Lecture 26 Nov. 17 -2006

Data Visualization

STAT 442 / 890, CM 462

Lecture: Ali Ghodsi

1 Using Partial Distance Side Information

We will now discuss the use of the second kind of side information mentioned at the beginning of this chapter. Recall that in this case, we assume that we have prior side information that gives exact distances between some pairs of points in some space natural to the data. This type of side information was motivated by cases where no class structure is available, but some prior distance information can be obtained.

In cases like this, partial distance information can be used to inform our learned metric. Given a set of similarities in the form of pairs for which distances are known

$$S: \quad (x_i, x_j) \in \mathcal{S} \quad \text{if the target distance } d_{ij} \text{is known}$$

the following cost function is employed, which attempts to preserve the set of known distances

$$L(A) = \sum_{(x_i,x_j)\in\mathcal{S}} \left\| \|x_i - x_j\|_A^2 - d_{ij} \right\|^2$$

The optimization problem is then

$$\begin{aligned} &\min_A L(A) \\ \text{s.t.} \quad & A \succeq 0 \end{aligned}$$

A convenient form for this optimization problem is obtained using the same approach for quadratic terms used earlier.

$$\begin{aligned} L(A) &= \sum_{(x_i,x_j)\in\mathcal{S}} \left\| \|x_i - x_j\|_A^2 - d_{ij} \right\|^2 \\ &= \sum_{(x_i,x_j)\in\mathcal{S}} \left\| \mathrm{vec}(A)^T \mathrm{vec}((x_i - x_j)(x_i - x_j)^T) - d_{ij} \right\|^2 \\ &= \sum_{(x_i,x_j)\in\mathcal{S}} \left\| \mathrm{vec}(A)^T \mathrm{vec}(B_{ij}) - d_{ij} \right\|^2 \\ &= \sum_{(x_i,x_j)\in\mathcal{S}} \mathrm{vec}(A)^T \mathrm{vec}(B_{ij}) \mathrm{vec}(B_{ij})^T \mathrm{vec}(A) + d_{ij}^2 - 2d_{ij}\mathrm{vec}(A)^T \mathrm{vec}(B_{ij}) \end{aligned}$$

where $B_{ij} = (x_i - x_j)(x_i - x_j)^T$. Note that the d_{ij}^2 term in the above is a constant so it can be dropped for the purposes of minimization, and rewrite the loss as

$$\begin{aligned} L(A) &= \mathrm{vec}(A)^T \left[\sum_{(x_i,x_j)\in\mathcal{S}} \mathrm{vec}(B_{ij}) \mathrm{vec}(B_{ij})^T \mathrm{vec}(A) - 2 \sum_{(x_i,x_j)\in\mathcal{S}} d_{ij} \mathrm{vec}(B_{ij}) \right] \\ &= \mathrm{vec}(A)^T \left[Q\mathrm{vec}(A) - 2R \right] \end{aligned}$$

where $Q = \sum_{(x_i,x_j)\in\mathcal{S}} \mathrm{vec}(B_{ij})\mathrm{vec}(B_{ij})^T$ and $R = \sum_{(x_i,x_j)\in\mathcal{S}} d_{ij}\mathrm{vec}(B_{ij})$. This objective is still quadratic but a linear objective can be obtained via the Schur complement [1]. We will briefly outline this approach.

The Schur complement relates the positive semidefiniteness of the matrix on the left and the expression on the right by an if-and-only-if relation

$$\begin{bmatrix} X & Y \\ Y^T & Z \end{bmatrix} \succeq 0 \quad \Leftrightarrow \quad Z - Y^T X^{-1} Y \succeq 0$$

By decomposing $Q = S^T S$, a matrix can be constructed of the form

$$J = \begin{bmatrix} I & S\text{vec}(A) \\ (S\text{vec(A)})^T & 2\text{vec(A)}^T R + t \end{bmatrix}$$

and by the Schur complement, if $J \succeq 0$, then the following relation holds

$$2\text{vec}(A)^T R + t - \text{vec}(A)^T S^T S\text{vec}(A) \quad \geq \quad 0$$

Note that the left-hand side of this relation is scalar. So, as long as J is positive semidefinite, the scalar t is an upper bound on the loss

$$\begin{aligned} \text{vec}(A)^T S^T S\text{vec}(A) - 2\text{vec}(A)^T R &= \text{vec}(A)^T Q\text{vec}(A) - 2\text{vec}(A)^T R \\ &\leq t \end{aligned}$$

Therefore, minimizing t subject to $J \succeq 0$ also minimizes the objective. This produces the final optimization problem, which can be readily solved by standard semidefinite programming software

$$\begin{aligned} &\min_A t \\ \text{s.t.} \quad &A \succeq 0 \\ &J \succeq 0 \end{aligned}$$

2 Kernelizing Metric Learning

Not uncommonly, one needs to consider nonlinear transformations of data in order to apply learning algorithms. One efficient method for doing this is via a kernel that computes a similarity measure between any two data points. In this section, we show how a distance metric can be learned in the feature space implied by a kernel, allowing our use of side information to be extended to nonlinear mappings of the data.

Conceptually, the points are being mapped into a feature space by some nonlinear mapping $\Phi()$ and then a distance metric is learned in that space. Actually performing the mapping is typically undesirable (the feature vectors may have infinite dimension, for example), so we employ the well known *kernel trick,* using some kernel $K(x_i, x_j)$ that computes inner products between feature vectors without explicitly constructing them.

The squared distances in our objective have the form

$$(x_i - x_j)^T A (x_i - x_j)$$

Because A is positive semidefinite, it can be decomposed into $A = WW^T$. This W matrix can then be reexpressed as a linear combination of the data points, $W = X\beta$, via the kernel trick. Rewriting the squared distance

$$\begin{aligned}
(x_i - x_j)^T A (x_i - x_j) &= (x_i - x_j)^T W W^T (x_i - x_j) \\
&= (x_i - x_j)^T X \beta \beta^T X^T (x_i - x_j) \\
&= (x_i^T X - x_j^T X) \beta \beta^T (X^T x_i - X^T x_j) \\
&= (X^T x_i - X^T x_j)^T \mathcal{A} (X^T x_i - X^T x_j)
\end{aligned}$$

where $\mathcal{A} = \beta\beta^T$, we have now expressed the distance in terms of the matrix to be learned, $\mathcal{A}$, and inner products between data points, which can be computed via the kernel, K. The optimization of $\mathcal{A}$ then proceeds just as in the non-kernelized version presented earlier, with one additional constraint. The rank of $\mathcal{A}$ must be t because β is t by n and $\mathcal{A} =\beta\beta^T$. However, this constraint is problematic, and so we drop it during the optimization and then find a rank t approximation of $\mathcal{A}$ using singular value decomposition.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, New York, 2004.