

# Lecture 6: Simple image recognition algorithm

Mnist dataset

Isabel Casas  
icasas@deusto.es

# Image recognition

- Computers do not recognise images as easily as we do.
- Any image is a set of pixels whose shade or color is represented by a number, for example in the RGB scale.
- In other words, an image is a numeric matrix (or array) for a computer.

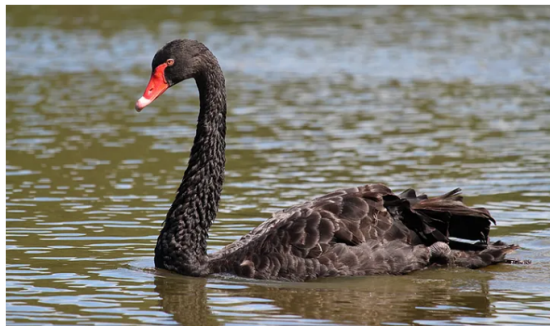
# Converting an Image Into a Matrix

- We convert colors and shades of grey into numbers and images into matrices of numbers.

## Video

- A matrix of matrices is called a tensor in mathematics and *array* in R.
- Even for simple images, the amount of data to analyse grows really quickly?

## Activity 1 in groups



- This is a 680x401 pixels image (we can discover this using Gimp, Photoshop or similar applications). Answer the following:
  - ▶ What type of color scale would you use to convert this image into numbers?
  - ▶ How many matrices would we have?
  - ▶ What dimension matrices?
  - ▶ How many numbers do we need to describe this image?

- A classical problem is the classification of number images from the MNIST dataset: tens of thousands of handwritten digits ranging from zero to nine. Each image is of size 28x28 pixels.

# MNIST dataset

Figure 1 shows a few images of all handwritten numbers appearing in MNIST.



# MNIST dataset

- MNIST is used to motivate the use of convolutional neural networks which is a complicated algorithm with a few layers to detect patterns.
- Because MNIST is not specially complicated, we can also use a random forest to classify those numbers.
- The MNIST data set consists of many  $28 \times 28$  matrices, one for each image of a number. Those  $28 \times 28$  are the 784 pixels of each image.
- We end up with a matrix with 784 numbers, each with a number representing a shade of grey

# MNIST dataset

- Each cell of the matrix has a number between 0 and 255 referring to the shade of grey of that pixel (0 = white, 255 = black).
- The current data is a transformation of the original data. The matrix is written in vector format (one column after another), so we have a single matrix with 785 columns.
- The first column contains the label (the number in the image) for training and the rest the shade of each pixel.



# Activity 1

- The train and testing sets of this converted dataset can be downloaded from [this link](#).
- Download the training and a testing sets into your computer
- Do these files have headers? Open them in R.
- See their dimension

## Activity 1 - Answer

A quick look at these datasets with any text editor shows that they do not contain column names in the first row and columns are separated by commas. Let us open them in R.

```
mnist.train <- read.table("mnist_train.csv", header = FALSE,  
                          sep=",")  
mnist.test  <- read.table("mnist_test.csv", header = FALSE,  
                          sep=",")
```

```
dim(mnist.train)
```

```
## [1] 60000    785
```

```
dim(mnist.test)
```

```
## [1] 10000    785
```

# MNIST dataset

- First column has the label (numbers 0, 1, ..., 9) and the rest is the number of each pixel corresponding to that label.
- Therefore, the first thing we have to tell R is that the first column is not a number, but a label, a factor.

```
mnist.train$V1 <- as.factor(mnist.train$V1)
mnist.test$V1 <- as.factor(mnist.test$V1)
summary(mnist.train$V1)
```

```
##      0      1      2      3      4      5      6      7      8      9
## 5923 6742 5958 6131 5842 5421 5918 6265 5851 5949
```

# Model training

We are training a random forest using the training dataset and 50 trees. The *randomForest* function generates the prediction of the test sample when we indicate testing data set in *xtest*. Because this is a classification problem, the output is the confusion matrix and each label's estimation error.

**Warning:** Running the code below takes more than 10 minutes.

```
library(randomForest)
rf <- randomForest(x = mnist.train[, -1], y = mnist.train[,1],
                  xtest = mnist.test[, -1], ytest = mnist.test[, 1],
                  ntree = 50,
                  keep.forest = TRUE)
```

## Activity 2

- In your computer, run the previous slides R code to generate model *rf*.
- The class will continue while your program is running.

# Confusion Matrix

The confusion matrix is a good method to evaluate the classification performance of classification models. It is very well explained in this [webpage of Wikipedia](#).

		Total population P + N	Predicted	
Actual values			FALSE	TRUE
	NEGATIVE	TN	FN	
	POSITIVE	FP	TP	

# Model evaluation: Confusion Matrix of our estimation

```
rf$confusion
```

##	0	1	2	3	4	5	6	7	8	9	class.error
## 0	5826	1	12	4	6	9	23	3	35	4	0.01637684
## 1	1	6629	31	22	15	5	6	9	12	12	0.01676061
## 2	21	11	5720	37	35	11	26	39	49	9	0.03994629
## 3	15	10	92	5742	6	101	9	44	81	31	0.06344805
## 4	14	9	14	2	5601	4	33	14	18	133	0.04125300
## 5	25	8	13	91	12	5145	46	6	48	27	0.05091312
## 6	30	11	13	1	16	40	5788	0	19	0	0.02196688
## 7	8	20	59	12	44	4	0	6011	15	92	0.04054270
## 8	14	32	42	72	34	50	31	13	5489	74	0.06186977
## 9	22	10	21	72	92	25	5	67	51	5584	0.06135485

# Model evaluation: Accuracy and error rate

- The overall prediction error and accuracy of the model are:

```
mean(rf$err.rate)
```

```
## [1] 0.07752641
```

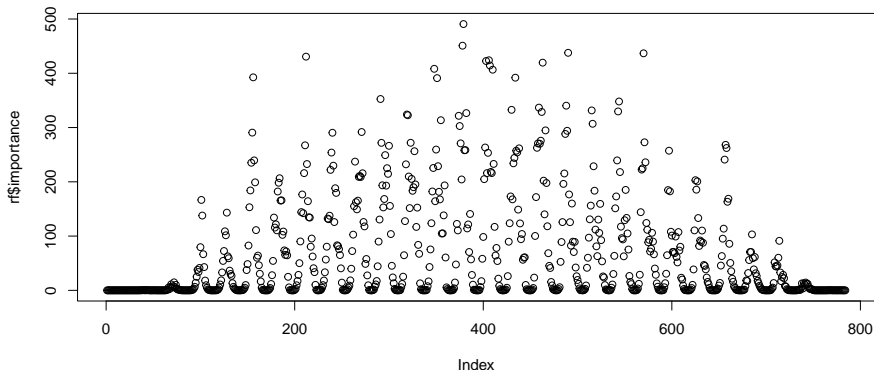
```
1 - mean(rf$err.rate)
```

```
## [1] 0.9224736
```



# Importance

```
plot(rf$importance)
```



Middle columns are more important than side columns for identification. It makes sense because darker pixels are in the center of each picture in the MNIST example

## Activity 3

- How many variables does your random forest model use at each tree?

**Hint:** Check the answer of `names(rf)` to know the possible values that one can get from this output model.

## Activity 3 - Answer

```
rf$mtry
```

```
## [1] 28
```

## Activity 4

Do this classification problem using the MLP.

Lecture 5 of Roger Grosse