ADS2 Practical 2.9:
Supervised learning: MNIST digit classification

Facilitators: GL, DS, JC, WZ

April 16, 2024

This and next week we are going to work on handwritten digits recognition – one of standard problems in machine learning that is particularly suitable for learning (yet is real).

Before we proceed with training our supervised learning model, the first step is to reduce dimensionality by computing features.

This week, our goal is to import, convert and visualize data in a useful way, as well as extract useful features. Next week we will try to perform ***simple*** supervised learning methods (no deep learning!) in order to classify the digits.

You can find the original dataset by Yann LeCun here:
http://yann.lecun.com/exdb/mnist/

Please download the data, read and try to understand its description.
Skip the part about various (mostly quite complicated) classifiers and their performance.

As data is not organized in a very convenient way, here is a nice tutorial how to reorganise it in a more useful way and visualize: http://varianceexplained.org/r/digit-eda/

There is also a CSV version of data available here, https://pjreddie.com/projects/mnist-in-csv/ which may be useful when working in R, but the file size is considerably larger.

Alternatively, if you find descriptions in these links too complicated, you could use a simpler description (and a tiny part of the dataset) provided by Melanie last year.

Dimensionality reduction, feature selection

Even though the dimensionality of MNIST dataset (784, corresponding to 28 x 28 pixels) is not too high for training a supervised learning model with raw data, if all 60000 examples are used, this could be too computationally intensive for at least some computers. Hence, in this practical we will use only the first 1000 examples, containing about 100 examples of each digit. For this reason, dimensionality reduction is needed. Although some generic approaches can be used, I would advise selecting hand designed features, *for example*:

- Average value on each row and each column (28 + 28 = 56 features)
- Number of transitions between low value (<128) and high value (>127) in each row and column – also 56 features. This effectively denotes contrast changes between light and dark pixels, once each column is traced from top to bottom and each row from left to right.

The first feature (average value) may be easier to compute, but the second feature (contrast changes) may be more powerful in classification. You can also come up with your own features.

The task:

1. Load the training data, take the first 1000 samples
2. Compute features that substantially reduce dimensionality from 784, while preserving essential information (hence simply taking the first 56 or random 56 pixels of each example/instance are not a good idea).

Pre-processing steps detailed in http://varianceexplained.org/r/digit-eda/
(until centroids/atypical instances) are quite useful, as they convert pixel data to x and y coordinates, which can be operated more easily in feature calculation. However, if you can compute it differently, you are welcome to do so.

Load the data, for example using

```
library(readr)
library(dplyr)
mnist_raw <- read_csv("mnist_train.csv", col_names = FALSE)
```

Here observations/rows are examples.
The first column denotes digit label (0, 1, …, 9) and the other 784 columns denote pixel value, between 0 (white) and 255 (black). As you could see from the histogram, most values are near 0 or 255, with not a lot in between.

If you would like to use the suggested transformation, you can use the following code

```
library(tidyr)
pixels_gathered <- mnist_raw %>%
  head(1000) %>%
  rename(label = X1) %>%
  mutate(instance = row_number()) %>%
  gather(pixel, value, -label, -instance) %>%
  tidyr::extract(pixel, "pixel", "(\\d+)", convert = TRUE) %>%
  mutate(pixel = pixel - 2,
         x = pixel %% 28,
         y = 28 - pixel %/% 28)
```

If you operate on `pixels_gathered` from now on, you could `rm(mnist_raw)` to clear memory.
If you don't, you may keep only the first 1000 rows of `mnist_raw` for the same purpose.

In the transformed dataset, `label` is digit label (0..9), `instance` is example number (1..1000), `pixel` is pixel number (0..783), equal to column numbers in the original dataset minus 2, `value` is pixel value (0..255) and `x` (0..27) and `y` (1..28) are pixel coordinates that can help you compute statistics over pixels in the same column or row. Note that unlike in the original dataset, observations are no longer examples, but example-pixel pairs.

Based on the features you selected, create and compute a `features` dataframe, where rows are examples and variables/columns are different features (e.g. 56 of them).

Here is an example code to compute features based on row & column averages (it takes at least several minutes to run, up to an hour with slower computers):

```
features = data.frame(label = mnist_raw$X1[1:1000])
# set labels (of 1000 examples) as the first column in features dataframe

for (i in 1:56)
  features = cbind(features, fi = c(1:1000)*0)
# create 56 features (28 for rows and 28 for column) for the 1000 examples

for (i in 1:28)
# loop over 28 rows and 28 columns
  for (j in 1:1000)
  # compute row & column means for each digit example using pixels_gathered
  { features[j,i+1] = mean(pixels_gathered$value[pixels_gathered$instance==j & pixels_gathered$y==i]);
    # first 28 features: row means (each row has fixed y)
    features[j,i+29] = mean(pixels_gathered$value[pixels_gathered$instance==j & pixels_gathered$x==i-1]);
    # next 28 features: column means (each row has fixed x)
  }
```