HW08: Canonical Discriminant Analysis

Stat 154, Fall 2019

Introduction

In this assignment you will be working with the famous `iris` data set, collected by Edgar Anderson in 1935. The main purpose is to work around the underlying concepts of Canonical Discriminant Analysis. More especifically, how total dispersion (i.e. total sum of squares) can be broken down into between-groups and within-groups dispersion. These concepts are the root of linear discriminant analysis and quadratic discriminant analysis.

1) Functions for Sum-of-Squares

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: $k = 1, \ldots, K$; each class is of size n_k; and $n = n_1 + \cdots + n_K$

1.1) 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)
```

1.2) 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
- check that `x` and `y` have equal length, otherwise `stop()` execution

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

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

1.3) Function `wss()`

Write a function `wss()` that computes the within 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
- check that `x` and `y` have equal length, otherwise `stop()` execution

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

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

2) Functions for Ratios of Sum-of-Squares

2.1) Function `cor_ratio()`

Use `bss()` and `tss()` to write a function `cor_ratio()` that computes the correlation ratio η^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)
```

2.2) 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)
```

---

3) Discriminant Power of Predictors

For this part of the lab, you will rank the predictors (e.g. `Sepal.Length`, `Sepal.Width`, `Petal.Length`, and `Petal.Width`) using two approaches: 1) correlation ratios, and 2) F-ratios.

The first approach consists of computing correlation ratios:

- 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 second approach consists of computing F-ratios:

- 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.

---

4) Variance functions

4.1) Function `total_variance()`

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

The variance-covariance matrix $\mathbf{V}$ is given by (assuming mean-centered data $\mathbf{X}$):

$$\mathbf{V} = \frac{1}{n-1}\mathbf{X}^\mathsf{T}\mathbf{X}$$

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

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

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

4.2) 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 $\mathbf{B}$. You are NOT allowed to use the function `var()` to create `between_variance()`.

The matrix $\mathbf{B}$ is given by:

$$\mathbf{B} = \sum_{k=1}^{K} \frac{n_k}{n-1}(\mathbf{g_k} - \mathbf{g})(\mathbf{g_k} - \mathbf{g})^\mathsf{T}$$

where:

- $\mathbf{g}$ is the overall centroid

- $\mathbf{g_k}$ is the centroid of the k-th class

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

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

4.3) 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()`.

The matrix $\mathbf{W}$ is given by:

$$\mathbf{W} = \sum_{k=1}^{K} \frac{n_k - 1}{n - 1} \mathbf{W_k}$$

where:

- $\mathbf{W_k}$ is the variance matrix for each group (mean-centered $\mathbf{X_k}$ with respect to $\mathbf{g_k}$):

$$\mathbf{W_k} = \frac{1}{n_k - 1} \mathbf{X_k^T X_k}$$

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

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

