Assignment 10

Vishal

2022-12-07

1. Linear discriminant analysis

Q1.

Probabilistic LDA is a generative model which assumes that given data samples are generated from a distribution (e.g. Gaussian distribution). We need to find the parameters of model which best describe the training data.

```
library(Stat2Data)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.2.2
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(ggplot2)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.2.2
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-6
```

```
library(QSARdata)
data(Hawks)
```

```
hawks_total <- Hawks %>%
  dplyr::select(Weight, Wing, Hallux, Tail, Species) %>%
  filter(Species =='SS' | Species =='CH') %>% drop_na() %>%
  mutate(Species = as.numeric(Species =='SS'))

head(hawks_total)
```

```
##   Weight Wing Hallux Tail Species
## 1    470  265   23.5  220       0
## 2    170  205   14.3  157       1
## 3    180  205   15.0  164       1
## 4    100  193  101.0  144       1
## 5     88  171   11.5  136       1
## 6    324  233   19.3  191       0
```

```
num_total <- hawks_total %>% nrow()
num_train <- floor(num_total * 0.6)
num_test <- num_total-num_train
set.seed(0)
test_inds <- sample(seq(num_total), num_test)
train_inds <- setdiff(seq(num_total), test_inds)
hawks_train <- hawks_total %>%
  filter(row_number() %in% train_inds)

hawks_test <- hawks_total %>%
  filter(row_number() %in% test_inds)
```

```
hawks_train_x <- hawks_train %>%
  dplyr::select(-Species)
hawks_train_y <- hawks_train %>%
  pull(Species)

hawks_test_x <- hawks_test %>%
  dplyr::select(-Species)
hawks_test_y <- hawks_test %>%
  pull(Species)
```

```
lda_model <- MASS::lda(Species ~ ., hawks_train)
```

Q2.

```
lda_train_predicted_y <- predict(lda_model, hawks_train_x)$class %>%
  as.character() %>% as.numeric()

lda_train_error <- mean(abs(lda_train_predicted_y - hawks_train_y))
lda_train_error
```

```
## [1] 0.04639175
```

```
lda_test_predicted_y <- predict(lda_model, hawks_test_x)$class %>%
  as.character() %>% as.numeric()

lda_test_error <- mean(abs(lda_test_predicted_y - hawks_test_y))
lda_test_error
```

```
## [1] 0.01538462
```

Q3.

```
data("iris")

iris_total <- iris %>%  drop_na() %>%
  mutate(Species = as.numeric(Species))

head(iris_total)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2       1
## 2          4.9         3.0          1.4         0.2       1
## 3          4.7         3.2          1.3         0.2       1
## 4          4.6         3.1          1.5         0.2       1
## 5          5.0         3.6          1.4         0.2       1
## 6          5.4         3.9          1.7         0.4       1
```

```
num_total <- iris_total %>% nrow()
num_train <- floor(num_total * 0.6)
num_test <- num_total - num_train
set.seed(0)
test_inds <- sample(seq(num_total), num_test)
train_inds <- setdiff(seq(num_total), test_inds)
iris_train <- iris_total %>%
  filter(row_number() %in% train_inds)

iris_test <- iris_total %>%  filter(row_number() %in% test_inds)
```

```r
iris_train_x <- iris_train %>% dplyr::select(-Species)
iris_train_y <- iris_train %>% pull(Species)

iris_test_x <- iris_test %>% dplyr::select(-Species)
iris_test_y <- iris_test %>% pull(Species)
```

```r
lda_model <- MASS::lda(Species ~ ., iris_train)
```

```r
lda_train_predicted_y <- predict(lda_model, iris_train_x)$class %>%
  as.character() %>% as.numeric()

lda_train_error <- mean(abs(lda_train_predicted_y - iris_train_y))
lda_train_error
```

```
## [1] 0.01111111
```

```r
lda_test_predicted_y <- predict(lda_model, iris_test_x)$class %>%
  as.character() %>% as.numeric()

lda_test_error <- mean(abs(lda_test_predicted_y - iris_test_y))
lda_test_error
```

```
## [1] 0.01666667
```

2. Logistic regression

Q1.

