Large-Scale Similarity and Distance Metric Learning

Aurélien Bellet
Télécom ParisTech

Joint work with K. Liu, Y. Shi and F. Sha (USC), S. Clémençon and I. Colin (Télécom ParisTech)

Séminaire Criteo
March 31, 2015

A bit about me

- PhD in Computer Science (Dec 2012)
  - Université Jean Monnet, Saint-Etienne
  - Advisors: Marc Sebban, Amaury Habrard
- Postdoc in 2013–2014 (18 months)
  - University of Southern California, Los Angeles
  - Working with Fei Sha
- Joined Télécom ParisTech in October
  - Chaire "Machine Learning for Big Data"
  - Working with Stéphan Clémençon

Outline of the talk

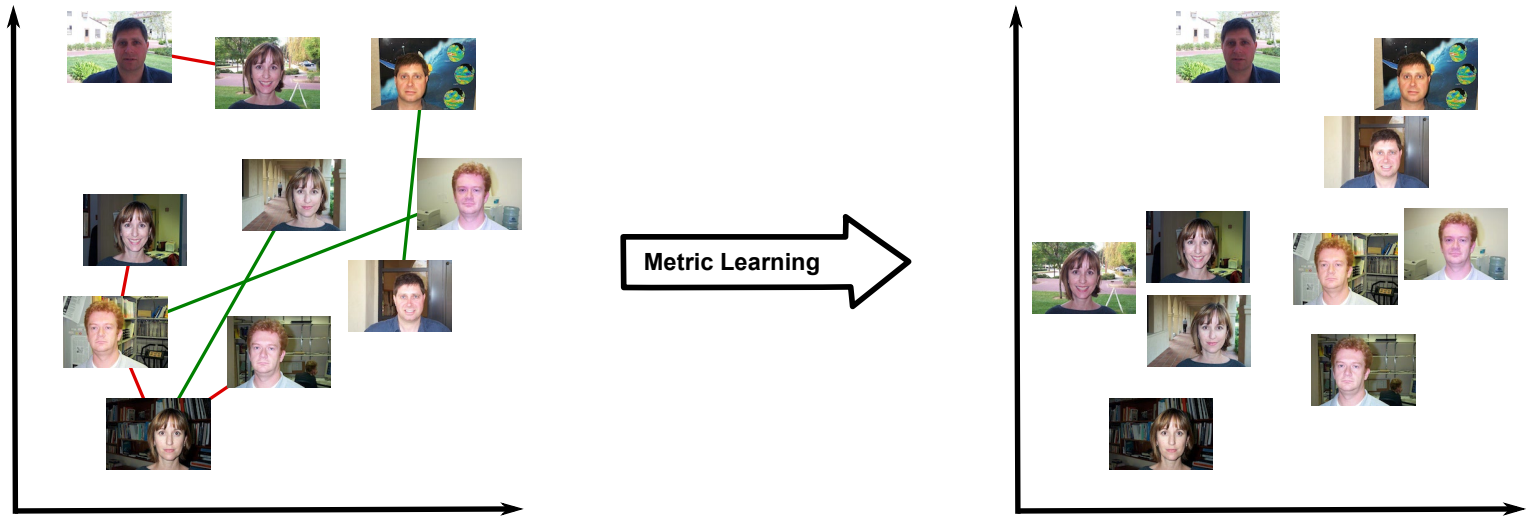
1. Introduction to Metric Learning
2. Learning (Infinitely) Many Local Metrics
3. Similarity Learning for High-Dimensional Sparse Data
4. Scaling-up ERM for Metric Learning

Introduction

Metric learning

Motivation

- How to assign a similarity score to a pair of objects?
  - Basic component of many algorithms: k-NN, clustering, kernel methods, ranking, dimensionality reduction, data visualization
  - Crucial to performance of the above
  - Obviously problem-dependent
- The machine learning way: let's learn it from data!
  - We'll need some examples of similar / dissimilar things
  - Learn to approximate the latent notion of similarity
  - Can be thought of as representation learning



Metric learning

Basic recipe

1. Pick a parametric form of distance or similarity function
   - (Squared) Mahalanobis distance

     $d^2_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}-\boldsymbol{x}')^{\mathrm{T}} \boldsymbol{M} (\boldsymbol{x}-\boldsymbol{x}')$ with $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ symmetric PSD

   - Bilinear similarity

     $$S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}' \text{ with } \boldsymbol{M} \in \mathbb{R}^{d \times d}$$

2. Collect similarity judgments on data pairs/triplets
   - $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar (or dissimilar)
   - $\boldsymbol{x}_i$ is more similar to $\boldsymbol{x}_j$ than to $\boldsymbol{x}_k$

