Problem Set 5: Intro to Discriminant Analysis

Stat 154, Fall 2017, Prof. Sanchez

Due date: Sun Nov-05 (before midnight)

Introduction

In this assignment you will be working with one of the wine data sets from the UCI Machine Learning Repository. The main purpose is to work around the concepts of how total dispersion can be broken down into between-groups and within-groups dispersion. These concepts are the root of linear discriminant analysis and quadratic discriminant analysis.

Wine Data Set

The data set is the *Wine Data Set*:

https://archive.ics.uci.edu/ml/datasets/wine

I've downloaded a copy of the data file to the course github repository, inside the `data/` folder: `wine.data`.

This data set contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars (i.e type of plant). The analysis determined the quantities of 13 constituents (i.e. predictors) found in each of the three types of wines.

The variables in the `wine.data` file are:

- `class`
- `alcohol`
- `malic`
- `ash`
- `alcalinity`
- `magnesium`
- `phenols`
- `flavanoids`
- `nonflavanoids`
- `proanthocyanins`
- `color`
- `hue`
- `dilution`
- `proline`

1) Sum-of-Squares Dispersion Functions (10 pts)

Consider a variable X and a categorical variable Y. Assume that Y represents the class (or group) of each observation. Let k be an index for the classes: $k = 1, \dots, K$; each class is of size n_k.

Function `tss()`: write a function `tss()` that computes the total sum of squares of a given variable:

$$\text{TSS} = \sum_{i=1}^{n} (x - \bar{x})^2$$

The function `tss()` should take one argument `x`, the input vector.

Here's how you should be able to call `tss()`

```
tss(iris$Sepal.Length)
```

```
## [1] 102.1683
```

Function `bss()`: write a function `bss()` that computes the between groups sum of squares:

$$\text{BSS} = \sum_{k=1}^{K} n_k (\bar{x}_k - \bar{x})^2$$

where:

- n_k is the number of individuals in class k
- $\bar{x}_k$ is the average of class k

The function `bss()` takes two arguments:

- `x` = vector for the predictor variable
- `y` = vector (or factor) for the response variable
- *include a `stop()` statement if `x` and `y` have different lengths*

Here's how you should be able to call `bss()`

```
bss(iris$Sepal.Length, iris$Species)
```

```
## [1] 63.21213
```

Function `wss()`: write a function `wss()` that computes the wetween groups sum of squares:

$$\text{WSS} = \sum_{k=1}^{K} \sum_{i \in G_k} (x_{ik} - \bar{x}_k)^2$$

where:

- x_{ik} is the i-th individual in class k
- $\bar{x}_k$ is the average of group k
- G_k represents the group of individuals in class k

The function `wss()` takes two arguments:

- `x` = vector for the predictor variable
- `y` = vector (or factor) for the response variable
- *include a `stop()` statement if `x` and `y` have different lengths*

Here's how you should be able to call `wss()`

```
wss(iris$Sepal.Length, iris$Species)
```

```
## [1] 38.9562
```

2) Sum-of-Squares Ratio Functions (10 pts)

Function `cor_ratio()`: use `bss()` and `tss()` to write a function `cor_ratio()` that computes the correlation ratio *eta*^{2} between a variable `x` and a response `y`.

$$\eta^2(x, y) = \frac{\text{BSS}}{\text{TSS}}$$

Here's how you should be able to call `cor_ratio()`

```
cor_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 0.6187057
```

Function `F_ratio()`: use `bss()` and `tss()` to write a function `F_ratio()` that computes the F-ratio between a variable `x` and a response `y`.

$$F = \frac{\text{BSS}/(k-1)}{\text{WSS}/(n-k)}$$

Here's how you should be able to call `F_ratio()`

```
F_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 119.2645
```

---

3) Discriminant Power of Predictors (30 pts)

This part of the assignment involves ranking the predictors using three approches: simple logistic regressions, correlation ratios, and F-ratios.

The first approach consists of running simple logistic regressions (10 pts):

- Run simple logistic regressions for each predictor and the response, and store the values of the AIC statistic.
- Make a table (e.g. data frame) with the predictors ranked by AIC value in increasing order. The smallest the AIC, the more discriminant the predictor.
- Display the AICs in a barchart.

The second approach consists of computing correlation ratios (10 pts):

- Calculate correlation ratios for each predictor and the response.
- Make a table (e.g. data frame) with the predictors ranked by η^2 value in increasing order. The largest the η^2, the more discriminant the predictor.
- Display the η^2's in a barchart.

The third approach consists of computing F-ratios (10 pts):

- Calculate F-ratios for each predictor and the response.
- Make a table (e.g. data frame) with the predictors ranked by F-value in increasing order. The largest the F, the more discriminant the predictor.
- Display the F-values in a barchart.

How do the AIC values from the logistic regression compare to the sum of squares ratios?

---

4) Variance functions (30 pts)

Function `total_variance()`: Write a function `total_variance()` that takes a matrix of predictors, and returns the (sample) variance-covariance matrix $\mathbf{V}$. Do NOT use `var()` to create `total_variance()`.

