BIOS 635: Linear and Quadratice Discriminant Analysis

Kevin Donovan

1/28/2021

Review

- Homework 2 due on 2/4 at 11PM through GitHub Classroom
- Article Evaluation 1 assigned, due on 2/9 through GitHub Classroom
- Last lecture: logistic regression

Classification

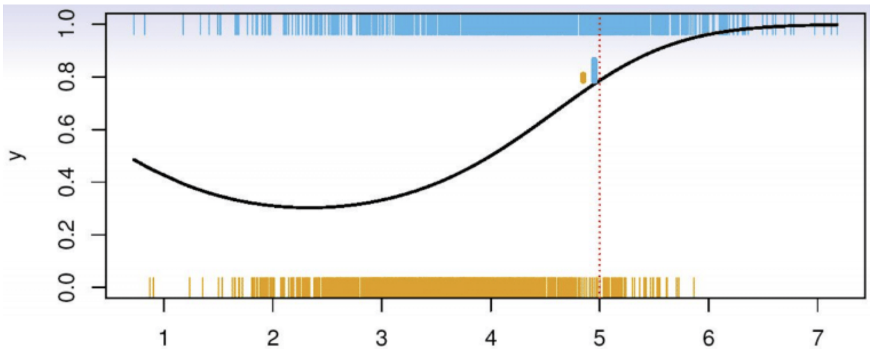
Let response $Y \in \{0,1\}$

Goal: Predict Y using features X

What to model?:

Let $p_k(x) = \Pr(Y = k|X = x)$, $k = 0, 1$

Denoted as the **conditional class probabilities** at x



Discriminant analysis

- **Idea**: Instead of looking at $f(y|x)$, look at $f(x|y)$ for each class $y = 1, 2, \ldots, K$
- Try to see if distribution of features X **differs** among the response classes
- Then use *Bayes theorem* to estimate $f(y|x) = \Pr(Y = y|X = x)$
- How to model distribution of $f(x|y)$ given many features X?
- *Linear/Quadratic Discriminant Analysis* (L/QDA) $\rightarrow$ normal distribution modeled
    - *Can be used with other distributions as well*

Bayes Theorm

- Suppose Y and X are discrete/categorical random variables
- **Bayes Theorem**:

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) * \Pr(Y = k)}{\Pr(X = x)}$$

- Also holds for Y and or X continuous:

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{f(x)}$$

where

- $f(x|k)$ denotes the conditional density of $X|Y$
- $f(x)$ denotes the density of X

Bayes Theorm for Classification

- We can reformulate this for discriminant analysis:

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x|l) * \Pr(Y = l)}$$

- $f(x|l)$ modeled using a chosen distribution (Normal in our case)
- In Bayes, $\Pr(Y = k)$ denoted as the *prior* probability for class *k*
    - *i.e., probability not based on features* $X = x$

Posterior Probability

- In Bayes, $\Pr(Y = k|X = x)$ denoted as *posterior probability*
  - *i.e. probability of being in class k based on features $X = x$ ("post" seeing data)*
- **Idea**: To classify an observation with feature set x_0, choose class with max posterior probability
- i.e., $\hat{y_0} = \underset{k=1,...,K}{\operatorname{argmax}} \frac{f(x_0|k)*\Pr(Y=k)}{\sum_{l=1}^{K} f(x_0|l)*\Pr(Y=l)}$
- This rule is the same rule as used in logistic regression, KNN, etc.
  - ***Difference**: Computing conditional probability differently*
- **Note**: Denominator is the same for each posterior probability
  - $\rightarrow \hat{y_0} = \underset{k=1,...,K}{\operatorname{argmax}} f(x_0|k) * \Pr(Y = k)$

Why discriminant analysis?

- Logistic regression limitations:
  - *Classes are well-separated $\rightarrow$ logistic regression model unstable*
  - *Not well suited for multi-category response prediction (required many models)*
- Discriminant analysis (DA) improves on stability and well-suited for multi-category response
- If n is small and $X \sim$ Normal in each class, DA more stable

LDA when $p = 1$

- Univariate Normal density:

$$f(x|k) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-0.5(\frac{x-\mu_k}{\sigma_k})^2}$$

where $\mu_k = \mathrm{E}(X|Y=k)$ and $\sigma_k = \sqrt{\mathrm{Var}(X|Y=k)} = \mathrm{SD}(X|Y=k)$

- **With LDA** assume $\sigma_k = \sigma$ for all $k = 1, \ldots, K$
  - *i.e. assume variance/SD in feature same in all response classes*

