Untitled

Nishith Ranjon Roy

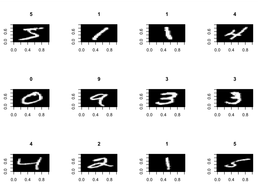
1/17/2022

Loading the dataset and basic structures

```
mnist <- readRDS("mnist.Rds")
str(mnist)
```

```
## List of 2
##  $ train:List of 2
##   ..$ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ y: int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4 ...
##  $ test :List of 2
##   ..$ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ y: int [1:10000(1d)] 7 2 1 0 4 1 4 9 5 9 ...
```

```
par(mfcol = c(3,4))
for(i in 1:12){
mnist$train$x[i,,] %>%
    apply(MARGIN = 2,rev) %>%
    t() %>%
    image(col=gray((0:255)/255), main = mnist$train$y[i])
}
```



Distribution of the disit

```
(mnist$train$y %>%
    table() %>%
    prop.table() %>%
    round(digits = 4))*100
```

```
##  0  1  2  3  4  5  6  7  8  9
## 9.87 11.24  9.93 10.22  9.74  9.04  9.86 10.44  9.75  9.92
```

Split the data set

```
train_x <- mnist$train$x
train_y <- mnist$train$y
test_x <- mnist$test$x
test_y <- mnist$test$y
```

Defining the model

```
tensorflow::tf$random$set_seed(123)
```

```
## Loaded Tensorflow version 2.7.0
```

```
model <-
    keras_model_sequential() %>%
    layer_dense(units = 300, activation = "relu", input_shape = 28*28) %>%
    layer_dense(units = 50, activation = "relu") %>%
    layer_dense(units = 10, activation = "softmax")

model
```

```
## Model
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_2 (Dense)                    (None, 300)                     235500
##
##  dense_1 (Dense)                    (None, 50)                      15050
##
##  dense (Dense)                      (None, 10)                      510
##
## ================================================================================
## Total params: 251,060
## Trainable params: 251,060
## Non-trainable params: 0
## ________________________________________________________________________________
```

Adding the compiler

```
model %>%
    compile(optimizer = "adam",
            loss = "categorical_crossentropy",
            metrics = c("accuracy"))
```

Preprocessing the data

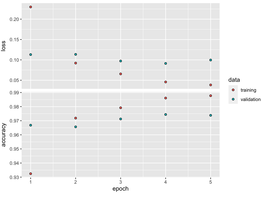
```
train_x <- array_reshape(train_x, dim = c(60000,28*28))/255
test_x <- array_reshape(test_x, dim = c(10000,28*28))/255

train_y <- to_categorical(train_y)
test_y <- to_categorical(test_y)
```

Train

```
plot <-
    model %>%
    fit(train_x, train_y, epochs = 5, batch_size = 20, validation_split = 0.2)

plot(plot)
```



Evaluation

```
model %>%
    evaluate(test_x, test_y)
```

```
##      loss  accuracy
## 0.07965852 0.97670007
```

CNN

Reshaping the array for the training

```
train_x <- array_reshape(mnist$train$x, c(dim(mnist$train$x),1))
train_x <- train_x/255
test_x <- array_reshape(mnist$test$x, c(dim(mnist$test$x),1))
test_x <- test_x/255
```

Building the model

```
tensorflow::tf$random$set_seed(123)

model <-
    keras_model_sequential() %>%

    # Adding a 2d tensor layer
    layer_conv_2d(filters = 32,
                  kernel_size = c(5,5),
                  activation = "relu",
                  input_shape = c(28,28,1)) %>%


    # Adding a 2d tensor layer
    layer_conv_2d(filters = 32,
                  kernel_size = c(3,3),
                  activation = "relu") %>%


    # Adding a max pulling layer
    layer_max_pooling_2d(pool_size = c(3,3)) %>%


    # Adding a 2d tensor layer
    layer_conv_2d(filters = 32,
                  kernel_size = c(3,3),
                  activation = "relu") %>%


    # Adding a max pulling layer
    layer_max_pooling_2d(pool_size = c(5,5)) %>%

    # Adding a flatten layer
    layer_flatten() %>%

    # Adding a dense layer
    layer_dense(units = 16,
                activation = "relu") %>%

    # Adding a dense layer
    layer_dense(units = 10,
                activation = "softmax",
                name = "Output")

model
```

```
## Model
## Model: "sequential_1"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  conv2d_2 (Conv2D)                  (None, 24, 24, 32)              832
##
##  conv2d_1 (Conv2D)                  (None, 22, 22, 32)              9248
##
##  max_pooling2d_1 (MaxPooling2D)     (None, 7, 7, 32)                0
##
##  conv2d (Conv2D)                    (None, 5, 5, 32)                9248
##
##  max_pooling2d (MaxPooling2D)       (None, 1, 1, 32)                0
##
##  flatten (Flatten)                  (None, 32)                      0
##
##  dense_3 (Dense)                    (None, 16)                      528
##
##  Output (Dense)                     (None, 10)                      170
##
## ================================================================================
## Total params: 20,026
## Trainable params: 20,026
## Non-trainable params: 0
## ________________________________________________________________________________
```

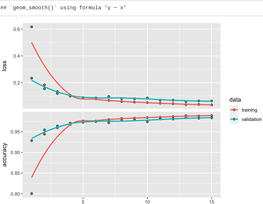
Adding the compiler

```
model %>%
    compile(
        optimizer = "adam",
        loss = "categorical_crossentropy",
        metrics = "accuracy"
    )

plot <-
    model %>%
    fit(train_x, train_y, epoch = 15, batch_size = 128, validation_split = .2)

plot(plot)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Evaluation

```
model %>%
    evaluate(test_x, test_y)
```

```
##      loss  accuracy
## 0.05447746 0.98280007
```