We know that, in Logistic regression target variable follows Bernoulli distribution. Probability mass function of Bernoulli distribution is

$$\mathbb{P}(y) = p^y * (1-p)^{(1-y)}$$

We also know that, Generalized Linear Model is of the form:

$$y = w^T x + \epsilon i.e. f(E(y)) = w^T x$$

Logit function is a link function in this kind of Generalized Linear Models. Logit function is defined as : log(odds) i.e.
odds = (success/failure).

$$odds = \frac{p}{(1-p)} log(odds) = log(\frac{p}{1-p}) \Rightarrow f(p) = log(\frac{p}{1-p})$$

For Bernoulli distribution $E(y) = p$. Hence equation becomes, $f(E(y)) = f(p)$

$f(p) = f(E(y))log(fracp1 - p) = w^T x$Taking exponential, $\frac{p}{1-p} = exp(w^T x)p = (1-p)exp(w^T x)p = exp(w^T x) - p.exp(w^T x)$

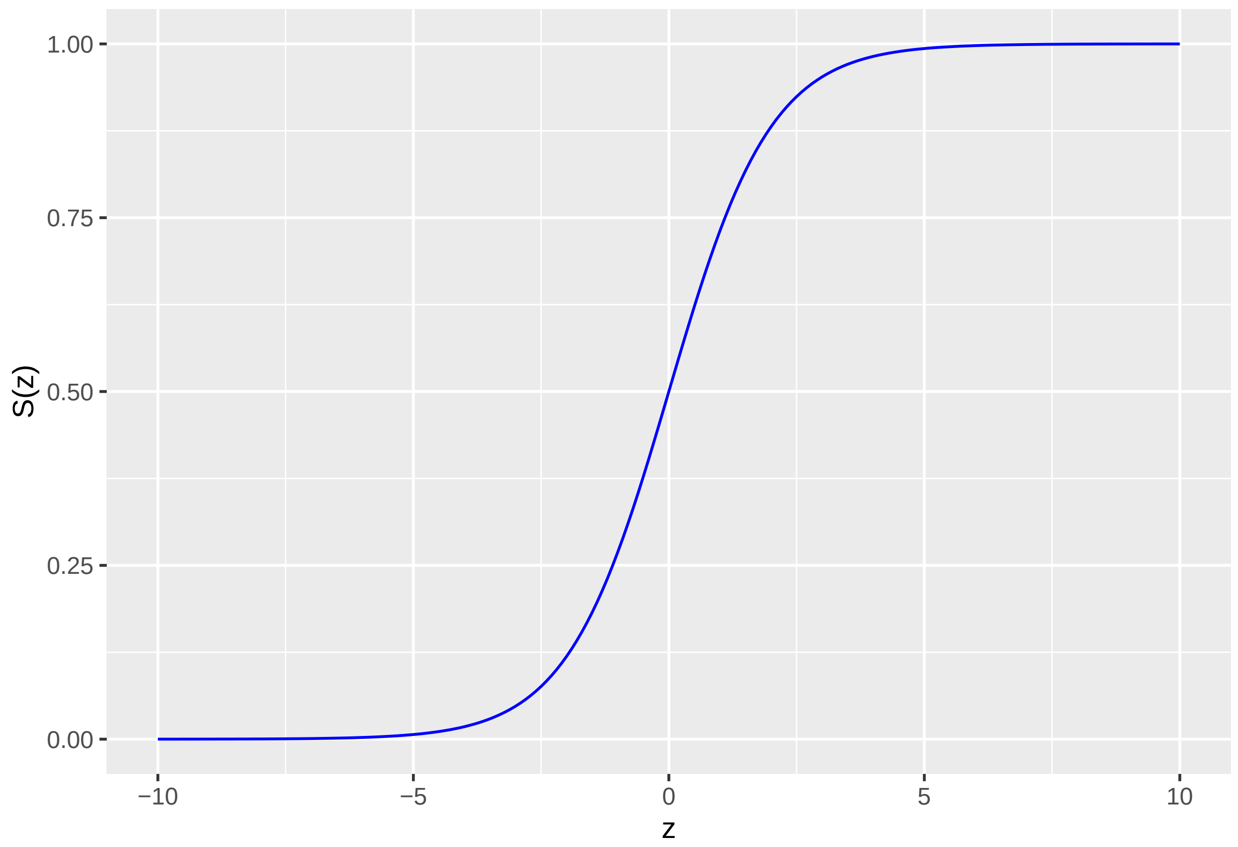
Q2.

```
sigmoid <- function(z) {
  S_z <- 1 / (1 + exp(-z))
  return(S_z)
}

z <- seq(-10, 10, 0.1)

temp_df <- data.frame(z = seq(-10, 10, 0.1)) %>%
  mutate(S_z = map_dbl(z, sigmoid))

ggplot(temp_df, aes(x = z, y = S_z)) +
  geom_line(color = "blue") +
  labs(y = 'S(z)')
```



```
  theme_bw()
```

Q3.

```r
logistic_model <- glmnet(x = hawks_train_x %>% as.matrix(),
                      y = hawks_train_y,
                      family = 'gaussian',
                      alpha = 0,
                      lambda = 0)
```

```r
logistic_train_predicted_y <- predict(logistic_model, hawks_train_x %>%
                                        as.matrix(),
                                      type = 'class') %>%
  as.integer()
logistic_train_error <- mean(abs(logistic_train_predicted_y - hawks_train_y))

logistic_train_error
```

```
## [1] 0.4536082
```

```r
logistic_test_predicted_y <- predict(logistic_model, hawks_test_x %>%
                                       as.matrix(),
                                     type = 'class') %>%
  as.integer()

logistic_test_error <- mean(abs(logistic_test_predicted_y - hawks_test_y))

logistic_test_error
```

```
## [1] 0.4846154
```

3. Basic concepts in regularisation

Q1.

