SIMILARITY AND DISTANCE METRIC LEARNING

Aurélien Bellet (Inria)

Research School on Uncertainty in Scientific Computing (ETICS 2019)
September 26, 2019

This morning

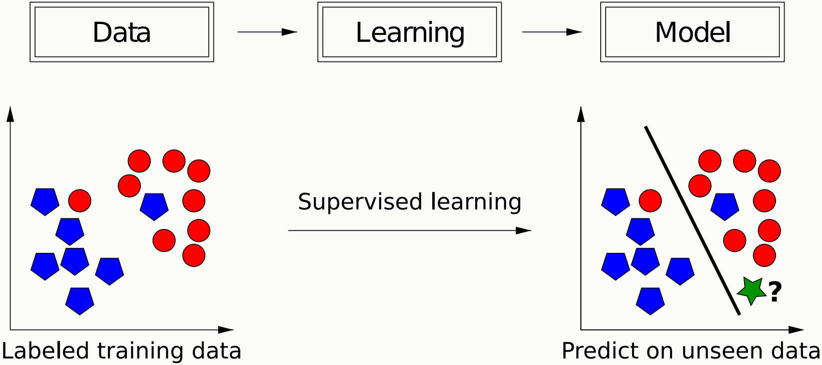
1. A brief formal introduction to machine learning
2. Similarity and distance metric learning

Tomorrow morning

Practical session in Python

A BRIEF FORMAL INTRODUCTION TO ML

SUPERVISED MACHINE LEARNING



SUPERVISED LEARNING: NOTATIONS

- Labeled data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- $\mathcal{X} \subset \mathbb{R}^q$: representation space (features)
- $\mathcal{Y}$: discrete (classification) or continuous (regression)
- A predictive model is a function $f : \mathcal{X} \to \mathcal{Y}$
- We measure the discrepancy between the prediction $f(x)$ and the true label y using a loss function $\ell(f; x, y)$

- We have access to a training dataset $\mathcal{S}_n = \{(x_i, y_i)\}_{i=1}^n$ of n labeled points
- A supervised ML algorithm takes $\mathcal{S}_n$ as input and outputs a model $f : \mathcal{X} \to \mathcal{Y}$
- The learned model f can then be used to predict a label $y \in \mathcal{Y}$ for any (new) data point $x \in \mathcal{X}$

The goal of ML is to generalize to unseen data

$\to$ **need an assumption to relate training data and future data**

SUPERVISED LEARNING: STATISTICAL SETTING

- **Assumption:** all data points $(x, y) \in \mathcal{X} \times \mathcal{Y}$ follows some unknown but fixed distribution μ (specific to the task)
- This is assumed to hold for both training and unseen data
- **Goal**: learn a model f in some model family $\mathcal{F}$ from training data which has small expected loss over μ:

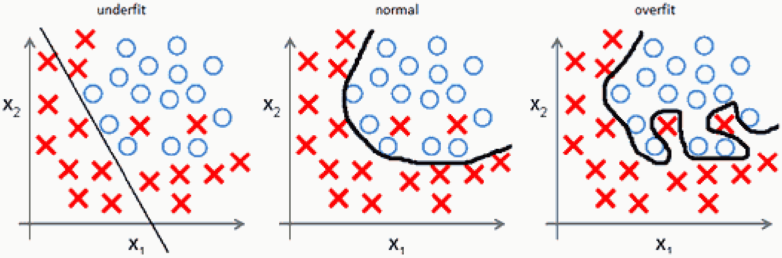
$$R(f) = \mathop{\mathbb{E}}_{(x,y)\sim\mu} \ell(f; x, y)$$

- But μ is unknown, so cannot compute $R(f)$

- Intuitive idea: minimize average loss on training data

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f; x_i, y_i)$$

- The hope is that an accurate model on training data will also do well on unseen data
- Why do we care about the model family $\mathcal{F}$? Can't we use a very expressive family which can model any data?



- $\mathcal{F}$ too simple $\rightarrow$ underfitting
- $\mathcal{F}$ too complex $\rightarrow$ overfitting
- Note: the complexity of $\mathcal{F}$ also impacts the algorithmic complexity of the learning procedure

STATISTICAL LEARNING GUARANTEES

- This trade-off is well-explained by statistical learning theory
- One can prove results of the form: for any $f \in \mathcal{F}$, w.p. $1 - \delta$

$$R(f) \leq \hat{R}(f) + \sqrt{\frac{C_{\mathcal{F}} \log(1/\delta)}{n}}$$

  where $C_{\mathcal{F}}$ is a measure of complexity of the model class $\mathcal{F}$
- $C_{\mathcal{F}}$ can simply be $|\mathcal{F}|$ when model family is finite
- Note: regularization can be used to penalize complexity within $\mathcal{F}$

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \hat{R}(f) + \lambda \Omega(f)$$