LDA when $p = 1$

- We can plug the above into our posterior probability formula from before:

$$\Pr(Y = k|X = x) = \frac{\Pr(Y = k)\frac{1}{\sqrt{2\pi}\sigma_k}e^{-0.5(\frac{x-\mu_k}{\sigma_k})^2}}{\sum_{l=1}^{K}\Pr(Y = l)\frac{1}{\sqrt{2\pi}\sigma_l}e^{-0.5(\frac{x-\mu_l}{\sigma_l})^2}}$$

LDA Simplifications

- As done with maximum likelihood, can simplify this max procedure by taking the log
  - $\log[f(x)]$ *is **monotonic**, so if* x_0 *maxes* $\log[f(x)] \rightarrow$ *maxes* $f(x)$
- Can also discard terms which don't involve k (as these are the same for all classes)
- Results in transformation of posterior: $\delta_k(x)$

$$\delta_k(x) = x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

where $\delta_k(x) = \max\{\delta_1(x), \ldots, \delta_K(x)\}$

$\leftrightarrow$

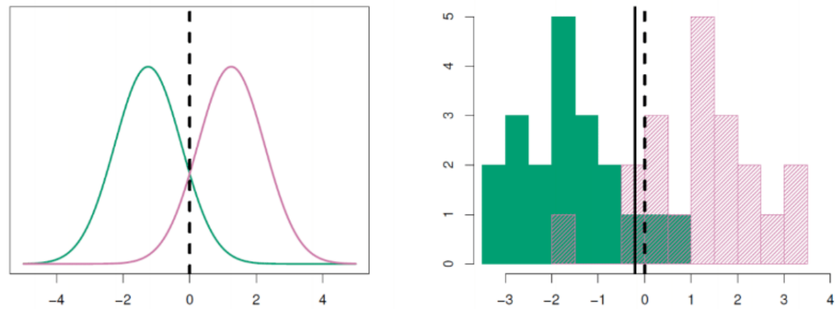
$$\Pr(Y = k|X = x) = \max_{l=1,\ldots,K}\{\Pr(Y = l|X = x)\}$$

LDA Simplifications

- Thus, can work with $\delta_k(x)$, denoted *discriminant score* instead
- Can see $\delta_k(x)$ is *linear* function of x (hence *linear* DA)
- Can show if $K = 2$ and priors $\Pr(Y = 1) = \Pr(Y = 2) = 0.5$, *decision boundary* is at

$$x = \frac{\mu_1 + \mu_2}{2}$$

LDA Visualization



Left (true distribution); Right (estimated from data)

- **Idea**: Feature distributions between classes differ, find differences in data using classes labels
- Need to estimate parameters (example $\mu_1 = -1.5, \mu_2 = 1.5, \pi_1 = \pi_2 = 0.5, \sigma^2 = 1$)

LDA Estimation

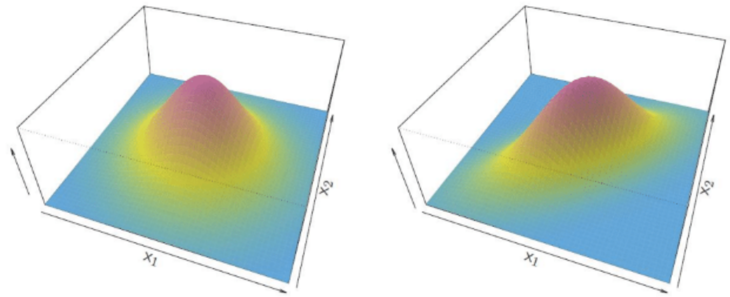
$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:\, y_i = k} x_i$$

$$\begin{aligned} \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:\, y_i = k} (x_i - \hat{\mu}_k)^2 \\ &= \sum_{k=1}^{K} \frac{n_k - 1}{n - K} \cdot \hat{\sigma}_k^2 \end{aligned}$$

- where $\hat{\sigma_k^2} = \frac{1}{n_k - 1} \sum_{i:y_i=k} (x_i - \mu_k)^2$ is the usual estimator for variance in class k
  - *Pool estimate over all classes due to* $\sigma_1^2 = \ldots = \sigma_K^2 = \sigma^2$

LDA with $p > 1$



$$\text{Density: } f(x) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \boldsymbol{\Sigma}^{-1}(x-\mu)}$$

$$\text{Discriminant function: } \delta_k(x) = x^T \boldsymbol{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T \boldsymbol{\Sigma}^{-1}\mu_k + \log \pi_k$$

Despite its complex form,
$\delta_k(x) = c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \ldots + c_{kp}x_p$ — a linear function.

