# Lab 5 - LDA and QDA in Python

February 24, 2016

This lab on Logistic Regression is a Python adaptation of p. 161-163 of "Introduction to Statistical Learning with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Adapted by R. Jordan Crouser at Smith College for SDS293: Machine Learning (Spring 2016).

```
In [ ]: import pandas as pd
        import numpy as np

        from sklearn.lda import LDA
        from sklearn.qda import QDA
        from sklearn.metrics import confusion_matrix, classification_report, precision_score

        %matplotlib inline
```

## 1 4.6.3 Linear Discriminant Analysis

Let's return to the Smarket data from ISLR.

```
In [ ]: df = pd.read_csv('Smarket.csv', usecols=range(1,10), index_col=0, parse_dates=True)
        df.head()
```

Now we will perform LDA on the Smarket data from the ISLR package. In Python, we can fit a LDA model using the LDA() function, which is part of the lda module of the sklearn library. As we did with logistic regression and KNN, we'll fit the model using only the observations before 2005, and then test the model on the data from 2005.

```
In [ ]: X_train = df[:'2004'][['Lag1','Lag2']]
        y_train = df[:'2004']['Direction']

        X_test = df['2005':][['Lag1','Lag2']]
        y_test = df['2005':]['Direction']

        lda = LDA()
        model = lda.fit(X_train, y_train)

        print(model.priors_)
```

The LDA output indicates prior probabilities of $\hat{\pi}_1 = 0.492$ and $\hat{\pi}_2 = 0.508$; in other words, 49.2% of the training observations correspond to days during which the market went down.

```
In [ ]: print(model.means_)
```

The above provides the group means; these are the average of each predictor within each class, and are used by LDA as estimates of $\mu_k$. These suggest that there is a tendency for the previous 2 days' returns to be negative on days when the market increases, and a tendency for the previous days' returns to be positive on days when the market declines.

```
In [ ]: print(model.coef_)
```

The coefficients of linear discriminants output provides the linear combination of `Lag1` and `Lag2` that are used to form the LDA decision rule.

If $-0.0554 \times \texttt{Lag1} - 0.0443 \times \texttt{Lag2}$ is large, then the LDA classifier will predict a market increase, and if it is small, then the LDA classifier will predict a market decline. **Note**: these coefficients differ from those produced by `R`.

The `predict`() function returns a list of LDA's predictions about the movement of the market on the test data:

```
In [ ]: pred=model.predict(X_test)
        print(np.unique(pred, return_counts=True))
```

The model assigned 70 observations to the "Down" class, and 182 observations to the "Up" class. Let's check out the confusion matrix to see how this model is doing. We'll want to compare the **predicted class** (which we can find in `pred`) to the **true class** (found in `y_test`).

```
In [ ]: print(confusion_matrix(pred, y_test))
        print(classification_report(y_test, pred, digits=3))
```

We can also get the predicted probabilities using the `predict_proba`() function:

```
In [ ]: pred_p = model.predict_proba(X_test)
```

Applying a 50% threshold to the posterior probabilities allows us to recreate the predictions:

```
In [ ]: print(np.unique(pred_p[:,1]>0.5, return_counts=True))
```

Notice that the posterior probability output by the model corresponds to the probability that the market will **increase**:

```
In [ ]: print np.stack((pred_p[10:20,1], pred[10:20])).T
```

If we wanted to use a posterior probability threshold other than 50% in order to make predictions, then we could easily do so. For instance, suppose that we wish to predict a market decrease only if we are very certain that the market will indeed decrease on that day—say, if the posterior probability is at least 90%:

```
In [ ]: print(np.unique(pred_p[:,1]>0.9, return_counts=True))
```

No days in 2005 meet that threshold! In fact, the greatest posterior probability of decrease in all of 2005 was 54.2%:

```
In [ ]: max(pred_p[:,1])
```

# 2 4.6.4 Quadratic Discriminant Analysis

We will now fit a QDA model to the `Smarket` data. QDA is implemented in `sklearn` using the `QDA`() function, which is part of the `qda` module. The syntax is identical to that of `LDA`().

```
In [ ]: qda = QDA()
        model2 = qda.fit(X_train, y_train)
        print model2.priors_
        print model2.means_
```

The output contains the group means. But it does not contain the coefficients of the linear discriminants, because the QDA classifier involves a quadratic, rather than a linear, function of the predictors. The `predict`() function works in exactly the same fashion as for LDA.

```
In [ ]: pred2=model2.predict(X_test)
        print(np.unique(pred2, return_counts=True))
        print(confusion_matrix(pred2, y_test))
        print(classification_report(y_test, pred2, digits=3))
```

Interestingly, the QDA predictions are accurate almost 60% of the time, even though the 2005 data was not used to fit the model. This level of accuracy is quite impressive for stock market data, which is known to be quite hard to model accurately.

This suggests that the quadratic form assumed by QDA may capture the true relationship more accurately than the linear forms assumed by LDA and logistic regression. However, we recommend evaluating this method's performance on a larger test set before betting that this approach will consistently beat the market!

# 3 An Application to Carseats Data

Let's see how the `LDA/QDA` approach performs on the `Carseats` data set, which is included with `ISLR`.

Recall: this is a simulated data set containing sales of child car seats at 400 different stores.

```
In [ ]: df2 = pd.read_csv('Carseats.csv')
        df2.head()
```

See if you can build a model that predicts `ShelveLoc`, the shelf location (Bad, Good, or Medium) of the product at each store. Don't forget to hold out some of the data for testing!

```
In [ ]: # Your code here
```

To get credit for this lab, please post your answers to the following questions:

- What was your approach to building the model?
- How did your model perform?
- Was anything easier or more challenging than you anticipated?

to Piazza: https://piazza.com/class/igwiv4w3ctb6rg?cid=23