1. Hyper-parameter:

Hyper-parameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. They are used to to control the learning process and the model parameters that result from it.
Learning rate & number of epochs are examples of hyper-parameters.

2. Validation data:

A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's hyper-parameters.
It is used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models.

3. The train-validation-test split:

The train-test-validation-test split is a technique to evaluate the performance of a machine learning model.
A given dataset is divided into three subsets for different purposes.
Train datset is used to train the model initially.
Validation dataset is used to provide an unbiased evaluation of a model fitted on the training dataset while tuning hyper-parameters.
Whereas, test dataset is used to provide an unbiased evaluation of a final model fitted on the training dataset.

Q2.

l_2 **norm:**
The Euclidean norm of a vector which is a point on a line, surface, or hyper-surface may be interpreted geometrically as the distance between this point and the origin.

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_n^2}$$

$l_1 norm$:
l_1 norm is also known as "Manhattan Distance or Taxicab norm." It is the sum of the magnitudes of the vectors in a space i.e. sum of absolute difference of the components of vectors.

$$||x||_1 = |x_1| + |x_2| + |x_3| + ... + |x_n|$$

Q3.

Lasso Regression Regularization:
It is a shrinkage technique. It stands for Least Absolute Shrinkage and Selection Operator. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2 + \lambda\sum_{j=1}^{m}|\hat{\beta}_j|$$

Ridge Regression:
Similar to the lasso regression, ridge regression puts a similar constraint on the coefficients by introducing a penalty factor. However, while lasso regression takes the magnitude of the coefficients, ridge regression takes the square.

$$L_{hridge}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2 + \lambda\sum_{j=1}^{m}w_j\hat{\beta}_j^{\,2}$$

Ridge regression is also referred to as L2 Regularization.

4. An investigation into ridge regression for high-dimensional regression

```
data(MeltingPoint)
```

```
mp_data_total <- MP_Descriptors %>%
  mutate(melting_pt=MP_Outcome)
```

There are **203** variables in the **mp_data_total** data frame. And there are **4401** examples.

Q2.

```
spec = c(train = .5, test = .25, val = .25)

g = sample(cut(
  seq(nrow(mp_data_total)),
  nrow(mp_data_total) * cumsum(c(0, spec)),
  labels = names(spec)
))

res <- split(mp_data_total, g)

mp_data_train <- res$train
mp_data_test <- res$test
mp_data_val <- res$val
```

Q3.

```
val_error_calculator <- function(train_data, val_data, ld) {

  train_x <- train_data %>%
    dplyr::select(-melting_pt)

  train_y <- train_data %>%
    pull(melting_pt)

  val_x <- val_data %>%
    dplyr::select(-melting_pt)

  val_y <- val_data %>%
    pull(melting_pt)

  logistic_model <- glmnet(x = train_x %>% as.matrix(),
                          y = train_y,
                          family = 'gaussian',
                          alpha = 1,
                          lambda = ld)

  logistic_val_predicted_y <- predict(logistic_model, val_x %>%
                                        as.matrix(),
                                      type = 'class') %>%
    as.integer()

  logistic_val_error <- mean(abs(logistic_val_predicted_y - val_y))

  return(logistic_val_error)
}

val_error_calculator(mp_data_train, mp_data_val, 0.1)
```

```
## [1] 35.85468
```

Q4.

```
mySeq <- seq(0, 70)
lambdas <- c()
for (i in mySeq) {
  temp <- (1.25 ^ i) * (10 ^ -5)
  lambdas <- c(lambdas, temp)
}

lambdas
```

