Linear Discriminant Analysis & Naive Bayes

D2K Course Staff

3/5/2019

Introduction

When we are working with discrete outcome variables and our task is prediction, we call this a **classification** problem. But how do you decide what class to predict? How do you determine rules for placing a new data point in a specific class? One of the first techniques developed (along with logistic regression) to accomplish this was Linear Discriminant Analysis (LDA). In essence, it is an extension of linear regression in the sense that it is using linear equation to model the data. However, instead of predicting the data points along the line, it uses the line as a decision boundary for prediction.

The assumptions of linear decision boundaries allows for computational efficiency and interpretability, but it is important to note that the large number of assumptions that go into LDA also contribute to its relative poor predictive performance compared to other methods. There are many cases where LDA works very well and is able to make fast predictions on large amounts of data, but there are **more** cases where its predictive ability is simply out-performed by more modern methods. As we have mentioned before, it is important to try many different models and compare them in order to find the right model for your specific task.

Furthermore, in this module we will cover two other methods that are closely related to LDA: Quadratic Discriminant Analysis and Naive Bayes. We will use a query set to compare the relative performance of each model before reporting our final error on the test set.

Linear Discriminant Analysis

The concept of LDA begins with the idea that we should be able to optimally classify our data if we know the **posterior** distribution of the class probabilities $P(G = k|X = x)$ for all $k = 1, ..., K$ classes. We know that we can derive posterior distributions from the data and the prior distributions using Bayes Theorem. LDA uses this logic to compute posterior distributions and determine linear decision boundaries for the different classes of data.

We suppose that $f_k(x)$ is the density of the data X conditioned on the class $G = k$. We let π_k be the prior probability distribution of class k and we know that $\Sigma_{k=1}^{K}\pi_k = 1$. Using this information, we are able to model the posterior distribution.

$$P(G = k|X = x) = \frac{f_k(x)\pi_k}{\Sigma_{j=1}^{K} f_j(x)\pi_j}$$

We then make the assumption that we can model each class density as a multivariate gaussian distribution.

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

We then again assume that the covariance matrices for each class are the same, $\Sigma_k = \Sigma \forall k$ and take the log-ratio of classes to compute the decision boundary between two classes k and j.

$$log\frac{P(G=k|X=x)}{P(G=j|X=x)} = log\frac{f_k(x)}{f_j(x)} + log\frac{\pi_k}{\pi_j}$$
$$= log\frac{\pi_k}{\pi_j} - \frac{1}{2}(\mu_k+\mu_j)^T\Sigma^{-1}(\mu_k-\mu_j) + x^T\Sigma^{-1}$$

This log-ratio gives us a linear equation over the p-dimensions of x. Furthermore, for any two classes, we get a linear decision boundary that we can use as a classification rule for new data where $P(G=k|X=x) = P(G=j|X=x)$. However, since we do not know the parameters of our distributions, we must estimate them from the training data.

- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the number of observations of class k
- $\hat{\mu}_k = \frac{\Sigma_{g_i=k}x_i}{N_k}$
- $\hat{\Sigma} = \Sigma_{k=1}^{K}\Sigma_{g_i=k}\frac{(x_i-\hat{\mu}_k)(x_i-\hat{\mu}_k)^T}{(N-K)}$

Naturally, we must come up with an explicit decision rule for classifying new data. We can determine one from the computed decision boundary using the log-ratio of classes.

$$G(x) = \underset{k}{\text{argmax}} x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k$$

We now show an implementation of this prediction task in R using the `iris` dataset.

```r
# Load in data
data(iris)
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

We can see that we have 3 outcome classes of different flower species and 4 predictor variables from measurements about the flowers. We segment the data into training, query, and a test set for model comparison and evaluation.

```r
# Set seed for reproducibility
set.seed(123)

# Create training/test index
tr_idx <- sample(1:nrow(iris), nrow(iris) * 0.6, replace = FALSE)

# Create training/validation set
iris_train <- iris[tr_idx, ]

iris_test <- iris[-tr_idx, ]
```

```r
# Create query set from test set
qr_idx <- sample(1:nrow(iris_test), nrow(iris_test) * 0.5, replace = FALSE)

iris_query <- iris_test[qr_idx, ]

iris_test <- iris_test[-qr_idx, ]
```

Now that we have chunked our data into the appropriate segments, we can train an LDA model on the training data and evaluate it on the query set.

```r
# Set seed for reproducibility
set.seed(123)

# Load MASS package for lda function
library(MASS)

# Train LDA model
lda_model <- lda(Species ~ ., data = iris_train)
```

Now that we have trained our model we can evaluate its predictive ability on the query set. We will use the metric of misclassification error to evaluate how well this model performs relative to other models.

