SIMILARITY AND DISTANCE METRIC LEARNING

MDI 341, MS Big Data, Télécom ParisTech

Aurélien Bellet

Inria Lille

March 5, 2018

OUTLINE

1. Introduction
2. Linear metric learning
3. Nonlinear extensions
4. Large-scale metric learning
5. Metric learning for structured data

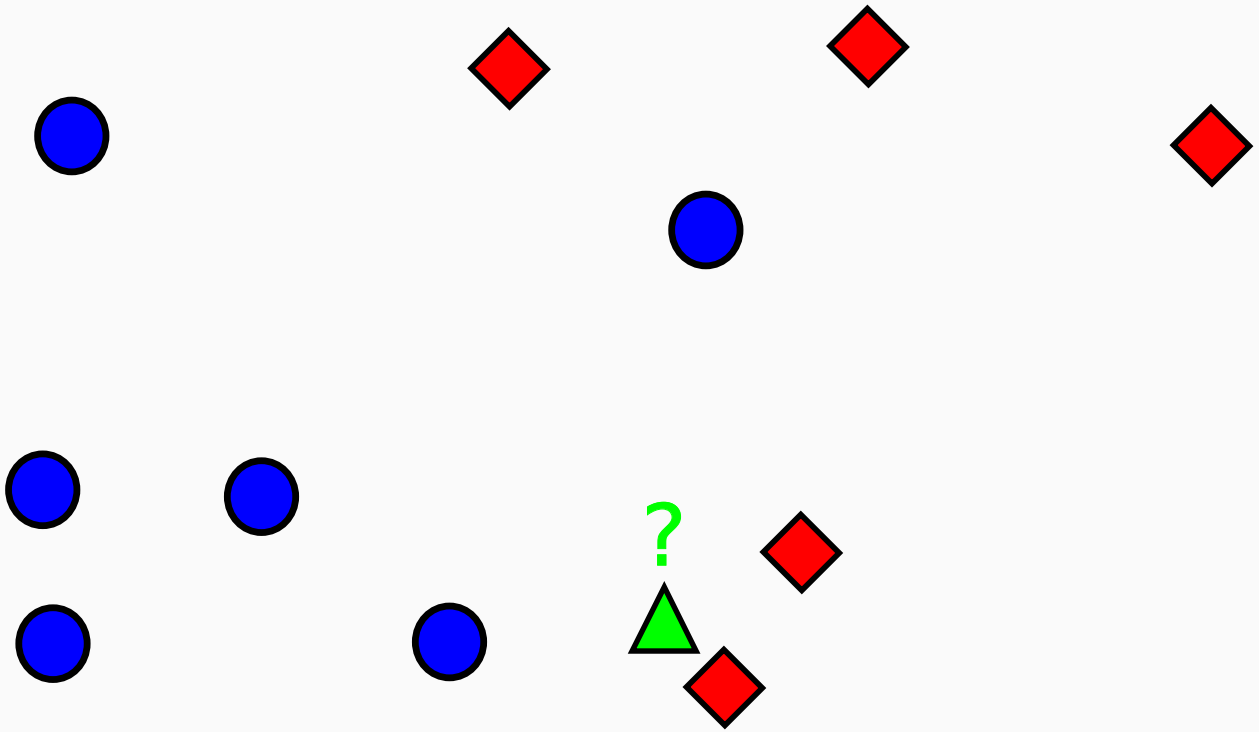
INTRODUCTION

FUNDAMENTAL QUESTION OF THIS LECTURE

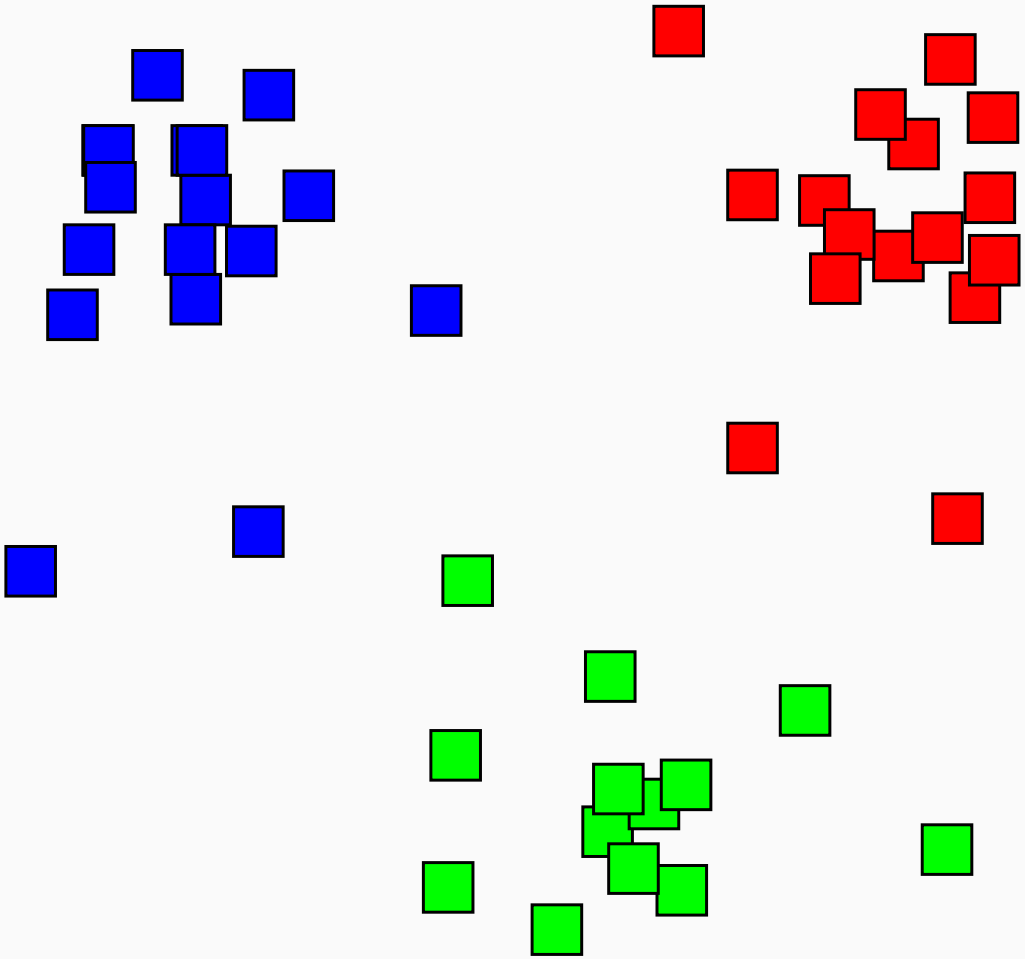
How to appropriately measure similarity or distance between things depending on the context?

- We (humans) are good at this [Tversky, 1977, Goldstone et al., 1997]
  - Recognize similar objects, sounds, ideas, etc, from past experience
  - Adapt the notion of similarity to the context
- AI systems need to do it too!
  - Categorize / retrieve data based on similarity to known examples
  - Detect situations similar to past experience

