Assignment 9 for Statistical Computing and Empirical Methods (Answers)

Dr. Henry WJ Reeve

Teaching block 1 2021

Introduction

This document describes your ninth assignment for Statistical Computing and Empirical Methods (Unit EMATM0061) on the MSc in Data Science. Before starting the assignment it is recommended that you first watch video lectures 21, 22 and 23.

1 Basic concepts in classification

(Q) Write down your explanation of each of the following concepts. Give an example where appropriate.

1. A classification rule

(A) A classification rule, also known as a classifier, is a mapping $\phi : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ is a space of feature vectors and $\mathcal{Y}$ is a set of discrete class labels, typically finite in number. As an example, consider a classifier which maps images of cats and dogs, represented as a vector, into a single label corresponding to whether the image is of a dog or a cat.

2. A learning algorithm

(A) In the context of classification, a learning algorithm is an algorithm which takes as input a set of training data and outputs a classification rule. This classification rule may then be applied to previously unseen data. Examples include linear discriminant analysis and logistic regression.

3. Training data

(A) Training data is a sequence of ordered pairs of the form $((X_1, Y_1), \cdots, (X_n, Y_n))$ where X_i is a feature vector and Y_i is an associated class label. For example, when trying to learn a cat/dog image classifier, X_i corresponds to an image of a dog or cat and Y_i is the label - "cat" or "dog".

4. Feature vector

(A) A feature vector X is a vector of containing one or more variables (also known as features) which we shall use to predict the class label, in the context of a classification rule. For example, in the context of the cat/dog image classifier the feature vector is a vector corresponding to a grey-scale image where each element corresponds to the numerical value of a particular pixel. As another example, in the case of penguin classification the feature vector is a vector containing several morphological properties such as the bill length, the flipper length, the weight etc.

5. Label

(A) The label Y is a unique identifier that signifies that the corresponding feature vector X is associated with an item belonging to a particular class. For example Y could be 1 when associated with an image X of a dog and 0 when associated with an image X of a cat. The set of labels are in one-to-one correspondence with the number of classes eg. if there are three classes then there must be three labels.

6. Test error

(A) The test error is the average number of mistakes a classification rule makes on unseen data. More precisely, suppose we have a classification rule $\phi : \mathcal{X} \to \mathcal{Y}$ then given a distribution P which describes the distribution over random pairs (X, Y) where X is a feature vector and Y is a label, the test error is

$$\mathcal{R}(\phi) = \mathbb{P}_{(X,Y)}\left[\phi(X) \neq Y\right].$$

This is typically estimated by the average error on a particular test data set.

7. Train error

(A) The train error is the average number of mistakes made by a classifier on the training data. More precisely, suppose we have set of training data $((X_1, Y_1), \cdots, (X_n, Y_n))$ and a classifier $\phi : \mathcal{X} \to \mathcal{Y}$, the train error is given by

$$\hat{\mathcal{R}}(\phi) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left\{\phi(X_i) \neq Y_i\right\}.$$

8. The train test split

(A) The train test split refers to a split of your data set into two pieces - containing two groups of examples: The training data and the testing data. The training data is used as an input into the learning algorithm. Based on the training data (and not the test data), the learning algorithm generates a classification rule. The test data plays no role in the learning of the classification rule. Instead the test data is used to assess the classification rule's performance on previously unseen data, not used within the training process. Hence, we compute the number of mistakes on the test data, which gives rise to an estimate for the expected test data.

