Supervised Learning from Disparate Dissimilarity Information

Michael W. Trosset

Department of Statistics

Indiana University

This research was supported by a grant from the Office of Naval Research.

Supervised Learning

Consider the problem of constructing a classifier from n labelled objects. How we proceed may depend on the type of information that we possess about the objects.

If we can measure features, then we might perform linear discriminant analysis (LDA).

If we can compute pairwise inner products, then we might construct a support vector machine (SVM).

If we can compute pairwise dissimilarities, then we might classify using k nearest neighbors (kNN).

In fact, each possibility is available because we can convert information of one type to information of another type.

Euclidean Distances and Inner Products

A dissimilarity matrix $A = [a_{ij}]$ is a *Type-2 Euclidean distance matrix* (EDM-2) iff there exist $x_1, \ldots, x_n \in \Re^p$ such that $a_{ij} = \|x_i - x_j\|^2$. The smallest such p is the *embedding dimension* of the EDM-2.

A matrix $B = [b_{ij}]$ is an inner product matrix (IPM) iff there exist $x_1, \ldots, x_n \in \Re^p$ such that $b_{ij} = \langle x_i, x_j \rangle$ iff $B = XX^t$ iff $B \geq 0$. Such a B is centered iff $X^t e = 0$.

There is a linear equivalence between Type-2 EDMs and centered IPMs:

- If A is EDM-2, then
$$B = \tau(A) = -PAP/2$$
is a centered IPM, where $P = I - ee^t/n$.

- If $B = XX^t$, then
$$A = \kappa(B) = \mathsf{diag}(B)ee^t - 2B + ee^t\mathsf{diag}(B)$$
is EDM-2. Note that $a_{ij} = \|x_i - x_j\|^2$. If $X^t e = 0$, then $\tau(\kappa(B)) = B$.

Classical MDS

Suppose that we want to embed fallible dissimilarities $\Delta = [\delta_{ij}]$ in $\Re^d$.

If the dissimilarity matrix $\Delta_2 = [\delta_{ij}^2]$ is *not* EDM-2 with embedding dimension $\leq d$, then $B = \tau(\Delta_2)$ is not psd with rank $\leq d$. Hence, we cannot factor B to obtain a d-dimensional configuration of points. Classical MDS (CMDS) circumvents this difficulty by replacing B with $\bar{B}$, the nearest IPM with rank $\leq d$.

Thus, we might construct a classifier from Δ as follows:

1. Use CMDS to embed Δ in $\Re^d$, thereby obtaining X.

2. Perform LDA on X.

Anderson & Robinson. Generalized discriminant analysis based on distances. *Australian & New Zealand Journal of Statistics*, 45:301–318, 2003.

Trosset, Priebe, Park, Miller. Semisupervised learning from dissimilarity data. *Computational Statistics and Data Analysis*, 52:4643–4657, 2008.

Combining Dissimilarities

Traditional multivariate data analysis combines features by forming products:

$$\left.\begin{array}{c} X_1 \\ \vdots \\ X_m \end{array}\right\} \rightarrow [X_1|\cdots|X_m] = X$$

By the Pythagorean Theorem,

$$D_2(X) = D_2(X_1) + \cdots + D_2(X_m);$$

hence, if $\Delta^{(1)}, \ldots, \Delta^{(m)}$ are EDM-1, then the following are equivalent:

$$\left.\begin{array}{ccc} \Delta^{(1)} & \rightarrow & X_1 \\ & \vdots & \\ \Delta^{(m)} & \rightarrow & X_m \end{array}\right\} \rightarrow [X_1|\cdots|X_m] = X$$

$$\Delta_2^{(1)} + \cdots + \Delta_2^{(m)} = \Delta_2 \rightarrow X$$

A Kernel Perspective

If $Xe = 0$, $B = XX^t$, and $\Delta = D(X)$, then $B = \tau(\Delta_2)$ and $\Delta_2 = \kappa(B)$.

