

Summary of Multi-classification Methods

Yuejing Ding
Department of Statistics
Columbia University
10/27/2005

Overview

There are generally two basic strategies in doing multi-classification problems.

- Classify all at once, based on posterior probability or distance metric
- First use binary classification methods and then combine binary classification results

Classification all at once, based on posterior probability or distance metric

Parametric classification methods

- Linear Discriminant Analysis(LDA)
- Quadratic Discriminant Analysis(QDA)
- Logistic Regression

Classification all at once, based on posterior probability or distance metric

Parametric classification methods: General Framework

- K classes ($k = 1, \dots, K$), training sample $(x_1, g_1), \dots, (x_N, g_N)$,
x: feature, g: class. Samples in class k come from density f_k .
- Prior distribution for each class is π_1, \dots, π_k .
- Given a new sample feature X, the posterior probability that x belongs to class k is:
$$P(G = k|X) = \frac{f_k(x)\pi_k}{\sum_{l=1}^k f_l(x)\pi_l}$$
- Then classify X to class j which produces highest posterior probability.

Classification all at once, based on posterior probability or distance metric

Gaussian Based Discriminant Analysis

$$f_k(x) = MVN(\mu_k, \Sigma_k)$$

- Linear Discriminant Analysis(LDA)
All $\Sigma_k = \Sigma$
- It can also be shown that LDA is equivalent to Least Square boundary.
- Quadratic Discriminant Analysis(QDA)
 Σ_k 's are different.

Classification all at once, based on posterior probability or distance metric

Logistic Regression

- Fit multi-class logistic regression:

$$\log\left(\frac{P(G=k|X)}{P(G=K|X)}\right) = \beta_{0k} + \beta_{1k}X, k = 1, \dots, K - 1$$

Classification all at once, based on posterior probability or distance metric

Nonparametric classification methods

- Flexible Discriminant Analysis (FDA)
- Penalized Discriminant Analysis (PDA)
- Prototype Methods
- K Nearest Neighbor (KNN)
- Neural Network

Classification all at once, based on posterior probability or distance metric

FDA

The basic idea of FDA is to extend the model of LDA by allowing the following:

- Working with a set of "optimal scoring" function $\theta_l(g), l = 1, \dots, L, L < K$ instead of original class g or posterior probability of g .
- Using non-linear functions instead of linear functions
- Considering a regularizer to smooth the boundary

Classification all at once, based on posterior probability or distance metric

FDA

Basic framework of FDA is as following:

- K classes, samples (x_i, g_i) , try to find a set of optimal scoring functions $\theta_1, \dots, \theta_L, \theta : \mathcal{G} \rightarrow \mathcal{R}, L < K$, to minimize

$$ASR = \sum_{l=1}^L [\sum_{i=1}^N (\theta_l(g_i) - \eta_l(x_i))^2 + \lambda J(\eta_l)]$$

where $\eta_l(x)$ is some non-linear smooth function, and $J(\eta_l)$ is regularizer, λ is tuning parameter.

- For class k, calculate the centroid $\bar{\eta}_l^k$ for all x_i 's in that class.
- Then, for a new sample X, classify it by finding the nearest centroid using distance:

$$\delta_J(x, \hat{\mu}_k) = \sum_{l=1}^{K-1} \omega_l (\bar{\eta}_l(x) - \bar{\eta}_l^k)^2 + D(x), \text{ where } \omega_l = \frac{1}{r_l^2(1-r_l^2)}$$

Classification all at once, based on posterior probability or distance metric

PDA

PDA is a special case of FDA.

- It specifies the non-linear function $\eta_l = h(x)^T \beta_l$, where $h(x)$ is the basis expansion of some kernel.
- It also specifies the regularizer $J(\eta_l) = \beta_l^T \Omega \beta_l$, where Ω is some constraint smooth matrix.

Classification all at once, based on posterior probability or distance metric

Prototype Methods

- K-mean clustering prototype
- Learning Vector Quantization (LVQ)

Classification all at once, based on posterior probability or distance metric

K-mean clustering prototype

Basic idea:

- Suppose we have K classes, for each class, we fit R prototypes based on K -mean clustering algorithm
- Assign labels to each prototype.
- For a new sample X , we find the nearest prototype and hence classify X to be in the same class as this prototype.

Classification all at once, based on posterior probability or distance metric

Learning Vector Quantization (LVQ)

- This method is an improvement of classical K-mean clustering prototype.
- Then basic idea is: we want those prototype to be far away from the classification boundary in order to minimize the misclassification rate.