9. Linear classifier

(A) A linear classifier is a particular type of classification rule where the decision corresponds to the value of the feature vectors multiplied by some weights. To be precise, let's suppose that our feature space is $\mathbb{R}^d$, so we have

d continuous predictive variables. in the context of binary classification, a linear classifier $\phi : \mathbb{R}^d \to \{0, 1\}$ is a mapping of the form

$$\phi(x) = \begin{cases} 1 & \text{if } w^0 + w^1 x^1 + \cdots + w^d x^d \geq 0 \\ 0 & \text{if } w^0 + w^1 x^1 + \cdots + w^d x^d < 0, \end{cases}$$

for all $x = (x^1, \cdots, x^d) \in \mathbb{R}^d$, where $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ is a vector of weights and w^0 is a bias term. Here (w, w^0) is a $d + 1$-dimensional parameter, which paramterises the linear classifer. We can also write ϕ as

$$\phi(x) = \mathbf{1}\left\{w\ x^\top + w^0 \geq 0\right\}.$$

2 The train test split

Suppose you want to build a classifier to predict whether a hawk belongs to either the "Sharp-shinned" or the "Cooper's" species of hawks. The feature vector will be a four dimensional row vector containing the weight, and the lengths of the wing, the tail and the hallux. The labels will be binary - 1 if the hawk is "Sharp-shinned" and 0 if the hawk belongs to "Cooper's" species.

(Q) Begin by loading the "Hawks" data frame from the "Stat2Data" library. Now extract a subset of the data called "hawks_total" with five columns - "Weight,"Wing","Hallux","Tail" and "Species". The data frame should only include rows corresponding to hawks from either the "Sharp-shinned" or the "Cooper's" species, and not the "Red-tailed" species. Convert the Species column to a binary variable with a 1 if the hawk belongs to the sharp-shinned species and 0 if the hawk belongs to the Cooper's species. Finally remove any rows with missing values from one of the relevant columns.

(A)

```
library(tidyverse)
library(Stat2Data)

data("Hawks")


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

(Q) Now implement a train test split for your "hawks_total" data frame. You should use 60% of your data within your training data and 40% in your test data. You should create a data frame consisting of training data called "hawks_train" and a data frame consisting of test data called "hawks_test". Display the number of rows in each data frame.

(A)

```
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(123) # 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

hawks_train%>%nrow()
```

```
## [1] 194
```

```
hawks_test%>%nrow()
```

```
## [1] 130
```

(Q) Next extract a data frame called "hawks_train_x" from your training data (from "hawks_train") containing the feature vectors and no labels. In addition extract a vector called "hawks_train_y" consisting of labels from your training data. Similarly, create data frames called "hawks_test_x" and "hawks_test_y" corresponding to the feature vectors and labels within the test set, respectively.

(A)

```
hawks_train_x<-hawks_train%>%select(-Species) # train feature vectors
hawks_train_y<-hawks_train%>%pull(Species) # train labels

hawks_test_x<-hawks_test%>%select(-Species) # test feature vectors
hawks_test_y<-hawks_test%>%pull(Species) # test labels
```

(Q) Now let's consider a very simple (and not very effective) classifier which entirely ignores the feature vectors. Instead the classifier simply predicts a single fixed value $y \in \{0, 1\}$. Hence, your classifier is of the form $\phi_y(x) \equiv y$ for all $x \in \mathbb{R}^4$. Begin by choosing a value $\hat{y} \in \{0, 1\}$ based on your training data - choose the value which minimises the training error.

(A)

```
train_error_phi_0<-mean(abs(hawks_train_y-0))
train_error_phi_1<-mean(abs(hawks_train_y-1))

if(train_error_phi_0<train_error_phi_1){
  y_hat<-0
}else{
  y_hat<-1
}

y_hat<-as.numeric(mean(hawks_train_y)>=0.5)

y_hat
```

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

(Q) Next compute the train and test error of $\phi_{\hat{y}}$

(A)

