Local Distance Functions

MISC Reading Group
November 2011

Santosh Divvala
What is this object?
‘Standard’ Approach

Model

Training Data

“Bike-Riding Mower”
‘Standard’ Approach

Training Data

Adaboost, SVM, LR,…

“Bike-Riding Mower”
Approach

Nearest Neighbor

“Bike-Riding Mower”
Approach

Nearest Neighbor

Choice of the Distance Function

“Bike-Riding Mower”
Nearest Neighbor

Choice of the Distance Function
LMNN (ICCV’07)

SVM-KNN (CVPR’07)

Similarity Metrics (PAMI’09)

Local Metric Learning (NIPS’07)

LDA-NN (CVPR’08)

Exemplars (CVPR’08)

NBNN (CVPR’08)

Metric Learning (ICCV’07)

LM-PCA (NIPS’07)

Similarity Metrics (ICCV’09)
Local Distance Functions: A Taxonomy, New Algorithms and an Evaluation
PAMI 2011

Similarity Metrics (ICCV'09)

Exemplars (CVPR'08)

Local Metric Learning (NIPS'07)

LMNN (ICCV'07)

NBNN (CVPR'08)

SVM-KNN (CVPR'07)

Local Metric Learning (NIPS'07)

LDA-NN (CVPR'08)

SimMetric (ICCV'09)
What is a distance function?
Lets set the stage...

• K-dimensional feature space $\mathbb{R}^k$
• Training points $x_i$, test point $x$
• Define an operation, $\mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$

$$\text{Dist}_M(x,x_i) = (x_i - x)^T M (x_i - x)$$

– where $M$ is a $k \times k$ symmetric, positive definite
– $D(x_i,x_j) + D(x_j,x_k) > D(x_i,x_k); \ D(x_i,x_j) = D(x_j,x_i); \ D(x_i,x_i) = 0$
– E.g., Euclidean distance function $L_2$

• Why?
Geodesic Distance

- Metric Tensor $MT(x_i)$
  - Define a metric $M$ at each $x_i$
  - Allows defining distances on a manifold

\[
\text{Length}(c(\lambda)) = \int_0^1 \left[ \frac{dc}{d\lambda} \right]^T MT(c(\lambda)) \frac{dc}{d\lambda} \, d\lambda
\]

\[
(x_i - x)^T M (x_i - x)
\]

\[
(c(\lambda_2) - c(\lambda_1))^T MT(c(\lambda)) (c(\lambda_2) - c(\lambda_1))
\]
Geodesic distance

- Shortest curve between two points $x, x_i$ i.e.,
  \[ \text{min } \text{Length} \left( c(\lambda) \right) \]
  - where $c(0) = x$ and $c(1) = x_i$

\[ \text{Length}(c(\lambda)) = \int_{0}^{1} \left[ \frac{dc}{d\lambda} \right]^T MT(c(\lambda)) \frac{dc}{d\lambda} d\lambda \]
Computing Geodesics is hard

- Most local distance functions $\text{Dist}_M(x, x_i)$ are an approximation of the geodesic distance defined by a metric tensor.
Taxonomy Overview

- How
- When
- Where

Metric Learning
LDA
PCA
How to estimate $\text{Dist}_M(x, x_i)$

1. Dimensionality Reduction
   a) PCA (unsupervised)

$M_{\text{PCA}} = P^T P$, where $P$ is the PCA projection matrix

(figure from http://www.nlpca.de/pca.html)
How to estimate $\text{Dist}_M(x,x_i)$

1. Dimensionality Reduction
   b) LDA (supervised)

$$M_{\text{LDA}} = V^T V,$$
where $V$ is the LDA basis matrix

(figure from http://www.dtreg.com/lda.htm)
How to estimate $\text{Dist}_M(x, x_i)$

2. Metric Learning

$M$ such that $L(M)$ is minimized

$$L(M) = \sum_{\{ij\} \in \text{Targ}} w_i \text{Dist}_M(x_i, x_j) +$$

$C \sum_{\{ijk\} \in \text{Imp}} w_i h(\text{Dist}_M(x_i, x_k) - \text{Dist}_M(x_i, x_j)).$ (LMNN)
How to estimate $\text{Dist}_M(x,x_i)$

2. Metric Learning
   - Common problem is to define $\text{Targ}(x_i)$
   - Use global metric e.g., $L_2$ as initialization

(figure from Malisiewicz & Efros, 2008)
How to estimate $\text{Dist}_M(x,x_i)$

3. Hybrid
   - Combining the best of both worlds
   - LMNN is powerful to populate a shortlist from large training data
   - LDA is efficient to estimate a local metric
Taxonomy Overview

How

Metric Learning

When

Where

Global

Per-class

Per-exemplar

Test

PCA

LDA

How
Where to estimate $\operatorname{Dist}_M(x,x_i)$

1. Global
   - assume $MT(x)$ is constant over the whole space i.e., $MT(x_i) = M_{\text{global}}$
Where to estimate $\text{Dist}_M(x, x_i)$