3. Estimate parameters such that metric best satisfies them
   - Convex optimization and the like

Metric learning

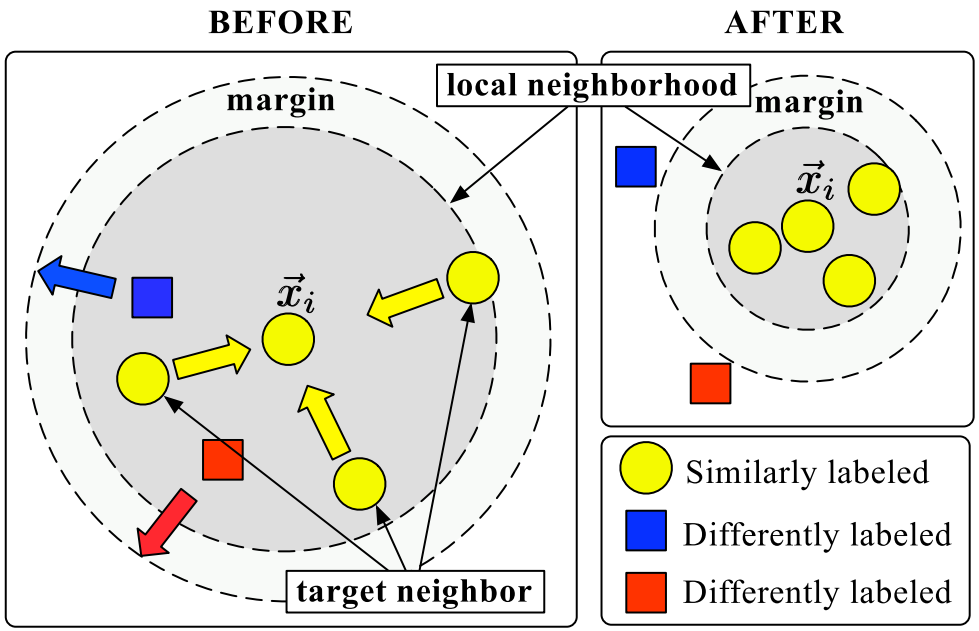
A famous algorithm: LMNN [Weinberger et al., 2005]

Large Margin Nearest Neighbors

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

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

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



Metric learning

Large-scale challenges

- How to efficiently learn multiple metrics?
  - Multi-task learning
  - Local linear metrics
- How to efficiently learn a metric for high-dimensional data?
  - Full $d \times d$ matrix not feasible anymore
- How to deal with large datasets?
  - Number of terms in the objective can grow as $O(n^2)$ or $O(n^3)$

Learning (Infinitely) Many Local Metrics

[Shi et al., 2014]

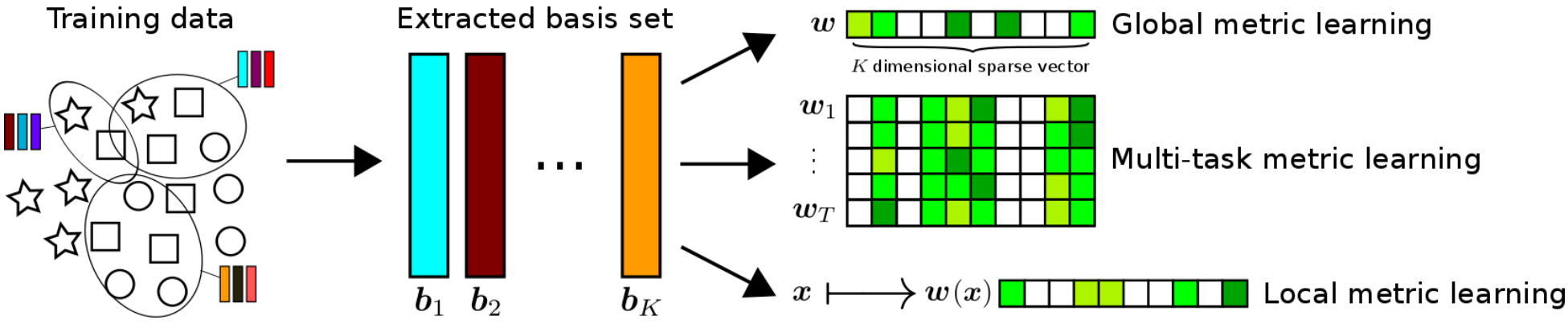
Learning (infinitely) many local metrics

Main idea

- Assume we are given a basis dictionary $B = \{\boldsymbol{b}_k\}_{k=1}^K$ in $\mathbb{R}^d$
- We will learn $d_{\boldsymbol{w}}^2(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{x}')$ where

$$\boldsymbol{M} = \sum_{k=1}^{K} w_k \boldsymbol{b}_k \boldsymbol{b}_k^{\mathrm{T}}, \quad \boldsymbol{w} \geq 0$$

- Use sparsity-inducing regularizer to do basis selection



Learning (infinitely) many local metrics

Global and multi-task convex formulations

$$\mathcal{C} = \left\{ \boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^- : d^2_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_i^+) \text{ should be smaller than } d^2_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_i^-) \right\}_{i=1}^{C}$$

SCML-Global

$$\min_{\boldsymbol{w} \in \mathbb{R}^K_+} \quad \frac{1}{C} \sum_{i=1}^{C} \left[1 + d^2_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_i^+) - d^2_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_i^-) \right]_+ \quad + \quad \beta \|\boldsymbol{w}\|_1,$$

where $[\cdot]_+ = \max(0, \cdot)$

mt-SCML : extension to T tasks

$$\min_{\boldsymbol{W} \in \mathbb{R}^{T \times K}_+} \quad \sum_{t=1}^{T} \frac{1}{C_t} \sum_{i=1}^{C_t} \left[1 + d^2_{\boldsymbol{w}_t}(\boldsymbol{x}_i, \boldsymbol{x}_i^+) - d^2_{\boldsymbol{w}_t}(\boldsymbol{x}_i, \boldsymbol{x}_i^-) \right]_+ + \beta \|\boldsymbol{W}\|_{2,1}$$

