Prototype and Nearest Neighbor Classification

Stephen Vardeman

Analytics Iowa LLC

ISU Statistics and IMSE

Classification to the nearest instance

Linear methods of classification can reduce to classification of input $\mathbf{x}$ to the class with fitted mean "closest" in some appropriate sense. A related notion is to represent classes each by several "prototype" vectors of inputs, and to classify to the class with closest prototype.

So consider a K-class classification problem (where y takes values in $\mathcal{G} = \{1, 2, \ldots, K\}$) and suppose that the coordinates of input $\mathbf{x}$ have been standardized (according to training means and standard deviations). For each class $k = 1, 2, \ldots, K$ represent the class by prototypes

$$\mathbf{z}_{k1}, \mathbf{z}_{k2}, \ldots, \mathbf{z}_{kR}$$

belonging to $\Re^p$ and consider a classifier/predictor of the form

$$f(\mathbf{x}) = \arg\min_k \min_l \|\mathbf{x} - \mathbf{z}_{kl}\|$$

(classify to the class with a prototype closest to $\mathbf{x}$).

Selection of prototypes

The most obvious question in using such a rule is "How does one choose the prototypes?" One standard (admittedly ad hoc, but not unreasonable) method is to use the so-called "K-means (clustering) algorithm" one class at a time. (The "K" in the name of this algorithm has nothing to do with the number of classes in the present context. In fact, here the "K" naming the clustering algorithm is our present R, the number of prototypes used per class. And the point in applying the algorithm is not so much to see exactly how training vectors aggregate into "homogeneous" groups/ clusters as it is to find a few vectors to represent them.)

“*R*-means” selection of prototypes

For $T_k = \{\mathbf{x}_i$ with corresponding $y_i = k\}$ an "R "-means algorithm might proceed by

1. randomly selecting R different elements from T_k say

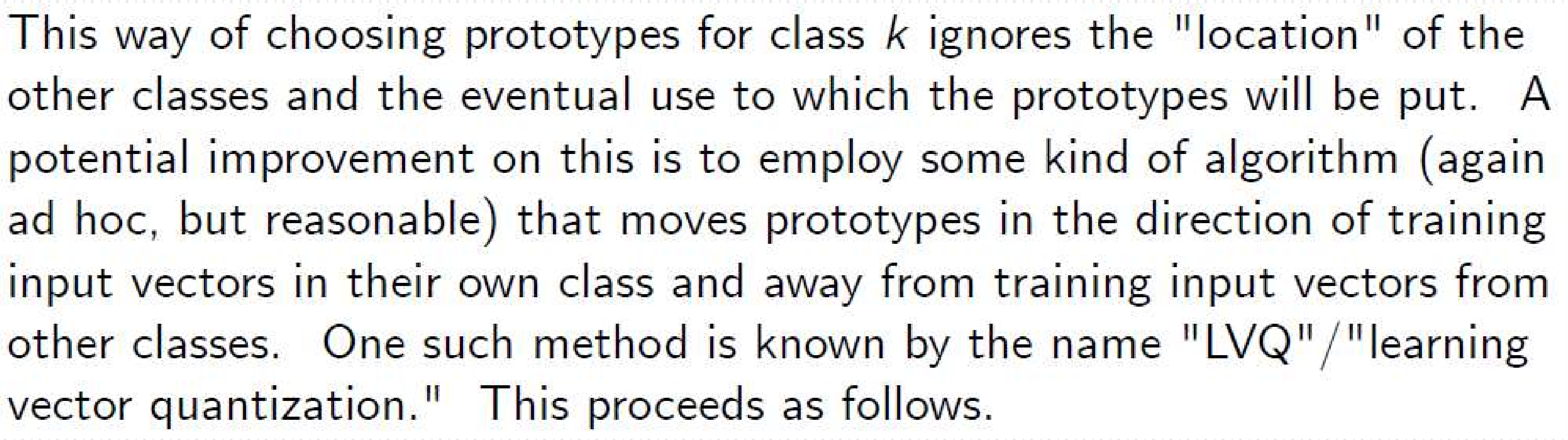
$$\mathbf{z}_{k1}^{(1)}, \mathbf{z}_{k2}^{(1)}, \ldots, \mathbf{z}_{kR}^{(1)}$$

2. then for $m = 2, 3, \ldots$ letting

$$\mathbf{z}_{kl}^{(m)} = \left\{ \begin{array}{l} \text{the mean of all } \mathbf{x}_i \in T_k \text{ with} \\ \left\| \mathbf{x}_i - \mathbf{z}_{kl}^{(m-1)} \right\| < \left\| \mathbf{x}_i - \mathbf{z}_{kl'}^{(m-1)} \right\| \text{ for all } l \neq l' \end{array} \right.$$

iterating until convergence.

Considering the locations of other prototypes



This way of choosing prototypes for class k ignores the "location" of the other classes and the eventual use to which the prototypes will be put. A potential improvement on this is to employ some kind of algorithm (again ad hoc, but reasonable) that moves prototypes in the direction of training input vectors in their own class and away from training input vectors from other classes. One such method is known by the name "LVQ"/"learning vector quantization." This proceeds as follows.