2. Per-Class

- assume $MT(x)$ is constant for each class

$$\text{Dist}_{PC}(x, x_i) = (x_i - x)^T M_{\text{Per-Class}}^{\text{Class}(x_i)} (x_i - x)$$

- Calibration issue: $\text{Dist}_{PC}(x, x_i)$ vs $\text{Dist}_{PC}(x, x_k)$

$$L(\{M_c\}) = \sum_{\{ij\} \in \text{Targ}} \text{Dist}_{M_{y_i}}(x_i, x_j) +$$

$$C \sum_{\{ijk\} \in \text{Imp}} h(\text{Dist}_{M_{y_i}}(x_i, x_k) - \text{Dist}_{M_{y_i}}(x_i, x_j))$$
Where to estimate $\text{Dist}_M(x,x_i)$

3. Per-Exemplar

- assume $\text{MT}(x)$ is constant within the neighborhood($x_i$)

\[
\text{Dist}_{\text{PE}}(x, x_i) = (x_i - x)^T M_{\text{Per-Exemplar}}^{x_i} (x_i - x)
\]

- defining neighborhood($x_i$)
- calibration issue
Where to estimate $\text{Dist}_M(x,x_i)$

4. At test point

- $\text{MT}(x) = M^x_{\text{Lazy}}$

$$\text{Dist}_{\text{Lazy}}(x, x_i) = (x_i - x)^T M^x_{\text{Lazy}} (x_i - x)$$

- Find training points $x_i$ close to $x$ (using any $M_{\text{global}}$) for optimizing $L(M)$

$$L(M) = \sum_{\{ij\} \in \text{Targ}} w_i \text{Dist}_M(x_i, x_j) + \sum_{\{ijk\} \in \text{Imp}} w_i h(\text{Dist}_M(x_i, x_k) - \text{Dist}_M(x_i, x_j)).$$

(LMNN)
Taxonomy Overview

- Metric Learning
- LDA
- PCA

Where
- Global
- Per-class
- Per-exemplar
- Test

When
- Online
- Offline

How
When to estimate $\text{Dist}_M(x,x_i)$

1. Offline
   - $M_{\text{global}} < M_{\text{per-class}} < M_{\text{per-exemplar}}$
     (increasing order of space + time complexity)

2. Online
   - $M_{\text{lazy}}$
     (best suited for Google-scale datasets)
When to estimate $\text{Dist}_M(x,x_i)$

3. Interpolated
- Offline step: select reference points $x_{RP_i}$, compute $M_{RP_i}$
- Online: Given test point $x$, find reference points $x_{RP_i}$ close to $x$ (using any $M_{\text{global}}$)

$$MT_{RBF}(x) = \sum_{i=1}^{R} w_i(x) M_{RP_i}^{x_{RP_i}}$$

$$w_i(x) = \frac{e^{-\frac{1}{\sigma} \text{Dist}_{\text{Global}}(x,x_{RP_i})}}{\sum_{j=1}^{R} e^{-\frac{1}{\sigma} \text{Dist}_{\text{Global}}(x,x_{RP_j})}}$$

Assumption: Metric tensor varies smoothly & thus can be interpolated!
Results
Theorem

• “The optimal distance function for 1-NN is the probability that the pair of examples belong to different classes”

S. Mahamud and M. Hebert, CVPR 2003
Datasets

CMU MultiPIE  Caltech 101  MNIST
Implementation Details

• Candidate Pruning

Using top 20 samples suffices!
Implementation Details

- Indexing Structure

*Using KD Tree offers great speedup with minimal performance drop!*
LDA is best given its computational efficiency (easier to regularize)
Results (‘where’, using LDA)

1. Per-Class is best given its lower computational load
2. Why per-exemplar is poor: a) overfitting b) calibration
3. Why ‘Test’ works well?
Results (‘where’, using LDA)

1. Per-Class is best given its lower computational load
2. Why per-exemplar are poor: a) overfitting b) calibration
3. Why Test works well?
1. **LMNN** prone to overfitting in case of a few training samples
2. **Hybrid (LMNN+LDA)** best as reaps best of both worlds
Result (‘When’, given Hybrid)

If efficiency is vital, then use Interp!
Summary

Where
- Global
- Per-class
- Per-exemplar
- Test

When
- Online
- Offline

How
- LDA
- LMNN
- PCA
Thank You!

- How
  - LDA
  - LMNN
  - PCA

- Where
  - Global
    - LDA-NN (CVPR'08)
  - Per-class
    - SVM-KNN (CVPR'07)
  - Per-exemplar
    - NBNN (CVPR'08)
  - Similarity Metrics (PAMI'09)

- When
  - Online
    - Exemplars (CVPR'08)
  - Offline
    - LM-PCA (NIPS'07)
  - Test
    - LM-PCA (NIPS'07)
  - Metric Learning (ICCV'07)
In defense of the NN algorithm

Parametric

Adaboost, SVM, LR...