ERM example 1: linear regression

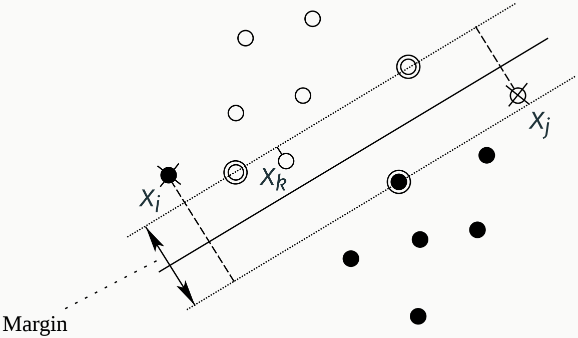
- Real labels $\mathcal{Y} = \mathbb{R}$
- Linear model $f_\theta(x) = \theta^T x$ parameterized by $\theta \in \mathbb{R}^q$
- Quadratic loss $\ell(f_\theta; x, y) = (y - f_\theta(x))^2$
- ERM problem is a simple least-square problem:

$$\hat{\theta} \in \underset{\theta \in \mathbb{R}^p}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2, \text{ equivalent to } \hat{\theta} \in \underset{\theta \in \mathbb{R}^p}{\arg\min} \|Y - X\theta\|_2^2$$

- Common regularization terms:
  - Squared L2 norm: $\|\theta\|_2^2$
  - L1 norm: $\|\theta\|_1$ (sparsity inducing, cf LASSO)

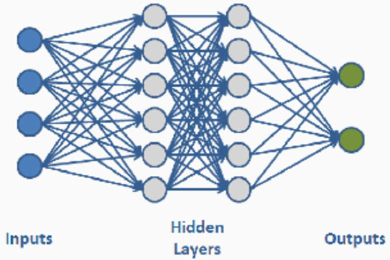
ERM EXAMPLE 2: LINEAR CLASSIFICATION WITH HINGE LOSS

- Binary labels $\mathcal{Y} = \{-1, 1\}$, linear model $f_\theta(x) = \text{sign}[\theta^T x]$
- Hinge loss $\ell(f_\theta; x, y) = \max(0, 1 - y\theta^T x)$ to enforce a safety margin
- ERM problem with L2 regularization is Support Vector Machine



ERM EXAMPLE 3: DEEP NEURAL NETS

- Feed-forward, fully connected DNN:
  - First layer is the input $x_0 = x$
  - Intermediate layers: $x_l = \sigma(W_l x_{l-1})$ with σ nonlinear mapping
  - Last layer: linear model on previous layer + loss



- Specialized networks: CNNs (images), LTSMs/RNNs (sequences)...
- High model complexity, but can still generalize well in practice!
- A lot of ongoing work to better understand this theoretically

- Assume $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$, the ERM problem is

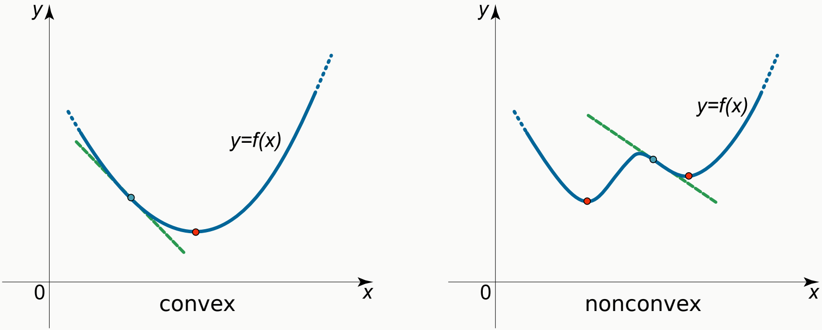
$$\min_{\theta \in \mathbb{R}^p} \hat{R}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; x_i, y_i)$$

- We typically work with loss functions that are differentiable in θ
- The workhorse of ML is first-order optimization methods: iteratively refine θ based on (an estimate of) the gradient

$$\nabla \hat{R}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(\theta; x_i, y_i)$$

SOLVING ERM PROBLEMS: GRADIENT-BASED METHODS

- Gradient Descent (GD):
  - Initialize to some $\theta(0) \in \mathbb{R}^p$
  - For $t = 0, \dots, T$: update $\theta(t+1) = \theta(t) - \gamma \nabla \hat{R}(\theta(t))$
- Stochastic Gradient Descent (SGD):
  - Initialize to some $\theta(0) \in \mathbb{R}^p$
  - For $t = 0, \dots, T$: pick random index $i_t \in \{1, \dots, n\}$ and update $\theta(t+1) = \theta(t) - \gamma(t) \nabla_\theta \ell(\theta; x_{i_t}, y_{i_t})$
- γ is the step size (or learning rate) to be tuned
- In ML, we typically do not care about high-precision solutions
- For large datasets, SGD has much cheaper iterations and converges faster to a solution with reasonable precision



- For convex objective functions, gradient-based methods will converge to the global minimum (under appropriate step size)
- Nonconvex case: convergence only to a local minimum
- Convergence rate depends on properties of the objective
- Note: optimization for ML is a very active topic

