# Report

## Practical assignment

## Introduction

This paper reports the result of our analysis of a data set of handwritten digits. We have divided the paper into paragraphs, where each paragraph describes a step in our analysis. All the experiments have been conducted in R language. To allow the reader to replicate the experiments, we have included the code in the “Appendix” paragraphs

## Exploratory analysis of the data

The data are composed of 784 features, which correspond to the 784 pixels an image has. We have in total 42000 images where the corresponding class (the digit that is written in the image) has been hand labelled in the data set as first column. The values of the feature represent the colour of the pixel on the grayscale, where 0 represents completely white, and 255 represents completely black. For each of them, we extract the minimum value it takes in the data set, the maximum, the average, the number of times the pixel takes value 0 (it’s “white”) and the number of times it takes value 255 (it’s “black”).

Then we have computer the mean of all these vectors.

|  |  |  |
| --- | --- | --- |
| Mean.minValue | Mean.maxValue | Mean.avgValue |
| 0 | 217.676 | 33.40891 |

|  |  |
| --- | --- |
| Mean.nwhite | Mean.nblack |
| 33955.76 | 284.8342 |

After that, we compute:

* The number of pixels which take value 0 at least once: 784 (all of them)
* The number of pixels which take value 255 at least once: 591

All these results suggest that we are dealing with a sparse matrix, where there are some pixels which always take value 0. In some sense they are “useless”, i.e. they do not bring any information we can use to build our predictive model. Moreover, we define as useless all the pixels who always take the same value in the data set.

We do so some analysis to spot those useless pixels. They are the following:

"pixel0" "pixel1" "pixel2"

"pixel3" "pixel4" "pixel5"

"pixel6" "pixel7" "pixel8"

"pixel9" "pixel10" "pixel11"

"pixel16" "pixel17" "pixel18"

"pixel19" "pixel20" "pixel21"

"pixel22" "pixel23" "pixel24"

"pixel25" "pixel26" "pixel27"

"pixel28" "pixel29" "pixel30"

"pixel31" "pixel52" "pixel53"

"pixel54" "pixel55" "pixel56"

"pixel57" "pixel82" "pixel83"

"pixel84" "pixel85" "pixel111"

"pixel112" "pixel139" "pixel140"

"pixel141" "pixel168" "pixel196"

"pixel392" "pixel420" "pixel421"

"pixel448" "pixel476" "pixel532"

"pixel560" "pixel644" "pixel645"

"pixel671" "pixel672" "pixel673"

"pixel699" "pixel700" "pixel701"

"pixel727" "pixel728" "pixel729"

"pixel730" "pixel731" "pixel754"

"pixel755" "pixel756" "pixel757"

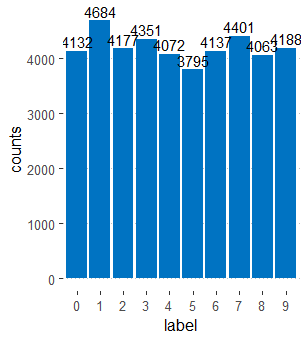
"pixel758" "pixel759" "pixel760"

"pixel780" "pixel781" "pixel782"

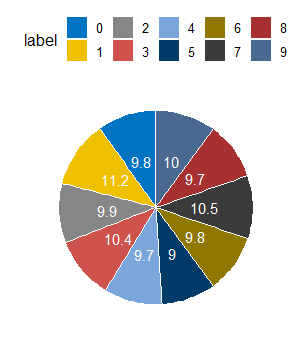
"pixel783"

They are 76 in total. All of them take always value 0. By, inspection, we find that there are no pixels which take always the same non-zero value.

Then we analyse the class distribution: we have in total 9 classes, each corresponding to a digit. We plot the following graph, with the classes on the x-axis and the counts of images with that digit on the y-axis.

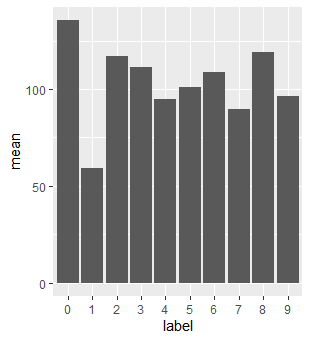


We plota pie chart as well, where “label” stands for “class”



The majority class is the class corresponding to the digit “1”: if we predicted only that class, we would have a percentage of correct classifications of 11,2%.