```r
# Predict flower species on query set
lda_pred <- predict(lda_model, newdata = iris_query)$class

# Compute misclassification error
lda_error <- sum(lda_pred != iris_query$Species)/nrow(iris_query)
lda_error
```

```
[1] 0
```

Wow! In this particular case, LDA perfectly predicts the outcome classes for the `iris` dataset. This is because the assumptions of normality and shared covariances hold for the 3 classes of flower species. However, let us recognize that rarely do we encounter data this perfectly linearly separable in the real world.

Quadratic Discriminant Analysis

A variation of LDA arises naturally when we relax the assumption that the covariances are the same for each outcome class. Solving for the decision boundary without covariances that are assumed to be equal produces Quadratic Discriminant Analysis and yields the quadratic equation below.

$$G(x) = \underset{k}{\operatorname{argmax}} - \frac{1}{2}|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + log\pi_k$$

Thus, instead of determining the linear boundaries between the classes, we determine quadratic boundaries. Often, classes will not be linearly separable so it is important to try many methods before determining a final model. Even though we already know that LDA perfectly separates the `iris` data, we will show an implementation of QDA in R below.

```r
# Set seed for reproducibility
set.seed(123)

# Train QDA model
qda_model <- qda(Species ~ ., data = iris_train)
```

```r
# Predict flower species on query set
qda_pred <- predict(qda_model, newdata = iris_query)$class

# Compute misclassification error
qda_error <- sum(qda_pred != iris_query$Species)/nrow(iris_query)
qda_error
```

```
[1] 0
```

We can see from the model predictions above that QDA **also** perfectly predicts the outcome labels for the query set. This means that the data is perfectly separable by both linear and quadratic equations.

Naive Bayes

Another classifier that is based on Bayes' Theorem is the Naive Bayes classifier. Unlike LDA, Naive Bayes makes the assumption that for any given class the features are independent and that the class densities are products of marginal densities of each feature. We can show this mathematically below,

$$f_j(X) = \Pi_{k=1}^{p} f_{jk}(X_k)$$

Even though this is rarely true, it allows us to simplify our computation quite a bit and is very useful for situations where p is very, very large. We can extend this logic to derive the logit-transform and thus create a decision boundary for our classes based on the posterior probabilities.

$$\begin{aligned} log\frac{P(G=k|X)}{P(G=J|X)} &= log\frac{\pi_k f_k(X)}{\pi_J f_J(X)} \\ &= log\frac{\pi_k \Pi_{i=1}^{p} f_{ki}(X_i)}{\pi_J \Pi_{i=1}^{p} f_J i(X_i)} \\ &= log\frac{\pi_k}{\pi_J} + \Sigma_{i=1}^{p} log\frac{f_{ki}(X_i)}{f_{Ji}(X_i)} \\ &= \alpha_k + \Sigma_{i=1}^{p} g_{ki}(X_i) \end{aligned}$$

We can show an implementation of the Naive Bayes classifier in R below on the `iris` data.

```r
# Set seed for reproducibility
set.seed(123)

# Load e1071 library for naiveBayes function
library(e1071)

# Fit naive bayes model on the training data
nb_model <- naiveBayes(Species ~ ., data = iris_train)

# Predict on query set
nb_pred <- predict(nb_model, newdata = iris_query)

# Compute misclassification error
nb_error <- sum(nb_pred != iris_query$Species)/nrow(iris_query)
nb_error
```

```
[1] 0.03333333
```

We can see that the Naive Bayes model did not perform quite as well as the LDA or QDA models, however, it still performed very well overall. It only misclassified 1 out of 30 flower labels!

Final Model and Conclusion

On this dataset, these traditional linear classifiers perform extremely well. However, it is **crucial** to remind you of the assumptions these models are making. All of the data needs to be normal **and** linearly (or quadratically) separable for these models to perform well. This is an example dataset, *most data in the real world are not like this!*

Finally, we report our misclassification error using our best-performing model (we arbitrarily choose LDA over QDA).

```r
# Combine training and query sets
iris_train_query <- data.frame(rbind(iris_train, iris_query))

# Set seed for reproducibility
set.seed(123)

# Train LDA model
lda_final_model <- lda(Species ~ ., data = iris_train_query)

# Predict on test set
final_pred <- predict(lda_final_model, newdata = iris_test)$class

# Report final test misclassification error
test_error <- sum(final_pred != iris_test$Species)/nrow(iris_test)
test_error
```

```
[1] 0.03333333
```

References

Hastie, Trevor, et al. The Elements of Statistical Learning Data Mining, Inference, and Prediction. Springer, 2009.