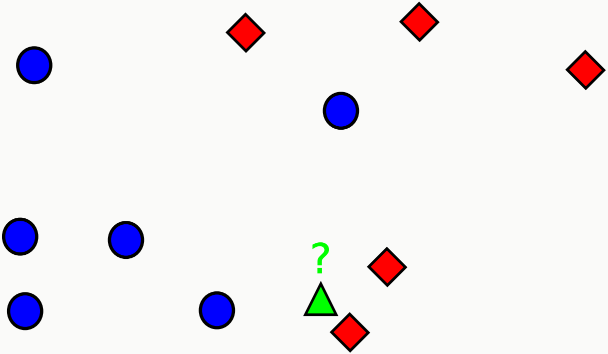
SIMILARITY AND DISTANCE METRIC LEARNING

FUNDAMENTAL QUESTION OF THIS PRESENTATION

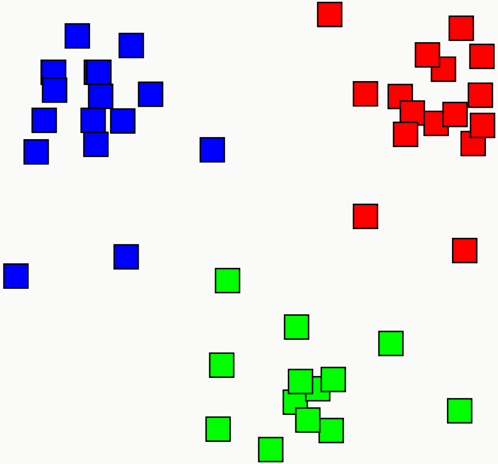
How to appropriately measure similarity or distance between things depending on the context?

- 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

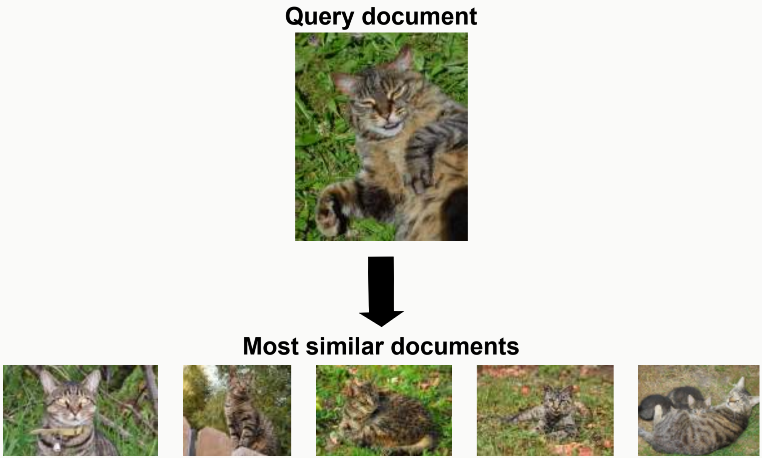
Nearest neighbor classification



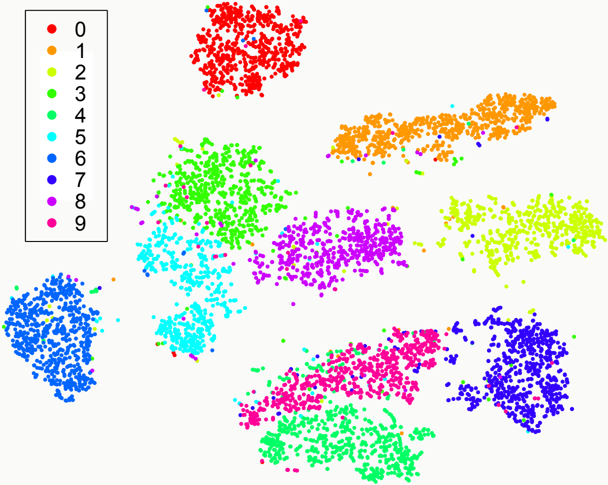
Clustering



Information retrieval

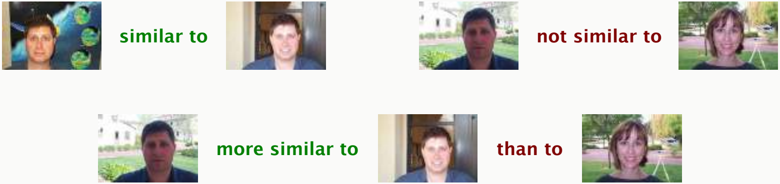


Data visualization



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

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



(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: invidivual labels are 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 all we need
  - Information retrieval (rank results by similarity to a query)

EXAMPLE APPLICATION: FACE VERIFICATION

- Face verification 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 ERM problem of the form

$$M^* = \arg\min_M \left[\underbrace{\hat{R}(M, \mathcal{S}, \mathcal{D}, \mathcal{R})}_{\text{empirical risk}} + \underbrace{\lambda \Omega(M)}_{\text{regularization}} \right]$$

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 ($p \geq 1$)