- $\|\boldsymbol{W}\|_{2,1}$: induce column-wise sparsity
- Metrics are task-specific but regularized to share features

Learning (infinitely) many local metrics

Local metric learning

- Learn multiple local metrics : more flexibility
  - One metric per region or class
  - One metric per training instance
- Many issues
  - How to define the local regions?
  - How to generalize at test time?
  - How to avoid a growing number of parameters?
  - How to do avoid overfitting?
- Proposed approach : learn a metric for each point in space

Learning (infinitely) many local metrics

Local nonconvex formulation

- We use the following parameterization :

$$d_w(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \left(\sum_{k=1}^{K} w_k(\boldsymbol{x}) \boldsymbol{b}_k \boldsymbol{b}_k^{\mathrm{T}} \right) (\boldsymbol{x} - \boldsymbol{x}'),$$

  - $w_k(\boldsymbol{x}) = (\boldsymbol{a}_k^{\mathrm{T}} \phi(\boldsymbol{x}) + c_k)^2$: weight of basis k
  - $\phi(\boldsymbol{x}) \in \mathbb{R}^{d'}$: (nonlinear) projection of $\boldsymbol{x}$ (e.g., kernel PCA)
  - $\boldsymbol{a}_k \in \mathbb{R}^{d'}$ and $c_k \in \mathbb{R}$: parameters to learn for each basis

SCML-Local

$$\min_{\tilde{\boldsymbol{A}} \in \mathbb{R}_+^{(d'+1) \times K}} \quad \frac{1}{C} \sum_{i=1}^{C} \left[1 + d_w^2(\boldsymbol{x}_i, \boldsymbol{x}_i^+) - d_w^2(\boldsymbol{x}_i, \boldsymbol{x}_i^-) \right]_+ + \beta \|\tilde{\boldsymbol{A}}\|_{2,1}$$

- $\tilde{\boldsymbol{A}}$: stack $\boldsymbol{A} = [\boldsymbol{a}_1 \dots \boldsymbol{a}_K]^{\mathrm{T}}$ and $\boldsymbol{c}$
- If $\boldsymbol{A} = \boldsymbol{0}$, recover the global formulation

Learning (infinitely) many local metrics

Efficient optimization

- Nonsmooth objective with large number of terms
  - Use stochastic proximal methods
- SCML-Global and mt-SCML are convex
  - Regularized Dual Averaging [Xiao, 2010]
- SCML-Local is nonconvex
  - Forward-backward algorithm [Duchi and Singer, 2009]
  - Initialize with solution of SCML-Global

Learning (infinitely) many local metrics

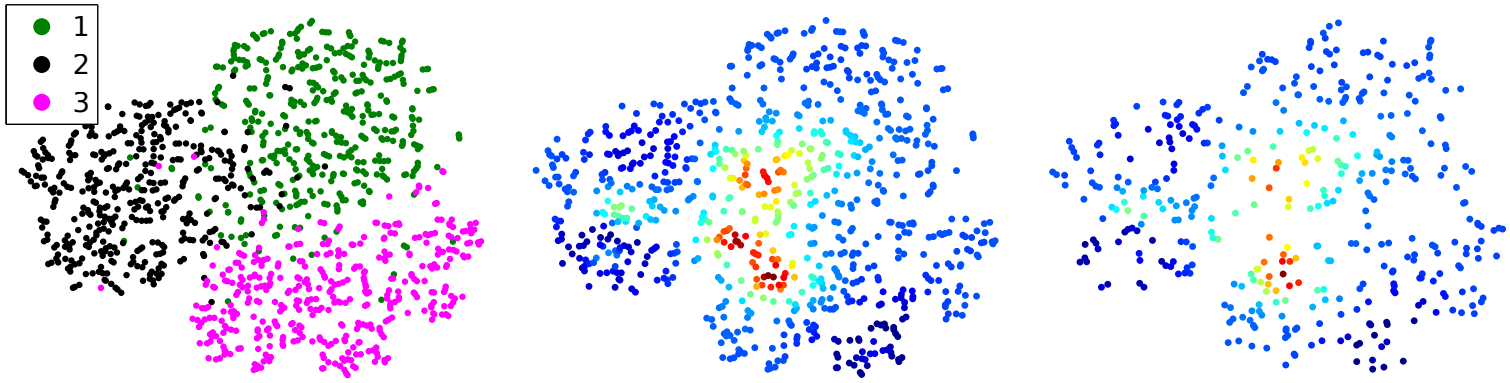
Experiments

Dataset	Euc	LMNN	BoostML	SCML-Global
Vehicle	29.7±0.6	23.5±0.7	**19.9±0.6**	21.3±0.6
Vowel	**11.1±0.4**	**10.8±0.4**	11.4±0.4	**10.9±0.5**
Segment	5.2±0.2	4.6±0.2	**3.8±0.2**	**4.1±0.2**
Letters	14.0±0.2	11.6±0.3	10.8±0.2	**9.0±0.2**
USPS	10.3±0.2	**4.1±0.1**	7.1±0.2	**4.1±0.1**
BBC	8.8±0.3	**4.0±0.2**	9.3±0.3	**3.9±0.2**
Avg. rank	3.3	2.0	2.3	**1.2**

