15. Discriminant Analysis (LDA and QDA)

Bruce E. Shapiro

Getting Started in Machine Learning

Last revised: March 22, 2019

Background & Assumptions

- Probabilistic Model based on Bayes' Rule

$$P(c_j|\mathbf{x}) = \frac{p(c_j)P(\mathbf{x}|c_j)}{\sum_{i=1}^{K} p(c_i)P(\mathbf{x}|c_i)}$$

- Assume Normal Distributions
- All features have same variance
- In multi-class discrimination:
  - LDA: All classes have same covariance matrix
  - QDA: Classes may have different covariance matrices

Normal Distributions

- Single Category: $P(x|c_j) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\mu_j)^2/(2\sigma^2)}$
- d Features, Same variances:
$$P(\mathbf{x}|c_j) = \frac{1}{\sqrt{(2\pi)^d \det \boldsymbol{\Sigma}}} e^{-(1/2)(\mathbf{x}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}(\mathbf{x}-\boldsymbol{\mu}_j)}$$
- Single Feature, Bayes' Rule (previous page):
$$P(c_j|x) = \frac{\frac{p(c_j)}{\sqrt{2\pi\sigma}} e^{-(x-\mu_j)^2/(2\sigma^2)}}{\sum_{i=1}^{K} \frac{p(c_i)}{\sqrt{2\pi\sigma}} e^{-(x-\mu_i)^2/(2\sigma^2)}} = \frac{1}{Z(x)} \frac{p(c_j)}{\sqrt{2\pi\sigma}} e^{-(x-\mu_j)^2/(2\sigma^2)}$$
where $Z(x)$ depends on x but not on j

Since $\ln f(u)$ is monotonically increasing with u, then u is maximized when $f(u)$ is maximized.

$$\ln P(c_j|x) = \ln p(c_j) - \frac{(x-\mu_j)^2}{2\sigma^2}$$
$$+ \text{terms that do not depend on } j$$

Define the **Discriminant**

$$\delta_j(x) = \ln p_j - \frac{(x-\mu_j)^2}{2\sigma^2}$$

For multiple factors,

$$\delta_j(x) = \mathbf{x}^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j - \frac{1}{2} \boldsymbol{\mu}_j^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j + \ln p(c_j)$$

LDA/QDA Python - Data Set - 2 Features

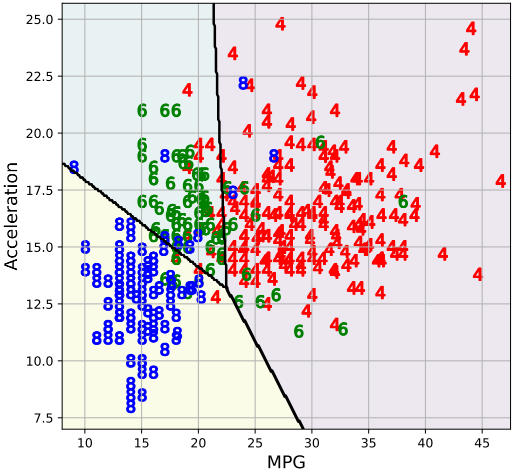
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data=pd.read_fwf("https://archive.ics.uci.edu/ml/
  machine-learning-databases/auto-mpg/auto-mpg.data",
  header=None,na_values="?")
data.columns=("mpg","cyl","displ","hp","weight",
  "accel","model","origin","carname")
data = data.dropna(axis=0)

cardata=np.array(data[["cyl","mpg","accel"]])
cars=np.array([line for line in cardata
   if line[0] in [4,6,8]])
Y=cars[:,0]/2-2
X=cars[:,1:]
```

LDA in Python (2D Car Cylinders)

```
from sklearn.discriminant_analysis import
        LinearDiscriminantAnalysis
errs=[]
nsplits=100
for j in range(nsplits):
        XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
        LDA = LinearDiscriminantAnalysis()
        LDA.fit(XTRAIN,YTRAIN)
        YP=LDA.predict(XTEST)
        errs.append(1-accuracy_score(YTEST,YP))
print("LDA mean error=%7.5f std=%7.5f"
        %(np.mean(errs),np.std(errs)))
```

```
LDA mean error=0.05199 std=0.02215
```

LDA decision boundary

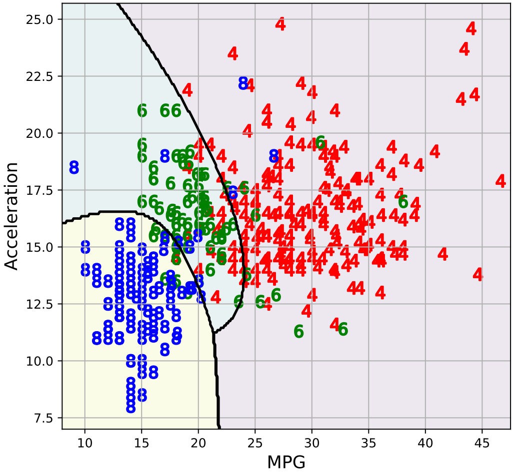


QDA in Python (Car Cylinders)

```
from sklearn.discriminant_analysis import
   QuadraticDiscriminantAnalysis as QDAerrs=[]
nsplits=100
for j in range(nsplits):
    XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
    model = QDA()
    model.fit(XTRAIN,YTRAIN)
    YP=model.predict(XTEST)
    errs.append(1-accuracy_score(YTEST,YP))
print("QDA mean=%7.5f std=%7.5f"
        %(np.mean(errs),np.std(errs)))
```

```
QDA mean=0.13835 std=0.02989
```

QDA Decision Boundary



Repeat with 4 features, same Classes

```
X=np.array(data[["cyl","mpg","displ","hp","accel"]])
X=np.array([line for line in X if line[0] in [4,6,8]])
Y=X[:,0]/2-2
X=X[:,1:]

... code for LDA here
print( ... LDA results ...)

... code for QDA here
print( ... QDA results ...)
```

```
LDA mean error=0.05220 std=0.02088
QDA mean error=0.03515 std=0.01429
```

References

1. MPG data from: Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.