Supervised Metric Learning

M. Sebban

Laboratoire Hubert Curien, UMR CNRS 5516
University of Jean Monnet Saint-Étienne (France)

AAFD’14, Paris 13, April, 2014
Outline

1. Intuition behind Metric Learning

2. State of the Art
   - Mahalanobis Distance Learning
   - Nonlinear Metric Learning
   - Online Metric Learning

3. Similarity Learning for Provably Accurately Linear Classification

4. Consistency and Generalization Guarantees

5. Experiments
The notion of **metric** plays an important role in many domains such as classification, regression, clustering, ranking, etc.
**Minkowski distances: family of distances induced by $\ell_p$ norms**

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

- For $p = 1$, the **Manhattan distance** $d_{man}(x, x') = \sum_{i=1}^{d} |x_i - x'_i|$.
- For $p = 2$, the “ordinary” **Euclidean distance**:
  \[ d_{euc}(x, x') = \left( \sum_{i=1}^{d} |x_i - x'_i|^2 \right)^{1/2} = \sqrt{(x - x')^T (x - x')} \]
- For $p \to \infty$, the **Chebyshev distance** $d_{che}(x, x') = \max_i |x_i - x'_i|$.
Key question

How to choose the right metric?

The notion of good metric is problem-dependent

Each problem has its own notion of similarity, which is often badly captured by standard metrics.
Metric learning
Adapt the metric to the problem of interest

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

It typically induces a change of representation space which satisfies constraints.
The Mahalanobis distance

∀\(x, x' \in \mathbb{R}^d\), the Mahalanobis distance is defined as follows:

\[
d_M(x, x') = \sqrt{(x - x')^T M (x - x')},
\]

where \(M \in \mathbb{R}^{d \times d}\) is a symmetric PSD matrix \((M \succeq 0)\).

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

Useful properties

If \(M \succeq 0\), then

- \(x^T M x \geq 0 \ \forall x\) (as a linear operator, can be seen as nonnegative scaling).
- \(M = L^T L\) for some matrix \(L\).
Using the decomposition $\mathbf{M} = \mathbf{L}^T \mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{k \times d}$, where $k$ is the rank of $\mathbf{M}$, one can rewrite $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$.

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

$$= \sqrt{(\mathbf{Lx} - \mathbf{Lx}')^T (\mathbf{Lx} - \mathbf{Lx}')}}.$$

**Mahalanobis distance learning = Learning a linear projection**

If $\mathbf{M}$ is learned, a Mahalanobis distance implicitly corresponds to computing the **Euclidean distance after a learned linear projection** of the data by $\mathbf{L}$ in a $k$-dimensional space.
Metric learning in a nutshell

General formulation

Given a metric, find its parameters $M^*$ as

$$M^* = \arg \min_{M \succeq 0} [\ell(M, S, D, R) + \lambda R(M)],$$

where

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

State of the art methods essentially differ by the choice of constraints, loss function and regularizer on $M$. 
Main Idea

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

$$S = \{(x_i, x_j) : y_i = y_j \text{ and } x_j \text{ belongs to the } k\text{-neighborhood of } x_i\},$$

$$R = \{(x_i, x_j, x_k) : (x_i, x_j) \in S, y_i \neq y_k\}.$$
LMNN (Weinberger et al. 2005)

Formulation

\[
\begin{align*}
\min_{M \succeq 0} & \quad \sum_{(x_i, x_j) \in S} d^2_M(x_i, x_j) \\
\text{s.t.} & \quad d^2_M(x_i, x_k) - d^2_M(x_i, x_j) \geq 1 \quad \forall (x_i, x_j, x_k) \in \mathcal{R}.
\end{align*}
\]

Remarks

- **Advantages**: Convex, with a solver based on working set and subgradient descent. Can deal with millions of constraints and very popular in practice.
- **Drawback**: Subject to overfitting in high dimension.
Information-Theoretical Metric Learning (ITML) introduces LogDet divergence regularization. This Bregman divergence on PSD matrices is defined as:

\[ D_{ld}(M, M_0) = \text{trace}(MM_0^{-1}) - \log \det(MM_0^{-1}) - d. \]

where \(d\) is the dimension of the input space and \(M_0\) is some PSD matrix we want to remain close to. ITML is formulated as follows:

\[
\min_{M \succeq 0} D_{ld}(M, M_0) \\
\text{s.t.} \\
\quad d_M^2(x_i, x_j) \leq u \quad \forall (x_i, x_j) \in S \\
\quad d_M^2(x_i, x_j) \geq v \quad \forall (x_i, x_j) \in D,
\]