Classification all at once, based on posterior probability or distance metric

Learning Vector Quantization (LVQ) (Cont.)

The algorithm is as following:

1. Fit $K \times R$ prototypes as before
2. Sample a training point X_i from the training set with replacement, and find the prototype $m_j(k)$ nearest to X_i and then classify X_i .

If X_i is correctly classified, this prototype $m_j(k)$ is moved toward X_i by

$$m_j(k) = m_j(k) + \epsilon(X_i - m_j(k))$$

If X_i is misclassified, this prototype $m_j(k)$ is moved away from X_i by

$$m_j(k) = m_j(k) - \epsilon(X_i - m_j(k))$$

Here ϵ is called the "learning rate".

3. Repeat (1), (2), and gradually decrease ϵ to 0.

Classification all at once, based on posterior probability or distance metric

K Nearest Neighbor (KNN)

- The basic idea of KNN is: for a new sample X , first find the K "nearest neighbors" of X , then classify X according to the majority votes of its neighbors.
- Possible problems with this procedure:
 - How to choose K , the number of neighbors?
 - What distance metric should we use?
 - How to overcome the curse of dimensionality?

Classification all at once, based on posterior probability or distance metric

K Nearest Neighbor (KNN)

- For the first problem, usually we use cross-validation to determine the best K .
- The second and third problems are closely related.
 - In low dimension, usually we just use Euclidean distance.
 - In high dimension, however, sample data become so sparse that the "nearest neighbors" may not be so near, thus the classification information may be misleading.
 - Possible solution: adaptive metric.

Classification all at once, based on posterior probability or distance metric

K Nearest Neighbor (KNN) - Adaptive Metric

Basic idea:

- Some dimension or some direction provides little information for classification
- Thus, the adaptive metric should "stretch out" in the direction with little information, while put more emphasis on the direction with a lot of information.

Classification all at once, based on posterior probability or distance metric

K Nearest Neighbor (KNN) - Discriminant adaptive nearest neighbor (DANN) metric

- DANN metric is defined by $D(x, x_0) = (x - x_0)^T \Sigma (x - x_0)$
- $\Sigma = W^{-\frac{1}{2}} [W^{-\frac{1}{2}} B W^{-\frac{1}{2}} + \epsilon I] W^{-\frac{1}{2}}$
- W is the pooled within-class covariance matrix $\sum_{k=1}^K \pi_k W_k$, and B is the between class covariance matrix $\sum_{k=1}^K \pi_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T$, ϵ is a regularizer tuning parameter.
- W and B are calculated using only 50 nearest neighbors (in Euclidean sense) of x_0

Classification all at once, based on posterior probability or distance metric

Neural Network

Neural Network is a two-stage regression or classification framework. The basic idea is: we have

- Input layer: inputs X_1, \dots, X_N
- Hidden layer: "Neurons" Z_1, \dots, Z_M (possible multi-layers)
- Output layer Y_1, \dots, Y_k

Classification all at once, based on posterior probability or distance metric

Neural Network

Z_m is nonlinear function of linear combinations of the inputs, and the target Y_k is modeled as a function of linear combinations of the Z_m .

- $Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \dots, M$
- $T_k = \beta_{0k} + \beta_k^T Z, k = 1, \dots, K, Z = (Z_1, \dots, Z_M)$
- $f_k(X) = g_k(T), k = 1, \dots, K, T = (T_1, \dots, T_K)$
- Activation function $\sigma(v)$ is usually chosen to be the sigmoid
$$\sigma(v) = \frac{1}{1+e^{-v}}$$
- $g_k(T)$ is usually chosen to be $g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^K e^{T_l}}$

Classification all at once, based on posterior probability or distance metric

Neural Network

It can be proved that if M is taken arbitrarily large, this family can approximate any continuous function in \mathcal{R} arbitrarily well.

Binary classification methods and methods to combine binary classification results

Binary Classification methods

- Golub's Weighted Votes
- Support Vector Machine

Binary classification methods and methods to combine binary classification results

Golub's Weighted Votes

Basic idea:

- first we find a set of "informative genes" based on each gene's signal-to-noise ratio:

$$\rho(g, c) = \frac{\mu_1(g) - \mu_2(g)}{\sigma_1(g) + \sigma_2(g)}$$

- Then, use those "informative genes" to form "weighted votes" for class 1 or 2:

Define $a_g = \rho(g, c)$, $b_g = \frac{\mu_1(g) + \mu_2(g)}{2}$, then vote of the gene g of sample x is $V_g = a_g(x_g - b_g)$

Binary classification methods and methods to combine binary classification results

Golub's Weighted Votes (cont.)

