TU München
Fakultät für Informatik
PD Dr. Rudolph Triebel
John Chiotellis

Machine Learning for Computer Vision
Summer term 2017

July 1, 2017
Topic: Metric Learning, Boosting

Exercise 1: Metric Learning

a) *Given a valid metric D_M, is D_M^2 also a metric? Why?*

No, D_M^2 is not a metric, as we cannot guarantee that the triangle inequality is preserved. We can easily find counterexamples. For instance, in the one-dimensional case with the euclidean metric we have:

$$D(0,2) \leq D(0,1) + D(1,2)$$

but

$$D^2(0,2) > D^2(0,1) + D^2(1,2).$$

We can also prove that the triangle inequality does not (always) hold. For arbitrary vectors $x, y, z \in \mathbb{R}^d$, we have:

$$\begin{aligned}
D_M^2(x,z) + D_M^2(z,y) &= (x-z)^T M(x-z) + (z-y)^T(z-y) \\
&= x^T Mx + z^T Mz - 2x^T Mz + z^T Mz + y^T My - 2z^T My \\
&= x^T Mx + y^T My + 2z^T Mz - 2x^T Mz - 2z^T My \\
&= D_M^2(x,y) + 2x^T My + 2z^T Mz - 2x^T Mz - 2z^T My \\
&= D_M^2(x,y) + 2(x^T M(y-z) + z^T M(z-y)) \\
&= D_M^2(x,y) + 2(x^T M(y-z) - z^T M(y-z)) \\
&= D_M^2(x,y) + 2(x-z)^T M(y-z).
\end{aligned}$$

The term $(x-z)^T M(y-z)$ is an inner product of the vectors $x-z$ and $y-z$ in the new metric space:

$$(x-z)^T M(y-z) = ||x-z|| \cdot ||M(y-z)|| \cos(\angle(x-z, M(y-z))) = \gamma \cos\theta \qquad , \gamma > 0$$

Since the vectors x, y, z were arbitrary, $|\theta| \in [0, \pi]$. Therefore we have:

$$D_M^2(x,z) + D_M^2(z,y) \begin{cases} \geq D_M^2(x,y) & \text{if } |\theta| \in [0, \frac{\pi}{2}] \\ < D_M^2(x,y) & \text{if } |\theta| \in (\frac{\pi}{2}, \pi] \end{cases}$$

For $|\theta| \in (\frac{\pi}{2}, \pi]$ the triangle inequality is violated.

b) *Given a matrix X of n data points $x_i \in \mathbb{R}^d$, show how computing the eigendecomposition of the covariance of X is equivalent to computing the singular value decomposition of X .*

The covariance of X is a real symmetric semi-positive definite matrix $C = \frac{1}{n}X^TX$. Therefore we know it has an eigendecomposition of the form $C = \frac{1}{n}V\Lambda V^T$, where V contains the eigenvectors of C as columns and Λ contains the corresponding non-negative eigenvalues in the diagonal.
Assuming X has a singular value decomposition $X = U\Sigma W^T$, we can infer a decomposition for C as

$$C = \frac{1}{n}X^TX = \frac{1}{n}W\Sigma U^TU\Sigma W^T = \frac{1}{n}W\Sigma^2W^T$$

By inspection, we can see that the two decompositions are identical, with $W = V$ and $\Lambda = \Sigma^2$, namely the right singular vectors of X are the eigenvectors of C and the singular values of X are the eigenvalues of C squared.

c) *What is the difference between metric learning and kernel learning? When would you prefer to use one over the other?*

Metric learning methods attempt to find a linear embedding of a given set of input vectors, such that a clustering or classification objective is optimized in the transformed space. This embedding has usually fewer or at worst equal number of dimensions as the original input space.

In contrast, kernel methods attempt to implicitly find a higher-dimensional representation of the data, such that the classes can be linearly separated. This representation does not need to be found explicitly but is exploited by using only inner products of the high-dimensional features (kernel trick).

Therefore when the dataset size is larger than the dimensionality of the given inputs ($n > d$), it will be more efficient to work with a covariance-like matrix. Metric learning will try to find inherent object relationships in the essential dimensions and weight them appropriately. When the size of the dataset is smaller than the inputs dimensionality ($n < d$), kernel methods are a more appropriate choice, as it will be more efficient to work with the kernel matrix and more complex - non-linear - relationships can be expressed.

d) *In Neighborhood Component Analysis, we define a stochastic neighbor selection rule. The probability that a data point j is selected as neighbor of point i is given by:*

$$p_{ij} = \frac{\exp\{-||Lx_i - Lx_j||^2\}}{\sum_{k\neq i}\exp\{-||Lx_i - Lx_k||^2\}} \tag{1}$$

namely a softmax over the squared distances to all points in the transformed space. The goal is to maximize

$$f(L) = \sum_i \sum_{j\in C_i} p_{ij} \tag{2}$$

namely the probability that the neighbors that will be selected for each point i will belong to the same class C_i. Can you derive the gradient of $f(L)$?