Here's how yo should be able to invoke `total_variance()`, and compare it with the `var()` function. (10 pts)

```r
# test total_variance()
total_variance(iris[ ,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```r
# compare with var()
var(iris[ ,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

Function `between_variance()`: Write a function `between_variance()` that takes a matrix of predictors, and a response vector (or factor), and returns the (sample) Between-variance matrix $\mathbf{B}$. Do NOT use `var()` to create `between_variance()`.

Here's how yo should be able to invoke `between_variance()` on `iris` data (10 pts)

```r
# test between_variance()
between_variance(iris[ ,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.4242425 -0.13391051    1.1090497   0.4783848
## Sepal.Width    -0.1339105  0.07614049   -0.3841584  -0.1539105
## Petal.Length    1.1090497 -0.38415839    2.9335758   1.2535168
## Petal.Width     0.4783848 -0.15391051    1.2535168   0.5396868
```

Function `within_variance()`: Write a function `within_variance()` that takes a matrix of predictors, and a response vector (or factor), and returns the (sample) Within-variance matrix $\mathbf{W}$. Do NOT use `var()` to create `within_variance()`.

Here's how yo should be able to invoke `within_variance()` on `iris` data (10 pts)

```r
# test within_variance()
within_variance(iris[ ,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.26145101  0.09147651   0.16526577  0.03788591
## Sepal.Width    0.09147651  0.11383893   0.05450201  0.03227114
## Petal.Length   0.16526577  0.05450201   0.18270201  0.04209262
## Petal.Width    0.03788591  0.03227114   0.04209262  0.04131946
```

Confirm that $\mathbf{V} = \mathbf{B} + \mathbf{W}$

```r
# confirm V = B + W
Viris <- total_variance(iris[ ,1:4])
Viris
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```
# B + W
Biris <- between_variance(iris[ ,1:4], iris$Species)
Wiris <- within_variance(iris[ ,1:4], iris$Species)
Biris + Wiris
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

---

5) Canonical Discriminant Analysis

Cacnonical Discriminant Analysis seeks to obtain one or more linear combinations of the predictors $\mathbf{z_k} = \mathbf{X}\mathbf{u_k}$ such that they *separate* as much as possible the groups (conveyed by $\mathbf{y}$). The maximum number of linear combinations is $k - 1$.

The optimal separation can be found by maximizing the criterion:

$$\frac{\mathbf{u}^\mathsf{T}\mathbf{B}\mathbf{u}}{\mathbf{u}^\mathsf{T}\mathbf{W}\mathbf{u}} \quad \text{s.t.} \quad \mathbf{u}^\mathsf{T}\mathbf{W}\mathbf{u} = 1$$

The solution is given by:

$$\mathbf{W}^{-1}\mathbf{B}\mathbf{u} = \lambda\mathbf{u}$$

that is, $\mathbf{u}$ is eigenvector of $\mathbf{W}^{-1}\mathbf{B}$.

The problem is that $\mathbf{W}^{-1}\mathbf{B}$ is not a symmetric matrix. Hence, it is more convenient to work with $\mathbf{B} = \mathbf{C}\mathbf{C}^\mathsf{T}$ where matrix $\mathbf{C}$ has general term:

$$c_{jk} = \sqrt{\frac{n_k}{n-1}(\bar{x}_{kj} - \bar{x}_j)}$$

The $p \times p$ matrix $\mathbf{W}^{-1}\mathbf{B}$ and the $k \times k$ matrix $\mathbf{C}^\mathsf{T}\mathbf{W}^{-1}\mathbf{C}$ have the same eigenvalues.

Their eigenvectors are related by:

$$\mathbf{u} = \mathbf{W}^{-1}\mathbf{C}\mathbf{w}$$

Thus, we can diagonalize (i.e. EVD) the following symmetric matrix:

$$\mathbf{C}^\mathsf{T}\mathbf{W}^{-1}\mathbf{C}$$

and then use the eigenvector $\mathbf{w}$ to recover $\mathbf{u}$

Challenge (70 pts)

- Use the predictors and response of the wine data, to write code in R that allows you to find the eigenvectors $\mathbf{u_k}$. (20 pts)
- Obtain the linear combinations $\mathbf{z_k}$ and make a scatterplot of the wines. Add color to the dots indicating the different classes. (10 pts)
- Obtain a scatterplot of the wines but this time using the first two principal components on the standardized predictors. Add color to the dots indicating the different classes. How does this compare to the previous scatterplot? (10 pts)
- Calculate the correlations between $\mathbf{z_k}$ and the predictors. How do you interpret each score? (10 pts)
- Create a matrix of size $n \times K$, with the squared Mahalanobis distances $d^2(\mathbf{x_i}, \mathbf{g_k})$ of each observation $\mathbf{x_i}$ (i.e. each wine) to the each of the k centroids $\mathbf{g_k}$. The squared distance, with the Mahalanobis metric, is given by:

$$d^2(\mathbf{x_i}, \mathbf{g_k}) = (\mathbf{x_i} - \mathbf{g_k})^\mathsf{T}\mathbf{W}^{-1}(\mathbf{x_i} - \mathbf{g_k})$$

- Finally, assign each observation to the class G_k for which the Mahalanobis distance $d^2(\mathbf{x_i}, \mathbf{g_k})$ is the smallest. And create a confussion matrix comparing the actual `class` versus the predicted class. (20 pts)