```
train_error_simple<-mean(abs(y_hat-hawks_train_y)) # train error
test_error_simple<-mean(abs(y_hat-hawks_test_y)) # train error

train_error_simple
```

```
## [1] 0.2371134
```

```
test_error_simple
```

```
## [1] 0.1769231
```

(Q) What does this tell you about the relative sizes of your classes?

(A) The fact that this very simple classifier, which completely ignores the feature vectors, actually achieves a very low error rate demonstrates that the classes are very imbalanced. Here the large majority of the hawks in the data set belong to the sharp-shinned species. If the two classes were more balanced we wouldn't be able to do much better than a 50% error rate.

3 Linear discriminant analysis

Describe the probabilistic model that underpins linear discriminant analysis.

(Q) Train a linear discriminant analysis model to carry out the classification task described above. That is, to predict whether a hawk belongs to either the "Sharp-shinned" or the "Cooper's" species of hawks, based on a four-dimensional feature vector containing the weight, and the lengths of the wing, the tail and the hallux.

(A)

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

(Q) Compute and report the train error and the test error.

(A)

```
lda_train_predicted_y<-predict(lda_model,hawks_train_x)$class%>%
  as.character()%>%as.numeric() # get vector of predicted ys

lda_train_error<-mean(abs(lda_train_predicted_y-hawks_train_y)) # compute train error

lda_train_error
```

```
## [1] 0.03092784
```

```
lda_test_predicted_y<-predict(lda_model,hawks_test_x)$class%>%
  as.character()%>%as.numeric() # get vector of predicted ys

lda_test_error<-mean(abs(lda_test_predicted_y-hawks_test_y)) # compute test error

lda_test_error
```

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

(Q) As a challenging optional extra implement your own linear discriminant analysis model.

(A)

```
generate_lda_model<-function(training_data,y_col_name){

  data<-training_data%>%rename(y=!!sym(y_col_name))

  x_0<-data%>%filter(y==0)%>%select(-y)
  x_1<-data%>%filter(y==1)%>%select(-y)
  y<-data%>%pull(y)

  n_0<-x_0%>%nrow()
  n_1<-x_1%>%nrow()


  q<-mean(data$y)

  mu_0<-x_0%>%summarise(across(everything(),mean))%>%as.matrix()
  mu_1<-x_1%>%summarise(across(everything(),mean))%>%as.matrix()

  sigma_0<-cov(x_0)
  sigma_1<-cov(x_1)

  sigma<-((n_0-1)/(n_0+n_1-2))*sigma_0+((n_1-1)/(n_0+n_1-2))*sigma_1

  sigma_inv<-solve(sigma)

  weight_0<-0.5*(mu_0%*%sigma_inv%*%t(mu_0)-
                   mu_1%*%sigma_inv%*%t(mu_1))%>%as.numeric()+log(q/(1-q))

  weight<-(mu_1-mu_0)%*%sigma_inv


  phi_lda<-function(x){

    y_hat<-(x%*%t(weight)+weight_0)>0

    return(y_hat%>%as.numeric())

  }

  return(phi_lda)
```

```
}

lda_function<-generate_lda_model(training_data=hawks_train,y_col_name="Species")

our_lda_train_predicted_y<-lda_function(hawks_train_x%>%as.matrix())

sum(abs(our_lda_train_predicted_y-lda_train_predicted_y))
```

```
## [1] 0
```

```
our_lda_test_predicted_y<-lda_function(hawks_test_x%>%as.matrix())

