Challenge Tutorial: Supervised learning

Benoit Liquet

MNIST challenge

Consider the MNIST digit classification problem. We have previously learned how to classify the 10 digits using the *one versus all strategy* which consists to build 10 binary classifiers (one for each digit). Here our goal is to create a classifier between even and odd digits. Then, there are two possible classifiers:

- (I) use the classification of digits ' 0 '-' 9 ' (the one we have done) and then decide if even or odd.
- (II) train on binary classification between evens and odds digits.

1. Build in R or Python these two classifiers and train it on the training set (60, 000 images).
2. Evaluate the performance of the two classifier on the test set (10, 000 images) by providing the accuracy. You should provide the confusion matrix.
3. Which is better? Discuss and comment on the results?

To get the data

The first part is to get the data set and to build the classification of digits '0'-'9'

Read and create the data set

```
dim(images_training_set)
```

```
[1] 28 28 10
```

```
images_training_set <- extract_images(file_training_set_image, 60000)
images_test_set <- extract_images(file_test_set_image, 10000)
labels_training_set <- extract_labels(file_training_set_label)
labels_test_set <- extract_labels(file_test_set_label)
length(labels_training_set)
```

```
[1] 60000
```

```
vectorized_result <- function(j) {
  e <- as.matrix(rep(0, 10))
  e[j + 1] <- 1
  return(e)
}

Xtrain <- matrix(0,nrow=60000,ncol=784)
Ytrain <- matrix(0,nrow=60000,ncol=10)
for (i in 1:60000) {
  Xtrain[i,] <- as.vector(images_training_set[,,i]) / 256
  Ytrain[i,] <- t(vectorized_result(labels_training_set[i]))
}
Ytrain[which(Ytrain==0)] <- -1
```

```
Xtest <- matrix(0,nrow=10000,ncol=784)
Ytest <- matrix(0,nrow=10000,ncol=10)
for (i in 1:10000) {
  Xtest[i,] <- as.vector(images_test_set[,,i]) / 256
  Ytest[i,] <- t(vectorized_result(labels_test_set[i]))
}
```

We first compute the key element $(X^T X)^{-1} X^T$

As $(X^T X)$ is singular, we use the Moore-Penrose generalized inverse matrix of X through the `ginv` function from the `MASS` R package:

```
library(MASS)
mat <- ginv(Xtrain)
```

Then, we estimate the 10 linear classifier models

```
model <- matrix(0,nrow=784,ncol=10)
for(i in 1:10){
model[,i] <- mat%*%matrix(Ytrain[,i],ncol=1)
}
```

Now, we can create the prediction function for the linear classifier

```
pred.model <- function(x,Xtest=Xtest){sum(x*Xtest)}

linear.class <- function(model,Xtest){
  res <- apply(model,MARGIN=2,FUN=pred.model,Xtest=Xtest)
  pred <- which.max(res)-1
  return(pred)
}
```

Build classifier to discriminate even and odd digits based on the one versus all classifier

Based on the one versus all classifier we can build the classifier to discriminate even and odd digits.

```
One.versus.all.even.odd <- function(model,Xtest){
  # To complete
}
```

As an example, we try it on the first image of the test set

```
i <- 1
labels_test_set[i]
Xtest1 <- (as.vector(images_test_set[,,i]) / 256)
resi <- One.versus.all.even.odd(model,Xtest=Xtest1)
resi
```

Build direct binary classifier for even and odd digit.

We now focus on the direct binary classifier for even and odd digit. Then for each image i, we set $y_i = +1$ if the image i is even otherwise we set $y_i = -1$. This labels our data as classifying ***yes even*** vs. ***not even***. We then compute,

$$\hat{\beta}_{even} = (X^T X)^{-1} X^T y$$

Now for every image i, the inner product $\beta_{even}.x_i$ yields an estimate of how likely this image is an even digit. A very high value indicated a high likelihood and a low value is a low likelihood. We then classify an arbitrary image $\tilde{x}$ using the following rule

$$\hat{y}(\tilde{x}) = +1 \quad \text{if} \quad \beta_{even}.\tilde{x} \leq 0 \quad \text{otherwise} \quad \hat{y}(\tilde{x}) = -1 \tag{1}$$

```
i <- 1
y <- rep(1,60000)
for (i in 1:60000){
 if(labels_training_set[i] %in% c(1,3,5,7,9)) y[i] <- -1
}
```

We now get $\hat{\beta}_{even}$

```
beta.even <- # to complete
```

Using $\hat{\beta}_{even}$ we define our binary classifier

```
Binary.even.odd <- function(beta.even,Xtest){
  # to be completed
}
```

Evaluate both classifier

We now evaluate both classifier

```
# Your turn
```

We present the two confusion matrices and check the accuracy found.

```
# Your turn
```

Comparison of the two classifier

Compared the two classifier:

```
# Your turn
```