Moreover, we do some analysis taking into consideration the images in the data set and aggregating the results per each class of belonging (i.e., per row instead of per column, as we have always done before). We define “dark pixels” the pixels whit a value greater than 128. Per each image, we compute the number of “dark pixels” and then compute the mean per each class. The results are in the following graph. There are few (consider that they each image has 784 pixels).



We feel that dark pixels are useful to spot the skeleton of a handwritten digit, eliminating all the remaining ink which has simply spread in the paper where the digit was written. As an example, we compare the image number 380 in the data set and the same image where we have shrunk to 0 out all the pixel which take values between 0 and 128



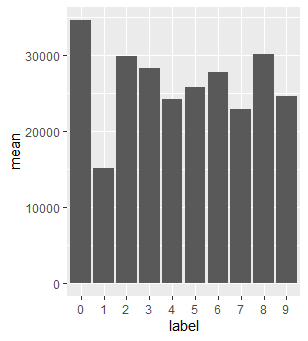
Figure 1: filteredl image

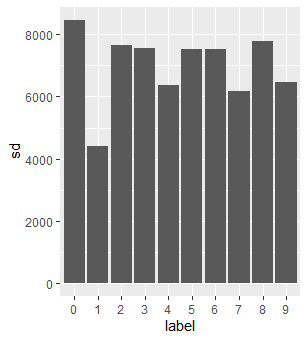


Figure 2: actual image

## How much ink

To measure “how much ink” a digit costs, we sum all the feature values per each element of the data we have (i.d., per each row of the dataframe). The results can be then in the interval [0; 255\*784 = 199920]. Then we aggregated the results per each digit, plotting the mean and the standard deviation. We called this feature “density”. As before, “label” means “class”.





This feature allows us to distinguish correctly between four groups: {class 1} [which has a mean below 20000], {class 4, class 7, class 9} [which have a mean between 20000 and 25000], {class 2, class 3, class 5, class 6, class 8} [which have a mean between 25000 and 3000 and more or less the same sd], {class 0} [which has a mean over 30000]. Moreover, we feel that only class 1 can be correctly distinguished from the other ones. However, big values of stardard deviations suggest that the feature is not very relevant to spot a class.

We add to our data set a column with density values as a new feature. We scale it. Then we train a multinomial logistic regression model with only the density of all the 42000 images as a feature. We test it using the same density values.

We get the following confusion matrix, where the rows represent the prediction of the model, and the columns the actual values. As expected, the model has the best performances on class 1, while it never predicts classes 4,5,6,8

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | sums |
| 0 | 2420 | 10 | 1496 | 1246 | 440 | 728 | 1057 | 325 | 1430 | 484 | 9636 |
| 1 | 83 | 3823 | 280 | 408 | 829 | 671 | 450 | 1190 | 192 | 763 | 8689 |
| 2 | 322 | 5 | 326 | 335 | 196 | 197 | 296 | 149 | 343 | 197 | 2366 |
| 3 | 805 | 101 | 1039 | 1037 | 886 | 846 | 982 | 819 | 1047 | 869 | 8431 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 384 | 722 | 874 | 1141 | 1496 | 1190 | 1145 | 1700 | 879 | 1651 | 11182 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 118 | 23 | 162 | 184 | 225 | 163 | 207 | 218 | 172 | 224 | 1696 |
| sums | 4132 | 4684 | 4177 | 4351 | 4072 | 3795 | 4137 | 4401 | 4063 | 4188 | 42000 |

The overall accuracy of the model is 0.2269048.

As an example, this the confusion matrix of the class 1 against all the others.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Image is of class 1** | |
|  |  | True | False |
| **Model predicts** | yes | 3823 | 4866 |
| **class 1.** | no | 861 | 32450 |

|  |  |  |
| --- | --- | --- |
| Accuracy | Precision | Recall |
| 0.8636 | 0.4400 | 0,8162 |

## New feature

To elaborate a new feature given an image, we deal with pixels as if they were points in a cartesian space. We try to find out a way of measuring different distributions of points, since each distribution corresponds to a different digit. Firstly, we associate to each pixel two cartesian coordinates which are basically the row and column numbers. Then, we filter only dark points, according to the definition previously given. Then, we select as a reference point the first point we encounter in the image, i.e. the one with smallest row and column values. Then we compute the mean distance between this point and all the other dark points. We repeat this procedure for each point, and then we have a new feature we can use in our multinomial model.

## Values

### Analysis 1 (single tree)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Defects are observed** | |
|  |  | True | False |
| **Model predicts** | Positive | 229 | 93 |
| **defect.** | Negative | 84 | 255 |

## Note

## Appendix 1: Metrics in the Eclipse Data Set