Because τ is linear,

$$\tau\left(\Delta_2^{(1)} + \cdots + \Delta_2^{(m)}\right) = \tau\left(\Delta_2\right)$$

and we can rewrite the preceding data fusion strategies as follows:

$$\left.\begin{array}{ccccccc} \Delta_2^{(1)} & \rightarrow & B_1 & \rightarrow & \sqrt{B_1} & = & X_1 \\ & \vdots & & \vdots & & \vdots & \\ \Delta_2^{(m)} & \rightarrow & B_m & \rightarrow & \sqrt{B_m} & = & X_m \end{array}\right\} \rightarrow [X_1|\cdots|X_m] = X$$

$$\left.\begin{array}{ccc} \Delta_2^{(1)} & \rightarrow & B_1 \\ & \vdots & \\ \Delta_2^{(m)} & \rightarrow & B_m \end{array}\right\} \rightarrow B_1 + \cdots + B_m = B \rightarrow \sqrt{B} = X$$

If the $\Delta^{(i)}$ are not EDM-1, then CMDS replaces each B_i with the nearest centered IPM of desired rank.

Summary

We can combine dissimilarities at any of three stages: the dissimilarities themselves, after transforming the dissimilarities to kernels, and after embedding the kernels.

- kNN operates on Δ. Combining $\Delta^{(1)}, \ldots, \Delta^{(m)}$ to obtain a suitable Δ can be viewed as a special case of *distance metric learning* (DML). DML is supervised: because kNN is scale dependent, DML must learn suitable weights for the $\Delta^{(i)}$. It is not obvious how to extract features directly from dissimilarities.

- SVMs operate on B. Combining $B_1, \ldots, B_m$ to obtain a suitable B is called *multiple kernel learning* (MKL). MKL is supervised: because SVMs are scale dependent, MKL must learn suitable weights for the B_i. It is possible to extract features directly from kernels.

- LDA operates on X, the construction of which is unsupervised. Because LDA is scale invariant, it does not matter if the $\Delta^{(i)}$ are measured on different scales. A variety of methods for feature selection are readily available, e.g.,

  McHenry. Variable selection in multivariate analysis. *Applied Statistics*, 27(3):291–296, 1978.

Combining Dissimilarities for Nearest Neighbors

Let $y_1, \ldots, y_N$ denote the labels of N objects and let

$$\pi_{ij} = \left(1, \delta_{ij}^{(1)}, \ldots, \delta_{ij}^{(m)}\right)^t.$$

The basic idea is to attempt to choose Δ^* to minimize the discontinuous objective function

$$f_1(\Delta) = \frac{1}{N} \sum_{(i,j,k) \in \mathcal{T}} I\left(\delta_{ij} > \delta_{ik}\right),$$

where

$$\mathcal{T} = \{(i,j,k) \ : \ j = \mathcal{N}(i),\, y_i \neq y_k\}$$

and $\mathcal{N}(i)$ is the nearest within-class neighbor of object i.

Restrict attention to certain quadratic forms, viz., $\delta_{ij} = \pi_{ij}^t A \pi_{ij}$ for A copositive. (A symmetric matrix A is copositive iff $v^t A V \geq 0$ for every v with nonnegative entries.)

Formulation

To obtain a tractable optimization problem:

1. Relax the copositive cone to

$$\mathcal{C} = \left\{ A \in \mathcal{S}_{m+1} \ : \ \pi_{ij}^t A \pi_{ij} \geq 0 \forall (i,j) \in \Pi \right\},$$

   where Π is a specified set of (i,j) pairs.

2. Replace f_1 with the continuous objective function

$$f_2(A) = \frac{1}{N} \sum_{(i,j,k) \in \mathcal{T}} \frac{1}{N - n_y} \left(\pi_{ij}^t A \pi_{ij} - \pi_{ik}^t A \pi_{ik} + 1 \right)_+ ,$$

   where n_y is the number of objects in class y.