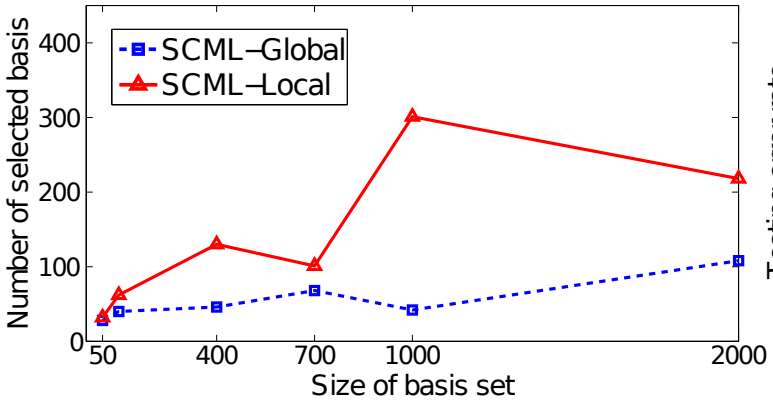
Dataset	M^2LMNN	GLML	PLML	SCML-Local
Vehicle	23.1±0.6	23.4±0.6	22.8±0.7	**18.0±0.6**
Vowel	6.8±0.3	**4.1±0.4**	8.3±0.4	6.1±0.4
Segment	**3.6±0.2**	**3.9±0.2**	**3.9±0.2**	**3.6±0.2**
Letters	9.4±0.3	10.3±0.3	**8.3±0.2**	**8.3±0.2**
USPS	**4.2±0.7**	7.8±0.2	4.1±0.1	**3.6±0.1**
BBC	4.9±0.4	5.7±0.3	**4.3±0.2**	**4.1±0.2**
Avg. rank	2.0	2.7	2.0	**1.2**

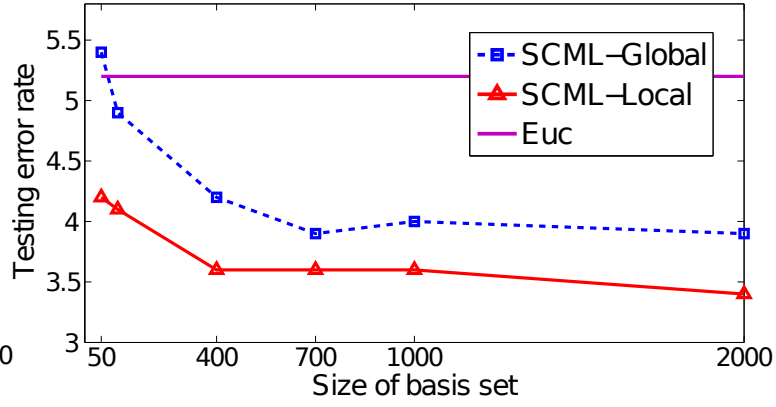
Learning (infinitely) many local metrics

Experiments



(a) Class membership (b) Trained metrics (c) Test metrics





Similarity Learning for High-Dimensional Sparse Data

[Liu et al., 2015]

Similarity learning for high-dimensional sparse data

Problem setting

- Assume data points are high-dimensional ($d > 10^4$) but D-sparse (on average) with $D \ll d$
  - Bags-of-words (text, image), bioinformatics, etc
- Existing metric learning algorithms fail
  - Intractable: training cost $O(d^2)$ to $O(d^3)$, memory $O(d^2)$
  - Severe overfitting
- Practitioners use dimensionality reduction (PCA, RP)
  - Poor performance in presence of noisy features
  - Resulting metric difficult to interpret for domain experts
- Contributions of this work
  - Learn similarity in original high-dimensional space
  - Time/memory costs independent of d
  - Explicit control of similarity complexity

Similarity learning for high-dimensional sparse data

A very large basis set

- We want to learn a similarity function $S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}'$
- Given $\lambda > 0$, for any $i, j \in \{1, \ldots, d\}$, $i \neq j$ we define

$$\boldsymbol{P}_{\lambda}^{(ij)} = \begin{pmatrix} \ddots & \vdots & \ddots & \vdots & \ddots \\ \cdots & \lambda & \cdots & \lambda & \cdots \\ \ddots & \vdots & \ddots & \vdots & \ddots \\ \cdots & \lambda & \cdots & \lambda & \cdots \\ \ddots & \vdots & \ddots & \vdots & \ddots \end{pmatrix} \quad \boldsymbol{N}_{\lambda}^{(ij)} = \begin{pmatrix} \ddots & \vdots & \ddots & \vdots & \ddots \\ \cdots & \lambda & \cdots & -\lambda & \cdots \\ \ddots & \vdots & \ddots & \vdots & \ddots \\ \cdots & -\lambda & \cdots & \lambda & \cdots \\ \ddots & \vdots & \ddots & \vdots & \ddots \end{pmatrix}$$

$$\mathcal{B}_{\lambda} = \bigcup_{ij} \left\{ \boldsymbol{P}_{\lambda}^{(ij)}, \boldsymbol{N}_{\lambda}^{(ij)} \right\}$$

$$\boldsymbol{M} \in \mathcal{D}_{\lambda} = \mathrm{conv}(\mathcal{B}_{\lambda})$$

- One basis involves only 2 features:

$$S_{\boldsymbol{P}_{\lambda}^{(ij)}}(\boldsymbol{x}, \boldsymbol{x}') = \lambda(x_i x_i' + x_j x_j' + x_i x_j' + x_j x_i')$$

$$S_{\boldsymbol{N}_{\lambda}^{(ij)}}(\boldsymbol{x}, \boldsymbol{x}') = \lambda(x_i x_i' + x_j x_j' - x_i x_j' - x_j x_i')$$

