# Assignment10

Zhen Liu

2022-12-07

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

# Linear Discriminant analysis

## Q1

Let's fit a Gaussian distribution to the data from each of the two classes.

Then we obtain a probabilistic model for the data. We can then use this probabilistic model to generate a rule based on the probability of the two classes.

## Q2

```
library(Stat2Data)
data(Hawks)

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

```
num_total <- hawks_total %>% nrow()
# number of penguin data
num_train <- floor(num_total*0.6)
# number of train examples
num_test <- num_total-num_train
# number of test samples
set.seed(0) # set random seed for reproducibility


test_inds <- sample(seq(num_total),num_test) # random sample of test indicies
train_inds <- setdiff(seq(num_total),test_inds) # training data indicies

hawks_train <- hawks_total %>% filter(row_number() %in% train_inds) # train data
hawks_test <- hawks_total %>% filter(row_number() %in% test_inds) # test data
```

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

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

lda_train_error<-mean(abs(lda_train_predicted-hawks_train$Species))

lda_train_error
```

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

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

lda_test_error<-mean(abs(lda_test_predicted-hawks_test$Species))

lda_test_error
```

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

# logistic regression

## Q1

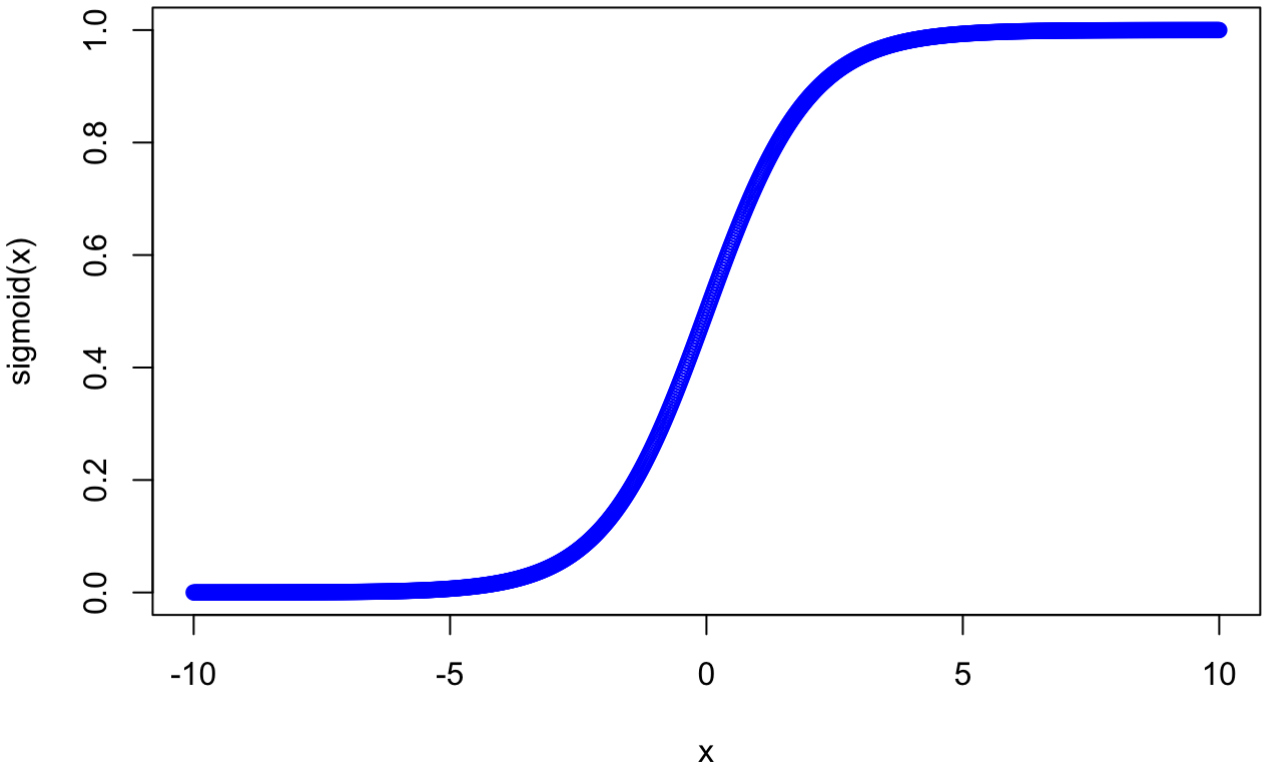
In logistic regression the target variable follow Bernoulli distribution. The probabilistic model is the probability mass function of Bernoulli distribution.

## Q2

```
sigmoid = function(x) {
   1 / (1 + exp(-x))
}
```

```
x<- seq(-10,10,0.02)
```

```
plot(x,sigmoid(x),col ='blue')
```




## Q3

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

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

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-6
```

```
library(tidyr)
logistic_model<- glmnet(x=hawks_train_x%>%as.matrix(),y=hawks_train_y,family = 'binom
ial',alpha =0,lambda=0)
```

```
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] 0.03608247
```

```
logistic_test_predicted_y<-predict(logistic_model,
                                    hawks_test_x%>%as.matrix(),type = "class")%>%as.i
nteger()

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



logistic_test_error
```

```
## [1] 0.03846154
```

# Basic concepts in regularisation

## Q1

1. In machine learning, a hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are derived via training.

Hyperparameters can be classified as model hyperparameters, that cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyperparameters, that in principle have no influence on the performance of the model but affect the speed and quality of the learning process. An example of a model hyperparameter is the topology and size of a neural network. Examples of algorithm hyperparameters are learning rate and batch size as well as mini-batch size.