sum(abs(our_lda_test_predicted_y-lda_test_predicted_y))
```

```
## [1] 0
```

4 Logistic regression

(Q) Describe the probabilistic model which underpins logistic regression.

(A) Logistic regression corresponds to a probabilistic model over random pairs (X, Y) where X is a random feature vector taking values in $\mathbb{R}^d$ and Y is a random binary class label taking values in $\{0, 1\}$. It is assumed that there exists a weight vector $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ and $w^0 \in \mathbb{R}$. The probabilistic model is given by

$$\mathbb{P}(Y = y|X = x) = \begin{cases} \mathrm{S}\left(w\, x^\top + w^0\right) & \text{if y=1} \\ 1 - \mathrm{S}\left(w\, x^\top + w^0\right) & \text{if y=0} \end{cases}$$

for all $x = (x^1, \cdots, x^d) \in \mathbb{R}^d$. Here $\mathrm{S} : \mathbb{R} \to (0, 1)$ denotes the logistic sigmoid function defined by $\mathrm{S}(z) = 1/(1 + e^{-z})$. Note that we can use the property $1 - \mathrm{S}(z) = \mathrm{S}(-z)$ to rewrite the above formulation as

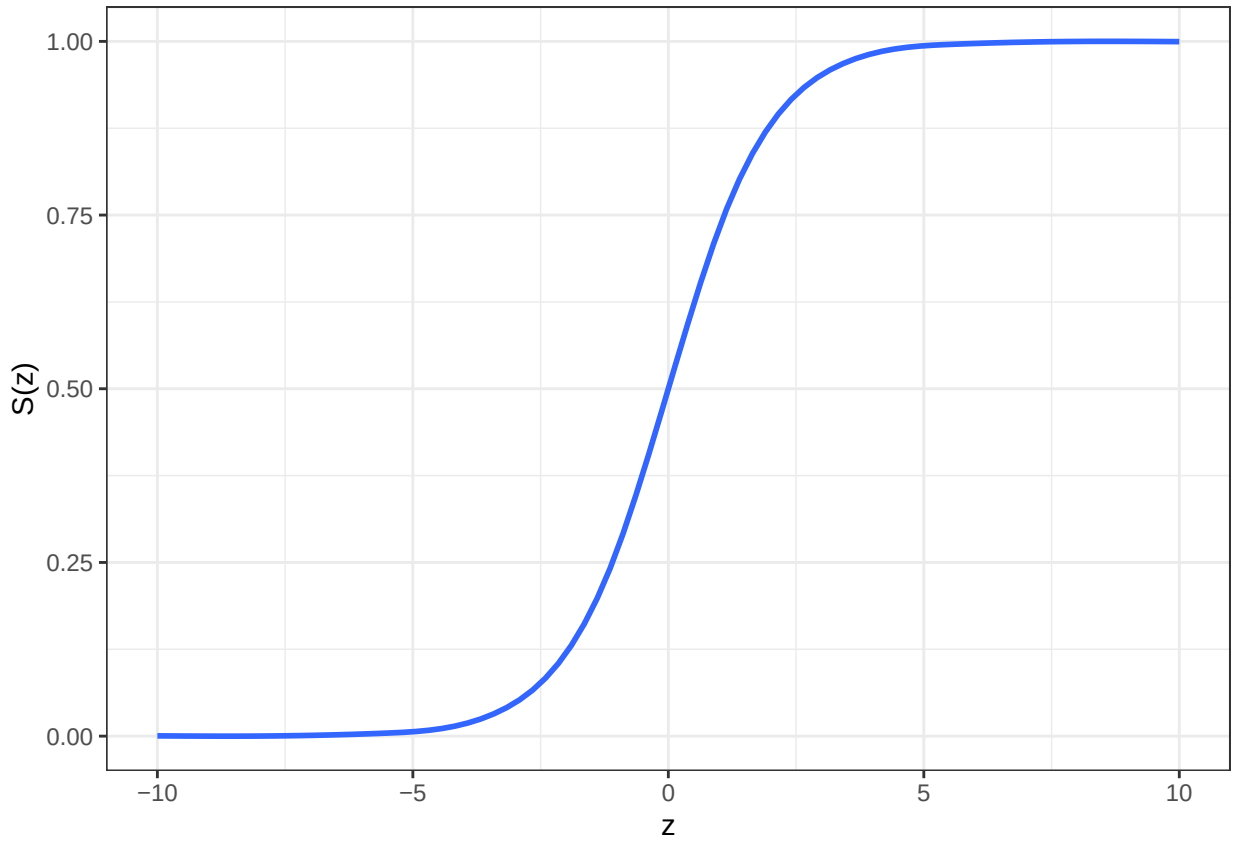
$$\mathbb{P}(Y = y|X = x) = \mathrm{S}\left((2y - 1)\left(w\, x^\top + w^0\right)\right).$$

(Q) Recall that the sigmoid function $\mathrm{S} : \mathbb{R} \to (0, 1)$ is defined by $\mathrm{S}(z) = 1/(1 + e^{-z})$. Generate the following plot which displays the sigmoid function:

(A)

