Statistics - Week 5

Jimmy Tsz Ming Yue[*]

University of Sydney

jyue6728@uni.sydney.edu.au

Semester 2 Statistics 2018

Contents

1 Basic Principles of Classification

For each object associated with a class label (or response) $y \in \{1, 2, \ldots, K\}$ and a feature vector (vector of predictor variables) of N measurements: $X = (x_1, \ldots, x_N)$ the aim is to classify y using X.

1.1 Classification vs Clustering

There is a difference between classification and clustering;

1. Clustering: classes are unknown and is the object of discovery from the data. It is a form of unsupervised learning

[*]440159151

2. Classification: classes are predefined, and the aim is tu se a training or learning set of labelled objects to form a classifier for classification of future observations. It is a form of supervised learning

1.2 Classification vs Regression

1. Regression: No class definition, the response variable is a continuous value. The relationship between explanatory variables and the response variable is the subject of the model.
2. Classification: Samples are predefined to be from a given class. Classification models produce a continous valued prediction, which is usually in the form of a probability, or probability distribution. A predicted class is required in order to make a decision.

1.3 Classification

INSERT PICTURE HERE:

Classification rule determined by following factors:

1. Classification procedure,
2. Feature selection,
3. Parameters (both pre-determined and estimable)
4. Distance measures
5. Aggregation methods

One can think of the classification rule as a black box, some methods may provide more insight into the box. Performance assessments needs to be looked at for all classification rules.

2 Two class classification

Examples of the two class problems include:

1. Email: Spam/Not Spam
2. Tumour: Malignant/Benigh

The general idea behind the two class problems is to label a sample as $y \in \{0, 1\}$, where

1. 0: " negative class"
2. 1: "positive class"

Generate a threshold classifier output $h_\theta(x)$ at 0.5, then:

1. if $h_\theta(x) > 0.5$, predict $y = 1$
2. if $h_\theta(x) < 0.5$, predict $y = 0$.

This generates the question as to why we do not simply use simple linear regression. This can be easily seen through the example: When y only takes on valuesof 0 and 1, standard linear regression will produce values greater than 1 and values of y between 0 and 1. There are other problems:

1. The regression line $\beta_0 + \beta_1 x$ can take on any value between negative and positive infinity. For example in the diagnosis problem, y can only take on two possible values 0 and 1. Therefore a simple regression line almost always predicts the wrong value for y in classification problems.

As such we need another type of regression; of which the Logistic Regression can be motivated;

3 Logistic Regression

Definition. The logistic regression model is defined as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \theta^T \vec{x} \tag{1}$$

For which solving for p we get;

$$\begin{aligned} p &= \Pr(y = 1|\vec{x}) && (2)\\ &= h_\theta(\vec{x}) && (3)\\ &= g(\theta^T(\vec{x})) && (4)\\ &= \frac{1}{1+e^{-\theta^T \vec{x}}} && (5) \end{aligned}$$

3.1 Logistic Regression: Decision boundary

We can construct the notion of a decision boundary from the above equations;
We predict $y = 1$ if $\theta^T x \geq 0$. NEEDS CLARIFICATION ON THIS POINT

3.2 Maximum likelihood estimation of theta with single predictor

From above we have that;

$$\Pr(y = 1|x) = g(\theta_0 + \theta_1 x) \tag{6}$$

Then from the axioms of probability the complementary event:

$$\begin{aligned} \Pr(y = 0|x) &= 1 - g(\theta_0 + \theta_1 x) && (7)\\ \Pr(y|x) &= [g(\theta_0 + \theta_1 x)]^y \, [1 - g(\theta_0 + \theta_1 x)]^{1-y} && (8) \end{aligned}$$

Then the likelihood function of n samples“

$$L = \prod_{i=1}^{n} [g(\theta_0 + \theta_1 x_i]^{y_i} \, [1 - g(\theta_0 + \theta_1 x_1)]^{1-y_i} \tag{9}$$