Nearest neighbor classification

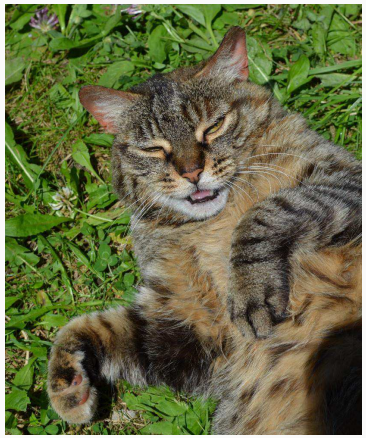


Clustering

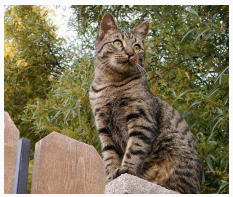


Information retrieval

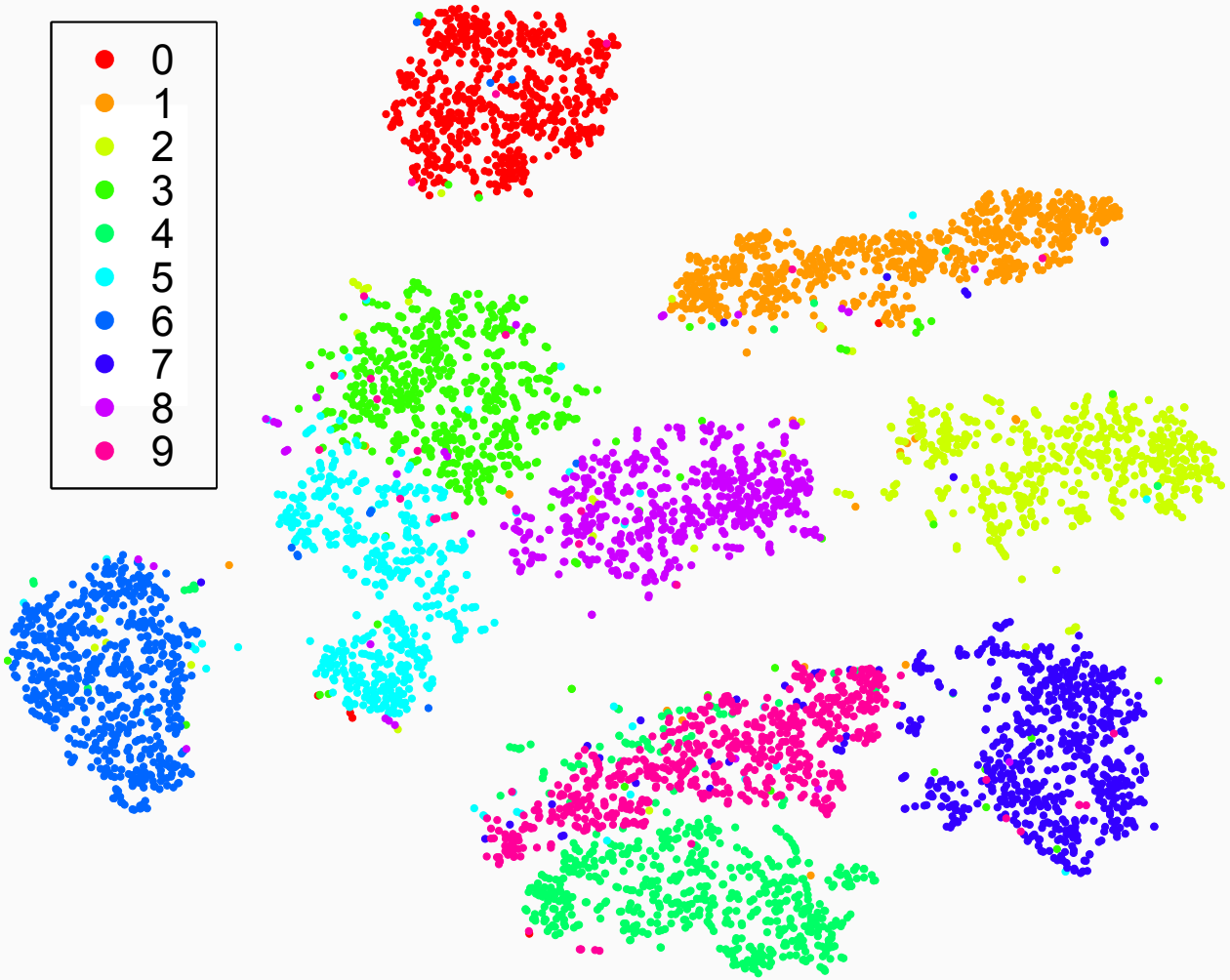
Query document



Most similar documents


Data visualization



(image taken from [van der Maaten and Hinton, 2008])

A GENERAL APPROACH: METRIC LEARNING

- Assume data represented in space $\mathcal{X}$ (e.g., $\mathcal{X} \subset \mathbb{R}^p$)
- We provide the system with some similarity judgments on data pairs/triplets for the task of interest


similar to



not similar to



more similar to

than to


(images taken from Caltech Faces dataset)

- The system uses this information to find the most "appropriate" pairwise distance/similarity function $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

(Note: I will refer to D as a metric regardless of its properties)

WHY NOT SIMPLY LEARN A CLASSIFIER?

- Case 1: huge number of classes (likely with class imbalance)
  - No need to learn many classifiers (as in 1-vs-1, 1-vs-all)
  - No blow-up in number of parameters (as in Multinomial Log. Reg.)
- Case 2: labels are unknown (and costly to obtain)
  - Similarity judgments often easier to label than individual points
  - Fully unsupervised generation possible in some applications
- Case 3: a pairwise metric is really what we need
  - Information retrieval (rank results by similarity to a query)

EXAMPLE APPLICATION: FACE IDENTIFICATION

- Face identification combines all of the above
  - Huge number of classes, with few instances in each class
  - Similarity judgments easy to crowdsource / generate
  - Given a new image, rank database by similarity and decide whether to match
- State-of-the-art results in empirical evaluations
  - Labeled Faces in the Wild [Zhu et al., 2015]
  - YouTube Faces [Hu et al., 2014]
- Popular in industry as well

(examples of positive pairs correctly classified from [Guillaumin et al., 2009])

Basic recipe

1. Pick a parametric distance or similarity function
   - Say, a distance $D_M(x, x')$ function parameterized by a matrix M
2. Collect similarity judgments on data pairs/triplets
   - $\mathcal{S} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ are similar}\}$
   - $\mathcal{D} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ are dissimilar}\}$
   - $\mathcal{R} = \{(x_i, x_j, x_k) : x_i \text{ is more similar to } x_j \text{ than to } x_k\}$
3. Estimate parameters s.t. metric best agrees with judgments
   - Solve an optimization problem of the form

$$M^* = \arg\min_M \Big[\underbrace{\ell(M, \mathcal{S}, \mathcal{D}, \mathcal{R})}_{\text{loss function}} + \underbrace{\lambda reg(M)}_{\text{regularization}} \Big]$$

LINEAR METRIC LEARNING

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)

- Note: a pseudo-distance satisfies the above except (2)

Minkowski distances

- A family of distances induced by L_p norms ($p \geq 1$)

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

- When $p = 2$: "ordinary" Euclidean distance

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

- When $p = 1$: Manhattan distance $d_{man}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{d} |x_i - x_i'|$
- When $p \to \infty$: Chebyshev distance $d_{che}(\boldsymbol{x}, \boldsymbol{x}') = \max_i |x_i - x_i'|$

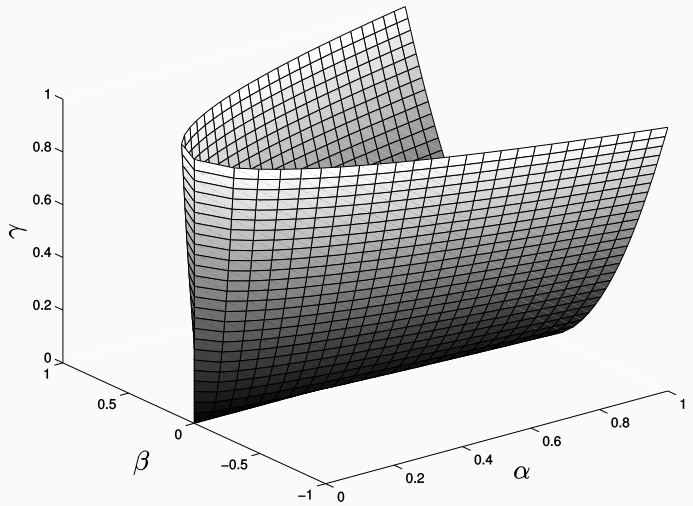
MAHALANOBIS DISTANCE

- Mahalanobis (pseudo) distance:

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

  where $M \in \mathbb{R}^{d \times d}$ is symmetric positive semi-definite (PSD)

- Denote by $\mathbb{S}_+^d$ the cone of symmetric PSD $d \times d$ matrices



MAHALANOBIS DISTANCE

- A symmetric matrix $\boldsymbol{M}$ is in $\mathbb{S}_+^d$ (also denoted $\boldsymbol{M} \succeq 0$) iff:
  - Its eigenvalues are all nonnegative
  - $\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x} \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^d$
  - $\boldsymbol{M} = \boldsymbol{L}^T \boldsymbol{L}$ for some $\boldsymbol{L} \in \mathbb{R}^{k \times d}$, $k \leq d$

- Equivalent to Euclidean distance after linear transformation:

$$D_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{L}^T \boldsymbol{L} (\boldsymbol{x} - \boldsymbol{x}')} = \sqrt{(\boldsymbol{L}\boldsymbol{x} - \boldsymbol{L}\boldsymbol{x}')^T (\boldsymbol{L}\boldsymbol{x} - \boldsymbol{L}\boldsymbol{x}')}$$

- If $\text{rank}(\boldsymbol{M}) = k \leq d$, then $\boldsymbol{L} \in \mathbb{R}^{k \times d}$ does dimensionality reduction

- For convenience, we often work with the squared distance

A first approach with pairwise constraints [Xing et al., 2002]