```
data.frame(z=seq(-10,10,0.001))%>%
  mutate(sigmoid=1/(1+exp(-z)))%>%
  ggplot(aes(x=z,y=sigmoid))+
  geom_smooth()+
  theme_bw()+
  labs(x="z",y="S(z)")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



(Q) Now train a logistic regression model to predict whether a hawk belongs to either the "Sharp-shinned" or the "Cooper's" species of hawks, based on a four-dimensional feature vector containing the weight, and the lengths of the wing, the tail and the hallux.

(A)

```
library(glmnet) # load the glmnet library
logistic_model<-glmnet(x=hawks_train_x%>%as.matrix(),y=hawks_train_y,
                       family ="binomial",alpha=0,lambda=0) # train a logistic model
```

(Q) Compute and report both the training error and the test error.

(A)

```
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)) # train error

logistic_train_error
```

```
## [1] 0.0257732
```

```
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)) # test error

logistic_test_error
```

```
## [1] 0.04615385
```

(Q) As an optional extra consider the following formula for the log-likelihood of the weights $w \in \mathbb{R}^d$ and bias $w^0 \in \mathbb{R}$, given data $\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$:

$$\log \ell(w, w^0) = \sum_{i=1}^{n} \log \mathrm{S}\left((2Y_i - 1) \cdot (wX_i^\top + w^0)\right)$$

Demonstrate the following formulas for the derivatives:

$$\frac{\partial}{\partial w} \log \ell(w, w^0) = \sum_{i=1}^{n} (2Y_i - 1)\mathrm{S}\left((1 - 2Y_i) \cdot (wX_i^\top + w^0)\right) X_i,$$
$$\frac{\partial}{\partial w^0} \log \ell(w, w^0) = \sum_{i=1}^{n} (2Y_i - 1)\mathrm{S}\left((1 - 2Y_i) \cdot (wX_i^\top + w^0)\right).$$

(A) Observe that since $\mathrm{S}(z) = 1/(1 + e^{-z})$ we have

$$\log\left(\mathrm{S}(z)\right) = \log\left(\frac{1}{1 + e^{-z}}\right) = -\log(1 + e^{-z}).$$

Taking derivatives we have

$$\begin{aligned}\frac{\partial}{\partial z}\left\{\log\left(\mathrm{S}(z)\right)\right\} &= -\frac{\partial}{\partial z}\log(1 + e^{-z}) \\ &= \left(-\frac{1}{1 + e^{-z}}\right) \times \left(-e^{-z}\right) \\ &= \frac{1}{1 + e^{z}} = \mathrm{S}(-z).\end{aligned}$$

That is, $\frac{\partial}{\partial z}\left\{\log\left(\mathrm{S}(z)\right)\right\} = \mathrm{S}(-z)$. Using this relationship the formulae from the derivatives follow straightforwardly via the chain rule.

(Q) Explain the role the above formula has in training a logistic regression model.

(A) Unlike the linear discriminant analysis, there is no closed solution to the weights and bias for the logistic regression model. Instead the solution is found through an iterative procedure. At a high level, the way this approach is works is by taking successive steps in the direction of the gradient. This algorithm is known as gradient ascent. Since gradient points in the direction towards a local maximum, the likelihood will continue to rise throughout the training process. This process is guaranteed to approach a local maximum in the limit. For logistic regression there is actually a single local maximum - the global maximum. Hence, in the limit the maximum likelihood is approached. In practice, a wide variety of different algorithms are used to train logistic regression. These have a range of different computational advantages, from lower memory use, through to better incorporating second order information. However, all such approaches in some sense build upon the basic theme of gradient ascent.

You can learn more about the the glmnet approach to logistic regression here.