If we then take the log of both sides to find the log-likelihood:

$$\log(L) = \sum_{i=1}^{n} y_i \log\left[g(\theta_0 + \theta_1 x_i\right] + (1 - y_i)\log\left[1 - g(\theta_0 + \theta_1 x_1)\right] \tag{10}$$

Then the partial derivatives:

$$\begin{aligned} \frac{\partial \log(L)}{\partial \theta_0} &= 0 && (11)\\ \frac{\partial \log(L)}{\partial \theta_1} &= 0 && (12) \end{aligned}$$

4 Linear Discriminant Analysis

Linear Discriminant Analysis or LDA, undertakes the same task as Logistic Regression. It classifies data based on categorical variables. For example;

1. Malignant or benign cancer;
2. Making profit in a business or not
3. Buying a product or not
4. Satisfised customer or not

4.1 Logistic Regression vs LDA formulation

Logistic Regressions model the probability of Y being from the k-th class as:

$$p(X) = \Pr(Y = k|X = x) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{13}$$

However through Bayes' Theorem, we have that:

$$p(X) = \Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi f_l(x)} \tag{14}$$

where:

1. π_k: Probability of coming from class k which is also called the prior
2. $f_k(x)$: Density function for X given that X is an observation from class k.

4.2 LDA estimations

LDA estimates π_k and $f_k(x)$. We can estimate these parameters to compute $p(X)$. The most common model for $f_k(x)$ is the normal density (LDA):

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2} \tag{15}$$

Using this density, we only need to estimate three quantities to compute $p(X)$:

1. μ_k
2. σ_k^2
3. π_k

4.3 Using training data set for estimation

The mean μ_k can be estimated through the averaging of all training observations from the k-th class. The variance σ_k^2 can be estimated as the weighted average of variances of all k classes. And lastly we have that π_k can be estimated as the proportion of the training observations that belong to the k-th class.

$$\hat{\mu_k} = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{16}$$

$$\hat{\sigma^2} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu_k})^2 \tag{17}$$

$$\hat{\pi_k} = \frac{n_k}{n} \tag{18}$$

Let us present a simple example:

> Suppose we have only one predictor, then let two normal density functions denoted $f_1(x)$ and $f_2(x)$ represent two distinct classes. These two density functions overlap, so there is some uncertainty about the class to which an observation with an unknown class belongs. The dashed vertical line represents Bayes' decision boundary;
> INSERT PICTURE HERE

4.4 Deriving LDA for one predictor

Assuming that we are working with only one predictor:

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2} \tag{19}$$

$$p_k(x) = \Pr(y = k|x) = \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2}} \tag{20}$$

Taking the log

$$\log(p_k(x)) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \tag{21}$$

4.5 LDA decision boundary

If $K = 2$ and $\pi_1 = \pi_2$, then assigns an observation to class 1 if:

$$\log(p_1(x)) > \log(p_2(x)) \tag{22}$$

$$x\frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \log(\pi_1) - x\frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \log(\pi_2) > 0 \tag{23}$$

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2 \tag{24}$$

Definition. Therefore we have the decision boundary for the LDA to be defined as:

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} \tag{25}$$

$$= \frac{\mu_1 + \mu_2}{2} \tag{26}$$

T

4.6 LDA for more than one learning feature

Definition. Now that we have defined LDA for more than one feature, let us attempt to extend this to multiple learning features; Let us first present again the one feature LDA:

$$\log(p_k(x)) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \tag{27}$$

Then we have for more than one feature:

$$\log(p_k(\vec{x})) = \vec{x}^T\vec{\Sigma}^{-1}\frac{\mu_k}{\sigma^2} - \frac{1}{2}\vec{\mu_k}\vec{\Sigma}^{-1}m\vec{u}_k + \log(\pi_k) \tag{28}$$

4.7 Why not use logistic regression

Logistic regression is unstable when the classes are well separated. In the case where n is small, and the distribution of predictors X is approximately normal, then LDA is more stable than Logistic Regression. Usually LDA is more popular when we have more than two response classes.

