Deep Learning Lecture 1 - MNIST: Densily connected NN

MA8701 General Statistical Methods

Thiago G. Martins, Department of Mathematical Sciences, NTNU

Spring 2019

MNIST dataset

The objective here is to classify the digit contained in a image.

Preparing the data

We can download the MNIST dataset using the `dataset_mnist` function available in the `keras` package.

```
# Downloading raw data
mnist <- dataset_mnist()
```

We then need to extract the training and test data from the `mnist` object.

```
# Training data
train_images <- mnist$train$x
train_labels <- mnist$train$y

# Test data
test_images <- mnist$test$x
test_labels <- mnist$test$y
```

The code below shows that the `train_images` is a tensor with 3 axis, `(samples, height, width)`. In this case, the channels axis was not necessary because the grey-scale images has only one channel.

```r
str(train_images)
```

```
##  int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
```

We can use slice operations to inspect specific images. Below we can see the content of the first training image.

```r
train_images[1,,]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [5,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##  [6,]    0    0    0    0    0    0    0    0    0     0     0     0     3
##  [7,]    0    0    0    0    0    0    0    0   30    36    94   154   170
##  [8,]    0    0    0    0    0    0    0   49  238   253   253   253   253
##  [9,]    0    0    0    0    0    0    0   18  219   253   253   253   253
## [10,]    0    0    0    0    0    0    0    0   80   156   107   253   253
## [11,]    0    0    0    0    0    0    0    0    0    14     1   154   253
## [12,]    0    0    0    0    0    0    0    0    0     0     0   139   253
## [13,]    0    0    0    0    0    0    0    0    0     0     0    11   190
## [14,]    0    0    0    0    0    0    0    0    0     0     0     0    35
## [15,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [16,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [17,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [18,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [19,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [20,]    0    0    0    0    0    0    0    0    0     0     0     0    39
## [21,]    0    0    0    0    0    0    0    0    0     0    24   114   221
## [22,]    0    0    0    0    0    0    0    0   23    66   213   253   253
## [23,]    0    0    0    0    0    0   18  171  219   253   253   253   253
## [24,]    0    0    0    0   55  172  226  253  253   253   253   244   133
## [25,]    0    0    0    0  136  253  253  253  212   135   132    16     0
## [26,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [27,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [28,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0
##  [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,]    18    18    18   126   136   175    26   166   255   247   127
##  [7,]   253   253   253   253   253   225   172   253   242   195    64
##  [8,]   253   253   253   253   251    93    82    82    56    39     0
##  [9,]   253   198   182   247   241     0     0     0     0     0     0
## [10,]   205    11     0    43   154     0     0     0     0     0     0
## [11,]    90     0     0     0     0     0     0     0     0     0     0
## [12,]   190     2     0     0     0     0     0     0     0     0     0
## [13,]   253    70     0     0     0     0     0     0     0     0     0
## [14,]   241   225   160   108     1     0     0     0     0     0     0
```

```
## [15,]    81   240   253   253   119    25     0     0     0     0     0
## [16,]     0    45   186   253   253   150    27     0     0     0     0
## [17,]     0     0    16    93   252   253   187     0     0     0     0
## [18,]     0     0     0     0   249   253   249    64     0     0     0
## [19,]     0    46   130   183   253   253   207     2     0     0     0
## [20,]   148   229   253   253   253   250   182     0     0     0     0
## [21,]   253   253   253   253   201    78     0     0     0     0     0
## [22,]   253   253   198    81     2     0     0     0     0     0     0
## [23,]   195    80     9     0     0     0     0     0     0     0     0
## [24,]    11     0     0     0     0     0     0     0     0     0     0
## [25,]     0     0     0     0     0     0     0     0     0     0     0
## [26,]     0     0     0     0     0     0     0     0     0     0     0
## [27,]     0     0     0     0     0     0     0     0     0     0     0
## [28,]     0     0     0     0     0     0     0     0     0     0     0
##       [,25] [,26] [,27] [,28]
##  [1,]     0     0     0     0
##  [2,]     0     0     0     0
##  [3,]     0     0     0     0
##  [4,]     0     0     0     0
##  [5,]     0     0     0     0

##  [6,]     0     0     0     0
##  [7,]     0     0     0     0
##  [8,]     0     0     0     0
##  [9,]     0     0     0     0
## [10,]     0     0     0     0
## [11,]     0     0     0     0
## [12,]     0     0     0     0
## [13,]     0     0     0     0
## [14,]     0     0     0     0
## [15,]     0     0     0     0
## [16,]     0     0     0     0
## [17,]     0     0     0     0
## [18,]     0     0     0     0
## [19,]     0     0     0     0
## [20,]     0     0     0     0
## [21,]     0     0     0     0
## [22,]     0     0     0     0
## [23,]     0     0     0     0
## [24,]     0     0     0     0
## [25,]     0     0     0     0
## [26,]     0     0     0     0
## [27,]     0     0     0     0
## [28,]     0     0     0     0
```

A plot can be made using the `as.raster` function.

```r
# Plotting a digit
digit <- train_images[5,,]
plot(as.raster(digit, max = 255))
```

The train labels is a vector holding the digit that the respective image is representing, going from zero to nine.

```
table(train_labels)
```

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

Defining the model

In this case we are using a `layer_dense` which expects an input tensor of rank equal to two `(sample, features)` where each `sample` should contain `28*28=784` pixels.

```
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28*28)) %>%
  layer_dense(units = 10, activation = "softmax")
```

Compiling the model

```
network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)
```

Data pre-processing

Before training, we'll preprocess the data by reshaping it into the shape the network expects and scaling it so that all values are in the [0, 1] interval.

Previously, our training images, for instance, were stored in an array of shape (60000, 28, 28) of type integer with values in the [0, 255] interval. We transform it into a double array of shape (60000, 28 * 28) with values between 0 and 1.

```
# Processing the data
train_images <- array_reshape(train_images, c(60000, 28 * 28))
train_images <- train_images / 255

train_labels <- to_categorical(train_labels)

test_images <- array_reshape(test_images, c(10000, 28 * 28))
test_images <- test_images / 255

test_labels <- to_categorical(test_labels)
```

Fitting the model

```
network %>% fit(train_images, train_labels, epochs = 5, batch_size = 128)
```

Recommended exercise 1

Perform model validation (plot loss and accuracy metric) and model assessment (accuracy in the test dataset) for the model defined here.