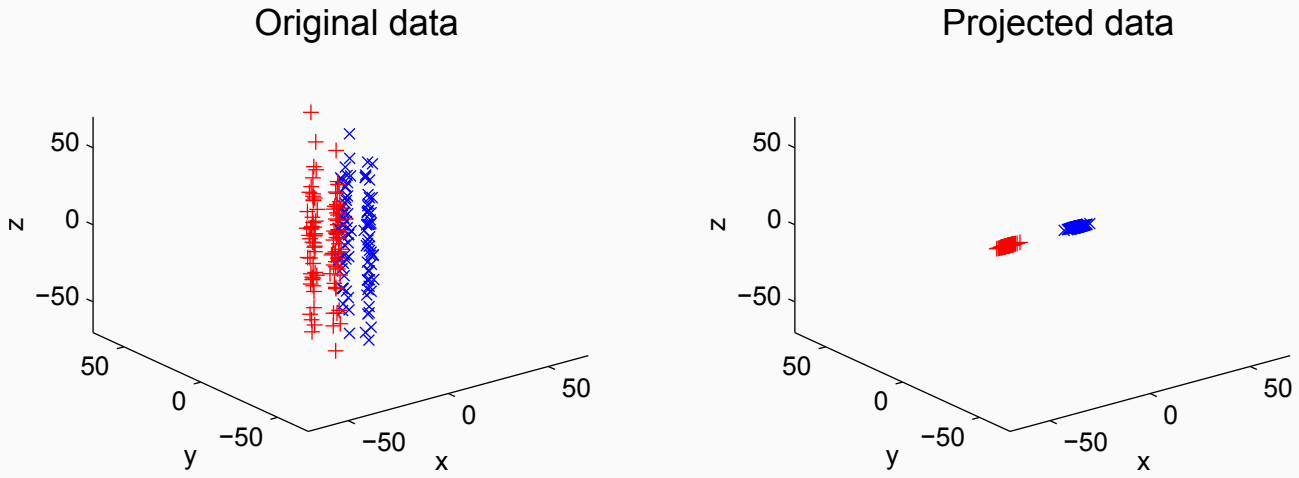
- Targeted task: clustering with side information

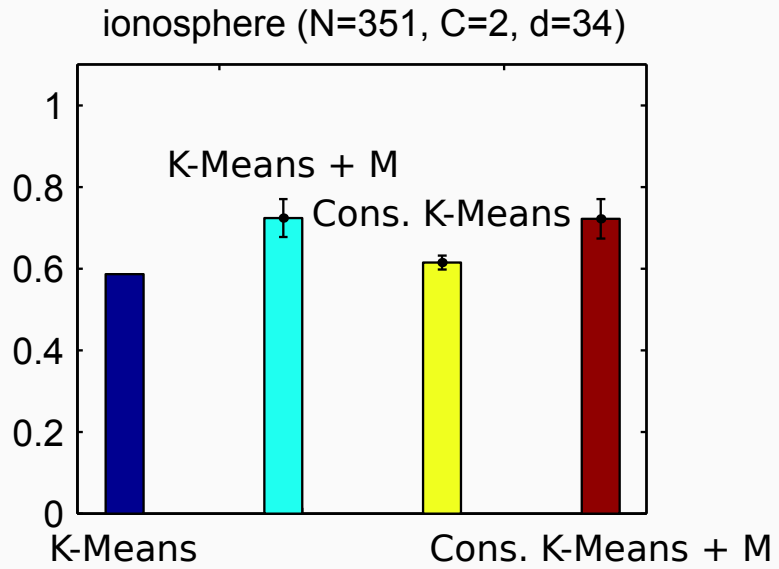
Formulation

$$\begin{aligned} \max_{M \in \mathbb{S}_+^d} \quad & \sum_{(x_i, x_j) \in \mathcal{D}} D_M(x_i, x_j) \\ \text{s.t.} \quad & \sum_{(x_i, x_j) \in \mathcal{S}} D_M^2(x_i, x_j) \leq 1 \end{aligned}$$

- Problem is convex in M and always feasible (take $M = 0$)
- Solved with projected gradient descent
  - Project onto distance constraint: $O(d^2)$ time
  - Project onto $\mathbb{S}_+^d$: $O(d^3)$ time
- Only look at sums of distances

A first approach with pairwise constraints [Xing et al., 2002]





A first approach with triplet constraints [Schultz and Joachims, 2003]

Formulation

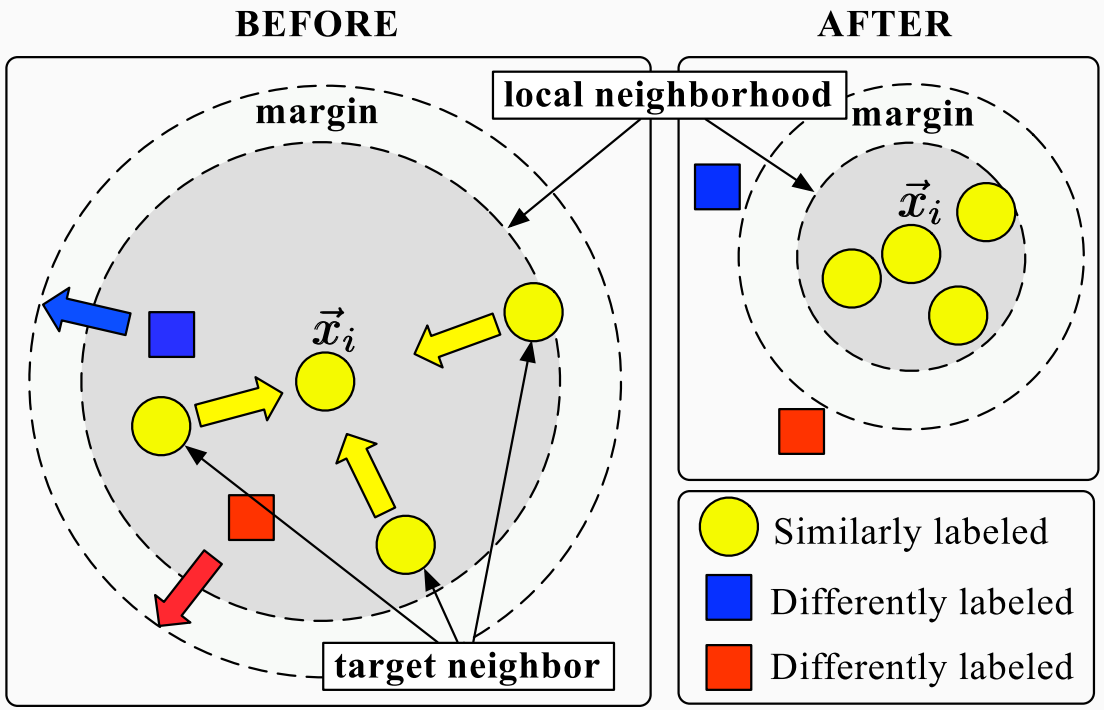
$$\min_{\boldsymbol{M} \in \mathbb{S}_+^d, \boldsymbol{\xi} \geq 0} \quad \|\boldsymbol{M}\|_{\mathcal{F}}^2 \quad + \quad \lambda \sum_{i,j,k} \xi_{ijk}$$

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

- Regularization by Frobenius norm $\|\boldsymbol{M}\|_{\mathcal{F}}^2 = \sum_{i,j=1}^d M_{ij}^2$
- Formulation is convex
- One large margin soft constraint per triplet
- Can be solved with similar techniques as SVM

Large Margin Nearest Neighbor [Weinberger et al., 2005]

- Targeted task: k-NN classification
- Constraints derived from labeled data
  - $\mathcal{S} = \{(x_i, x_j) : y_i = y_j, x_j \text{ belongs to } k\text{-neighborhood of } x_i\}$
  - $\mathcal{R} = \{(x_i, x_j, x_k) : (x_i, x_j) \in \mathcal{S}, y_i \neq y_k\}$



Large Margin Nearest Neighbor [Weinberger et al., 2005]

Formulation

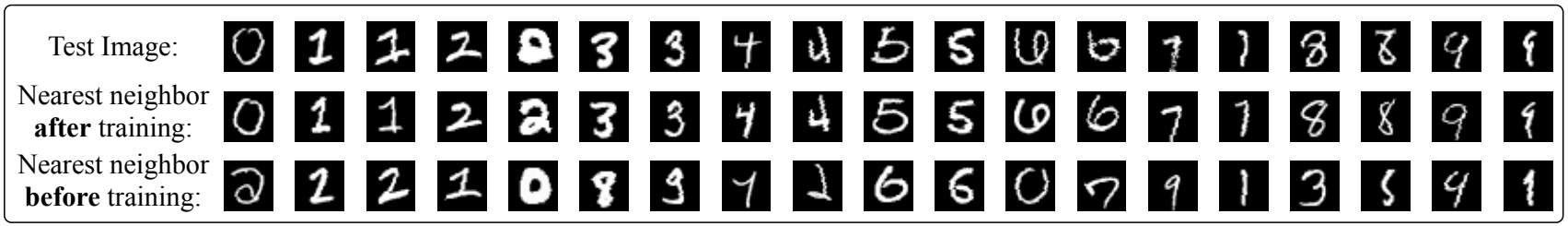
$$\min_{M \in \mathbb{S}_+^d, \boldsymbol{\xi} \geq 0} \quad (1-\mu) \sum_{(x_i, x_j) \in \mathcal{S}} D_M^2(x_i, x_j) \quad + \quad \mu \sum_{i,j,k} \xi_{ijk}$$