4.8 LDA vs Logistic Regression

Similarity: Both Logistic Regression and LDA produce linear boundaries.

Difference: LDA assumes that the observations are drawn from the normal distribution with common variance in each class, while logistic regression does not have this assumption.

LDA would do better than Logistic Regression if the assumption of normality holds, otherwise logistic regresion may outperform LDA

5 Classification evaluation based on class labels Training and testing errors

Definition. Training error rate

$$\frac{1}{N}\sum_{i=1}^{N} I(y_i \neq \hat{y}_i) \tag{29}$$

a

Definition. Test error rate

$$\frac{1}{T}\sum_{t=1}^{T} I(y_t \neq \hat{y}_i) \tag{30}$$

<++>

Often the best classifier is the one for which the test error is the smallest. It should be noted that test error not be used for classifcation evaluation

6 Tutorial

```
> # create positive class sample with 2 descriptive features
> set.seed(3)
> f1 <- rnorm(100, mean=6, sd = 1.2)
> set.seed(4)
> f2 <- rnorm(100, mean=6, sd = 1.2)
> P.data <- cbind(f1, f2)
> # create positive class sample with 2 descriptive features
> set.seed(7)
> f1 <- rnorm(300, mean=4, sd = 1.2)
> set.seed(8)
> f2 <- rnorm(300, mean=4, sd = 1.2)
> N.data <- cbind(f1, f2)
> # combine all samples
> data.mat <- data.frame(rbind(P.data, N.data), Class=rep(c(1, 0), time=c(nrow(P.data), nrow(N.data)
```

The above code will create a dataset with two class and two features, each follows a normal distribution.

1. Partition the data into 80% for model training (training set) and 20% for model testing (test set).

Solution. First load the caret package to deal with paritioning data sets:

```
> head(data.mat)

        f1       f2 Class
1 4.845680 6.260106     1
2 5.648969 5.349009     1
3 6.310546 7.069374     1
4 4.617442 6.715177     1
5 6.234939 7.962742     1
6 6.036149 6.827131     1

> dim(data.mat)

[1] 400   3

> library(caret)
> inTrain <- createDataPartition(data.mat$Class, p = .8)[[1]]
> data.train <- data.mat[inTrain, ]
> head(data.train)

        f1       f2 Class
1 4.845680 6.260106     1
2 5.648969 5.349009     1
3 6.310546 7.069374     1
5 6.234939 7.962742     1
6 6.036149 6.827131     1
8 7.339932 5.744227     1
```

```
> dim(data.train)

[1] 320   3

> data.test <- data.mat[-inTrain, ]
> dim(data.test)

[1] 80  3
```

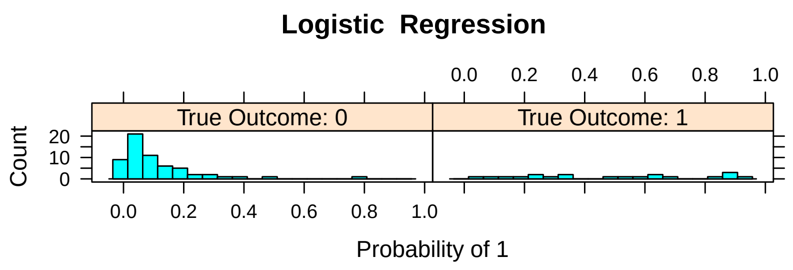
□

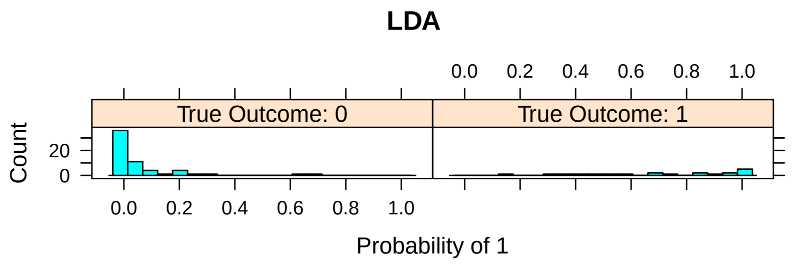
2. Train a Logistic Regression, a LDA and a kNN (try different *k* value) classifier using training dataset. Compare their performance using test dataset.