The LogDet divergence is finite iff \(M\) is PSD (cheap way of preserving a PSD matrix). It is also rank-preserving.
Nonlinear metric learning

The big picture

Three approaches

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

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

A solution: KPCA trick (Chatpatanasiri et al., 2010)

- Use KPCA (PCA in kernel space) to get a nonlinear but low-dimensional projection of the data.
- Then use unchanged algorithm!
Nonlinear metric learning
Learning a nonlinear metric: GB-LMNN (Kedem et al. 2012)

Main idea

- Learn a nonlinear mapping $\phi$ to optimize the Euclidean distance $d_\phi(x, x') = \|\phi(x) - \phi(x')\|_2$ in the transformed space.
- $\phi = \phi_0 + \alpha \sum_{t=1}^{T} h_t$, where $\phi_0$ is the mapping learned by linear LMNN, and $h_1, \ldots, h_T$ are gradient boosted regression trees.
- Intuitively, each tree divides the space into $2^p$ regions, and instances falling in the same region are translated by the same vector.
Nonlinear metric learning
Local metric learning

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

Pitfalls
- How to split the space?
- How to avoid a blow-up in number of parameters to learn, and avoid overfitting?
- How to obtain a proper (continuous) global metric?
- ...
If the number of training constraints is very large (this can happen even with a moderate number of training examples), previous algorithms become huge, possibly intractable optimization problems (gradient computation and/or projections become very expensive).

One solution: online learning

- In online learning, the algorithm receives training pairs of instances one at a time and updates the current hypothesis at each step.
- Performance typically inferior to that of batch algorithms, but allows to tackle large-scale problems.
- Often come with guarantees in the form of regret bounds stating that the accumulated loss suffered along the way is not much worse than that of the best hypothesis chosen in hindsight.
Mahalanobis distance learning
LEGO (Jain et al. 2008)

Formulation

At each step, receive \((x_t, x_t', y_t)\) where \(y_t\) is the target distance between \(x_t\) and \(x_t'\), and update as follows:

\[
M^{t+1} = \arg \min_{M \succeq 0} D_{ld}(M, M^t) + \lambda \ell(M, x_t, x_t', y_t),
\]

where \(\ell\) is a loss function (square loss, hinge loss...).

Remarks

- It turns out that the above update has a closed-form solution which maintains \(M \succeq 0\) automatically.
- Can derive a regret bound.
There exist many other metric learning approaches. Most of them are discussed at more length in our recent survey:

Limitations of the state of the art ML algorithms

Algorithmic limitations

Drawbacks of Mahalanobis distance learning:

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

**Cosine similarity**

The cosine similarity (widely used in data mining) measures the cosine of the angle between two instances, and can be computed as

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

**Bilinear similarity**

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

$$K_M(x, x') = x^T M x',$$

where $M$ is not required to be PSD nor symmetric.
Limitations of the state of the art ML algorithms

Theoretical limitations

Establishing theoretical guarantees for the metric learning algorithms has so far received very little attention. However, we may be interested in theoretical results on:

- the learned metric $d_M$ itself (optimized w.r.t. training data),
- and on the algorithm which makes use of it ("Plug and hope" strategy).

3 contributions:

1. Optimize a **similarity** function (bilinear similarity) rather than a true distance.

2. **Consistency guarantees** for the learned similarity: using the uniform stability framework.

3. **Generalization guarantees** for the algorithm using the similarity: optimizing the notion of **goodness** (theory of Balcan et al. 2008).
Definition (Balcan et al., 2008)

A similarity function $K \in [-1, 1]$ is $\textbf{(ε, γ, τ)-good}$ w.r.t. to an indicator function $R(x)$ defining a set of “reasonable points” if:

1. A $1 - \epsilon$ probability mass of examples $(x, y)$ satisfy:
   \[
   \mathbb{E}_{(x', y') \sim P} [yy'K(x, x') | R(x')] \geq \gamma.
   \]

2. $\Pr_{x'}[R(x')] \geq \tau$. \hspace{1cm} $\epsilon, \gamma, \tau \in [0, 1]$

- The first condition requires that a $1 - \epsilon$ proportion of examples $x$ are \textbf{on average} more similar to reasonable examples of the same class than to reasonable examples of the opposite class by a margin $\gamma$.