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

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

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

5) Canonical Discriminant Analysis (CDA)

As we saw in class, CDA involves looking for linear combinations of the predictors, $\mathbf{z_k} = \mathbf{X u_k}$, in order to obtain a low-dimensional space that separates the groups as much as possible. The desired vectors $\mathbf{u}$ are obtaining by maximizing one of the two equivalent *discriminant criteria*:

$$max \left\{ \frac{\mathbf{u^T B u}}{\mathbf{u^T V u}} \right\} \quad \Longleftrightarrow \quad max \left\{ \frac{\mathbf{u^T B u}}{\mathbf{u^T W u}} \right\}$$

It can be shown that $\mathbf{u}$ is eigenvector of $\mathbf{W^{-1}B}$

$$\mathbf{W^{-1}Bu} = \rho \mathbf{u}$$

The problem is that, in general, $\mathbf{W^{-1}B}$ is not symmetric. So what can we do? We can look for a similar matrix to $\mathbf{W^{-1}B}$ that is symmetric. How? One option is to factorize $\mathbf{B}$ as:

$$\mathbf{B} = \mathbf{C C^T}$$

where $\mathbf{C}$ has general term:

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

We can then use $\mathbf{C}$ to compute $\mathbf{C^T W^{-1} C}$. It turns out that the $p \times p$ matrix $\mathbf{W^{-1}B}$ and the $k \times k$ matrix $\mathbf{C^T W^{-1} C}$ have the same eigenvalues.

Their eigenvectors are related by:

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

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

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

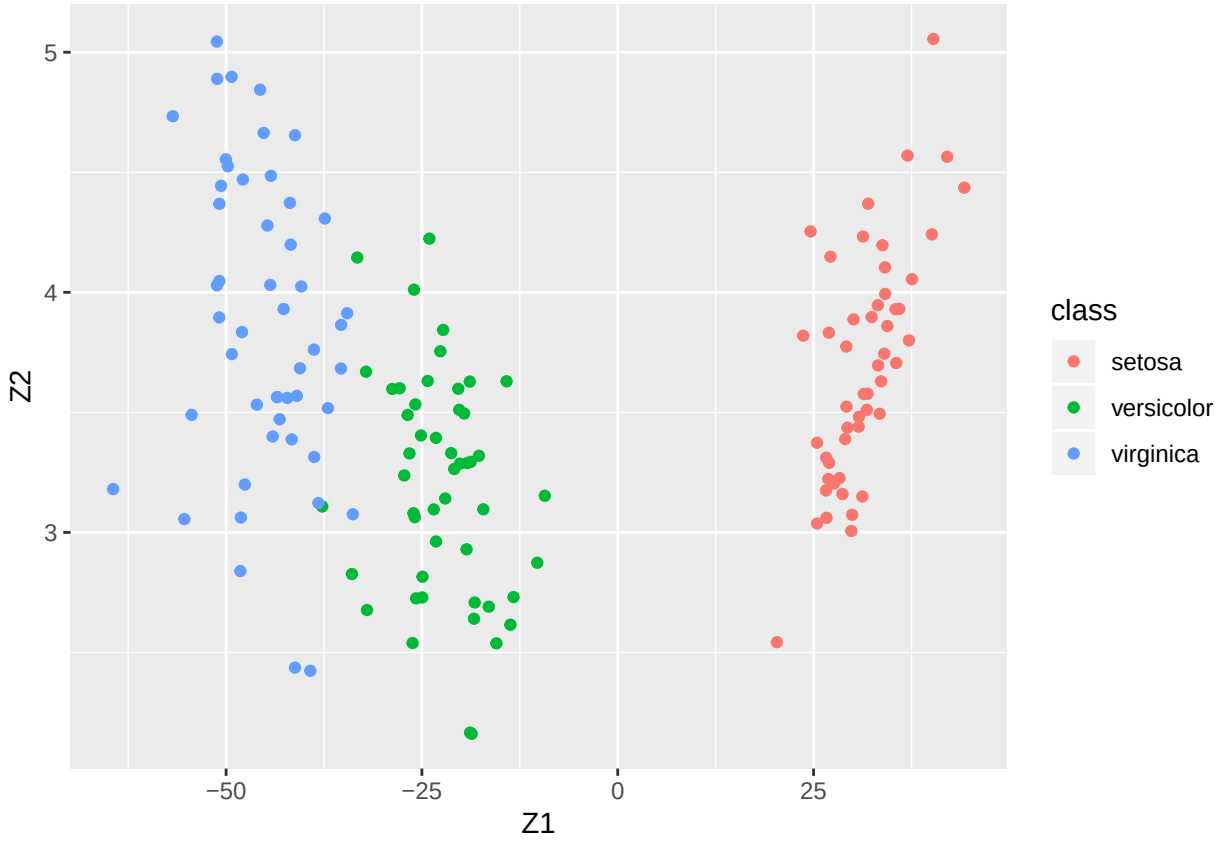
and then use the eigenvector $\mathbf{w}$ to recover $\mathbf{u}$

Coding CDA

Use the predictors and response of the `iris` data, to write code that allows you to find the eigenvectors $\mathbf{u_k}$. In other words:

- Create the matrix $\mathbf{C}$

- Obtain EVD of $\mathbf{C}^{\mathbf{T}}\mathbf{W}^{-\mathbf{1}}\mathbf{C}$; this will give you eigenvectors $\mathbf{w}$. Use eigenvectors $\mathbf{w}$ to obtain eigevectors $\mathbf{u} = \mathbf{W}^{-\mathbf{1}}\mathbf{C}\mathbf{w}$
- Compute canonical variables $\mathbf{z_1} = \mathbf{X}\mathbf{u_1}$ and $\mathbf{z_2} = \mathbf{X}\mathbf{u_2}$, and use them to plot the `iris` data: i.e. get a scatterplot in which iris classes are best separated.



It is possible that the scale of your scatterplot is different, or even that the shape is different (e.g. you may have an inverted image of my plot). The important thing is the relative position of the cloud of points.

- Obtain a scatterplot of the `iris` data 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?

6) CDA for Classification

As we also saw in class, CDA can also be used for prediction purposes. Each observation $\mathbf{x_i}$ is classed in the group G_k for which the distance to the centroid $\mathbf{g_k}$ is minimal, the distance being calculated using the Mahalanobis metric $\mathbf{W}^{-\mathbf{1}}$

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

The rule of classification in Canonical Discriminant Analysis is a geometric rule. Moreover, it is a linear rule with respect to the predictors (which is why some authors speak of CDA as a type of **linear discriminant analysis**).

Compute the distances to the centroids of *setosa*, *virginica*, and *versicolor* for the following test points, and determine the predicted class for each of them:

- $x_1 = (5.0, 3.0, 1.5, 0.5)$
- $x_2 = (5.5, 3.0, 6.0, 2.0)$
- $x_3 = (6.0, 3.0, 4.0, 1.0)$
- $x_4 = (5.0, 3.0, 1.0, 0.5)$