Lecture 24 and 25 Nov. 13 and 15 -2006

Data Visualization

STAT 442 / 890, CM 462

Lecture: Ali Ghodsi

1 Learning a Metric from Class-Equivalence Side Information

Given a set of t points, $\{x_i\}_{i=1}^{t} \subseteq R^n$, we identify two kinds of class-related side information. The first is a set of pairs of similar or class-equivalent points (they belong to the same class)

$$S: \quad (x_i, x_j) \in \mathcal{S} \quad \text{if } x_i \text{ and } x_j \text{ are similar}$$

and the second is a set of dissimilar or class-inequivalent pairs (they belong to different classes)

$$O: \quad (x_i, x_j) \in \mathcal{O} \quad \text{if } x_i \text{ and } x_j \text{ are dissimilar}$$

We then wish to learn a matrix A that induces a distance metric $D^{(A)}$ over the points

$$D^{(A)}(x_i, x_j) = \|x_i - x_j\|_A = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

where $A \succeq 0$.

We define the following loss function, which, when minimized attempts to minimize the squared induced distance between similar points and maximize the squared induced distance between dissimilar points

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

The optimization problem then becomes

$$\begin{aligned} & \min_A L(A) \\ \text{s.t.} \quad & A \succeq 0 \\ & \mathrm{Tr}(A) = 1 \end{aligned} \tag{1}$$

The first constraint (positive semidefiniteness) ensures a valid Euclidean metric. The second constraint excludes the trivial solution where all distances are zero. The constant in this constraint is arbitrary and changing it simply scales the resulting space.

This objective will be optimized using standard semidefinite programming software and so it must be converted to a linear objective. Expanding the loss function

$$L(A) \quad = \sum_{(x_i,x_j)\in\mathcal{S}} (x_i - x_j)^T A (x_i - x_j) - \sum_{(x_i,x_j)\in\mathcal{O}} (x_i - x_j)^T A (x_i - x_j)$$

each squared distance term must be converted. We start by observing that $\mathrm{vec}(XYZ) = (Z^T \otimes X)\mathrm{vec}(Y)$, where vec() simply rearranges a matrix into a vector by concatenating columns and $\otimes$ is the Kronecker product. Note that $(x_i - x_j)^T A(x_i - x_j) = \mathrm{vec}((x_i - x_j)^T A(x_i - x_j))$ because the left-hand side is a scalar. Using this and the fact that $(a^T \otimes b^T) =$

$\text{vec}(ba^T)^T$, we can rewrite the squared distance terms as

$$\begin{aligned}
(x_i - x_j)^T A (x_i - x_j) &= \text{vec}((x_i - x_j)^T A (x_i - x_j)) \\
&= ((x_i - x_j)^T \otimes (x_i - x_j)^T)\text{vec}(A) \\
&= \text{vec}((x_i - x_j)(x_i - x_j)^T)^T \text{vec}(A) \\
&= \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T)
\end{aligned}$$

The linear loss function is then

$$\begin{aligned}
L(A) &= \sum_{(x_i,x_j)\in\mathcal{S}} \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) - \sum_{(x_i,x_j)\in\mathcal{O}} \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) \\
&= \text{vec}(A)^T \left[\sum_{(x_i,x_j)\in\mathcal{S}} \text{vec}((x_i - x_j)(x_i - x_j)^T) - \sum_{(x_i,x_j)\in\mathcal{O}} \text{vec}((x_i - x_j)(x_i - x_j)^T) \right]
\end{aligned}$$

This form, along with the two constraints from (1), can be readily submitted to an SDP solver to optimize the matrix A[1]. Aside from this convenient form, this formulation has other advantages over that used by Xing *et al.*, especially with respect to the side information we can convey. Xing *et al.* require at least one dissimilar pair in order to avoid the trivial solution where all distances are zero. The constraint on the trace that we employ means that do not place any restrictions on pairings. There can be only similar pairs, only dissimilar pairs, or any combination of the two, and the method will still avoid trivial solutions.

Furthermore, in the absence of specific information regarding dissimilarities, Xing *et al.* assume that all points not explicitly identified as similar are dissimilar. This information

[1] we use the MATLAB SDP solver SeDuMi [1]

may be misleading, forcing the algorithm to separate points that should in fact be similar. The formulation presented here allows one to specify only the side information one actually has, partitioning the pairings into similar, dissimilar, and unknown.

2 Multiple-Attribute Metric Learning with Class-Equivalence Side Information

In some cases, one may have more than one distinction to capture in data (e.g., glasses vs. no glasses **and** male vs. female). The method just presented can be extended to construct a distance metric using multiple sets of side information, each corresponding to a different criterion for similarity. Multiple metrics can be learned, one for each kind of side information, and then be combined to form a new distances metric.

Suppose there are k different sets of side information over the same set of points. Using the optimization described above, k transformations, $A_1, \cdots, A_k$ can be learned. From each A_i, the dominant eigenvector v_i can be taken and a new matrix assembled where each column is one of these eigenvectors

$$\bar{A} = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$$

This combined matrix defines a rank k linear subspace onto which we can project the data. Applying singular value decomposition to $\bar{A}$, one can form a matrix U from the k eigenvectors corresponding to the largest eigenvalues. The final distance transformation is then $\hat{A} = UU^T$, which is an orthonormal basis combining the distinctions drawn by each kind of side information.

References

[1] J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11/12(1-4):625–653, 1999.