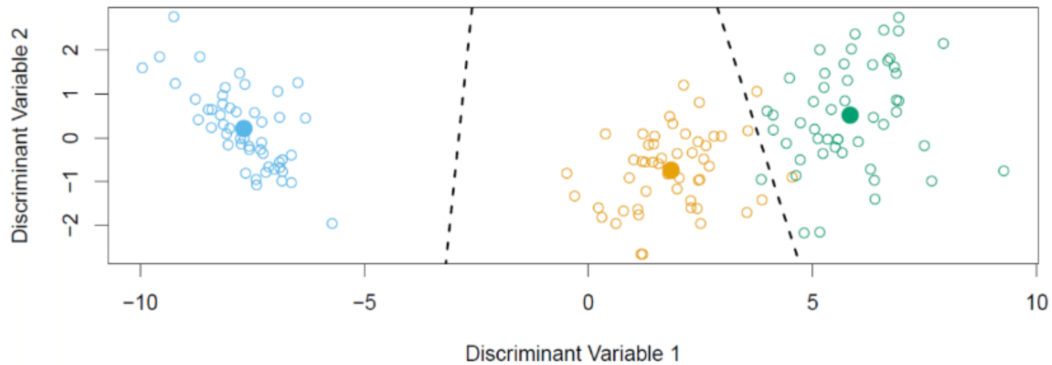
LDA: Estimating Probabilities

- Given $\hat{\delta}_k(x)$, can compute estimated class probabilities:

$$\hat{\Pr}(Y = k | X = x) = \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^{K} e^{\hat{\delta}_l(x)}}$$

- Just undoing log in probability equation from before by using e
- To classify, can use usual rule of largest $\hat{\delta}_k(x) \leftrightarrow$ largest $\hat{\Pr}(Y = k | X = x)$

Example with $p > 1$



LDA Example

- Heart disease prediction
- **NOTE**: Do our features follow normal distributions within heart disease status?

```
# Partition Data
set.seed(12)
train_test_indices <- createDataPartition(heart_data$heart_disease, p=0.6, list = FALSE)
heart_data_train <- heart_data[train_test_indices,]
heart_data_test <- heart_data[-train_test_indices,]

# Train
lda_fit <- train(heart_disease~Age+Sex+Chest_Pain+Resting_Blood_Pressure+Colestrol+
                MAX_Heart_Rate+Exercised_Induced_Angina,
                data = heart_data_train, method = "lda")

# Add in test set predictions
heart_data_test$estimated_prob_heart_disease <-
  predict(lda_fit, newdata=heart_data_test, type = "prob")$Yes

heart_data_test <-
  heart_data_test %>%
  mutate(pred_heart_disease =
         relevel(factor(ifelse(estimated_prob_heart_disease>0.5, "Yes", "No")),
                ref = "No"))

# Compute confusion matrix
confusionMatrix(data = heart_data_test$pred_heart_disease,
                reference = heart_data_test$heart_disease,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  53  19
##        Yes 12  36
##
##                Accuracy : 0.7417
##                  95% CI : (0.6538, 0.8172)
##     No Information Rate : 0.5417
##     P-Value [Acc > NIR] : 5.135e-06
##
##                   Kappa : 0.4746
##
```

```
##  Mcnemar's Test P-Value : 0.2812
##
##             Sensitivity : 0.6545
##             Specificity : 0.8154
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7361
##              Prevalence : 0.4583
##          Detection Rate : 0.3000
##    Detection Prevalence : 0.4000
##       Balanced Accuracy : 0.7350
##
##        'Positive' Class : Yes
##
```

