Metric Learning

16 March 2010

Plan

Plan

1. Motivation

2. Metric Learning for Clustering with Constraints
   - Xing's Method

Plan

1. Motivation
2. Metric Learning for Clustering with Constraints
   - Xing's Method
3. Metric Learning for Classification
   - Neighbourhood Components Analysis (NCA)
   - Large Margin Nearest Neighbour (LMNN)

Plan

1. Motivation
2. Metric Learning for Clustering with Constraints
   - Xing's Method
3. Metric Learning for Classification
   - Neighbourhood Components Analysis (NCA)
   - Large Margin Nearest Neighbour (LMNN)
4. Conclusions

Outline

Introduction

- The notion of (dis-)similarity is fundamental for many data mining / machine learning algorithms, such as:
  - Instance-based supervised algorithms (e.g. kNN)
  - Clustering algorithms (e.g. k-means)
  - Kernel-based algorithms (e.g. SVM)
  - ...
- The traditional approach is to specify the (dis-)similarity a-priori (i.e. before learning)
- This lecture is on adapting (learning) the (dis-)similarities from training data

(Dis-)similarity Learning in Different Settings

Different forms of information to exploit while learning (dis-)similarity:

- Class labels for all the training instances (supervised setting)
- Some constraints on the data (we call it *side-information*)
  - Must-link (ML) and cannot-link (CL) instance level constraints
  - Only ML constraints
  - $\mathbf{x}_i$ is closer to $\mathbf{x}_j$ than to $\mathbf{x}_k$
  - ...
- No additional information, only the data itself (a.k.a. unsupervised dimensionality reduction or manifold learning)

The Focus in This Lecture

- We will learn Mahalanobis distance metric
- We will exploit information from
  - ML and CL constraints
  - Class labels

The subjects of the next lectures will be:

- Kernel learning
- Unsupervised metric learning
- Feature selection (strongly related with metric learning)

Mahalanobis Metric

- In this work we will learn the Mahalanobis (quadratic) metric parametrized by a symmetric, squared matrix $\mathbf{A}$:

$$d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_j)$$

  - $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^p$
  - $\mathbf{A} \in \mathbf{R}^{p \times p}$
- To keep $d_{\mathbf{A}}$ a valid (pseudo-)metric $\mathbf{A}$ has to be positive demi-definite ($\mathbf{A} \succeq 0$), i.e.
  - $\forall_{\mathbf{x} \in \mathbb{R}^p} \quad \mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$
  - equivalently, $\mathbf{A}$ has non-negative eigenvalues
- Generalizes squared Euclidean metric ($\mathbf{A} = \mathbf{I}$)
- Often $\mathbf{A}$ is the inverse of the data covariance matrix
- $\mathbf{A}$ can be constrained to be diagonal

Mahalanobis Metric $\Longleftrightarrow$ Linear Transformation

- We can rewrite $d_{\mathbf{A}}$ using the eigendecomposition of $\mathbf{A} = \mathbf{W}^T\mathbf{W}$ (why is it always possible?):

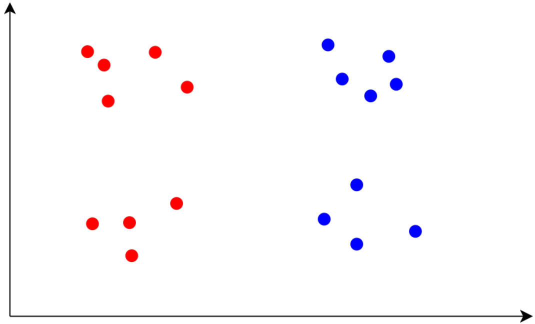
$$\begin{aligned} d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j)^T (\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j) \\ &= (\mathbf{x}'_i - \mathbf{x}'_j)^T (\mathbf{x}'_i - \mathbf{x}'_j) \end{aligned}$$

- $d_{\mathbf{A}}$ is equivalent to applying a simple (spherical) Euclidean metric to the points $\{\mathbf{x}'_i = \mathbf{W}\mathbf{x}_i\}$
- We also start directly with a non-squared $\mathbf{W} \in \mathbb{R}^{d \times p}$, $d < p$

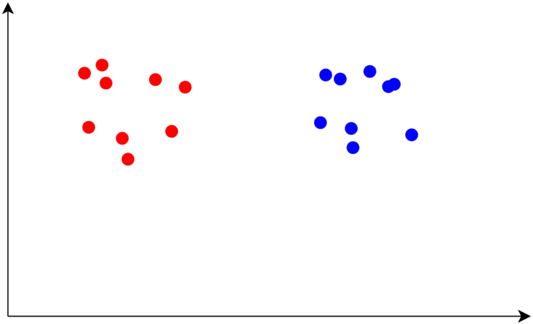
Mahalanobis Metric $\Longleftrightarrow$ Linear Transformation (II)

