A tutorial on Metric Learning with some recent advances

Nakul Verma
HHMI
The Machine Learning Pipeline

Some interesting phenomenon

Collect various measurements of observations

Preprocess: data cleaning/curation, and design a more useful representation

Do machine learning: SVMs, nearest neighbors, perceptrons, decision trees, random forests, etc.

Try to interpret the results…
Discover key factors in data
Encode higher order interactions
Provide simple description of complicated phenomenon

Hand design representations can only get you so far…
perhaps learn a good representation?

This study has created several specialized subfields in machine learning:
Manifold learning, Metric Learning, Deep learning
So What is this Metric Learning?

A type of **mechanism to combine features** to effectively compare observations.

**Unsupervised**
- Principal Component Analysis
- Random Projections
- IsoMap
- Locally Linear Embeddings
- Laplacian Eigenmaps
- Stochastic Neighbor Embedding

**Supervised**
- Mahalanobis Metric for Clustering
- Large Margin Nearest Neighbor
- Neighborhood Component Analysis
- Information Theoretic Metric Learning
How to compare observations?

\[ \rho(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2} \]

\[ = \sqrt{\left( \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_d \end{array} \right) - \left( \begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_d \end{array} \right) \cdot \left( \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_d \end{array} \right) - \left( \begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_d \end{array} \right) \right) \]

\[ = \sqrt{(x - y)\mathsf{T}(x - y)} \]
But...

Not all features are created equal.

Often some features are noisy or uninformative.

A priori, we don’t know which features are relevant for the prediction task at hand.

\[
\rho(x, y) = \sqrt{w_1(x_1 - y_1) + w_2(x_2 - y_2)}
\]

\[
= \sqrt{W \left( \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix} \right) \cdot W \left( \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix} \right)}
\]

\[
= \sqrt{[W(x - y)]^T [W(x - y)]} = \sqrt{(x - y)^T [W^T W](x - y)} = \sqrt{(x - y)^T M(x - y)}
\]
So what is metric learning?

Given data of interest, *learn a metric* \( M \), which helps in the prediction task.

\[
\rho_M(x, y) = \sqrt{(x - y)^\top M (x - y)}
\]

How?

Want:

Given some annotated data, want to find an \( M \) such that examples from the same class get small distance than examples from opposite class.

So:

Create an appropriate optimization problem and optimize for \( M \)!
The basic optimization

\[ \rho_M(x, y) = \sqrt{(x - y)^T M (x - y)} \]

Attempt: Let’s create two sets of pairs: similar set \( S \), dissimilar set \( D \).

want \( M \) such that:
\[ \rho_M(x, x') \quad \text{large, for } (x, x') \in D \]
\[ \rho_M(x, x') \quad \text{small, for } (x, x') \in S \]

Create cost/energy function:
\[ \Psi(M) = \lambda \sum_{(x, x') \in S} \rho_M^2(x, x') - (1 - \lambda) \sum_{(x, x') \in D} \rho_M^2(x, x') \]

Minimize \( \Psi(M) \) with respect to \( M \)!
Detour: How do we minimize?

Its an optimization problem!
- Take the gradient
- Find the stationary points

Things to consider:
- There are constraints
- The function is high dimensional

Of course, some certain constraint optimization problems are easier to minimize than others
The basic optimization

Attempt: Let’s create two sets of pairs: similar set $S$, dissimilar set $D$.

want $M$ such that: \[ \rho_M(x, x') \text{ large, for } (x, x') \in D \]
\[ \rho_M(x, x') \text{ small, for } (x, x') \in S \]

Create cost/energy function: \[ \Psi(M) \]
\[ \Psi(M) = \lambda \sum_{(x, x') \in S} \rho_M^2(x, x') - (1 - \lambda) \sum_{(x, x') \in D} \rho_M^2(x, x') \]

Minimize \( \Psi(M) \) with respect to $M$!
A Variation...

maximize$_M$ \[ \sum_{(x,x') \in D} \rho_M^2(x, x') \]

constraint: \[ \sum_{(x,x') \in S} \rho_M^2(x, x') \leq 1 \]

\[ M \in \text{PSD} \]

Advantages:
- Problem formulation is convex, so efficiently solvable!
- Tight convex clusters, can help in clustering!