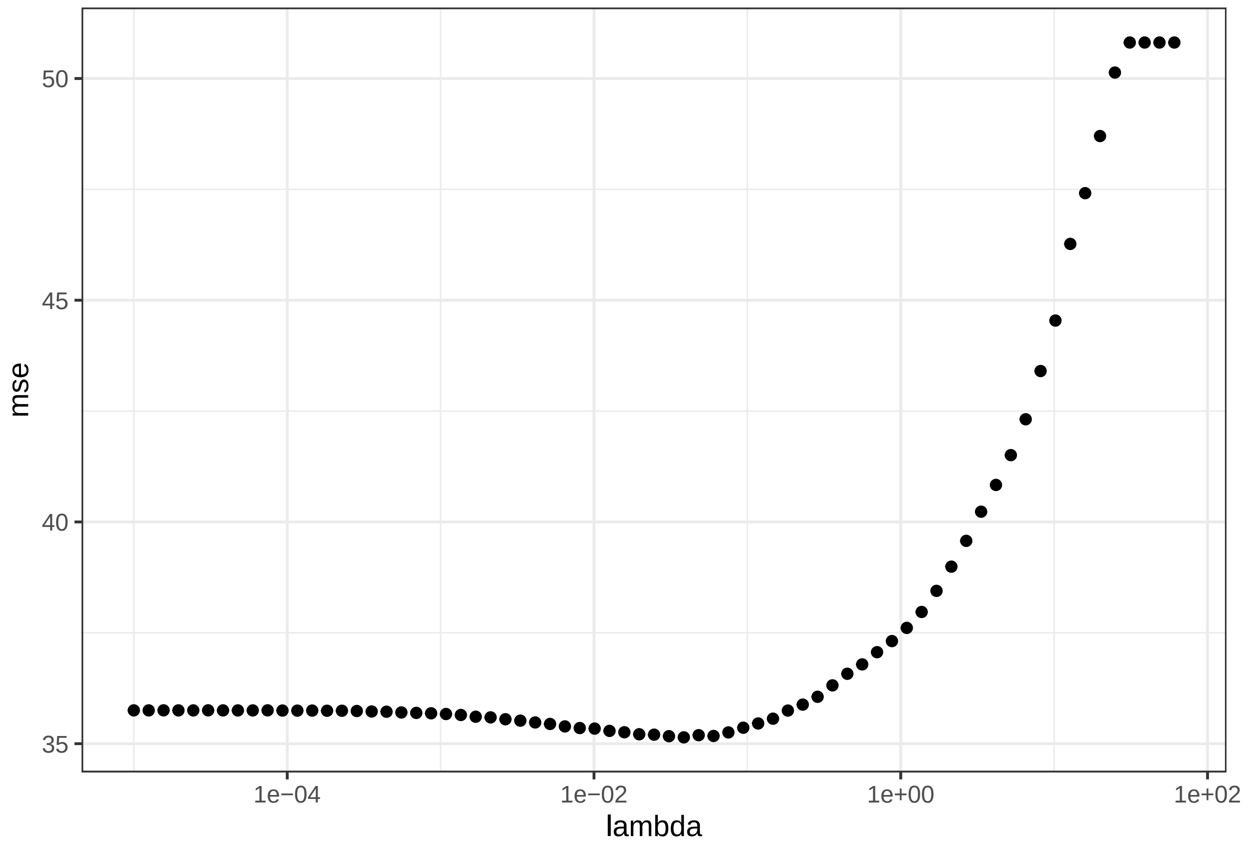
```
##  [1] 1.000000e-05 1.250000e-05 1.562500e-05 1.953125e-05 2.441406e-05
##  [6] 3.051758e-05 3.814697e-05 4.768372e-05 5.960464e-05 7.450581e-05
## [11] 9.313226e-05 1.164153e-04 1.455192e-04 1.818989e-04 2.273737e-04
## [16] 2.842171e-04 3.552714e-04 4.440892e-04 5.551115e-04 6.938894e-04
## [21] 8.673617e-04 1.084202e-03 1.355253e-03 1.694066e-03 2.117582e-03
## [26] 2.646978e-03 3.308722e-03 4.135903e-03 5.169879e-03 6.462349e-03
## [31] 8.077936e-03 1.009742e-02 1.262177e-02 1.577722e-02 1.972152e-02
## [36] 2.465190e-02 3.081488e-02 3.851860e-02 4.814825e-02 6.018531e-02
## [41] 7.523164e-02 9.403955e-02 1.175494e-01 1.469368e-01 1.836710e-01
## [46] 2.295887e-01 2.869859e-01 3.587324e-01 4.484155e-01 5.605194e-01
## [51] 7.006492e-01 8.758115e-01 1.094764e+00 1.368456e+00 1.710569e+00
## [56] 2.138212e+00 2.672765e+00 3.340956e+00 4.176195e+00 5.220244e+00
## [61] 6.525304e+00 8.156631e+00 1.019579e+01 1.274474e+01 1.593092e+01
## [66] 1.991365e+01 2.489206e+01 3.111508e+01 3.889385e+01 4.861731e+01
## [71] 6.077163e+01
```

Q5.

```
df <- data.frame(lambda = lambdas) %>%
  mutate(mse = map_dbl(.x = lambda, .f = ~val_error_calculator(mp_data_train, mp_data_test, .x)))
```

Q6.

```
ggplot(df, aes(x = lambda, y = mse)) +
  geom_point() +
  scale_x_log10() +
  theme_bw()
```



Q7.

```
df[which.min(df$mse), ]
```

```
##       lambda    mse
## 38 0.0385186 35.143
```

```
val_error_calculator(mp_data_train, mp_data_test, 0.0385186)
```

```
## [1] 35.143
```