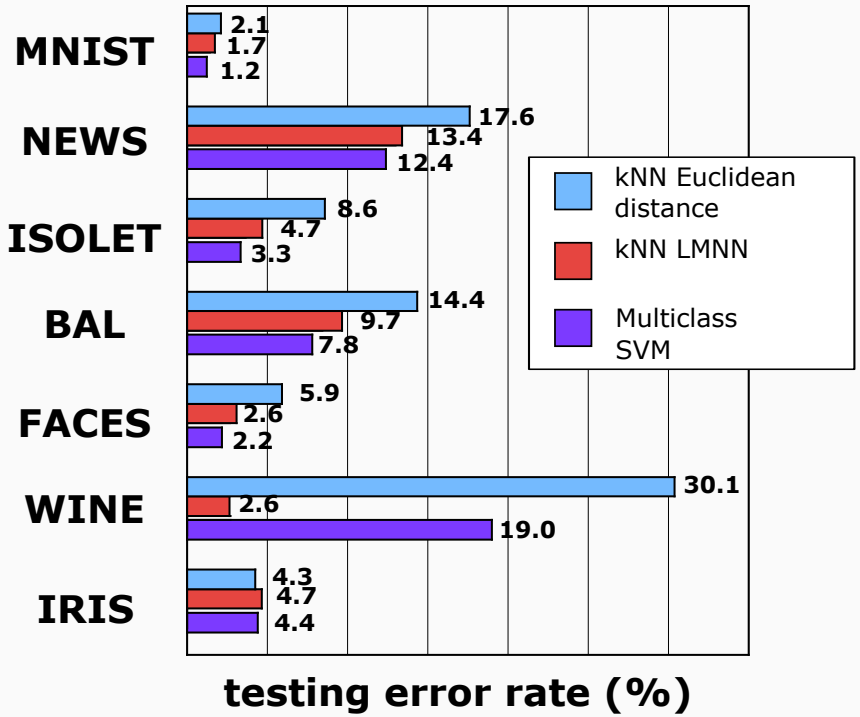
$$\text{s.t.} \quad D_M^2(x_i, x_k) - D_M^2(x_i, x_j) \geq 1 - \xi_{ijk} \quad \forall (x_i, x_j, x_k) \in \mathcal{R}$$

$\mu \in [0, 1]$ trade-off parameter

- Convex formulation, unlike NCA [Goldberger et al., 2004]
- Number of constraints in the order of kn^2
  - Solver based on projected gradient descent with working set
  - Simple alternative: only consider closest "impostors"
- Chicken and egg situation: which metric to build constraints?

Large Margin Nearest Neighbor [Weinberger et al., 2005]





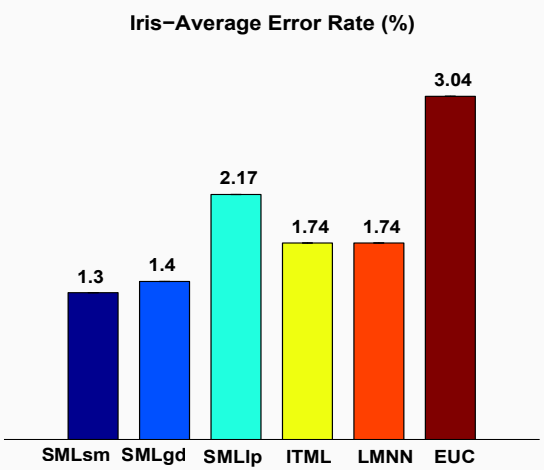
Pointers to metric learning algorithms for other tasks

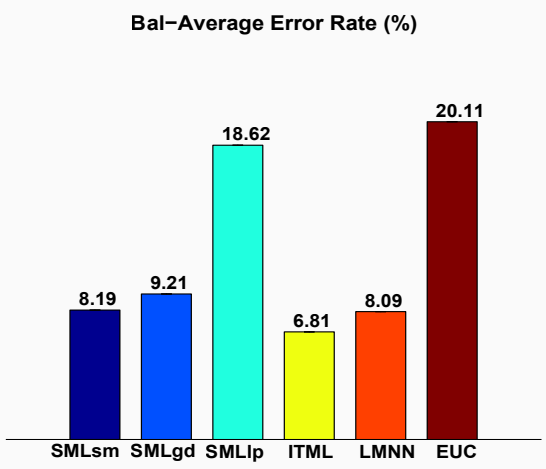
- Learning to rank [McFee and Lanckriet, 2010]
- Multi-task learning [Parameswaran and Weinberger, 2010]
- Transfer learning [Zhang and Yeung, 2010]
- Semi-supervised learning [Hoi et al., 2008]

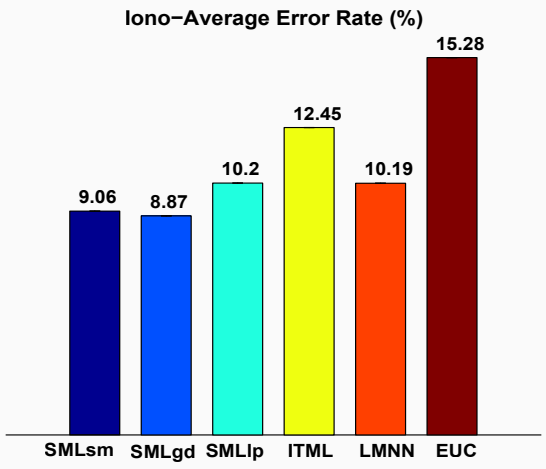
Interesting regularizers

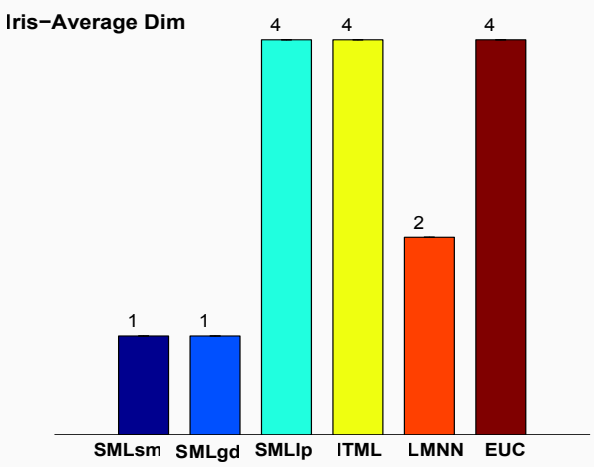
- Add regularization term to prevent overfitting
- We have already seen the Frobenius norm $\|\boldsymbol{M}\|_{\mathcal{F}}^2 = \sum_{i,j=1}^{d} M_{ij}^2$
  - Convex, smooth $\rightarrow$ easy to optimize
- Mixed $L_{2,1}$ norm: $\|\boldsymbol{M}\|_{2,1} = \sum_{i=1}^{d} \|\boldsymbol{M}_i\|_2$
  - Tends to zero-out entire columns $\rightarrow$ feature selection
  - Convex but nonsmooth
  - Efficient proximal gradient algorithms
- Trace (or nuclear) norm: $\|\boldsymbol{M}\|_* = \sum_{i=1}^{d} \sigma_i(\boldsymbol{M})$
  - Favors low-rank matrices $\rightarrow$ dimensionality reduction
  - Convex but nonsmooth
  - Efficient Frank-Wolfe algorithms