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

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

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

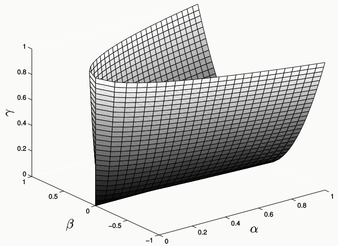
- When $p = 1$: Manhattan distance $d_{man}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} |x_i - x_i'|$
- When $p \to \infty$: Chebyshev distance $d_{che}(\mathbf{x}, \mathbf{x}') = \max_i |x_i - x_i'|$

- 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



- 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:

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

- If $\mathrm{rank}(\boldsymbol{M}) = k \leq d$, 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]

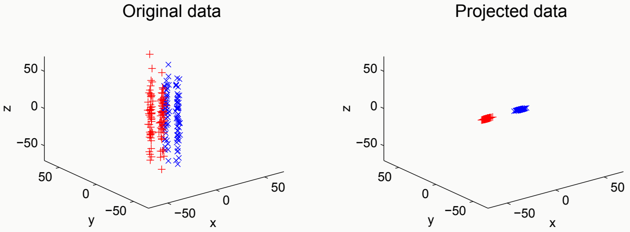
- Targeted task: clustering with side information

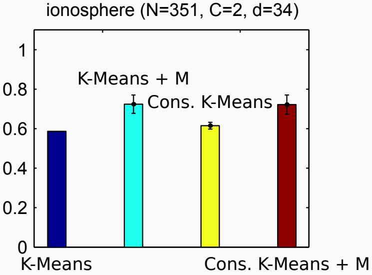
Formulation

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

- Problem is convex in M and always feasible (take $M = \mathbf{0}$)
- 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]

- Targeted task: information retrieval

Formulation

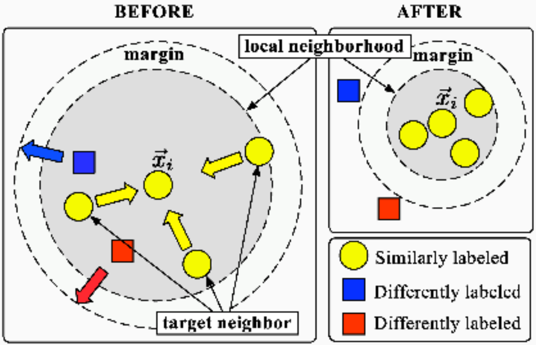
$$\min_{\mathbf{M}\in\mathbb{S}_+^d,\boldsymbol{\xi}\geq 0} \quad \|\mathbf{M}\|_{\mathcal{F}}^2 \quad + \quad \lambda \sum_{i,j,k} \xi_{ijk}$$

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

- Regularization by Frobenius norm $\|\mathbf{M}\|_{\mathcal{F}}^2 = \sum_{i,j=1}^{d} M_{ij}^2$
- 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
  - $\mathcal{S} = \{(x_i, x_j) : y_i = y_j, x_j$ belongs to k-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

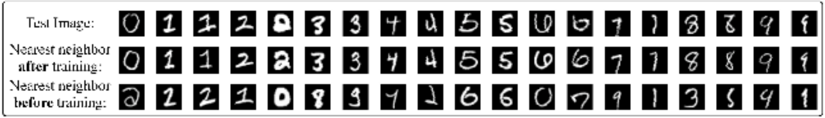
$$\min_{M \in \mathbb{S}_+^d, \boldsymbol{\xi} \geq 0} \quad (1-\mu) \sum_{(x_i, x_j) \in \mathcal{S}} D_M^2(x_i, x_j) \quad + \quad \mu \sum_{i,j,k} \xi_{ijk}$$

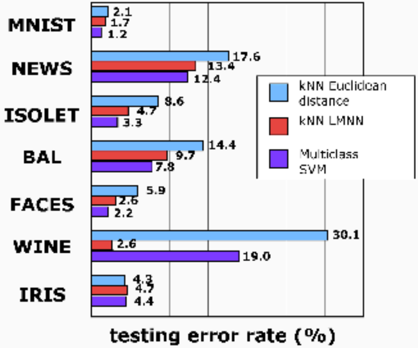
$$\text{s.t.} \quad D_M^2(x_i, x_k) - D_M^2(x_i, x_j) \geq 1 - \xi_{ijk} \quad \forall (x_i, x_j, x_k) \in \mathcal{R}$$

$\mu \in [0, 1]$ 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

- 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

- LogDet divergence (used in ITML [Davis et al., 2007])