LDA Example: Varying Threshold

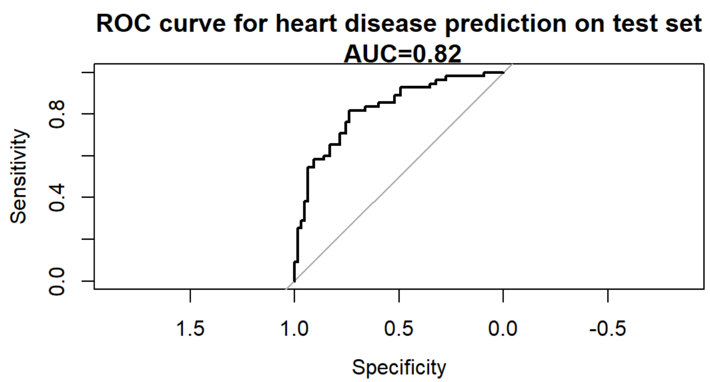
- Used a threshold of 0.5, but may see "better" performance using a different one
- Can analyze all thresholds using ROC curve

```r
# Using pROC, add ROC curve using estimated probabilities of heart disease in test set
roc_obj <-
  roc(response = heart_data_test$heart_disease,
    predictor = heart_data_test$estimated_prob_heart_disease)

# Print obj
roc_obj
```

```
##
## Call:
## roc.default(response = heart_data_test$heart_disease, predictor = heart_data_test$estimated_prob_heart_disease)
##
## Data: heart_data_test$estimated_prob_heart_disease in 65 controls (heart_data_test$heart_disease No) < 55 cases (heart_data_test$heart_disease
## Area under the curve: 0.8246
```

```r
# Plot curve
plot(roc_obj, main = paste0("ROC curve for heart disease prediction on test set\n AUC=",
                            round(auc(roc_obj),2)))
```



LDA Example: Training Set Performance

- Looking back at training set performance, expect this to be biased upward

```
# Add in train set predictions
heart_data_train$estimated_prob_heart_disease <-
  predict(lda_fit, newdata=heart_data_train, type = "prob")$Yes

heart_data_train <-
  heart_data_train %>%
  mutate(pred_heart_disease =
           relevel(factor(ifelse(estimated_prob_heart_disease>0.5, "Yes", "No")),
                   ref = "No"))

# Compute confusion matrix
confusionMatrix(data = heart_data_train$pred_heart_disease,
                reference = heart_data_train$heart_disease,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  88  24
##        Yes 11  60
##
##                Accuracy : 0.8087
##                  95% CI : (0.7442, 0.863)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 2.987e-14
##
##                   Kappa : 0.6103
##
##  Mcnemar's Test P-Value : 0.04252
##
##             Sensitivity : 0.7143
##             Specificity : 0.8889
##          Pos Pred Value : 0.8451
##          Neg Pred Value : 0.7857
##              Prevalence : 0.4590
##          Detection Rate : 0.3279
##    Detection Prevalence : 0.3880
##       Balanced Accuracy : 0.8016
##
##        'Positive' Class : Yes
##
```

LDA Example: Training Set Performance

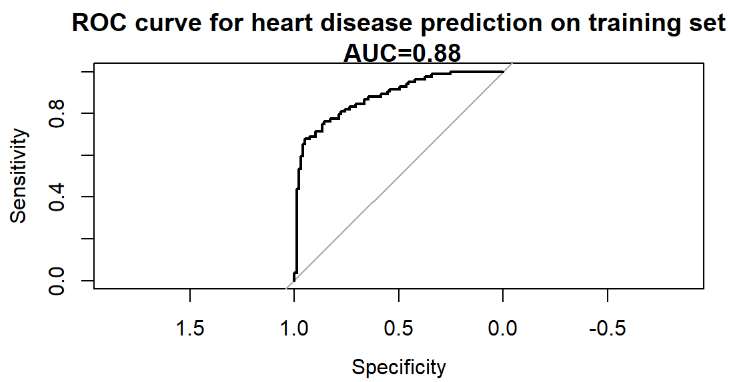
- Training set ROC curve:

```r
# Using pROC, add ROC curve using estimated probabilities of heart disease in test set
roc_obj <-
  roc(response = heart_data_train$heart_disease,
    predictor = heart_data_train$estimated_prob_heart_disease)

# Print obj
roc_obj
```

```
##
## Call:
## roc.default(response = heart_data_train$heart_disease, predictor = heart_data_train$estimated_prob_heart_disease)
##
## Data: heart_data_train$estimated_prob_heart_disease in 99 controls (heart_data_train$heart_disease No) < 84 cases (heart_data_train$heart_dise
## Area under the curve: 0.882
```

```r
# Plot curve
plot(roc_obj, main = paste0("ROC curve for heart disease prediction on training set\n AUC=",
                            round(auc(roc_obj),2)))
```



Other forms of discriminant analysis

- **Recall** starting formula for posterior class probabilities:

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x|l) * \Pr(Y = l)}$$

- For LDA, used Normal densities for $f(x|k)$, but could use different distribution to obtain different model
- For LDA, under Normal density model, also assumed $\Sigma_k = \Sigma$ for all k (classes)
  - *That is, assumed all classes have same covariance matrix for features*
  - *May not be reasonable*
  - *Normal densities but different Σ_k om each class $\rightarrow$* ***quadratic discriminant analysis***
  - *If we additional assume features on independent in each class, i.e. $f(x|k) = \prod_{j=1}^{p} f(x_j|k)$, obtain* ***naive Bayes***