Similarity learning for high-dimensional sparse data

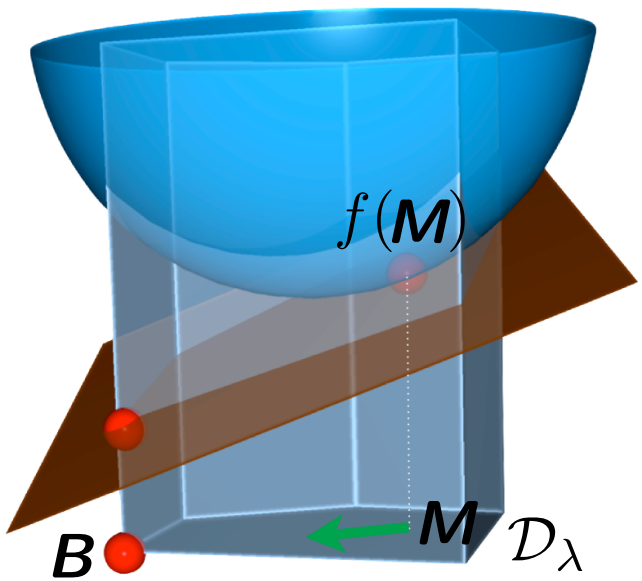
Problem formulation and algorithm

Optimization problem (smoothed hinge loss ℓ)

$$\min_{\boldsymbol{M} \in \mathbb{R}^{d \times d}} \quad f(\boldsymbol{M}) = \frac{1}{C} \sum_{i=1}^{C} \ell\left(1 - \boldsymbol{x}_i^T \boldsymbol{M} \boldsymbol{x}_i^+ + \boldsymbol{x}_i^T \boldsymbol{M} \boldsymbol{x}_i^-\right)$$

$$\text{s.t.} \quad \boldsymbol{M} \in \mathcal{D}_\lambda$$

- Use a Frank-Wolfe algorithm [Jaggi, 2013] to solve it



Let $\boldsymbol{M}^{(0)} \in \mathcal{D}_\lambda$

for $k = 0, 1, \ldots$ **do**

$\boldsymbol{B}^{(k)} = \arg\min_{\boldsymbol{B} \in \mathcal{B}_\lambda} \left\langle \boldsymbol{B}, \nabla f(\boldsymbol{M}^{(k)}) \right\rangle$

$\boldsymbol{M}^{(k+1)} = (1 - \gamma)\boldsymbol{M}^{(k)} + \gamma \boldsymbol{B}^{(k)}$

end for

Figure adapted from [Jaggi, 2013]

Similarity learning for high-dimensional sparse data

Convergence and complexity analysis

Convergence

Let $L = \frac{1}{C}\sum_{i=1}^{C} \|\boldsymbol{x}_i(\boldsymbol{x}_i^+ - \boldsymbol{x}_i^-)^T\|_F^2$. At any iteration $k \geq 1$, the iterate $\boldsymbol{M}^{(k)} \in \mathcal{D}_\lambda$ of the FW algorithm:

- has at most rank $k+1$ with $4(k+1)$ nonzero entries
- uses at most $2(k+1)$ distinct features
- satisfies $f(\boldsymbol{M}^{(k)}) - f(\boldsymbol{M}^*) \leq 16L\lambda^2/(k+2)$

- An optimal basis can be found in $O(CD^2)$ time and memory
- An approximately optimal basis can be found in $O(mD^2)$ with $m \ll C$ using a Monte Carlo approximation of the gradient
  - Or even $O(mD)$ using a heuristic (good results in practice)
- Storing $\boldsymbol{M}^{(k)}$ requires only $O(k)$ memory
  - Or even the entire sequence $\boldsymbol{M}^{(0)}, \ldots, \boldsymbol{M}^{(k)}$ at the same cost

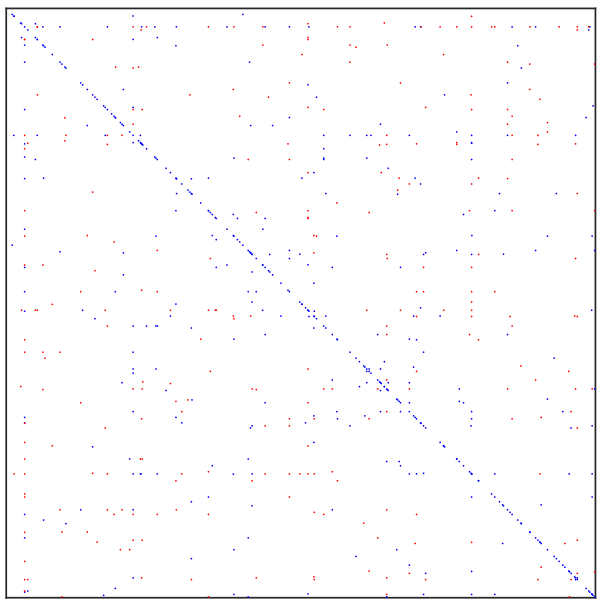
Similarity learning for high-dimensional sparse data