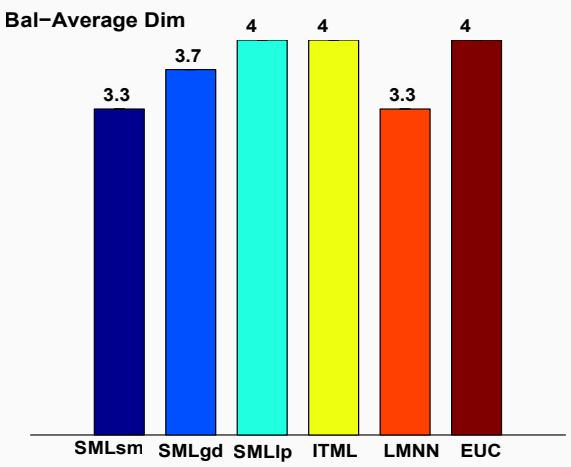
$L_{2,1}$ norm illustration

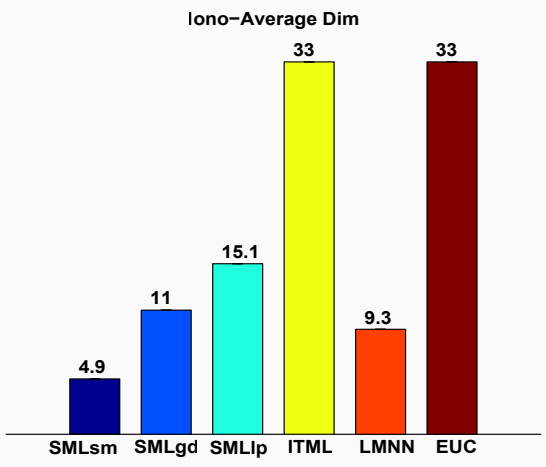








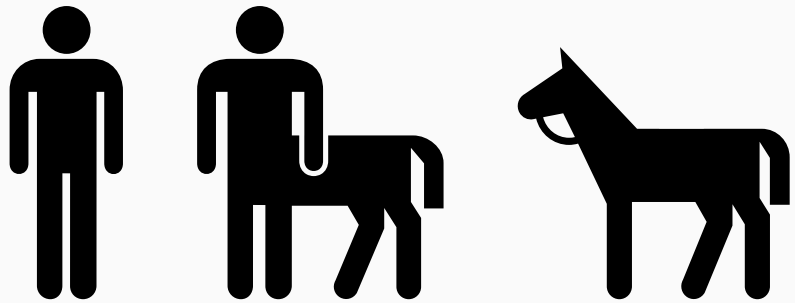




(image taken from [Ying et al., 2009])

LINEAR SIMILARITY LEARNING

- Mahalanobis distance satisfies some distance properties
  - Nonnegativity, symmetry, triangle inequality
  - Natural regularization, required by some applications
- In practice, these properties may not be satisfied
  - By human similarity judgments



  - By some good visual recognition systems
- Alternative: learn bilinear similarity function $S_M(x, x') = x^T M x'$
  - Example: OASIS algorithm (presented later)
  - No PSD constraint on $M \rightarrow$ computationally easier

NONLINEAR EXTENSIONS

Definition (Kernel function)

A symmetric function K is a kernel if there exists a 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 $\mathbb{H}$:

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

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 for metric learning

- Notations
  - Kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, training data $\{\boldsymbol{x}_i\}_{i=1}^n$
  - $\boldsymbol{\phi}_i \stackrel{def}{=} \phi(\boldsymbol{x}_i) \in \mathbb{R}^D$, $\boldsymbol{\Phi} \stackrel{def}{=} [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{n \times D}$

- Mahalanobis distance in kernel space

$$D_M^2(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) = (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^T \boldsymbol{M} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j) = (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^T \boldsymbol{L}^T \boldsymbol{L} (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)$$

- Setting $\boldsymbol{L}^T = \boldsymbol{\Phi} \boldsymbol{U}^T$, where $\boldsymbol{U} \in \mathbb{R}^{D \times n}$, we get

$$D_M^2(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')) = (\boldsymbol{k} - \boldsymbol{k}')^T \boldsymbol{M} (\boldsymbol{k} - \boldsymbol{k}')$$

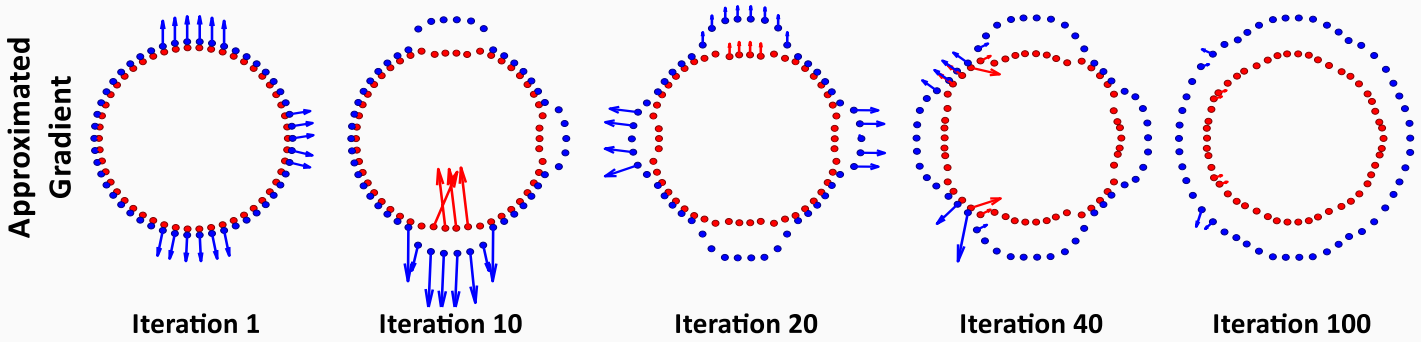
  - $\boldsymbol{M} = \boldsymbol{U}^T \boldsymbol{U} \in \mathbb{R}^{n \times n}$, $\boldsymbol{k} = \boldsymbol{\Phi}^T \phi(\boldsymbol{x}) = [K(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x})]^T$

- Theoretically justified (representer theorem)

Kernel trick for metric learning

- Similar trick as kernel SVM
  - Use a nonlinear kernel (e.g., Gaussian RBF)
  - Inexpensive computations through the kernel
  - Nonlinear metric learning while retaining convexity
- Need to learn $O(n^2)$ parameters
- Linear metric learning algorithm must be kernelized
  - Interface to data limited to inner products only
  - Several algorithms shown to be kernelizable
- General trick (simple and works well in practice):
  1. Kernel PCA: nonlinear mapping to low-dimensional space
  2. Apply linear metric learning algorithm to transformed data

LEARNING A NONLINEAR METRIC



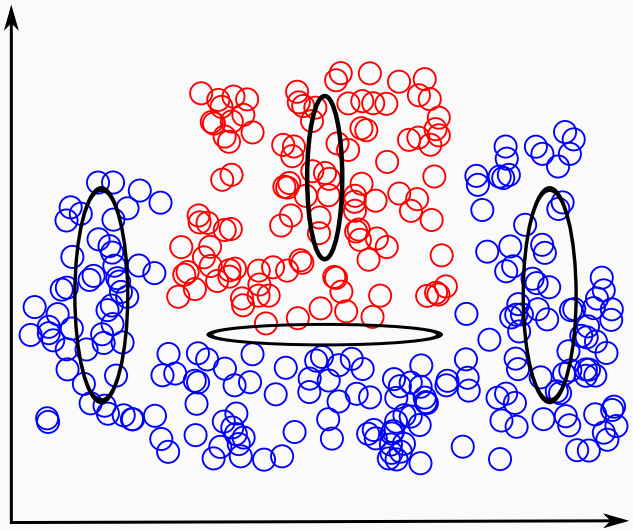
- More flexible approach: learn nonlinear mapping ϕ to optimize

$$D_\phi(x, x') = \|\phi(x) - \phi(x')\|_2$$

- Possible parameterizations for ϕ:
  - Gradient boosted regression trees [Kedem et al., 2012]
  - Deep neural nets [Hu et al., 2014, Wang et al., 2014, Song et al., 2016]
  - ...
- Typically nonconvex formulations

LEARNING MULTIPLE LOCAL METRICS

- Simple linear metrics perform well locally
- Idea: different metrics for different parts of the space



Multiple Metric LMNN [Weinberger and Saul, 2009]

- Group data into C clusters
- Learn a metric for each cluster in a coupled fashion