2. Data validation means checking the accuracy and quality of source data before using, importing or otherwise processing data. Different types of validation can be performed depending on destination constraints or objectives. Data validation is a form of data cleansing.

3.Train test split is a model validation procedure that allows you to simulate how a model would perform on new/unseen data. Split the data set into two pieces — a training set and a testing set.Train the model on the training set and test the model on the testing set.

## Q2

The length of a vector is most commonly measured by the “square root of the sum of the squares of the elements,” also known as the Euclidean norm. It is called the 2-norm because it is a member of a class of norms known as p -norms。

The L1 norm that is calculated as the sum of the absolute values of the vector. The L2 norm that is calculated as the square root of the sum of the squared vector values.

### Q3

This is a regularization technique used in feature selection using a Shrinkage method also referred to as the penalized regression method. Lasso is short for Least Absolute Shrinkage and Selection Operator, which is used both for regularization and model selection. If a model uses the L1 regularization technique, then it is called lasso 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.

- Increasing in Ridge will only shrink parameters towards 0
- Increasing in Lasso can shrink parameters 0 (natural feature selector).

# An investigation into ridge regression for high-dimensional regression

```
library(QSARdata)
data(MeltingPoint)
```

### Q1

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

```
dim(mp_data_total)
```

```
## [1] 4401  203
```

variable numebrs: 893,403 examples: 4401

### Q2

```
num_total<-mp_data_total%>%nrow()
num_train<-floor(0.5*num_total)
num_validate<-floor(0.25*num_total)
num_test<-num_total-num_train-num_validate
```

```
set.seed(123)

test_inds<-sample(seq(num_total),num_test)
validate_inds<-sample(setdiff(seq(num_total),test_inds),num_validate)

train_inds<-setdiff(seq(num_total),union(validate_inds,test_inds))
```

```
mp_data_train<-mp_data_total%>%
  filter(row_number() %in% train_inds)
mp_data_validate<-mp_data_total%>%
  filter(row_number() %in% validate_inds)
mp_data_test<-mp_data_total%>%
  filter(row_number() %in% test_inds)
```

```
mp_data_train_x<-mp_data_train%>%
  select(-melting_pt)%>%as.matrix()
mp_data_train_y<-mp_data_train%>%
  pull(melting_pt)


mp_data_validate_x<-mp_data_validate%>%
  select(-melting_pt)%>%as.matrix()
mp_data_validate_y<-mp_data_validate%>%
  pull(melting_pt)


mp_data_test_x<-mp_data_test%>%
  select(-melting_pt)%>%as.matrix()
mp_data_test_y<-mp_data_test%>%
  pull(melting_pt)
```

```
length(mp_data_train_y)
```

```
## [1] 2200
```

## Q3

```
compute_train_validate_error_ridge<-function(train_x,train_y,validate_x,validate_y,la
mbda){

 glmRidge= glmnet(x=train_x,y=train_y,alpha =0, lambda =lambda)

 train_y_test<-predict(glmRidge,newx = train_x)
 train_error = mean((train_y - train_y_test)^2)

 validate_y_test<-predict(glmRidge,newx = validate_x)
 validate_error = mean((validate_y-validate_y_test)^2)


 return(list(train_error=train_error,validate_error =validate_error))


}
```

## Q4

```
lambda_min = 0.00001

lambdas = 0.00001*(1.25^seq(1,70, by=1))
```

## Q5

```
ridge_results_df<-data.frame(lambda=lambdas)%>%
  mutate(out=map(lambda,~compute_train_validate_error_ridge(train_x = mp_data_train_
x,train_y =mp_data_train_y, validate_x = mp_data_validate_x, validate_y = mp_data_val
idate_y,lambda = .x)))%>%
  mutate(train_error = map_dbl(out,~((.x)$train_error)), validate_error =map_dbl(out,
~((.x)$validate_error)))%>%
  select(-out)
```