Experiments

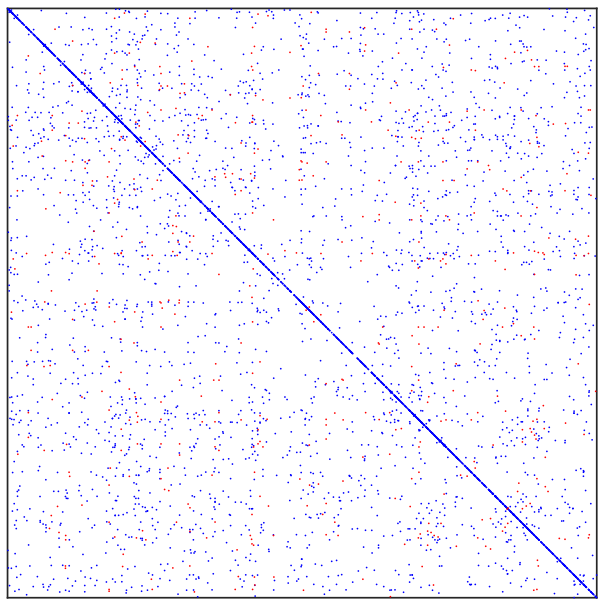
- K-NN test error on datasets with d up to 10^5

Datasets	IDENTITY	RP+OASIS	PCA+OASIS	DIAG-ℓ_2	DIAG-ℓ_1	HDSL
dexter	20.1	24.0	9.3	8.4	8.4	**6.5**
dorothea	9.3	11.4	9.9	6.8	6.6	**6.5**
rcv1_2	6.9	7.0	4.5	3.5	3.7	**3.4**
rcv1_4	11.2	10.6	6.1	6.2	7.2	**5.7**

- Sparsity structure of the matrices



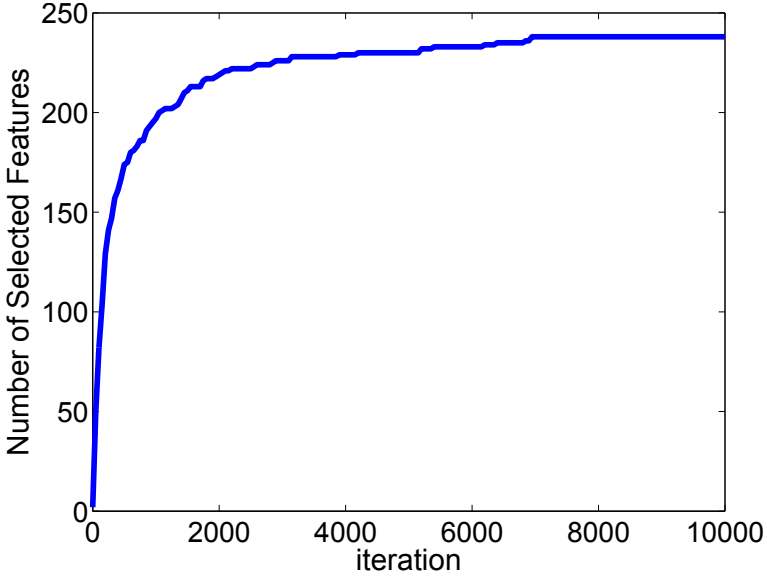
(a) dexter ($20,000 \times 20,000$ matrix, 712 nonzeros)



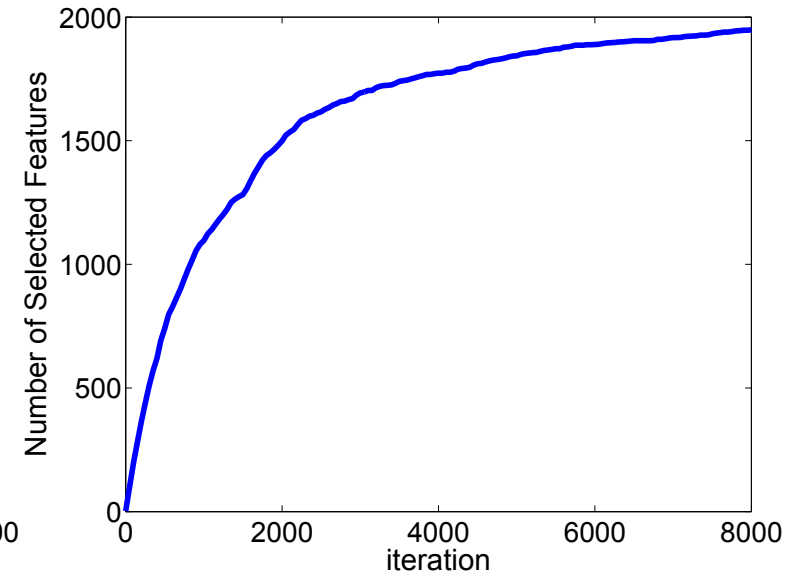
(b) rcv1_4 ($29,992 \times 29,992$ matrix, 5263 nonzeros)