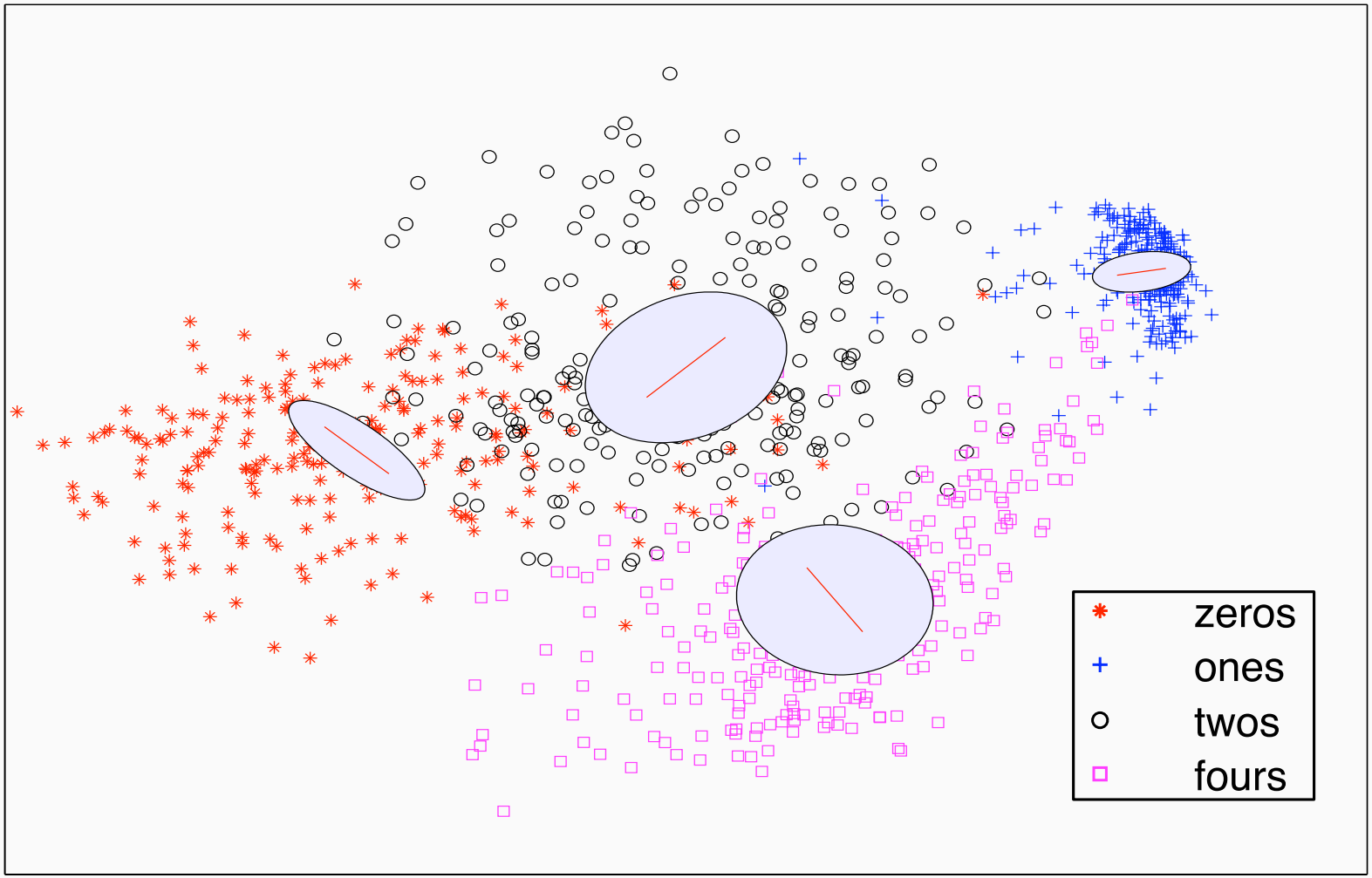
Formulation

$$\min_{\substack{M_1,\dots,M_C \\ \boldsymbol{\xi}\geq 0}} \quad (1-\mu)\sum_{(x_i,x_j)\in\mathcal{S}} D^2_{M_{C(x_j)}}(x_i,x_j) \quad + \quad \mu\sum_{i,j,k}\xi_{ijk}$$

$$\text{s.t.} \quad D^2_{M_{C(x_k)}}(x_i,x_k) - D^2_{M_{C(x_j)}}(x_i,x_j) \geq 1-\xi_{ijk} \quad \forall(x_i,x_j,x_k)\in\mathcal{R}$$

- Remains convex
- Computationally more expensive than standard LMNN
- Subject to overfitting (many parameters)

Multiple Metric LMNN [Weinberger and Saul, 2009]



LARGE-SCALE METRIC LEARNING

MAIN CHALLENGES

- How to deal with large datasets?
  - Number of similarity judgments can grow as $O(n^2)$ or $O(n^3)$
- How to deal with high-dimensional data?
  - Cannot store $d \times d$ matrix
  - Cannot afford computational complexity in $O(d^2)$ or $O(d^3)$

Online learning

- Online algorithm
  - Receive *one* similarity judgment
  - Suffer loss based on current metric
  - Update metric and iterate

- Goal: minimize regret

$$\sum_{t=1}^{T} \ell_t(\boldsymbol{M}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{M}^*) \leq f(T),$$

  - ℓ_t: loss suffered at time t
  - $\boldsymbol{M}_t$: metric learned at time t
  - $\boldsymbol{M}^*$: best metric in hindsight

OASIS [Chechik et al., 2010]

Formulation

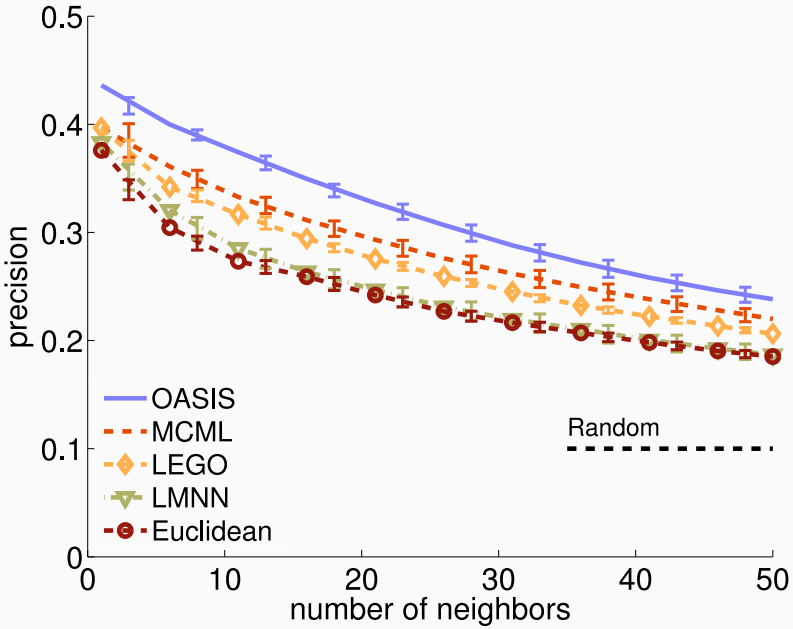
- Set $M^0 = I$
- At step t, receive $(x_i, x_j, x_k) \in \mathcal{R}$ and update by solving

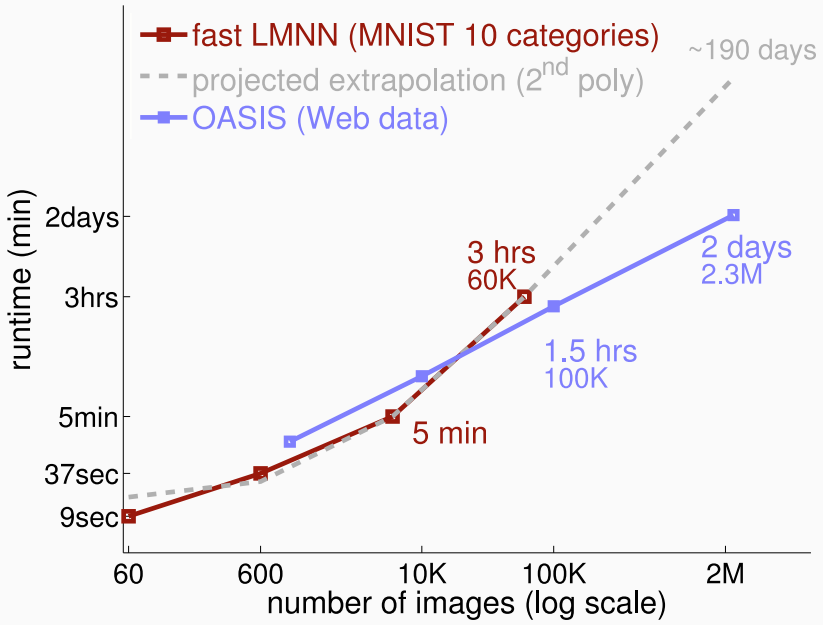
$$\begin{aligned} M^t = \quad \arg\min_{M,\xi} \quad & \frac{1}{2}\|M - M^{t-1}\|_{\mathcal{F}}^2 + C\xi \\ \text{s.t.} \quad & 1 - S_M(x_i, x_j) + S_M(x_i, x_k) \leq \xi \\ & \xi \geq 0 \end{aligned}$$

  - $S_M(x, x') = x^T M x'$, C trade-off parameter

- Simple closed-form solution at each iteration
- Trained with 160M triplets in 3 days on 1 CPU

OASIS [Chechik et al., 2010]





Stochastic and distributed optimization

- Assume metric learning problem of the form

$$\min_{M} \quad \frac{1}{|\mathcal{R}|} \sum_{(x_i,x_j,x_k)\in\mathcal{R}} \ell(M, x_i, x_j, x_k)$$

- Can use Stochastic Gradient Descent
  - Use a random sample (mini-batch) to estimate gradient
  - Better than full gradient descent when n is large
  - **We will implement this in the practical session**

- Can be combined with distributed optimization
  - Distribute triplets on workers
  - Each worker use a mini-batch to estimate gradient
  - Coordinator averages estimates and updates

Simple workarounds

- Learn a diagonal matrix
  - Learn d parameters
  - Only a weighting of features!
- Learn metric after dimensionality reduction (e.g., PCA)
  - Used in many papers
  - Potential loss of information
  - Learned metric difficult to interpret