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

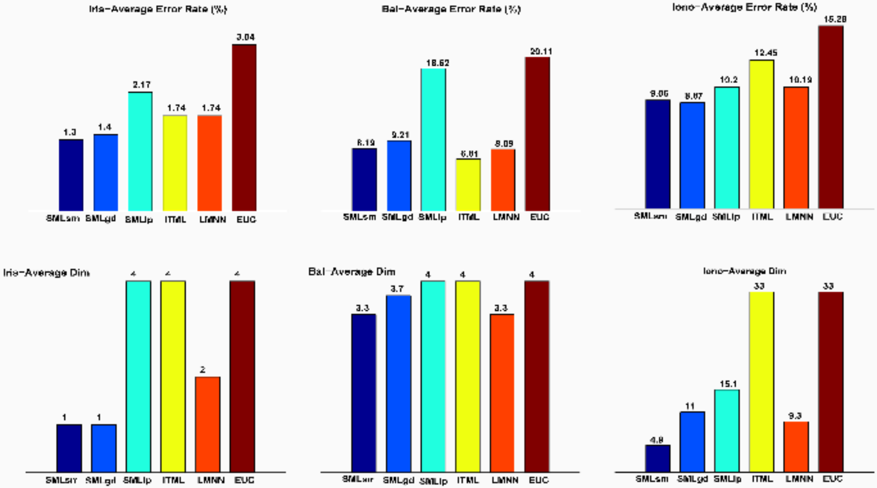
where $\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and $\boldsymbol{M}_0 = \boldsymbol{U}\boldsymbol{\Theta}\boldsymbol{U}^T$ is PD

  - Remain close to good prior metric $\boldsymbol{M}_0$ (e.g., identity)
  - Implicitly ensure that $\boldsymbol{M}$ is PD
  - Convex in $\boldsymbol{M}$ (determinant of PD matrix is log-concave)
  - Efficient Bregman projections in $O(d^2)$

Interesting regularizers

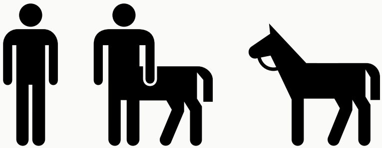
- Mixed $L_{2,1}$ norm: $\|M\|_{2,1} = \sum_{i=1}^{d} \|M_i\|_2$
    - Tends to zero-out entire columns $\rightarrow$ feature selection
    - Convex but nonsmooth
    - Efficient proximal gradient algorithms
- Trace (or nuclear) norm: $\|M\|_* = \sum_{i=1}^{d} \sigma_i(M)$
    - Favors low-rank matrices $\rightarrow$ dimensionality reduction
    - Convex but nonsmooth
    - Efficient Frank-Wolfe algorithms

$L_{2,1}$ norm illustration



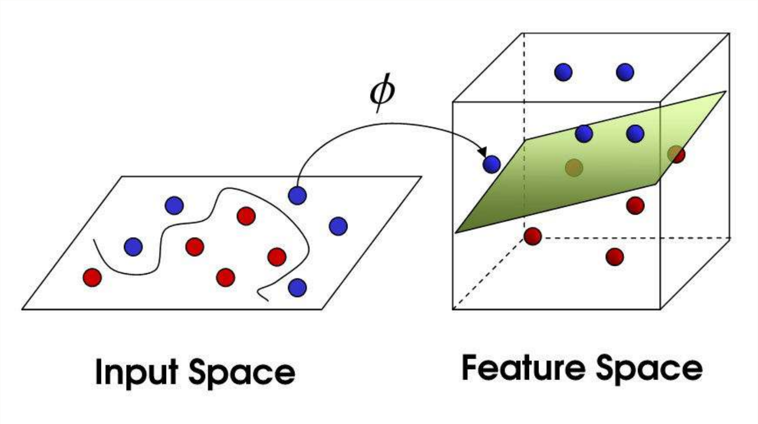
(image taken from [Ying et al., 2009])

- 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 [Tversky and Gati, 1982]



  - 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


ϕ
Input Space
Feature Space

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 $c_1, \ldots, c_n \in \mathbb{R}$.

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

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

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

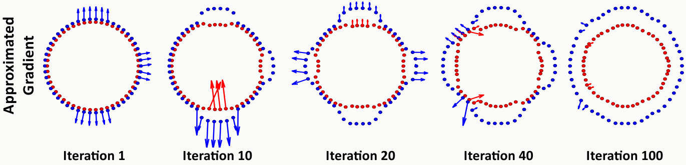
$$D_{\mathbf{M}}^2(\phi(\mathbf{x}), \phi(\mathbf{x}')) = (\mathbf{k} - \mathbf{k}')^T \mathbf{M} (\mathbf{k} - \mathbf{k}')$$

  - $\mathbf{M} = \mathbf{U}^T \mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{k} = \boldsymbol{\Phi}^T \phi(\mathbf{x}) = [K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_n, \mathbf{x})]^T$
- Justified by a representer theorem [Chatpatanasiri et al., 2010]

KERNELIZATION OF LINEAR METHODS

- 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 [Chatpatanasiri et al., 2010]:
  1. Kernel PCA: nonlinear mapping to low-dimensional space
  2. Apply linear metric learning algorithm to transformed data