Similarity learning for high-dimensional sparse data

Experiments



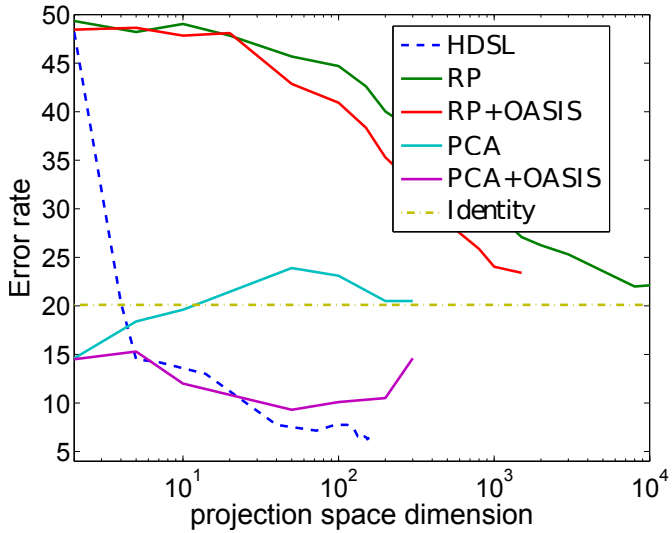
(a) dexter dataset



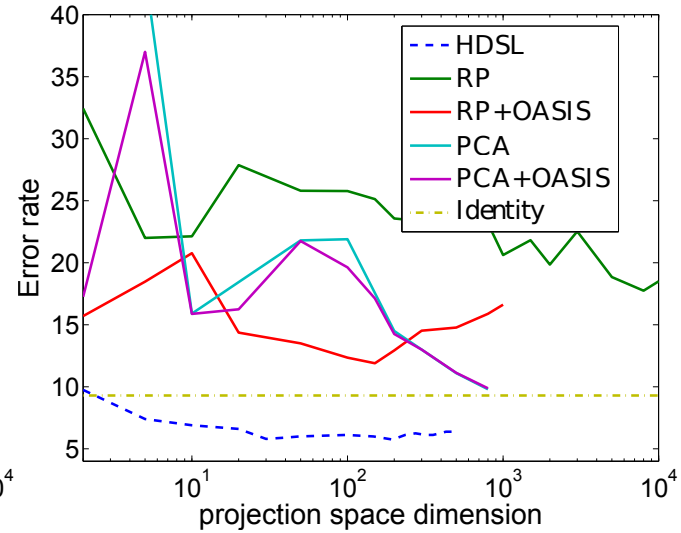
(b) rcv1_4 dataset

Similarity learning for high-dimensional sparse data

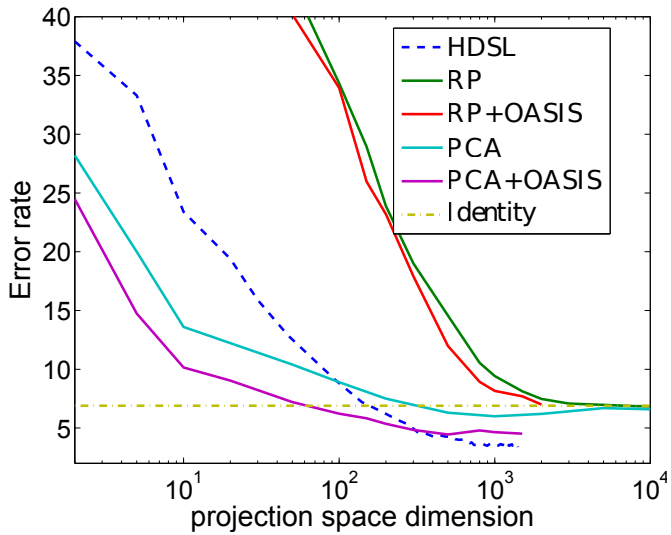
Experiments



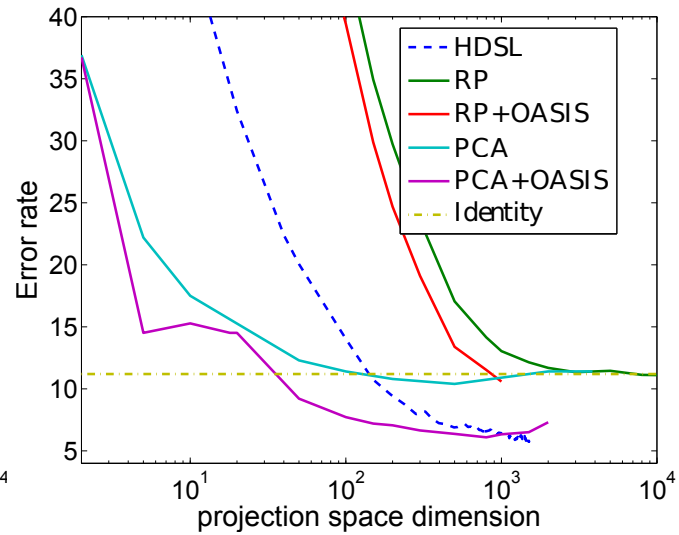
(a) dexter dataset



(b) dorothea dataset



(c) rcv1_2 dataset



(d) rcv1_4 dataset

Scaling-up ERM for Metric Learning

[Clémençon et al., 2015]

Scaling-up ERM for metric learning

Empirical Minimization of U-Statistics

- Given an i.i.d. sample $X_1, \ldots, X_n$, the U-statistic of degree 2 with kernel H is given by

$$U_n(H;\theta) = \frac{2}{n(n-1)} \sum_{i<j} H(X_i, X_j; \theta)$$

  - Can be generalized to higher degrees and multi-samples

- Empirical Minimization of U-statistic: $\min_{\theta \in \Theta} U_n(H;\theta)$
  - Applies to metric learning for classification
  - Also pairwise clustering, ranking, etc

