Homework 5

Marisa Blackman

4/20/2023

```r
#install_keras()
#library(keras)
#devtools::install_github("rstudio/keras")
#tensorflow::install_tensorflow(package_url = "https://pypi.python.org/packages/b8/d6/af3d52dd52150ec4a
#library(tensorflow)

#install_keras(method = "conda")
#install_keras(tensorflow = "gpu")
#tensorflow::install_tensorflow()

#library(keras)
# mnist <- dataset_mnist()
# x_train <- mnist$train$x
# y_train <- mnist$train$y
# x_test <- mnist$test$x
# y_test <- mnist$test$y

library(reticulate)
#path_to_python <- install_python()
#virtualenv_create("r-reticulate", python = path_to_python)
#library(keras)
 ##install_keras(envname = "r-reticulate") library(tensorflow)

#install.packages("tensorflow")
#library(reticulate)
 ##path_to_python <- install_python() ##virtualenv_create("r-reticulate", python = path_to_python)
library(keras)
 ##install_keras(envname = "r-reticulate")
library(tensorflow)
```

Goal: Get started using Keras to construct simple neural networks

Read through the "Image Classification" tutorial on the RStudio Keras website.

```r
library(keras)
```

Use the Keras library to create a convolutional neural network similar to (or more sophisticated than) "Net-5" described during lecture on 4/4 and also described in the ESL book section 11.7. See the ConvNet tutorial on the RStudio Keras website.

```
# Parameters --------------------------------------------------------------

batch_size <- 32
epochs <- 50


# Data Preparation --------------------------------------------------------

# See ?dataset_cifar10 for more info

load("~/Desktop/Spring 2023/data/zip.train.RData")
load("~/Desktop/Spring 2023/data/zip.test.RData")


# using 16x16 images with 10 potential classes
nclass = 10
img_rows = 16
img_cols = 16

x_test = zip.test[,-1]
x_train = zip.train[,-1]

y_test = zip.test[,1]
y_train = zip.train[,1]

# reshape data
x_train_array = array(dim = c(nrow(x_train), 16, 16))
for (i in 1:nrow(x_train)){
  x_train_array[i,,] = as.numeric(x_train[i,])
  x_train_array[i,,] = t(x_train_array[i,,])
}

x_test_array = array(dim = c(nrow(x_test), 16, 16))
for (i in 1:nrow(x_test)){
  x_test_array[i,,] = as.numeric(x_test[i,])
  x_test_array[i,,] = t(x_test_array[i,,])
}

x_train_array <- array_reshape(x_train_array, c(nrow(x_train), img_rows, img_cols, 1))
x_test_array <- array_reshape(x_test_array, c(nrow(x_test), img_rows, img_cols, 1))
input_shape <- c(img_rows, img_cols, 1)

# Convert class vectors to binary class matrices
y_train <- to_categorical(y_train, nclass)
y_test <- to_categorical(y_test, nclass)
```

```r
# Initialize sequential model
model <- keras_model_sequential()

# convolutional model using weight sharing
model <- model %>%
  # first hidden layer
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
                input_shape = input_shape) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  # another hidden layer
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
  # max pooling
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  # output corresponding to the 10 possible digits
  layer_dense(units = nclass, activation = 'softmax')

# summary
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  conv2d_1 (Conv2D)                  (None, 14, 14, 32)              320
##  max_pooling2d_1 (MaxPooling2D)     (None, 7, 7, 32)                0
##  conv2d (Conv2D)                    (None, 5, 5, 64)                18496
##  max_pooling2d (MaxPooling2D)       (None, 2, 2, 64)                0
##  dropout_1 (Dropout)                (None, 2, 2, 64)                0
##  flatten (Flatten)                  (None, 256)                     0
##  dense_1 (Dense)                    (None, 128)                     32896
##  dropout (Dropout)                  (None, 128)                     0
##  dense (Dense)                      (None, 10)                      1290
## ================================================================================
## Total params: 53,002
## Trainable params: 53,002
## Non-trainable params: 0
## ________________________________________________________________________________
```

Fit the CNN to the zipcode data from the authors website and create a figure similar to that from the slides that shows test error as a function of training epochs.

```r
# Compile model
model %>% compile(
  loss = loss_categorical_crossentropy,
  optimizer = "adam",
  metrics = c('accuracy')
)
```

```r
# Train model
hist <- model %>% fit(
  x_train_array, y_train,
  batch_size = batch_size,
  epochs = epochs,
  validation_split = 0.2
)
# evaluate model for accuracy
score <- model %>% evaluate(x_test_array, y_test)
cat('Test loss:', score["loss"], "\n")
```

```
## Test loss: 0.1746661
```

```r
cat('Test accuracy:', score["accuracy"], "\n")
```

```
## Test accuracy: 0.9661186
```

```r
#plot history
plot(hist)
```