LVQ algorithm

With a set of prototypes (chosen randomly, or from an R-means algorithm, or some other way)

$$\mathbf{z}_{kl} \quad k = 1, 2, \ldots, K \quad \text{and} \quad l = 1, 2, \ldots, R$$

in hand, at each iteration $m = 1, 2, \ldots$ for some sequence of "learning rates" $\{\epsilon_m\}$ with $\epsilon_m \geq 0$ and $\epsilon_m \searrow 0$

1. sample an $\mathbf{x}_i$ at random from the training set and find k, l minimizing $\|\mathbf{x}_i - \mathbf{z}_{kl}\|$
2. if $y_i = k$ (from 1.), update $\mathbf{z}_{kl}$ as

$$\mathbf{z}_{kl}^{\text{new}} = \mathbf{z}_{kl} + \epsilon_m \left(\mathbf{x}_i - \mathbf{z}_{kl}\right)$$

and if $y_i \neq k$ (from 1.), update $\mathbf{z}_{kl}$ as

$$\mathbf{z}_{kl}^{\text{new}} = \mathbf{z}_{kl} - \epsilon_m \left(\mathbf{x}_i - \mathbf{z}_{kl}\right)$$

iterating until convergence.

Nearest neighbor classification

Define for each $\mathbf{x}$ the l-neighborhood

$$n_l(\mathbf{x}) = \text{the set of } l \text{ inputs } \mathbf{x}_i \text{ in the training set closest to } \mathbf{x} \text{ in } \Re^p$$

A nearest neighbor method is to classify $\mathbf{x}$ to the class with the largest representation in $n_l(\mathbf{x})$ (possibly breaking ties at random). That is, define

$$\hat{f}(\mathbf{x}) = \arg\max_k \sum_{\mathbf{x}_i \in n_l(\mathbf{x})} I[y_i = k] \qquad (1)$$

l is a complexity parameter (potentially chosen by cross-validation). This can be effective (despite the curse of dimensionality), depending upon clever application-specific choice of feature vectors/functions, definition of appropriate "distance" to define "closeness" and the neighborhoods, and appropriate local or global dimension reduction.

DANN classification

A possibility for "local" dimension reduction is this. At $\mathbf{x} \in \Re^p$ one might use regular Euclidean distance to find, say, 50 neighbors of $\mathbf{x}$ in the training inputs to use to identify an appropriate local distance to employ in actually defining the neighborhood $n_l(\mathbf{x})$ to be employed in (1). The following produces a DANN (discriminant adaptive nearest neighbor) (squared) metric at $\mathbf{x} \in \Re^p$. Let

$$D^2(\mathbf{z}, \mathbf{x}) = (\mathbf{z} - \mathbf{x})' \mathbf{Q} (\mathbf{z} - \mathbf{x})$$

for

$$\begin{aligned}\mathbf{Q} &= \mathbf{W}^{-\frac{1}{2}} \left(\mathbf{W}^{-\frac{1}{2}} \mathbf{B} \mathbf{W}^{-\frac{1}{2}} + \epsilon \mathbf{I} \right) \mathbf{W}^{-\frac{1}{2}} \\ &= \mathbf{W}^{-\frac{1}{2}} \left(\mathbf{B}^* + \epsilon \mathbf{I} \right) \mathbf{W}^{-\frac{1}{2}}\end{aligned}$$

for some $\epsilon > 0$ where $\mathbf{W}$ and $\mathbf{B}^*$ are as follows.

DANN building blocks

$\mathbf{W}$ is a pooled within-class sample covariance matrix

$$\mathbf{W} = \sum_{k=1}^{K} \hat{\pi}_k \mathbf{W}_k = \sum_{k=1}^{K} \hat{\pi}_k \left(\frac{1}{n_k - 1} \sum (\mathbf{x}_i - \overline{\mathbf{x}}_k)(\mathbf{x}_i - \overline{\mathbf{x}}_k)' \right)$$

($\overline{\mathbf{x}}_k$ is the average $\mathbf{x}_i$ from class k in the 50 used to create the local metric), $\mathbf{B}$ is a weighted between-class covariance matrix of sample means

$$\mathbf{B} = \sum_{k=1}^{K} \hat{\pi}_k (\overline{\mathbf{x}}_k - \overline{\mathbf{x}})(\overline{\mathbf{x}}_k - \overline{\mathbf{x}})'$$

($\overline{\mathbf{x}}$ is a–typically weighted–average of the $\overline{\mathbf{x}}_k$) and

$$\mathbf{B}^* = \mathbf{W}^{-\frac{1}{2}} \mathbf{B} \mathbf{W}^{-\frac{1}{2}}$$

DANN squared “distance”

Notice that in

$$\mathbf{Q} = \mathbf{W}^{-\frac{1}{2}} \left(\mathbf{B}^{*} + \epsilon \mathbf{I}\right) \mathbf{W}^{-\frac{1}{2}}$$

the "outside" $\mathbf{W}^{-\frac{1}{2}}$ factors "sphere" $(\mathbf{z} - \mathbf{x})$ differences relative to the within-class covariance structure. $\mathbf{B}^{*}$ is then the between-class covariance matrix of sphered sample means. Without the $\epsilon\mathbf{I}$, the distance would then discount differences in the directions of the eigenvectors corresponding to large eigenvalues of $\mathbf{B}^{*}$ (allowing the neighborhood defined in terms of D to be severely elongated in those directions). The effect of adding the $\epsilon\mathbf{I}$ term is to limit this elongation to some degree, preventing $\mathbf{x}_i$ "too far in terms of Euclidean distance from $\mathbf{x}$" from being included in $n_l(\mathbf{x})$.

Global DANN classification

A global use of DANN thinking is the following. At each training input vector $\mathbf{x}_i \in \Re^p$, one might use regular Euclidean distance to find, say, 50 neighbors and compute a weighted between-class-mean covariance matrix $\mathbf{B}_i$ as above (for that $\mathbf{x}_i$). These might be averaged to produce

$$\overline{\mathbf{B}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{B}_i$$

Then for eigenvalues of $\overline{\mathbf{B}}$, say $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ with corresponding (unit) eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$ one might do nearest neighbor classification based on the first few features

$$v_j = \mathbf{u}_j' \mathbf{x}$$

and ordinary Euclidean distance (producing a nearest neighbor version of the reduced rank classification idea).