Training Data

“Bike-Riding Mower”

Non-Parametric

Nearest Neighbor

“Bike-Riding Mower”

Why?
Descriptor Quantization

Informative descriptors have low frequency, & thus high quantization error!
Image-to-Image vs. Image-to-Class

\[ KL(p_Q | p_1) = 17.54 \]
\[ KL(p_Q | p_2) = 18.20 \]
\[ KL(p_Q | p_3) = 14.56 \]
Naïve Bayes NN Algorithm

1. Compute descriptors $d_1, \ldots, d_n$ of the query image $Q$.
2. $\forall d_i \forall C$ compute the NN of $d_i$ in $C$: $\text{NN}_C(d_i)$.
3. $\hat{C} = \text{arg min}_C \sum_{i=1}^n \| d_i - \text{NN}_C(d_i) \|^2$. 
Results

• How does NBNN compare to other NN methods (on Caltech dataset)

<table>
<thead>
<tr>
<th>NN-based method</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPM NN Image [27]</td>
<td>42.1 ± 0.81%</td>
</tr>
<tr>
<td>GBDist NN Image [27]</td>
<td>45.2 ± 0.96%</td>
</tr>
<tr>
<td>GB Vote NN [3]</td>
<td>52%</td>
</tr>
<tr>
<td>SVM-KNN [30]</td>
<td>59.1 ± 0.56%</td>
</tr>
<tr>
<td>NBNN (1 Desc)</td>
<td>65.0 ± 1.14%</td>
</tr>
<tr>
<td>NBNN (5 Desc)</td>
<td>72.8 ± 0.39%</td>
</tr>
</tbody>
</table>
Results

- Comparison of NN vs Parametric methods
Analysis

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|}
\hline
 & No Quant. & With Quant. \\
\hline
“Image-to-Class” & 70.4\% & 50.4\% (-28.4\%) \\
\hline
“Image-to-Image” & 58.4\% (-17\%) & - \\
\hline
\end{tabular}
\end{table}

Both factors are crucial for good performance accuracy!
Need for ‘context’

• Key strength of bag-of-words methods
NBNN Kernel

1. Compute a set of features $X = \{x\}$.
2. $\forall x \forall c$ Compute the NN of $x$ in $c$: $NN^c(x)$, and its distance-to-class $d^c_x = ||x - NN^c(x)||^2$.
3. $\hat{c} = \operatorname{argmin}_{c \in C} \sum_{x \in X} d^c_x$.

1. Compute a set of features $X = \{x\}$.
2. $\forall x \forall c$ Compute the NN of $x$ in $c$: $NN^c(x)$, and its distance-to-class $d^c_x = ||x - NN^c(x)||^2$.
3. $\forall c \quad \Phi^c(X) = \sum_{x \in X} f(d^1_x, \ldots, d^{|C|}_x)$.
4. $\Phi(X) = [\Phi^1(X) \ldots \Phi^{|C|}(X)]^T$.
5. Repeat steps 1-4 for a second set of features $Y = \{y\}$.
6. $K(X, Y) = \Phi(X)^T \Phi(Y)$. 
How to estimate $\text{Dist}_M(x,x_i)$

1. Dimensionality Reduction
   b) LDA (supervised)

$M_{\text{LDA}} = V^T V$, where $V$ is the LDA basis matrix

$$
\arg \max_V \text{tr}(V^T \Sigma_B V) \quad \text{s.t.} \quad V^T (\Sigma_W + \lambda I) V = I \quad \text{(LDA)}
$$

$$
\Sigma_B = \frac{1}{C} \sum_j \mu_j \mu_j^T, \quad \Sigma_W = \frac{\sum_i w_i \bar{x}_i \bar{x}_i^T}{\sum_i w_i}, \quad \bar{x}_i = x_i - \mu y_i
$$

$$
\bar{\mu}_j = \mu_j - \frac{1}{C} \sum_j \mu_j, \quad \mu_j = \frac{\sum_{i:y_i=j} w_i x_i}{\sum_{i:y_i=j} w_i}
$$
Computing Geodesics is hard

• Most local distance functions $\text{Dist}_M(x, x_i)$ are an approximation of the geodesic distance defined by a metric tensor.

• For now, all you need to know is:

$$\text{Dist}_M(x, x_i) = (x_i - x)^T M (x_i - x)$$

$$\text{Dist}_{\text{Geo}}(x, x_i) = \min_{c(0)=x, c(1)=x_i} \int_0^1 \left[ \frac{dc}{d\lambda} \right]^T MT(c(\lambda)) \frac{dc}{d\lambda} d\lambda$$