- $d_{\mathbf{A}}$ (and $d_{\mathbf{W}}$) rotates and scales input data
- Good $\mathbf{A}$ ($\mathbf{W}$) should amplify informative and squash non-informative dimensions
- Possible to define non-linear transformations by exploiting (non-linear) kernel functions - this is not considered in this lecture

Example: Original Data



Example: After Linear Transformation



Outline

Method of Xing et al.'02

Simple idea:

- Keep points in ML as close as possible
- To avoid a trivial solution put some constraint on points in CL

More formally:

$$\min_{\mathbf{A}} \quad \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \mathsf{ML}} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) \tag{1}$$

$$\text{s.t.} \quad \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \mathsf{CL}} d_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \tag{2}$$

$$\mathbf{A} \succeq 0 \tag{3}$$

- This optimization problem is *convex*
- Note that (1) is linear while (2) is not
- If we make (2) linear (i.e. use $d^2_{\mathbf{A}}$) then $\mathbf{A}$ will be of rank 1 (similar to LDA)

How to Solve It?

Equivalent formulation

$$\max_{\mathbf{A}} \quad f(\mathbf{A}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \mathsf{CL}} d_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) \tag{4}$$

$$\text{s.t.} \quad g(\mathbf{A}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \mathsf{ML}} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) \leq 1 \quad \longrightarrow C_1 \tag{5}$$

$$\mathbf{A} \succeq 0 \quad \longrightarrow C_2 \tag{6}$$

Main idea:

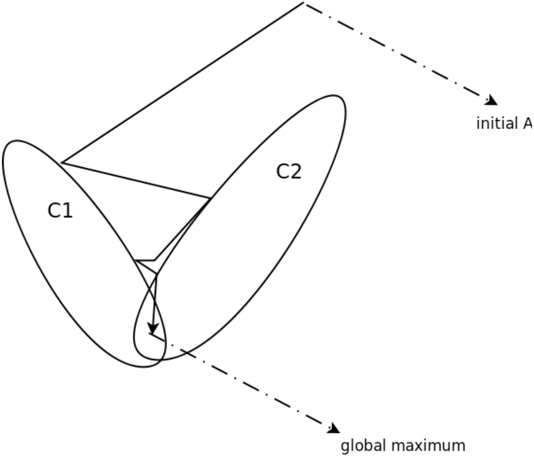
- To optimize (4): gradient ascent
- To satisfy constraints (5) and (6) use the iterative projection algorithm

Iterative Projections

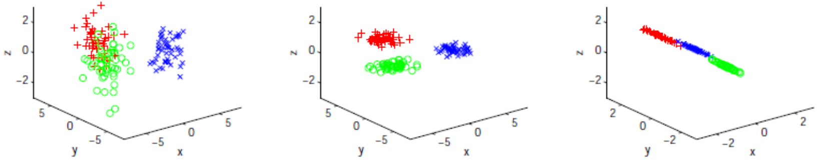
1: Initialize $\mathbf{A}$
2: **repeat**
3: **repeat**
4: $\mathbf{A} \leftarrow \mathrm{argmin}_{\mathbf{A}'}\{\|\mathbf{A}' - \mathbf{A}\|_F \ : \ \mathbf{A}' \in C_1\}$
5: $\mathbf{A} \leftarrow \mathrm{argmin}_{\mathbf{A}'}\{\|\mathbf{A}' - \mathbf{A}\|_F \ : \ \mathbf{A}' \in C_2\}$
6: **until** $\mathbf{A}$ converges
7: $\mathbf{A} \leftarrow \mathbf{A} + \nabla_{\mathbf{A}} f(\mathbf{A})$
8: **until** convergence

- First projection (Line 4)
  - (sparse) system of linear equations
- Second projection (Line 5)
  - $\mathbf{A} = \mathbf{X}^T \mathbf{\Lambda} \mathbf{X}$, $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$ - matrix of eigenvalues, $\mathbf{X}$ - matrix with eigenvectors as columns
  - $\mathbf{A}' = \mathbf{X}^T \mathbf{\Lambda}' \mathbf{X}$, $\mathbf{\Lambda}' = diag(\max(\lambda_1, 0), \ldots, \max(\lambda_n, 0))$
- $\| \cdot \|_F$: Frobenious norm over matrices

Iterative Projections: Schematic View



Toy Example



- First plot: original data (three clusters whose centroids differ only in the x and y directions)
- Second plot: $\mathbf{Wx}$ for diagonal $\mathbf{A}$
- Third plot: $\mathbf{Wx}$ for full $\mathbf{A}$