Recall:
\[ M = W^T W \]

Xing, Ng, Jordan, Russell, *NIPS 2002.*
Empirical performance

Left to right: $k$-means, $k$-means+diag MMC, $k$-means + full MMC.
Another Interesting Formulation...

\[
\Psi_{\text{pull}}(M) = \sum_{i,j(i)} \rho_M^2(x_i, x_j)
\]

\[
\Psi_{\text{push}}(M) = \sum_{i,j(i),l(i,j)} 1 + \rho_M^2(x_i, x_j) - \rho_M^2(x_i, x_l)
\]

\[
\Psi(M) = \lambda \, \Psi_{\text{pull}}(M) + (1 - \lambda) \, \Psi_{\text{push}}(M)
\]

Advantages:
- Local constraints, so directly improves nearest neighbor quality!

Empirical performance

Query

After learning

Original metric
## Empirical performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>k-NN best</th>
<th>LMNNN best</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>mnist</td>
<td>2.12</td>
<td>1.18</td>
<td>1.20</td>
</tr>
<tr>
<td>letters</td>
<td>4.63</td>
<td>2.67</td>
<td>3.21</td>
</tr>
<tr>
<td>isolet</td>
<td>5.90</td>
<td>3.40</td>
<td>3.40</td>
</tr>
<tr>
<td>yfaces</td>
<td>4.80</td>
<td>4.05</td>
<td>15.22</td>
</tr>
<tr>
<td>balance</td>
<td>10.82</td>
<td>5.86</td>
<td>1.92</td>
</tr>
<tr>
<td>wine</td>
<td>2.17</td>
<td>2.11</td>
<td>22.24</td>
</tr>
<tr>
<td>iris</td>
<td>4.00</td>
<td>3.68</td>
<td>3.45</td>
</tr>
</tbody>
</table>
Metric Learning for Multi-class Classification

**Observation:**
Categories in multiclass data are often part of a underlying *semantic taxonomy*.

**Goal:**
To learn *distance metrics* that leverage the class taxonomy to yield good classification performance.

\[
\begin{align*}
\mathbf{M}_{\text{owl}} &= M_1 + M_2 + M_4 \\
\mathbf{M}_{\text{horse}} &= M_1 + M_3 + M_6
\end{align*}
\]
Metric Learning for Multi-class Classification

Given a query, define its affinity to a class:  
\[ f(x_q; y) := \sum_{x \in \mathcal{N}_y(x_q)} \rho(x_q, x; M_y) \]

So, putting it in probabilistic framework:

\[ p(y|x, M_1, \ldots, M_T) := \frac{\exp(-f(x; y, M_y))}{\sum_{\tilde{y}} \exp(-f(x; \tilde{y}, M_{\tilde{y}}))} \]

Now, given training samples:  
\((x_1, y_1), \ldots, (x_n, y_n)\)

we obtain a good set of metrics  
\(M_1, \ldots, M_T\)  
by maximizing:

\[ \mathcal{L}(M_1, \ldots, M_T) := \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|x_i; M_1, \ldots, M_T) - \frac{\lambda}{2} \sum_t \text{trace}(M_t^T M_t) \]

Observations:

• Optimization is jointly convex.
• Geometrically, the likelihood is maximized by: pulling together the neighbors belonging to the same class, while pushing away the neighbors from different class.

V., Mahajan, Sellamanikam, Nair, CVPR 2012.
Empirical performance

20 Newsgroup dataset – 20 classes, with 20k articles.
Empirical performance

Performance (accuracy)

<table>
<thead>
<tr>
<th>20 Newsgroups</th>
<th>SVM</th>
<th>Euclid</th>
<th>Flat</th>
<th>Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data (16k)</td>
<td>0.86</td>
<td>0.80</td>
<td>0.87</td>
<td>0.94</td>
</tr>
<tr>
<td>Test Data (4k)</td>
<td>0.80</td>
<td>0.79</td>
<td>0.80</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Empirical performance

Visualizations:

representation in 'Recreation'

- automobile
- motorcycle
- baseball
- hockey

- rec
  - motorcycle
  - sport
  - auto
  - baseball
  - hockey
Empirical performance

Visualizations:

- rec
- sci
- motorcycle
- sport
- auto
- baseball
- hockey

Representation in 'Science'

Legend:
- automobile
- motorcycle
- baseball
- hockey
Empirical performance

Visualizations:

- comp
- rec
- sci
- religion

- IBM
- auto
- crypt
- misc

Representation in 'root'