Let us notate the denominator of p_{ij} as $Z(L) = \sum\limits_{k \neq i} \exp\{-||Lx_i - Lx_k||^2\}$.
We will make use of the fact that we can write squared distances as traces:

$$\begin{aligned} ||Lx_i - Lx_j||^2 &= (L(x_i - x_j))^T(L(x_i - x_j)) = (x_i - x_j)^T L^T L(x_i - x_j) \\ &= tr(L^T L(x_i - x_j)(x_i - x_j)^T) = tr(L^T L C_{ij}) \end{aligned}$$

where for convenience we defined $C_{ij} = (x_i - x_j)(x_i - x_j)^T$. And we will use the gradient w.r.t. L: $\nabla_L tr(L^T L C_{ij}) = 2LC_{ij}$.

First we can see that the gradient can go inside of the sums: $\nabla_L f(L) = \sum\limits_i \sum\limits_{j \in C_i} \nabla_L p_{ij}$.

Then we see that we have to apply the quotient rule, since L appears in both the nominator and denominator of p_{ij}:

$$\begin{aligned} \nabla_L p_{ij} &= \nabla_L \left(\frac{\exp\{-||Lx_i - Lx_j||^2\}}{Z(L)} \right) \\ &= \frac{\left(\nabla_L \exp\{-||Lx_i - Lx_j||^2\}\right) Z(L) - \left(\nabla_L Z(L)\right) \exp\{-||Lx_i - Lx_j||^2\}}{Z(L)^2} \end{aligned}$$

The gradient of the nominator is

$$\begin{aligned} \nabla_L \exp\{-||Lx_i - Lx_j||^2\} &= \exp\{-||Lx_i - Lx_j||^2\} \nabla_L(-||Lx_i - Lx_j||^2) \\ &= -\exp\{-||Lx_i - Lx_j||^2\} \nabla_L tr(L^T L C_{ij}) \\ &= -2\exp\{-||Lx_i - Lx_j||^2\} L C_{ij} \end{aligned}$$

The gradient of the denominator is similar:

$$\begin{aligned} \nabla_L Z(L) &= \nabla_L \sum_{k \neq i} \exp\{-||Lx_i - Lx_k||^2\} = \sum_{k \neq i} \nabla_L \exp\{-||Lx_i - Lx_k||^2\} \\ &= \sum_{k \neq i} \nabla_L \exp\{-||Lx_i - Lx_k||^2\} = -2L \sum_{k \neq i} \exp\{-||Lx_i - Lx_k||^2\} C_{ik} \end{aligned}$$

Therefore we have

$$\begin{aligned}
\nabla_L p_{ij} &= -2L \frac{\exp\{-||Lx_i - Lx_j||^2\} C_{ij} Z(L) - \sum_{k \neq i} \exp\{-||Lx_i - Lx_k||^2\} C_{ik} \exp\{-||Lx_i - Lx_j||^2\}}{Z(L)^2} \\
&= -2L \left(\frac{\exp\{-||Lx_i - Lx_j||^2\}}{Z(L)} C_{ij} - \sum_{k \neq i} \frac{\exp\{-||Lx_i - Lx_k||^2\}}{Z(L)} C_{ik} \frac{\exp\{-||Lx_i - Lx_j||^2\}}{Z(L)} \right) \\
&= -2L \left(p_{ij} C_{ij} - \sum_{k \neq i} p_{ik} C_{ik} p_{ij} \right) = 2L p_{ij} \left(\sum_{k \neq i} p_{ik} C_{ik} - C_{ij} \right)
\end{aligned}$$

And summing over all inputs again we get the total gradient

$$\begin{aligned}
\nabla_L f(L) &= \sum_i \sum_{j \in C_i} 2L p_{ij} \left(\sum_{k \neq i} p_{ik} C_{ik} - C_{ij} \right) & &= 2L \sum_i \sum_{j \in C_i} p_{ij} \left(\sum_{k \neq i} p_{ik} C_{ik} - C_{ij} \right) \\
&= 2L \sum_i \left(\sum_{j \in C_i} p_{ij} \sum_{k \neq i} p_{ik} C_{ik} - \sum_{j \in C_i} p_{ij} C_{ij} \right) & &= 2L \sum_i \left(p_i \sum_{k \neq i} p_{ik} C_{ik} - \sum_{j \in C_i} p_{ij} C_{ij} \right)
\end{aligned}$$

Exercise 2: Adaboost (Programming)

See code.