- Number of terms quickly becomes huge
  - MNIST dataset : $n = 60000 \rightarrow 2 \times 10^9$ pairs

Scaling-up ERM for metric learning

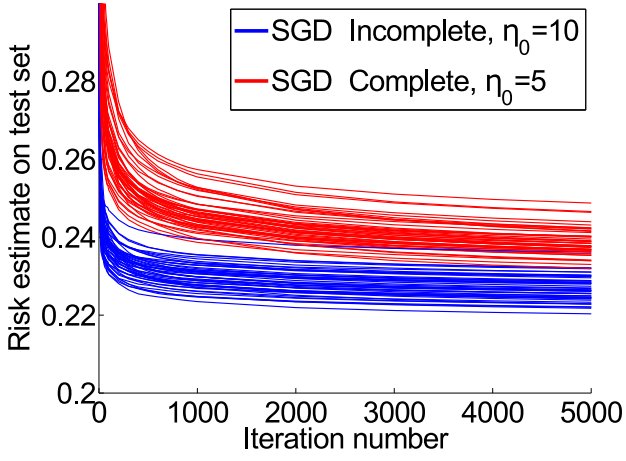
Main idea and results

- Approximate $U_n(H)$ by an incomplete version $\tilde{U}_B(H)$
  - Uniformly sample B terms (with replacement)
- Main result : if $B = O(n)$, the learning rate is preserved
  - $O(1/\sqrt{n})$ convergence in the general case
  - Objective function has only $O(n)$ terms v.s. $O(n^2)$ initially
  - Due to high dependence in the pairs
- Naive strategy : use complete U-statistic with fewer samples
  - Uniformly sample $O(\sqrt{n})$ samples
  - Form all possible pairs : $O(n)$ terms
  - Learning rate is only $O(1/n^{1/4})$
- Can use similar ideas in mini-batch SGD
  - Reduce variance of gradient estimates

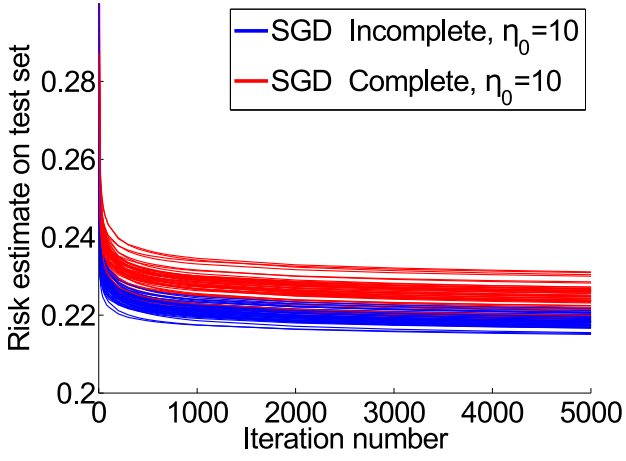
Scaling-up ERM for metric learning

Experiments

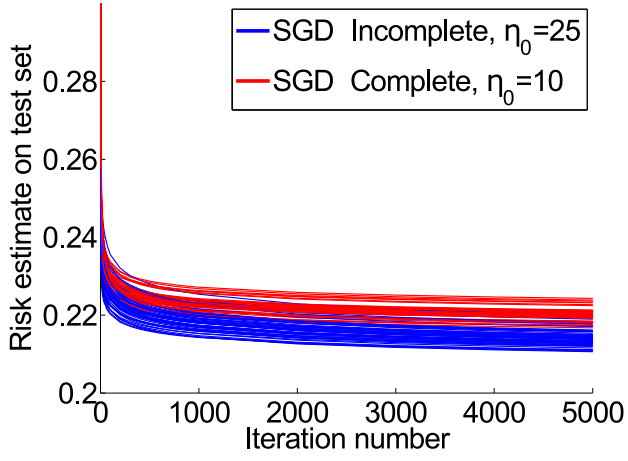
- MNIST dataset : $n = 60000 \rightarrow 2 \times 10^9$ pairs
- Mini-batch SGD
  - Step size tuning
  - 50 random runs



(a) $B = 10$



(b) $B = 55$



(c) $B = 253$

Conclusion and perspectives

- Metric learning: useful in a variety of setting
- Ideas to overcome some key limitations
  - Large datasets
  - High dimensionality
  - Learn many local metrics
- Details and more results in the papers
- One interesting challenge : distributed metric learning
  - Existing work [Xie and Xing, 2014] too restrictive

References I

[Clémençon et al., 2015] Clémençon, S., Bellet, A., and Colin, I. (2015).
Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics.
Technical report, arXiv:1501.02629.

[Duchi and Singer, 2009] Duchi, J. and Singer, Y. (2009).
Efficient Online and Batch Learning Using Forward Backward Splitting.
JMLR, 10:2899–2934.

[Jaggi, 2013] Jaggi, M. (2013).
Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization.
In *ICML*.

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

[Shi et al., 2014] Shi, Y., Bellet, A., and Sha, F. (2014).
Sparse Compositional Metric Learning.
In *AAAI*, pages 2078–2084.

[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.

References II

[Xiao, 2010] Xiao, L. (2010).
Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization.
JMLR, 11:2543–2596.

[Xie and Xing, 2014] Xie, P. and Xing, E. (2014).
Large Scale Distributed Distance Metric Learning.
Technical report, arXiv:1412.5949.