LEARNING A NONLINEAR TRANSFORMATION

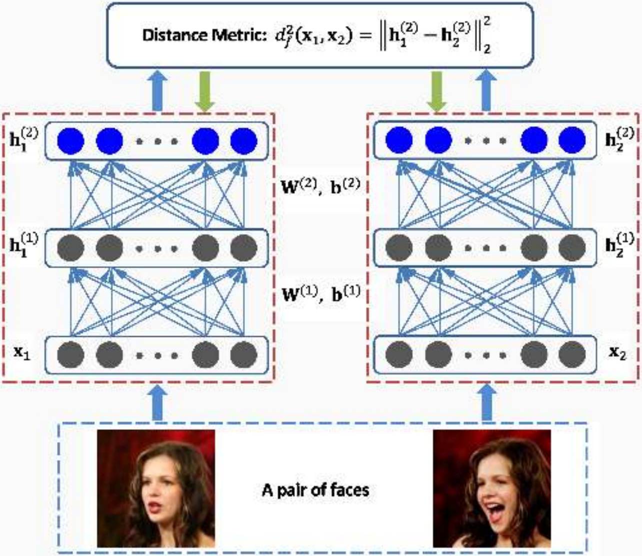


- More flexible approach: learn nonlinear mapping ϕ to optimize

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

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

- Typically nonconvex formulations



(image taken from [Hu et al., 2014])

Multiple Metric LMNN [Weinberger and Saul, 2009]

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

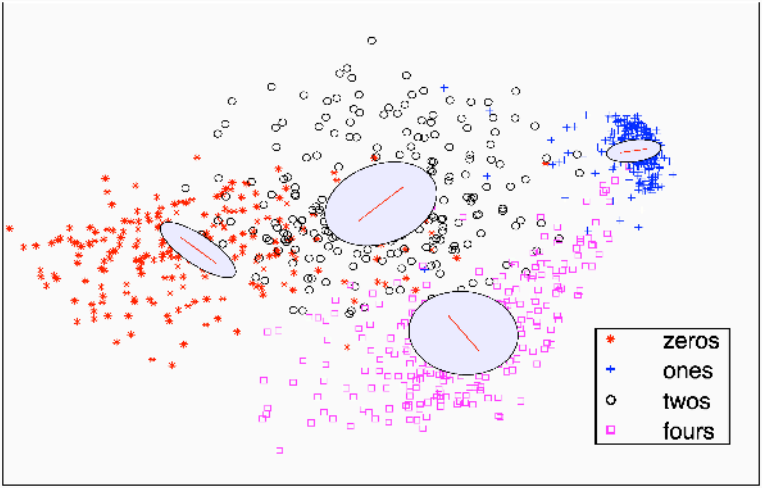
Formulation

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

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

- Remains convex
- Computationally more expensive than standard LMNN
- Subject to overfitting (many parameters)
- Other local approaches: [Wang et al., 2012, Shi et al., 2014]

Multiple Metric LMNN [Weinberger and Saul, 2009]



LARGE-SCALE METRIC LEARNING

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

- Goal: minimize regret

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

  - ℓ_t: loss suffered at time t
  - M_t: metric learned at time t
  - 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

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

  - $S_M(x, x') = x^T M x'$, $C \geq 0$ trade-off parameter

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

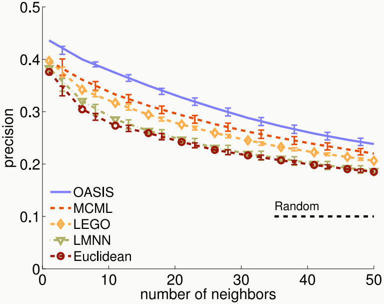
$$
\begin{aligned}
M^t = \quad \arg\min_{M,\xi} \quad & \frac{1}{2}\|M - M^{t-1}\|_{\mathcal{F}}^2 + C\xi \\
\text{s.t.} \quad & 1 - S_M(x_i, x_j) + S_M(x_i, x_k) \leq \xi \\
& \xi \geq 0
\end{aligned}
$$

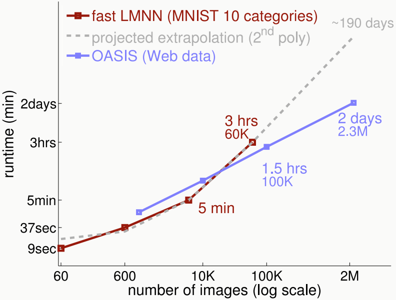
  - $S_M(x, x') = x^T M x'$, $C \geq 0$ trade-off parameter

- Denoting $V = x_i(x_j - x_k)^T$, solution is given by $M^t = M^{t-1} + \beta V$ with

$$\beta = \min\left(C, \frac{\max(0, 1 - S_M(x_i, x_j) + S_M(x_i, x_k))}{\|V\|_F^2}\right)$$

OASIS [Chechik et al., 2010]





- Trained with 160M triplets in 3 days on 1 CPU

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
- 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
  - Used in [Xing et al., 2002, Schultz and Joachims, 2003]
  - 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

CASE OF LARGE d

Matrix decompositions

- Low-rank decomposition $\boldsymbol{M} = \boldsymbol{L}^T\boldsymbol{L}$ with $\boldsymbol{L} \in \mathbb{R}^{r\times d}$
  - Used in [Goldberger et al., 2004]
  - Learn $r \times d$ parameters
  - Generally nonconvex, must tune r
- Rank-1 decomposition $\boldsymbol{M} = \sum_{i=1}^{K} w_k \boldsymbol{b}_k \boldsymbol{b}_k^T$
  - Used in SCML [Shi et al., 2014]
  - Learn K parameters
  - Must choose good basis set