- The second condition means that at least a $\tau$ proportion of the examples are reasonable.
Strategy

If $R$ is known, use $K$ to map the examples to the space $\phi$ of “the similarity scores with the reasonable points” (similarity map).

![Diagram showing the mapping of examples to the similarity space](image)
A trivial linear classifier

By definition of $(\epsilon, \gamma, \tau)$-goodness, we have a linear classifier in $\phi$ that achieves true risk $\epsilon$ at margin $\gamma$. 
Deriving generalization guarantees

Generalization guarantees for the classifier using the metric: $(\epsilon, \gamma, \tau)$-goodness

Theorem (Balcan et al., 2008)

If $R$ is unknown, given $K$ is $(\epsilon, \gamma, \tau)$-good and enough points to create a similarity map, with high probability there exists a linear separator $\alpha$ that has true risk $\epsilon$ at margin $\gamma$.

Question

Can we find this linear classifier in an efficient way?
Answer

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

\[
\min_{\alpha} \sum_{i=1}^{n} \left[ 1 - \sum_{j=1}^{n} \alpha_j y_i K(x_i, x_j) \right] + \lambda \|\alpha\|_1
\]

$L_1$ norm induces sparsity

L2 constraint

L1 constraint
The performance of the linear classifier theoretically depends on how well the similarity function satisfies the definition of goodness.

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

**SLLC optimizes the empirical goodness** of \( K \) over the training set.

### Formulation of SLLC

\[
\min_{M \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - y_i \frac{1}{|R|} \sum_{x_j \in R} y_j K_M(x_i, x_j) \right] + \beta \|M\|_F^2,
\]

where

\[ K_M(x, x') = x^T M x'. \]
Properties of SLLC

SLLC has a number of desirable properties:

- SLLC optimizes a link between the quality of the metric and the quality of the linear classifier.
- Unlike classic algorithms, which rely on pair or triplet-based constraints, SLLC satisfies constraints that are defined over an average of similarity scores.
- SLLC has only one constraint per training example, instead of one for each pair or triplet.
- We can derive consistency guarantees on the learned similarity.
Deriving consistency guarantees
Consistency guarantees for the learned metric: uniform stability

Definition (Uniform stability for metric learning)
A learning algorithm $\mathcal{A}$ has a **uniform stability** in $\kappa/n$, where $\kappa > 0$, if

$$\forall (T, x), \forall i, \sup_{x_1, x_2} |\ell(\mathcal{A}_T, x_1, x_2) - \ell(\mathcal{A}_{T^{i,x}}, x_1, x_2)| \leq \frac{\kappa}{n},$$

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

Theorem (Uniform stability bound)
For any algorithm $\mathcal{A}$ with uniform stability $\kappa/n$, with probability $1 - \delta$ over the random sample $T$, we have:

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

where $B$ is a problem-dependent constant.
Consistency and Generalization Guarantees

Stability of SLLC

Formulation of SLLC

\[
\min_{M \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - y_i \frac{1}{\gamma |R|} \sum_{x_j \in R} y_j K_M(x_i, x_j) \right] + \beta \|M\|_F^2,
\]

where

\[K_M(x, x') = x^T M x'.\]

Lemma

Let \( n \) and \(|R|\) be the number of training examples and reasonable points respectively, \(|R| = \hat{\tau} n \) with \( \hat{\tau} \in ]0, 1] \). SLLC has a uniform stability in \( \frac{\kappa}{n} \) with

\[\kappa = \frac{1}{\gamma} \left( \frac{1}{\beta \gamma} + \frac{2}{\hat{\tau}} \right),\]

where \( \beta \) is the regularization parameter and \( \gamma \) the margin.
Consistency and Generalization Guarantees