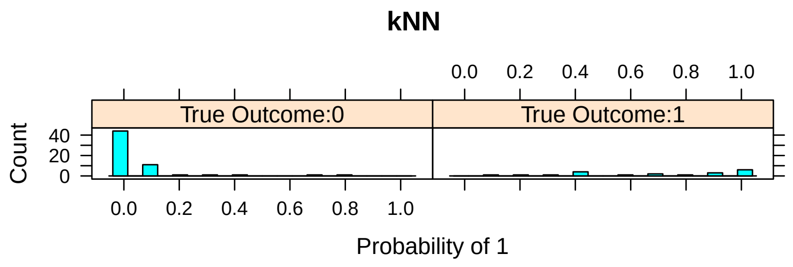
Solution. Let us first do the Logistic Regression using glm;

```
> set.seed(1)
> data.train$Class <- as.factor(data.train$Class)
> logisticRegFull <- train(Class ~ f1,
+                       data = data.train,
+                       method = "glm",
+                       trControl = trainControl(method = "repeatedcv",
+                                                repeats = 5))

> logisticRegFull

Generalized Linear Model

320 samples
  1 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 288, 288, 288, 288, 288, 288, ...
Resampling results:

  Accuracy  Kappa
  0.834375  0.5139071

> PredictFunc <- function(logisticReg) {
+   Results <- data.frame(obs = data.test$Class)
+   Results$prob <- predict(logisticReg, data.test, type = "prob")[, "1"]
+   Results$pred <- predict(logisticReg, data.test)
+   Results$Label <- ifelse(Results$obs == "1",
+                           "True Outcome: 1",
+                           "True Outcome: 0")
+   return(Results)
+ }
>
```

```
> ResultsFull <- PredictFunc(logisticRegFull)

> LDAFull <- train(Class ~ .,
+                     data =data.train,
+                     method = "lda",
+                     trControl = trainControl(method = "repeatedcv",
+                                              repeats = 5))
> summary(LDAFull)

            Length Class      Mode
prior       2      -none-     numeric
counts      2      -none-     numeric
means       4      -none-     numeric
scaling     2      -none-     numeric
lev         2      -none-     character
svd         1      -none-     numeric
N           1      -none-     numeric
call        3      -none-     call
xNames      2      -none-     character
problemType 1      -none-     character
tuneValue   1      data.frame list
obsLevels   2      -none-     character
param       0      -none-     list

> LDAFull$finalModel

Call:
lda(x, grouping = y)

Prior probabilities of groups:
   0    1
0.75 0.25

Group means:
        f1       f2
0 4.130876 3.904459
1 6.046112 6.104246

Coefficients of linear discriminants:
         LD1
f1 0.5763584
f2 0.6588399

> PredictFuncLDA <- function(model) {
+   LDAResults <- data.frame(obs =data.test$Class)
+   LDAResults$prob <- predict(model, data.test, type = "prob")[, "1"]
+   LDAResults$pred <- predict(model, data.test)
+   LDAResults$Label <- ifelse(LDAResults$obs == "1",
+                                 "True Outcome: 1",
+                                 "True Outcome: 0")
+   return(LDAResults)
+ }
> LDAresultsFull <- PredictFuncLDA(LDAFull)

> KNNFull <- train(Class ~ .,
+                       data = data.train,
+                       method = "knn",
+                       trControl = trainControl(method = "repeatedcv",
+                                                repeats = 5))
> KNNFull

k-Nearest Neighbors

320 samples
  2 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 288, 288, 288, 288, 288, 288, ...
Resampling results across tuning parameters:

  k  Accuracy  Kappa
  5  0.920625  0.7794484
  7  0.918125  0.7716414
  9  0.926250  0.7959865

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.

> PredictFuncKNN <- function(model) {
+   results <- data.frame(obs = data.test$Class)
+   results$prob <- predict(model, data.test, type = "prob")[, "1"]
+   results$pred <- predict(model, data.test)
+   results$Label <- ifelse(results$obs == "1",
+                               "True Outcome:1",
+                               "True Outcome:0")
+   return(results)
+ }
> KNNresultsFull <- PredictFuncKNN(KNNFull)

> library(gridExtra)
> hist1 <- histogram(~prob|Label, data = ResultsFull, layout = c(2, 1), nint = 20, xlab = "Pr
+           type = "count", main="Logistic  Regression")
> hist2 <- histogram(~prob|Label, data = LDAresultsFull, layout = c(2, 1), nint = 20, xlab =
+           type = "count", main="LDA")
> hist3 <- histogram(~prob|Label, data = KNNresultsFull, layout = c(2, 1), nint = 20, xlab =
+           type = "count", main="kNN")
> grid.arrange(hist1, hist2, hist3, nrow=3)
```