- The final classification is based on weighted majority votes: $V = \sum_g V_g$
If $V > 0$, x is classified to class 1, otherwise x is classified to class 2.
- This method also gives predicted strength:
$$PS = (V_{win} - V_{lose}) / (V_{win} + V_{lose})$$

Binary classification methods and methods to combine binary classification results

Support Vector Machine

Basic idea:

- Try to find the optimal "hyperplane" in a transformed feature space that can discriminant data
- Tolerant some misclassification rate in order to smooth the boundary.

Binary classification methods and methods to combine binary classification results

Support Vector Classifier

- Find the optimal "hyperplane" $f(x) = \beta_0 + \beta^T X$ which maximizes the margin of classification and tolerates some points to be within the margin with distance ξ_i :

$$\max_{\beta_0, \|\beta\|=1} C$$

$$\text{under the constraint } y_i(\beta_0 + \beta^T x_i) \geq C(1 - \xi_i)$$

$$\xi_i \geq 0$$

$$\sum_{i=1}^N \xi_i \leq M$$

Binary classification methods and methods to combine binary classification results

Support Vector Classifier (cont.)

- By setting $C = 1$ and relax the normalization condition of β_1 , the problem above can be reframed as the following:

$$\min \frac{1}{2} \|\beta\|^2$$

$$\text{subject to } y_i(\beta_0 + \beta^T x_i) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

$$\sum_{i=1}^N \xi_i \leq M$$

Binary classification methods and methods to combine binary classification results

Support Vector Classifier (cont.)

- This is a convex optimization problem and the result to this problem is:

$$\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i x_i$$

- $\hat{\alpha}_i$ is nonzero only for those observation i which is within the margin. Those observations are called *support vectors*.
- Given the solution β_0 and β , the classification function $\hat{G}(x) = \text{sign}[\hat{f}(x)]$

Binary classification methods and methods to combine binary classification results

From Support Vector Classifier to Support Vector Machine

- Support Vector Machine is the Support Vector Classifier in a transformed feature space, which is characterized by a set of basis function $h(x) = (h_1(x), \dots, h_m(x))$.
- Find the optimal "hyperplane" $f(x) = \beta_0 + \beta^T h(X)$ under the same constraint as in support vector classifier.
- The estimates of β is $\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i h(x_i)$
- Classification function $\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i y_i \langle h(x), h(x_i) \rangle + \beta_0$

Binary classification methods and methods to combine binary classification results

*From Support Vector Classifier to Support Vector Machine
(cont.)*

- From the form of $\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i y_i \langle h(x), h(x_i) \rangle + \beta_0$, we can see that if we define a kernel function:

$K(x, x') = \langle h(x), h(x') \rangle$, then $\hat{f}(x)$ only depends on $K(x, x')$:

$$\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i y_i K(x, x_i) + \beta_0$$

- Commonly used kernel functions:
 - Polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$
 - Radial basis: $K(x, x') = \exp(-\|x - x'\|^2 / c)$
 - Neural network basis: $K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

Binary classification methods and methods to combine binary classification results

Methods to combine binary classification results

Standard modern approach of combining binary classifier is through "output coding".

- If we have K classifiers, for a given sample X , we can output a K -dimension output vector.
- Then we use a "codebook" to decode this K -dimension vector and get classification result.
- There are two most widely used coding systems in genetic literature: Pairwise comparison and One vs All (OVA).

Binary classification methods and methods to combine binary classification results

All Pairs

- Coding matrix C : $K \times K$, where C_{ij} = signed confidence measure of the classifier for (i,j) .
- Decoding method:
$$\text{class} = \underset{i=1, \dots, k}{\operatorname{argmax}} \left(\sum_{j=1}^K C_{ij} \right)$$

Binary classification methods and methods to combine binary classification results

OVA

- Coding matrix: $K \times K$ diagonal matrix, where C_{ii} = signed confidence measure of the classifier for (i, not i), and all $C_{ij} = 0$
- Decoding method: $\text{class} = \underset{i=1, \dots, k}{\text{argmax}}(C_{ii})$