HDSL [Liu et al., 2015, Liu and Bellet, 2019]

- Learn similarity $S_M(x, x') = x^T M x'$
- Given $\lambda > 0$, for any $i, j \in \{1, \ldots, d\}$, $i \neq j$ we define

$$P_\lambda^{(ij)} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad N_\lambda^{(ij)} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & -\lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -\lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\mathcal{B}_\lambda = \bigcup_{ij} \left\{ P_\lambda^{(ij)}, N_\lambda^{(ij)} \right\}$$

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

- One basis involves only 2 features:

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

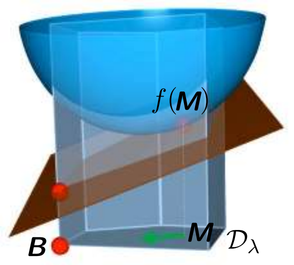
HDSL [Liu et al., 2015, Liu and Bellet, 2019]

Optimization problem (ℓ: smoothed hinge loss)

$$\min_{M \in \mathbb{R}^{d \times d}} \quad f(M) = \frac{1}{|\mathcal{R}|} \sum_{(x_i, x_j, x_k) \in \mathcal{R}} \ell\left(1 - x_i^T M x_j + x_i^T M x_k\right)$$

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

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



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

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

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

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

end for

Case of large d: case of s-sparse data

HDSL [Liu et al., 2015, Liu and Bellet, 2019]

Convergence

Let $L = \frac{1}{|\mathcal{R}|} \sum_{(x_i, x_j, x_k) \in \mathcal{R}} \|\boldsymbol{x}_i(\boldsymbol{x}_j - \boldsymbol{x}_k)^T\|_F^2$. 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 $O(|\mathcal{R}|s^2)$ time and memory
- 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

GENERALIZATION GUARANTEES

- Training dataset $\mathcal{S}_n = \{(x_i, y_i)\}_{i=1}^n$
- For ease of notation, denote a labeled point by $z = (x, y)$
- Minimize the regularized empirical risk

$$\hat{R}(M) = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n}^{n} \ell(M; z_i, z_j) + \lambda \Omega(M)$$

- Hope to achieve small expected risk

$$R(M) = \mathop{\mathbb{E}}_{z, z' \sim \mu} [\ell(M; z, z')]$$

- Note: this can be easily adapted to triplets

STATISTICAL VIEW OF SUPERVISED METRIC LEARNING

- Standard statistical learning theory: sum of i.i.d. terms
- In metric learning $\hat{R}(M)$ is a sum of dependent terms!
    - Each training point involved in several pairs
    - This is indeed the case in practice
- Need specific tools to go around this problem

Definition ([Jin et al., 2009])

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

$$\forall(\mathcal{S}_n, x, y), \forall i, \quad \sup_{z_1,z_2} |\ell(\hat{M}_{\mathcal{S}_n}, z_1, z_2) - \ell(\hat{M}_{\mathcal{S}_n^{i,z}}, z_1, z_2)| \leq \frac{\kappa}{n}$$

- $M_{\mathcal{S}_n}$: metric learned from $\mathcal{S}_n$
- $\mathcal{S}_n^{i,z}$: set obtained by replacing $z_i \in \mathcal{S}_n$ by z

- If $\Omega(M) = \|M\|_{\mathcal{F}}^2$, under mild conditions (ℓ Lipschitz, bounded domain), algorithm has uniform stability [Jin et al., 2009]
- Does not apply to other (sparse) regularizers

Theorem ([Jin et al., 2009])

For any metric learning algorithm with uniform stability κ/n, with probability $1-\delta$ over the random sample $\mathcal{S}_n$, we have:

$$R(M_{\mathcal{S}_n}) \leq \hat{R}(M_{\mathcal{S}_n}) + \frac{2\kappa}{n} + (2\kappa + B(d))\sqrt{\frac{\ln(2/\delta)}{2n}}$$

- Standard learning rate in $O(1/\sqrt{n})$
- Dependence on dimension: $B(d)$ is in $O(\sqrt{d})$

- Algorithmic robustness [Bellet and Habrard, 2015]
    - Wide applicability but loose bounds
- U-processes and Rademacher complexity [Cao et al., 2012]
    - Tighter bounds for several matrix norms
    - Example: $O(\sqrt{\log d})$ for $L_{2,1}$ norm
- U-processes and sparse greedy algorithms [Liu and Bellet, 2019]
    - $O(\sqrt{\log k})$ where k is the number of iterations