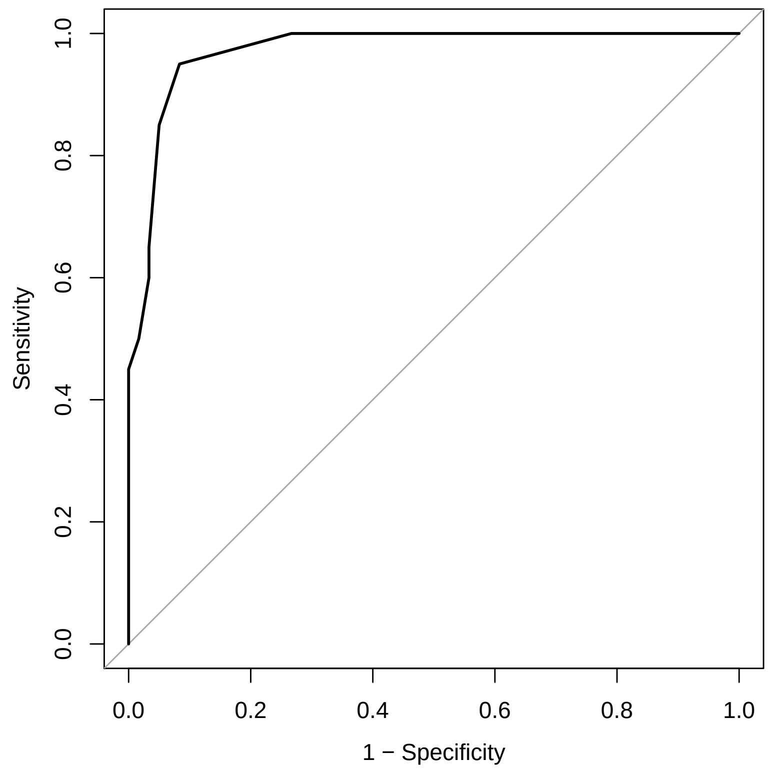


□

3. For kNN, identify optimal k value by minimising classification error on test set.

Solution:

```
> library(pROC)
> ROCFull <- roc(KNNresultsFull$obs, KNNresultsFull$prob)
> auc(ROCFull)

Area under the curve: 0.9717

> ci.auc(ROCFull)

95% CI: 0.9417-1 (DeLong)

> plot(ROCFull, legacy.axes = TRUE, col ="black")
```



As we can see from the Sensitivy Specificity Curve $k = 9$ is the optimal based on minimising classification errors. □

4. Now we used test set to select optimal k, is it still valid to use this test set to evaluate the performance of our optimised kNN classifier? Why or why not?

Solution. No, it is best to use a different test set drawn again from the sample, albeit ensuring that it is a balanced data set. □