Matrix decompositions

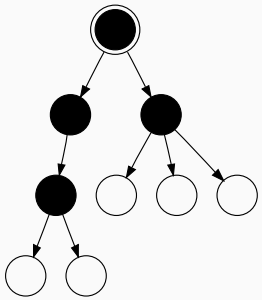
- Low-rank decomposition $M = L^T L$ with $L \in \mathbb{R}^{r \times d}$
  - Learn $r \times d$ parameters
  - Generally nonconvex, must tune r
- Rank-1 decomposition $M = \sum_{i=1}^{K} w_k b_k b_k^T$
  - Learn K parameters
  - Must choose good basis set
- Special case: sparse data [Liu et al., 2015]
  - Decomposition as rank-1 4-sparse matrices
  - Greedy algorithm incorporating a single basis at each iteration
  - Computational cost independent of d

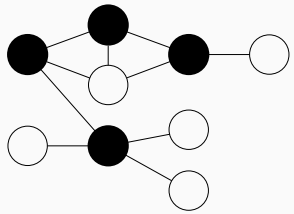
METRIC LEARNING FOR STRUCTURED DATA

MOTIVATION

- Each data instance is a structured object
  - Strings: words, DNA sequences
  - Trees: XML documents
  - Graphs: social network, molecules

ACGGCTT





- Metrics on structured data are convenient
  - Act as proxy to manipulate complex objects
  - Can use any metric-based algorithm

MOTIVATION

- Could represent each object by a feature vector
  - Idea behind many kernels for structured data
  - Could then apply standard metric learning techniques
  - Potential loss of structural information
- Instead, focus on edit distances
  - Directly operate on structured object
  - Variants for strings, trees, graphs
  - Natural parameterization by cost matrix

STRING EDIT DISTANCE

- Notations
  - Alphabet Σ: finite set of symbols
  - String x: finite sequence of symbols from Σ
  - $|x|$: length of string x
  - \$: empty string / symbol

Definition (Levenshtein distance)

The Levenshtein string edit distance between x and x′ is the length of the shortest sequence of operations (called an *edit script*) turning x into x′. Possible operations are insertion, deletion and substitution of symbols.

- Computed in $O(|x| \cdot |x'|)$ time by Dynamic Programming (DP)

Parameterized version

- Use a nonnegative $(|\Sigma| + 1) \times (|\Sigma| + 1)$ matrix C
  - C_{ij}: cost of substituting symbol i with symbol j

Example 1: 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)

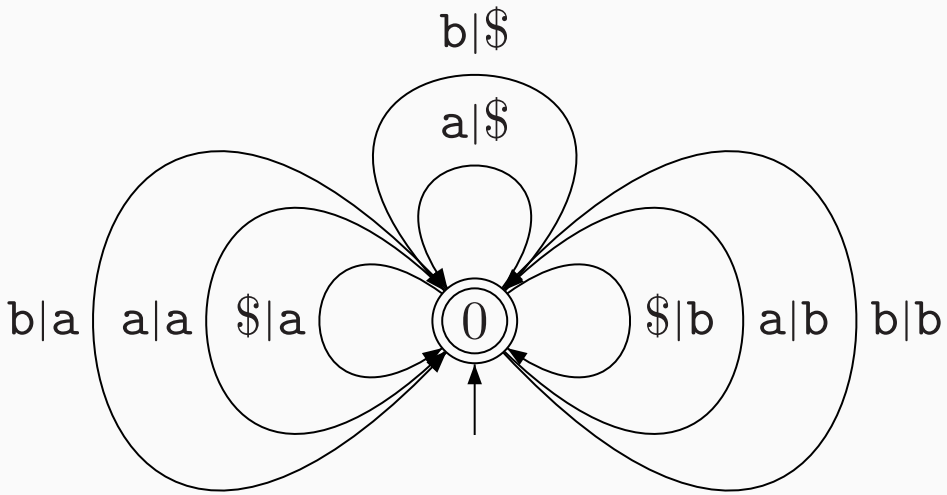
Example 2: specific costs

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

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

EDIT PROBABILITY LEARNING

- Interdependence issue
  - The optimal edit script depends on the costs
  - Updating the costs may change the optimal edit script
- Consider edit probability $p(x'|x)$ [Oncina and Sebban, 2006]
  - Cost matrix: probability distribution over operations
  - Corresponds to summing over all possible scripts
- Represent process by a stochastic memoryless transducer
- Maximize expected log-likelihood of positive pairs

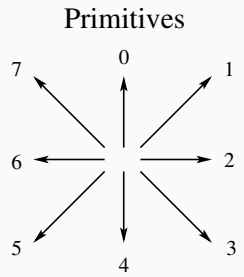


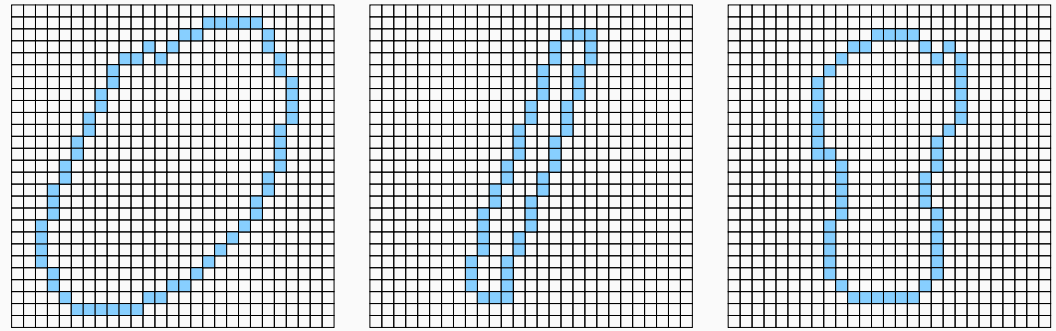
Iterative Expectation-Maximization algorithm [Oncina and Sebban, 2006]

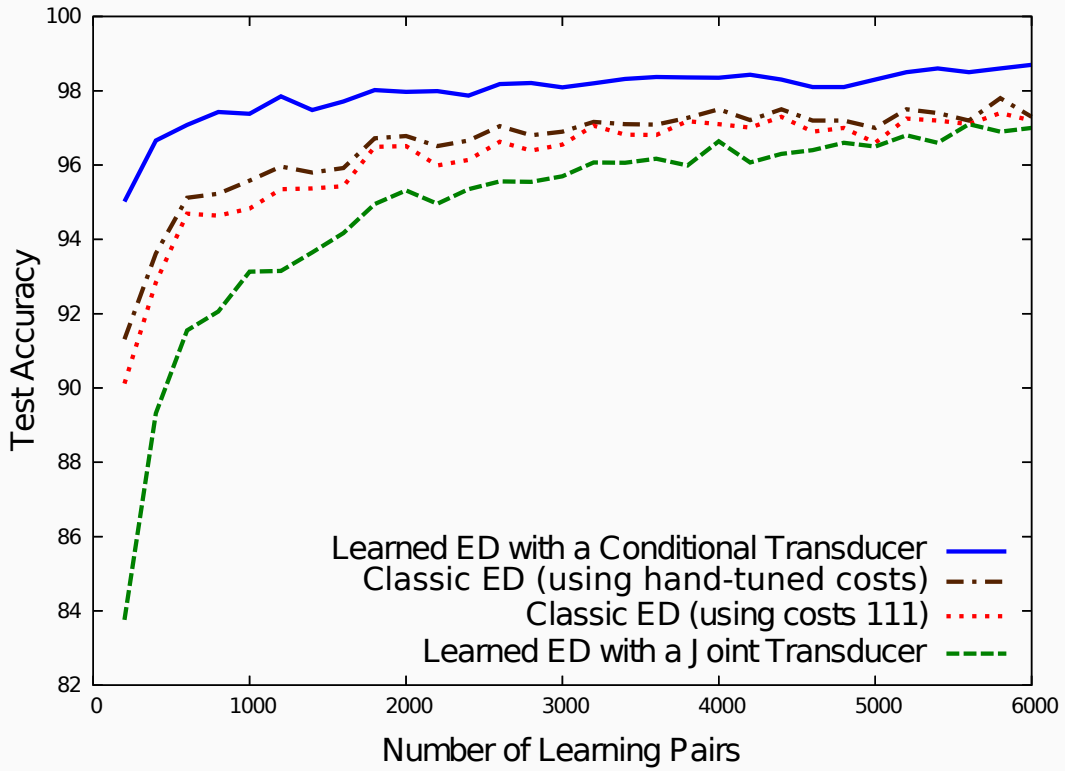
- Expectation step
  - Given edit probabilities, compute frequency of each operation
  - Probabilistic version of the DP algorithm
- Maximization step
  - Given frequencies, update edit probabilities
  - Done by likelihood maximization under constraints

$$\forall u \in \Sigma, \sum_{v \in \Sigma \cup \{\$\}} C_{v|u} + \sum_{v \in \Sigma} C_{v|\$} = 1, \quad \text{with} \sum_{v \in \Sigma} C_{v|\$} + \underbrace{c(\#)}_{\text{exit prob.}} = 1,$$