3. Add a regularization term,

$$\frac{\lambda}{2} \|A\|_F^2,$$

   to f_2.

Solution

The resulting optimization problem,

$$\begin{aligned} \text{minimize} \quad & f_3(A) = f_2(A) + \frac{\lambda}{2}\|A\|_F^2 \\ \text{subject to} \quad & A \in \mathcal{C}, \end{aligned}$$

closely resembles two recent formulations of DML.

Weinberger & Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

Jin, Wang, Zhou. Regularized distance metric learning: theory and algorithm. *Advances in Neural Information Processing Systems*, 2009.

Solutions can be computed by a projected stochastic gradient method in the spirit of Pegasos.

Shalev-Shwartz, Singer, Srebo. Primal estimated sub-gradient solver for svm. *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

Multiple Kernel Learning

Suppose that $y_i = \pm 1$. Let $e = (1, \ldots, 1)^t \in \Re^N$. Given $\alpha \in \Re^N$, let

$$[y\alpha] = (y_1\alpha_1, \ldots, y_N\alpha_N)^t.$$

Then the hard-margin SVM for a fixed kernel matrix K can be obtained by solving the following quadratic program:

$$\begin{array}{rl} \text{minimize} & W(\alpha) = \alpha^t e - [y\alpha]^t K[y\alpha]/2 \\ \text{subject to} & [y\alpha]^t e = 0, \alpha_i \geq 0 \end{array}$$

Now let $\mathcal{K}$ denote a closed convex set of possible kernel matrices. For example, we might consider K that satisfy

$$K = \sum_{i=1}^{m} \mu_i B_i, \quad \mu_i \geq 0, \quad \text{trace}(K) \leq c$$

for a fixed constant c.

We can learn $K^* \in \mathcal{K}$ (and its corresponding SVM) by solving

$$\begin{aligned} \text{minimize} \quad & W(\alpha, K) = \alpha^t e - [y\alpha]^t K [y\alpha]/2 \\ \text{subject to} \quad & [y\alpha]^t e = 0, y_i \geq 0, \\ & K \in \mathcal{K}. \end{aligned}$$

Lanckriet, Cristianini, Bartlett, El Ghaoui, Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Multiple Kernel Basis Extraction

Castle, Tang, Trosset. Feature extraction for multiple kernel learning. Technical Report 09-04, Department of Statistics, Indiana University, December 2009.

The basic idea is to extract a set of orthogonal vectors, $\mathcal{V} = \{v_1, \ldots, v_r\}$, then submit the rank-one kernels $v_i v_i^t$ to MKL.

To extract vectors with predictive value, we rely on the notion of kernel alignment,

$$A(K_1, K_2) = \frac{\text{trace}(K_1, K_2)}{\sqrt{\text{trace}(K_1, K_1)}\sqrt{\text{trace}(K_2, K_2)}},$$

the cosine of the angle between K_1 and K_2.

Initially, for each $i = 1, \ldots, m$, compute the $q_{1,i}$ that maximizes $q^t B_i q / q^t q$ among all $q \neq 0$. Set v_1 equal to the $q_{1,i}$ for which $q_{1,i} q_{1,i}^t$ is maximally aligned with yy^t.

Subsequently, having extracted $v_1, \ldots, v_k$, compute each $q_{k+1,i}$ that maximizes $q^t B_i q / q^t q$ among the $q \neq 0$ that are orthogonal to $v_1, \ldots, v_k$. Set v_{k+1} equal to the $q_{k+1,i}$ for which $q_{k+1,i} q_{k+1,i}^t$ is maximally aligned with yy^t.

Let C_k denote the $k \times N$ matrix with rows $v_1^t, \ldots, v_k^t$ and set

$$P_k = I_N - C_k^t \left(C_k C_k^t\right)^{-1} C_k.$$

It turns out that $q_{k+1,i}$ is the eigenvector associated with the largest eigenvalue of PB_iP.