Consistency guarantees of SLLC

Theorem

Let $\gamma > 0$, $\delta > 0$ and $n_T > 1$. With probability at least $1 - \delta$, for any model $M$ learned with SLLC, we have:

$$
\epsilon \leq \hat{\epsilon} + \frac{1}{n} \left( \frac{1}{\gamma} \left( \frac{1}{\beta \gamma} + \frac{2}{\hat{\tau}} \right) \right) + \left( \frac{1}{\gamma} \left( \frac{1}{\beta \gamma} + \frac{2}{\hat{\tau}} \right) + 1 \right) \sqrt{\frac{\ln 1/\delta}{2n}}
$$

where:

- $\hat{\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \left[ 1 - y_i \frac{1}{|R|} \sum_{k=1}^{|R|} y_k K_M(x_i, x_k) \right]_+.$
- $\epsilon = \mathbb{E}_{(x_i, y_i) \sim P} \left[ 1 - y_i \frac{1}{|R|} \sum_{k=1}^{|R|} y_k K_M(x_i, x_k) \right]_+.$
Experimental Results

Comparison between a kernelized version (using a KPCA) of SLLC and:

- Standard bilinear similarity.
- LMNN
- LMNN KPCA
- ITML
- ITML KPCA
### Experiments with linear classifiers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Breast</th>
<th>Iono.</th>
<th>Rings</th>
<th>Pima</th>
<th>Splice</th>
<th>Svmguide1</th>
<th>Cod-RNA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$K_i$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.57</td>
<td>89.81</td>
<td>100.00</td>
<td>75.62</td>
<td>83.86</td>
<td>96.95</td>
<td>95.91</td>
</tr>
<tr>
<td></td>
<td>20.39</td>
<td>52.93</td>
<td>18.20</td>
<td>25.93</td>
<td>362</td>
<td>64</td>
<td>557</td>
</tr>
<tr>
<td><strong>SLLC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.90</td>
<td>93.25</td>
<td>100.00</td>
<td>75.94</td>
<td>87.36</td>
<td>96.55</td>
<td>94.08</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td><strong>LMNN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.81</td>
<td>90.21</td>
<td>100.00</td>
<td>75.15</td>
<td>85.61</td>
<td>95.80</td>
<td>88.40</td>
</tr>
<tr>
<td></td>
<td>9.98</td>
<td>13.30</td>
<td>18.04</td>
<td>69.71</td>
<td>315</td>
<td>157</td>
<td>61</td>
</tr>
<tr>
<td><strong>LMNN KPCA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.01</td>
<td>86.12</td>
<td>100.00</td>
<td>74.92</td>
<td>86.85</td>
<td>96.53</td>
<td>95.15</td>
</tr>
<tr>
<td></td>
<td>8.46</td>
<td>9.96</td>
<td>8.73</td>
<td>22.20</td>
<td>156</td>
<td>82</td>
<td>591</td>
</tr>
<tr>
<td><strong>ITML</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.80</td>
<td>92.09</td>
<td>100.00</td>
<td>75.25</td>
<td>81.47</td>
<td>96.70</td>
<td>95.06</td>
</tr>
<tr>
<td></td>
<td>9.79</td>
<td>9.51</td>
<td>17.85</td>
<td>56.22</td>
<td>377</td>
<td>49</td>
<td>164</td>
</tr>
<tr>
<td><strong>ITML KPCA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96.23</td>
<td>93.05</td>
<td>100.00</td>
<td>75.25</td>
<td>85.29</td>
<td>96.55</td>
<td>95.14</td>
</tr>
<tr>
<td></td>
<td>17.17</td>
<td>18.01</td>
<td>15.21</td>
<td>16.40</td>
<td>287</td>
<td>89</td>
<td>206</td>
</tr>
</tbody>
</table>
Conclusion and Perspectives

Conclusion

- New metric learning algorithm with theoretical guarantees:
  - on the metric itself (using Uniform Stability).
  - on the learning algorithm making use of it (theory of good similarities).
- SLLC is robust to overfitting because of constraints based on an average of similarity scores.
- Good results with very spare linear classifiers.

Perspectives

- Full kernelization of SLLC.
- Adaptation of Balcan et al’s framework to local metrics for local classifiers (like kNN).