Application to handwritten digit recognition [Oncina and Sebban, 2006]







Some remarks

- Advantages
  - Elegant probabilistic framework
  - Enables data generation
  - Generalization to trees [Bernard et al., 2008]
- Drawbacks
  - Convergence to local minimum
  - Costly: DP algorithm for each pair at each iteration
  - Cannot use negative pairs

GESL [Bellet et al., 2012]

- Inspired from successful algorithms for non-structured data
  - Large-margin constraints
  - Convex optimization
- Requires key simplification: fix the edit script

$$e_C(x, x') = \sum_{u,v \in \Sigma \cup \{\$\}} C_{uv} \cdot \#_{uv}(x, x')$$

  - $\#_{uv}(x, x')$: nb of times $u \to v$ appears in Levenshtein script
- e_C is a linear function of the costs

GESL [Bellet et al., 2012]

Formulation

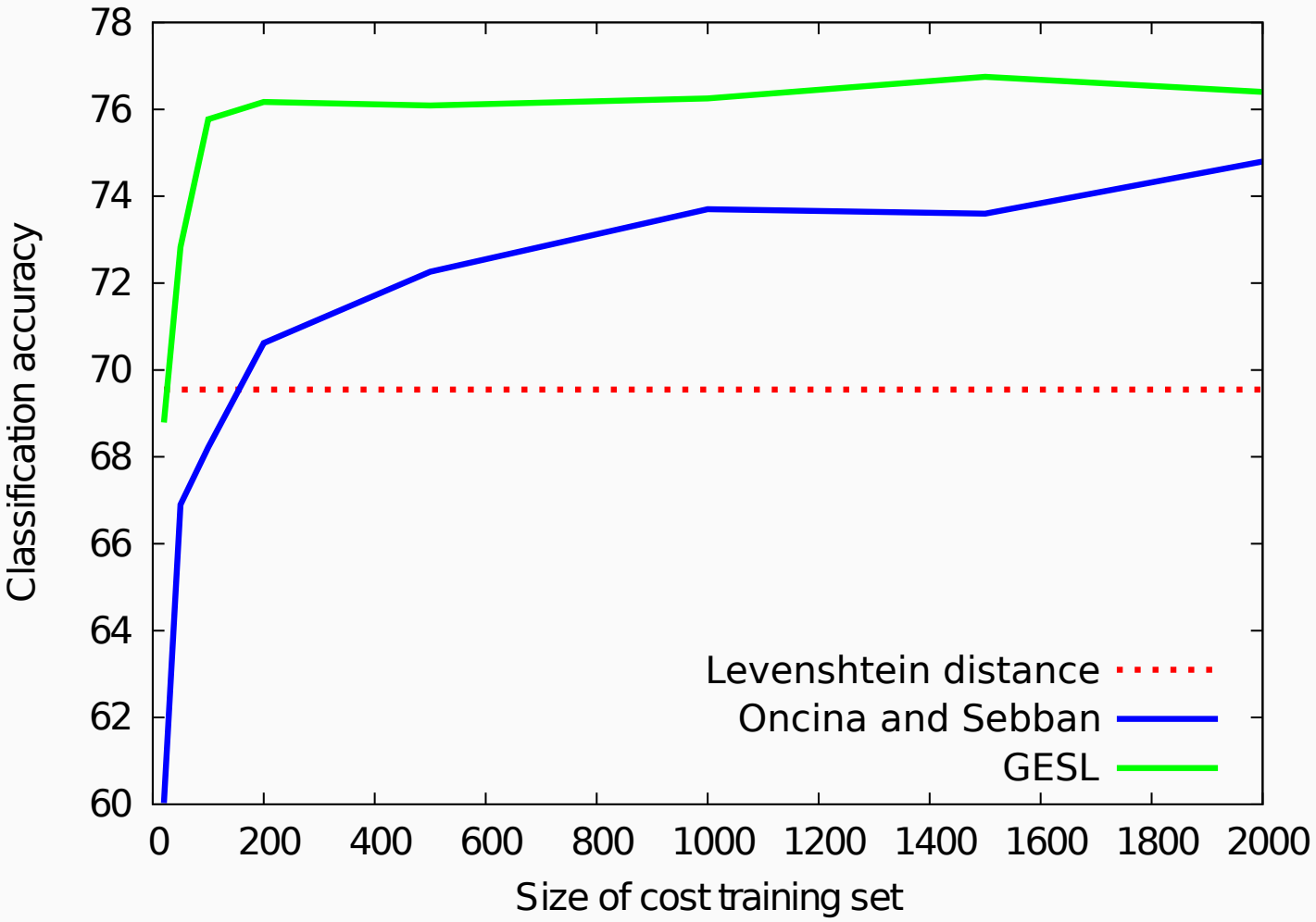
$$\min_{C\geq 0,\boldsymbol{\xi}\geq 0,B_1\geq 0,B_2\geq 0} \quad \sum_{i,j}\xi_{ij} \quad + \quad \lambda\|C\|_{\mathcal{F}}^2$$

$$\begin{aligned} \text{s.t.} \quad & e_C(x, x') \geq B_1 - \xi_{ij} && \forall (x_i, x_j) \in \mathcal{D} \\ & e_C(x, x') \leq B_2 + \xi_{ij} && \forall (x_i, x_j) \in \mathcal{S} \\ & B_1 - B_2 = \gamma \end{aligned}$$

γ margin parameter

- Convex, less costly and use of negative pairs
- Straightforward adaptation to trees and graphs
- Less general than proper edit distance
  - Chicken and egg situation similar to LMNN

Application to word classification [Bellet et al., 2012]



CONCLUSION

- Distance / similarity: key component of machine learning
- Metric learning often requires only weak supervision
- Many algorithms:
  - For classification, clustering, ranking...
  - Linear, nonlinear, local metrics
  - Scalable methods
- Very successful in practical applications
- More details: can refer to [Bellet et al., 2015]

REFERENCES I

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

[Bellet et al., 2015] Bellet, A., Habrard, A., and Sebban, M. (2015).
***Metric Learning*.**
Morgan & Claypool Publishers.

[Bernard et al., 2008] Bernard, M., Boyer, L., Habrard, A., and Sebban, M. (2008).
Learning probabilistic models of tree edit distance.
Pattern Recognition, 41(8):2611–2629.

[Chechik et al., 2010] Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010).
Large Scale Online Learning of Image Similarity Through Ranking.
Journal of Machine Learning Research, 11:1109–1135.

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

[Goldstone et al., 1997] Goldstone, R. L., Medin, D. L., and Halberstadt, J. (1997).
Similarity in context.
Memory & Cognition, 25(2):237–255.

References II

[Guillaumin et al., 2009] Guillaumin, M., Verbeek, J. J., and Schmid, C. (2009).
Is that you? Metric learning approaches for face identification.
In *ICCV*.

[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*.

[Hu et al., 2014] Hu, J., Lu, J., and Tan, Y.-P. (2014).
Discriminative Deep Metric Learning for Face Verification in the Wild.
In *CVPR*.

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

[Liu et al., 2015] Liu, K., Bellet, A., and Sha, F. (2015).
Similarity Learning for High-Dimensional Sparse Data.
In *AISTATS*.

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

References III

[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*.

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

[Song et al., 2016] Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. (2016).
Deep Metric Learning via Lifted Structured Feature Embedding.
In *CVPR*.

[Tversky, 1977] Tversky, A. (1977).
Features of similarity.
Psychological Review, 84(4):327–352.

[van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008).
Visualizing Data using t-SNE.
Journal of Machine Learning Research, 9:2579–2605.

References IV

[Wang et al., 2014] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014).
Learning Fine-Grained Image Similarity with Deep Ranking.
In *CVPR*.

[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*.

[Weinberger and Saul, 2009] Weinberger, K. Q. and Saul, L. K. (2009).
Distance Metric Learning for Large Margin Nearest Neighbor Classification.
Journal of Machine Learning Research, 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*.

[Ying et al., 2009] Ying, Y., Huang, K., and Campbell, C. (2009).
Sparse Metric Learning via Smooth Optimization.
In *NIPS*.

[Zhang and Yeung, 2010] Zhang, Y. and Yeung, D.-Y. (2010).
Transfer metric learning by learning task relationships.
In *KDD*.

References V

[Zhu et al., 2015] Zhu, X., Lei, Z., Yan, J., Yi, D., and Li, S. Z. (2015).
High-fidelity pose and expression normalization for face recognition in the wild.
In *CVPR*.