu, C. (2011).
DAML: Domain Adaptation Metric Learning.
IEEE Transactions on Image Processing, 20(10):2980–2989.

[Goldberger et al., 2004] Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004).
Neighbourhood Components Analysis.
In *NIPS*, pages 513–520.

References IV

[Hoi et al., 2008] Hoi, S. C., Liu, W., and Chang, S.-F. (2008).
Semi-supervised distance metric learning for Collaborative Image Retrieval.
In *CVPR*.

[Jain et al., 2008] Jain, P., Kulis, B., Dhillon, I. S., and Grauman, K. (2008).
Online Metric Learning and Fast Similarity Search.
In *NIPS*, pages 761–768.

[Jin et al., 2009] Jin, R., Wang, S., and Zhou, Y. (2009).
Regularized Distance Metric Learning: Theory and Algorithm.
In *NIPS*, pages 862–870.

References V

[Kedem et al., 2012] Kedem, D., Tyree, S., Weinberger, K., Sha, F., and Lanckriet, G. (2012).
Non-linear Metric Learning.
In *NIPS*, pages 2582–2590.

[Kulis et al., 2011] Kulis, B., Saenko, K., and Darrell, T. (2011).
What you saw is not what you get: Domain adaptation using asymmetric kernel transforms.
In *CVPR*, pages 1785–1792.

[McFee and Lanckriet, 2010] McFee, B. and Lanckriet, G. R. G. (2010).
Metric Learning to Rank.
In *ICML*, pages 775–782.

References VI

[Oncina and Sebban, 2006] Oncina, J. and Sebban, M. (2006).
Learning Stochastic Edit Distance: application in handwritten character recognition.
Pattern Recognition, 39(9):1575–1587.

[Parameswaran and Weinberger, 2010] Parameswaran, S. and Weinberger, K. Q. (2010).
Large Margin Multi-Task Metric Learning.
In *NIPS*, pages 1867–1875.

[Schultz and Joachims, 2003] Schultz, M. and Joachims, T. (2003).
Learning a Distance Metric from Relative Comparisons.
In *NIPS*.

[Wang et al., 2012] Wang, J., Woznica, A., and Kalousis, A. (2012).
Parametric Local Metric Learning for Nearest Neighbor Classification.
In *NIPS*, pages 1610–1618.

References VII

[Weinberger et al., 2005] Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2005).
Distance Metric Learning for Large Margin Nearest Neighbor Classification.
In *NIPS*, pages 1473–1480.

[Weinberger and Saul, 2009] Weinberger, K. Q. and Saul, L. K. (2009).
Distance Metric Learning for Large Margin Nearest Neighbor Classification.
JMLR, 10:207–244.

[Xing et al., 2002] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. J. (2002).
Distance Metric Learning with Application to Clustering with Side-Information.
In *NIPS*, pages 505–512.

References VIII

[Xu et al., 2012] Xu, H., Caramanis, C., and Mannor, S. (2012).
Sparse Algorithms Are Not Stable: A No-Free-Lunch Theorem.
TPAMI, 34(1):187–193.