QDA Example

- Again, heart disease example
- **NOTE**: Do our features meet the normal distribution assumption?

```r
# Train
qda_fit <- train(heart_disease~Age+Sex+Chest_Pain+Resting_Blood_Pressure+Colestrol+
                MAX_Heart_Rate+Exercised_Induced_Angina,
                data = heart_data_train, method = "qda")

# Add in test set predictions
heart_data_test$estimated_prob_heart_disease <-
  predict(qda_fit, newdata=heart_data_test, type = "prob")$Yes

heart_data_test <-
  heart_data_test %>%
  mutate(pred_heart_disease =
          relevel(factor(ifelse(estimated_prob_heart_disease>0.5, "Yes", "No")),
                 ref = "No"))

# Compute confusion matrix
confusionMatrix(data = heart_data_test$pred_heart_disease,
                reference = heart_data_test$heart_disease,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  55  19
##        Yes 10  36
##
##                Accuracy : 0.7583
##                  95% CI : (0.6717, 0.8318)
##     No Information Rate : 0.5417
##     P-Value [Acc > NIR] : 7.725e-07
##
##                   Kappa : 0.5071
##
##  Mcnemar's Test P-Value : 0.1374
##
##             Sensitivity : 0.6545
##             Specificity : 0.8462
##          Pos Pred Value : 0.7826
##          Neg Pred Value : 0.7432
```

```
##              Prevalence : 0.4583
##          Detection Rate : 0.3000
##    Detection Prevalence : 0.3833
##       Balanced Accuracy : 0.7503
##
##        'Positive' Class : Yes
##
```

QDA Example: Varying Threshold

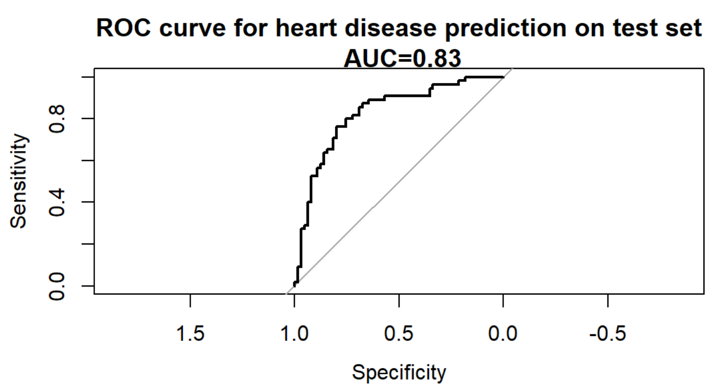
- Test set ROC curve

```
# Using pROC, add ROC curve using estimated probabilities of heart disease in test set
roc_obj <-
  roc(response = heart_data_test$heart_disease,
    predictor = heart_data_test$estimated_prob_heart_disease)

# Print obj
roc_obj
```

```
##
## Call:
## roc.default(response = heart_data_test$heart_disease, predictor = heart_data_test$estimated_prob_heart_disease)
##
## Data: heart_data_test$estimated_prob_heart_disease in 65 controls (heart_data_test$heart_disease No) < 55 cases (heart_data_test$heart_disease
## Area under the curve: 0.8285
```

```
# Plot curve
plot(roc_obj, main = paste0("ROC curve for heart disease prediction on test set\n AUC=",
                            round(auc(roc_obj),2)))
```



Discriminant analysis summary

- **General rule**:

$$\hat{y}_0 = \underset{k=1,...,K}{\operatorname{argmax}} \frac{f(x_0|k) * \Pr(Y=k)}{\sum_{l=1}^{K} f(x_0|l) * \Pr(Y=l)} = \underset{k=1,...,K}{\operatorname{argmax}} f(x_0|k) * \Pr(Y=k)$$

- LDA: assume all $f(x|k) \sim \text{Multivariate Normal}(\mu_k, \Sigma)$ for $k = 1, \ldots, K$
- QDA: assume $f(x|k) \sim \text{Multivariate Normal}(\mu_k, \Sigma_k)$ for $k = 1, \ldots, K$
  - *Can fit QDA in `caret` package with `train` function using `method="qda"`*

Song of the session

Bad Boy by BIGBANG

Blue by BIGBANG