Example and Figure taken from Xing et al. '02

Xing's method - conclusions

Pros:

- Simple idea
- Improves performance of standard k-means
- Can be used in supervised setting

Cons:

- Complex to solve
- Iterative projections can be computationally expensive

Outline

1. Motivation
2. Metric Learning for Clustering with Constraints
   - Xing's Method
3. Metric Learning for Classification
   - Neighbourhood Components Analysis (NCA)
   - Large Margin Nearest Neighbour (LMNN)
4. Conclusions

kNN

We focus on kNN since:

- It is easy to implement
- Decision boundaries are non-linear
- Its complexity does not depend on the number of classes
- In the distance metric is carefully selected : state-of-the-art results
- Only one parameter to tune (k)
- Some theoretical guaranties of good performance

Neighbourhood Components Analysis (NCA)

- We want to improve the performance of kNN by learning metric
  - the metric should reduce generalization error (E_{G})
- We will approximate the generalization error by the error estimated using leave-one-out cross validation procedure (E_{LOO})
  - We assume $E_{\mathrm{LOO}} \approx E_{\mathrm{G}}$
  - The goal is then to keep E_{LOO} as small possible
- However, E_{LOO} is a difficult objective function to optimize with respect to the metric
  - *highly discontinuous* function of the metric

NCA - Stochastic Neighbourhood Selection

Main idea:

1. For each instance select the neighbours in a *stochastic manner*
2. Look for the *expected votes* for each class

More formally:

1. Each instance $\mathbf{x}_i$ select another instance $\mathbf{x}_j$ as its neighbour with probability p_{ij}
   - p_{ij} should be higher for points that are closer according to $d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)$

$$p_{ij} = \frac{\exp -d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{k \neq i} \exp -d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k)}, \; p_{ii} = 0$$

   - p_{ij} quickly reaches 0 expect for points that are close
2. The probability p_i that instance $\mathbf{x}_i$ will be correctly classified:

$$p_i = \sum_{\{j \,:\, y_i = y_j\}} p_{ij}$$

NCA - Cost Function

- The expected (soft) leave-one-out classification performance:

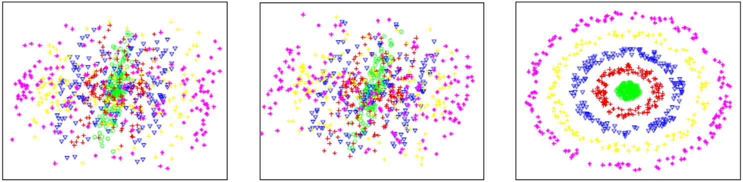
$$
\begin{aligned}
f(\mathbf{W}) &= \frac{1}{n}\sum_i p_i \\
&= \frac{1}{n}\sum_i \sum_{\{j \,:\, y_i = y_j\}} p_{ij} \\
&= \frac{1}{n}\sum_i \sum_{\{j \,:\, y_i = y_j\}} \frac{\exp -d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{k \neq i} \exp -d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k)}
\end{aligned}
$$

- $f(\mathbf{W})$ is continuous and differentiable, but not convex
- The goal is to find $\mathbf{W}$ that maximizes $f(\mathbf{W})$, i.e.

$$\max_{\mathbf{W}} f(\mathbf{W})$$

- We *do not need* the $\mathbf{W} \succeq 0$ constraint

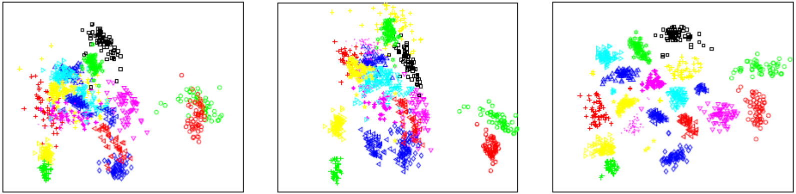
Toy Example



- Original data (in $\mathbb{R}^3$): Five classes organized in concentric rings in $\mathbb{R}^2$, the third dimension is an uncorrelated noise
- First plot: PCA
- Second plot: LDA
- Third plot: NCA (we learn $\mathbf{W} \in \mathbb{R}^{2\times3}$)

Example and Figure taken from Glodberger et al. '05

Real World Example



- Original data (in $\mathbb{R}^{560}$): images of 18 faces (100 images per face)
- First plot: PCA
- Second plot: LDA
- Third plot: NCA (we learn $\mathbf{W} \in \mathbb{R}^{2\times 560}$)

Example and Figure taken from Glodberger et al. '05

NCA - Conclusions

Pros

- Usually good performance
- No assumptions about the data such that unimodality, linear separability, etc.