OTHER TOOLS AND RESULTS

- U-processes and subsampling [Clémençon et al., 2016]
    - Approximation of empirical risk by sampling $O(n)$ pairs
    - Minimization of this incomplete risk preserves $O(1/\sqrt{n})$ rate
    - Fast rates in $O(1/n)$ under assumptions on data distribution
- Uniform stability and learning with similarity [Bellet et al., 2012b]
    - Similarity learning for linear classification
    - Generalization bounds for classifier based on learned similarity
    - Builds upon theory developed in [Balcan and Blum, 2006]

- 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
- Statistical learning guarantees
- Very successful in practical applications
- More on metric learning: can refer to [Bellet et al., 2015]

AN OPEN SOURCE PACKAGE: `metric-learn`

- `metric-learn`: Metric Learning Algorithms in Python

  GitHub repo: `https://github.com/scikit-learn-contrib/metric-learn`

  Doc: `http://contrib.scikit-learn.org/metric-learn/`

- Implements popular supervised and weakly supervised algorithms within a unified API
- Compatible with scikit-learn (part of scikit-learn-contrib)
- Open source package, high test coverage
- Last major release in July 2019
- See [de Vazelhes et al., 2019] for more technical details
- You're welcome to contribute!

Thank you for your attention!

Questions?

REFERENCES I

[Balcan and Blum, 2006] Balcan, M.-F. and Blum, A. (2006).
On a Theory of Learning with Similarity Functions.
In *ICML*.

[Bellet and Habrard, 2015] Bellet, A. and Habrard, A. (2015).
Robustness and Generalization for Metric Learning.
Neurocomputing, 151(1):259–267.

[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*.

[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.

[Cao et al., 2012] Cao, Q., Guo, Z.-C., and Ying, Y. (2012).
Generalization Bounds for Metric and Similarity Learning.
Technical report, University of Exeter.

[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., 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.

[Clémençon et al., 2016] Clémençon, S., Colin, I., and Bellet, A. (2016).
Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics.
Journal of Machine Learning Research, 17(76):1–36.

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

[de Vazelhes et al., 2019] de Vazelhes, W., Carey, C., Tang, Y., Vauquier, N., and Bellet, A. (2019).
metric-learn: Metric Learning Algorithms in Python.
Technical report, arXiv:1908.04710.

[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.

[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*.

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

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

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

[Liu and Bellet, 2019] Liu, K. and Bellet, A. (2019).
Escaping the curse of dimensionality in similarity learning: Efficient frank-wolfe algorithm and generalization bounds.
Neurocomputing, 333:185–199.

[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*.

[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*.

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

[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.

[Tversky and Gati, 1982] Tversky, A. and Gati, I. (1982).
Similarity, separability, and the triangle inequality.
Psychological Review, 89(2):123–154.

[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.

[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*.

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

[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*.

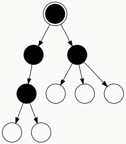
[Zhang and Yeung, 2010] Zhang, Y. and Yeung, D.-Y. (2010).
Transfer metric learning by learning task relationships.
In *KDD*.

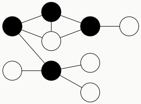
[Zhu et al., 2015] Zhu, X., Lei, Z., Yan, J., Yi, D., and Li, S. Z. (2015).
High-fidelity pose and expression normalization for face recognition in the wild.
In *CVPR*.

BONUS: METRIC LEARNING FOR STRUCTURED DATA

- 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

- 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

- 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)

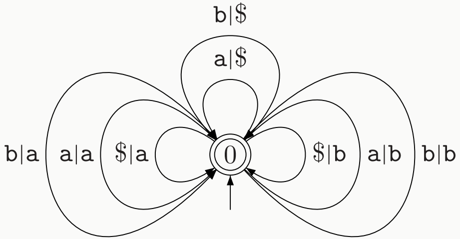
Example 2: specific costs

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

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

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

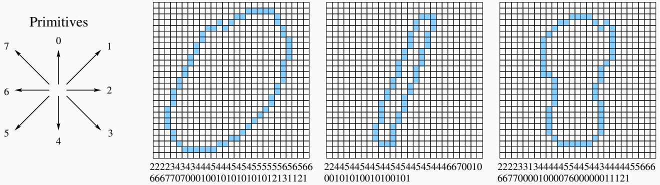


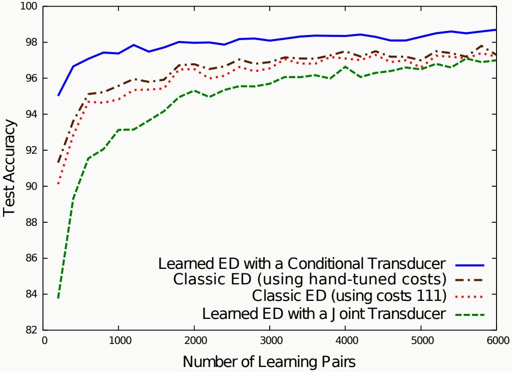
Iterative Expectation-Maximization algorithm [Oncina and Sebban, 2006]

- 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., 2012a]

- 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 $u \rightarrow v$ appears in Levenshtein script
- e_C is a linear function of the costs

GESL [Bellet et al., 2012a]

Formulation

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

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

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., 2012a]