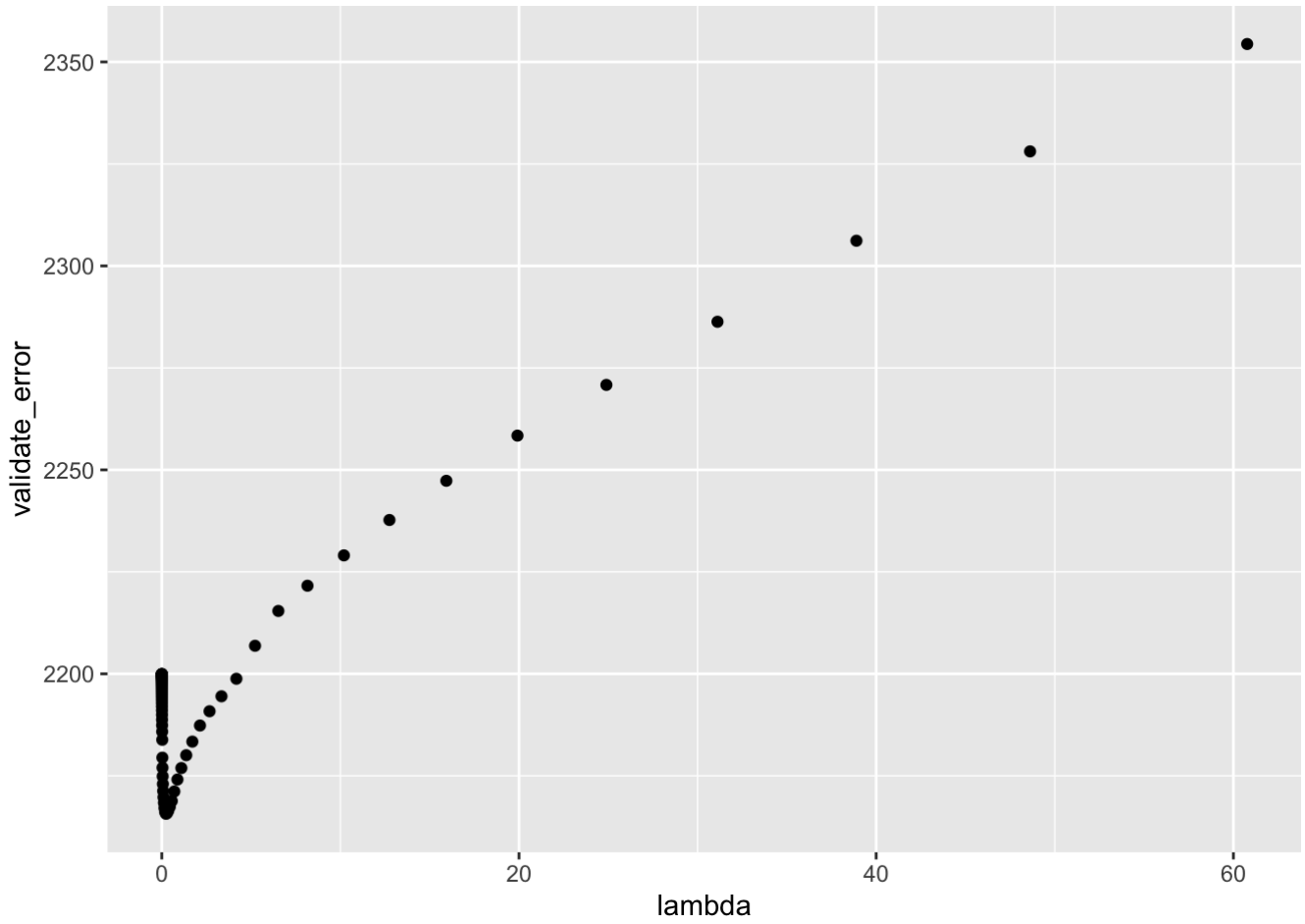
```
head(ridge_results_df)
```

```
##         lambda train_error validate_error
## 1 1.250000e-05    1712.476       2199.948
## 2 1.562500e-05    1712.488       2199.936
## 3 1.953125e-05    1712.503       2199.920
## 4 2.441406e-05    1712.519       2199.901
## 5 3.051758e-05    1712.541       2199.877
## 6 3.814697e-05    1712.569       2199.847
```

## Q6

```
p1 <- ggplot(ridge_results_df, aes(lambda, validate_error)) +
  geom_point()

p1 + scale_x_continuous()
```




## Q7

```
min_validation_error<-ridge_results_df%>%
  pull(validate_error)%>%min()
```

```
optimal_lambda<-ridge_results_df%>%
  filter(validate_error==min_validation_error)%>%
  pull(lambda)
```

```
optimal_lambda
```

```
## [1] 0.2295887
```

```
final_ridge_model<-glmnet(x = mp_data_train_x,y=mp_data_train_y, alpha = 0, lambda =
optimal_lambda)
```

```
final_ridge_test_y_est<-predict(final_ridge_model,
                                newx = mp_data_test_x)


final_ridge_test_error = mean((mp_data_test_y-final_ridge_test_y_est)^2)
```

```
final_ridge_test_error
```

```
## [1] 2109.132
```