Cons:

- Not convex
- Computational complexity
  - $O(n^2p^2)$ for full matrices $\mathbf{W}$
  - $O(n^2p)$ for rectangular "thin" matrices $\mathbf{W}$

Large Margin Nearest Neighbour (LMNN)

- For each learning instance set a number of closest instances (*target neighbours*) that share the same class
- The target neighbours are defined using standard Euclidean metric
- Similarly to NCA optimizes a surrogate function for LOO error

For each instance:

1. Pull target neighbours closer
2. Push away instances of different classes that are in the "neighbourhood" (i.e. *impostors*)
   - By a large margin

LMNN - Schematic View

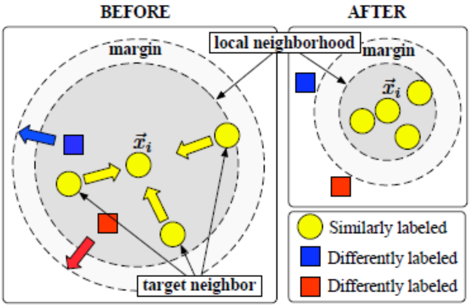
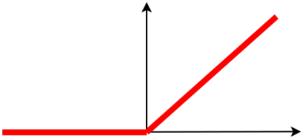


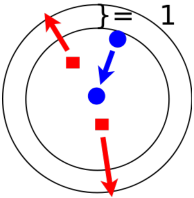
Figure taken from Weinberger et al. '06

Notation

- $\eta_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is a target neighbour of } \mathbf{x}_j, \\ 0 & \text{otherwise} \end{cases}$
  - need to be specified before learning
- $y_{ij} = \begin{cases} 1 & \text{if } y_i = y_j, \\ 0 & \text{otherwise} \end{cases}$
- $[k]_+ = \max(k, 0)$: hinge loss

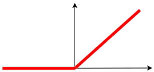


Formal Definition



$$\min_{\mathbf{A}}\{\mathrm{pull}(\mathbf{A}) + \mathrm{push}(\mathbf{A})\}$$

- $\mathrm{pull}(\mathbf{A}) = \sum_{ij} \eta_{ij} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j)$
  - pull together the neighbours
- $\mathrm{push}(\mathbf{A}) = \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_l)]_+$
  - Push away impostors by a margin of 1
  - hinge loss



Formal Definition (2)

$$
\begin{aligned}
\min_{\mathbf{A}} \quad & \sum_{ij} \eta_{ij} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) \\
& + \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_l)]_+ \\
s.t. \quad & \mathbf{A} \succeq 0
\end{aligned}
$$

Equivalent semi-definite program (convex):

$$
\begin{aligned}
\min_{\mathbf{A}} \quad & \sum_{ij} \eta_{ij} d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + \sum_{ijl} \eta_{ij}(1 - y_{il})\xi_{ijl} \\
s.t. \quad & d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_l) \geq 1 - \xi_{ijl} \\
& \xi_{ijl} \geq 0, \mathbf{A} \succeq 0
\end{aligned}
$$

Possible to solve it efficiently even though the number of constraints scales as $O(n^2 k)$

- Possible to solve problems with $n \approx 60,000$ in a reasonable time

LMNN - Some Empirical Results

Estimated error:

	MNIST	News	Isolet	Bal	Faces	Wine	Iris
p	784	30000	617	4	1178	13	4
p after PCA	164	200	172	-	30	-	-
n (train)	60000	16000	6238	445	280	126	106
Euclidean kNN	1.9	20.0	9.4	14.1	8.2	30.0	4.3
LMNN kNN	**1.2**	**11.0**	**4.7**	**10.0**	**0.3**	**1.1**	**3.5**

Results taken from Weinberger et al. '06

LMNN - Conclusions

Pros

- State-of-the-art metric learning results
- No assumptions about the data such that unimodality, linear separability, etc.
- Convex problem
- Can be solved very efficiently
- Because of the hinge-loss it is similar to SVM

Cons:

- Sensitive to outliers
- Target neighbours need to be specified a-priori
- Too many constraints - this might affect the quality of the solution

Outline

Metric Learning - Conclusions

Pros

- Unified framework for learning a linear transformation that better separates classes
- In most cases improves over standard Euclidean metric
- Often reported state-of-the-art results
- Can be non-linear (not discussed here)
- Can be very efficient

Cons:

- Scales as $O(p^2)$ ($O(p)$) - so data usually preprocessed by e.g. PCA
- Convex methods are less flexible; flexible methods and non-convex