- IBM (comp)
- automobile (rec)
- crypt (sci)
- misc (religion)
Empirical performance

Visualizations:

- sci
  - crypt
  - elec
  - med
  - space

Representation in 'Root'

- crypt (sci)
- elec (sci)
- med (sci)
- space (sci)
Empirical performance

[Graph showing 0-1 classification accuracy for different classes and models, comparing the absolute improvement over AggkNN.]
Metric Learning for Information Retrieval

**Problem:**
Information retrieval: find most relevant examples for a given query.

**Goal:**
Learn *distance metric* that can rank the examples in a database effectively.

**Observations:**
Output has a structure (ranking) associated with it.

\[
q \in \mathcal{X} \quad y \in \mathcal{Y} \quad y_q^* \in \mathcal{Y} \quad \psi(q, y)
\]

\[
\text{(input space) \quad (output space) \quad (optimal output) \quad (joint feature space)}
\]

\[
\text{minimize}_w \quad \sum_{q \in \mathcal{X}} \xi_q + \lambda \ \text{reg}(w)
\]

\[
\langle w, \psi(q, y_q^*) \rangle \quad \geq \quad \langle w, \psi(q, y) \rangle + \Delta(y_q^*, y) \quad - \quad \xi_q
\]

\[
\text{score(good ranking) \quad score(bad ranking) \quad loss(bad ranking)}
\]
Feature representation for rankings

\[
\psi(q, y) = \sum_{i \in \mathcal{X}_q^+} \sum_{j \in \mathcal{X}_q^-} a_{i,j} \frac{\phi(q, i) - \phi(q, j)}{|\mathcal{X}_q^+||\mathcal{X}_q^-|}
\]

\[
a_{i,j} = \begin{cases} 
+1 & \text{if } i \text{ before } j \\
-1 & \text{if } i \text{ after } j
\end{cases}
\]

\[
\phi(i, j) = (q - i)(q - i)^T
\]

why does this work? \[
\rho^2_M(x, y) = (x - y)^T M (x - y) = \langle M, (x - y)(x - y)^T \rangle_F
\]

minimize \[
\sum_{q \in \mathcal{X}} \xi_q + \lambda \text{ trace}(M)
\]

\[
\langle M, \psi(q, y^*_q) \rangle_F \geq \langle M, \psi(q, y) \rangle_F + \Delta(y^*_q, y) - \xi_q
\]

score(good ranking) \quad score(bad ranking) \quad loss(bad ranking)

McFee, Lanckriet, ICML 2010.
Empirical performance

eHarmony dataset: ~250k users, ~450k matchings, feature representation in R

<table>
<thead>
<tr>
<th></th>
<th>eHarmony</th>
<th>SVM-MAP</th>
<th>Euclidean</th>
<th>MLR-MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.614</td>
<td>0.522</td>
<td></td>
<td>0.624</td>
</tr>
<tr>
<td>MAP</td>
<td>0.447</td>
<td>0.394</td>
<td></td>
<td>0.453</td>
</tr>
<tr>
<td>MRR</td>
<td>0.467</td>
<td>0.414</td>
<td></td>
<td>0.474</td>
</tr>
</tbody>
</table>
The “Theory” of Metric Learning

How hard/easy is it to learn $M$ as a function of key properties of data?
- Presence of uninformative features or noisy features?
- How does dimension plays into the sample complexity of learning?

Key result (work in progress)

**Theorem:** For all data distributions in $\mathbb{R}^D$, given $m$ random samples from it:

$$
err(M^*) \leq err(\hat{M}_m) + O(D/\sqrt{m})
$$

**Theorem:** for all data distributions in $\mathbb{R}^D$ with $d$ relevant features, given $m$ random samples from it:

$$
err(M^*) \leq err(\hat{M}_m) + O(d \log D/\sqrt{m})
$$

Summary

- **Metric Learning:**
  A powerful technique to combine features for effective comparison between observations.

- The basic technique has been extended for multiple learning problems
  Large multi-class classification, information retrieval, multi-task learning, domain adaptation, semi-supervised learning

- **Interesting questions:**
  Adapting to changing data, account for multiple ways to compare data, incorporating more sophisticated structure/geometry into account.
Thank You!

SFML-MG (Tony and David)

Flurry: for hosting the event

The Audience: for patiently listening!