Homework8

Steven Black

4/5/2019

IS 677: Introduction to Data Science Spring 2019

Homework Assignment 8 (Due: April 14, 2019, midnight EST)

Use the fashion_mnist data that comes with Keras to answer the questions below. The fashion_mnist dataset has 60,000 training images and 10,000 test images. All images are 28*28 arrays and are grey scale images. The images are individual articles of clothing labeled as 0-9 as follows:

- 0: T-shirt/top,
- 1: Trouser,
- 2: Pullover,
- 3: Dress,
- 4: Coat,
- 5: Sandal,
- 6: Shirt,
- 7: Sneaker,
- 8: Bag,
- 9: Ankle boot.

You can follow the example from last week to answer questions 1-6.

```
library(keras)
```

1. Load the data into a variable called fashion. (5 points)

```
fashion <- dataset_fashion_mnist()
```

2. Load the train_images, train_labels, test_images, and test_labels into appropriate variables. (5 points)

```
train_images <- fashion$train$x
train_labels <- fashion$train$y
test_images <- fashion$test$x
test_labels <- fashion$test$y
```

3. Reshape and scale the training and test images. (5 points)

```
train_images_reshaped <- array_reshape(train_images, c(60000, 28, 28, 1))
train_images_reshaped <- train_images_reshaped / 255

test_images_reshaped <- array_reshape(test_images, c(10000, 28, 28, 1))
test_images_reshaped <- test_images_reshaped / 255
```

4. Create one-hot encoding of the training and test labels. (5 points)

```
train_labels_categorical <- to_categorical(train_labels)
test_labels_categorical <- to_categorical(test_labels)
```

5. Build a convolutional NN for classifying the images:

- layer 1: 2D CONV 32 filters size 3*3, with RELU activation.
- Layer 2: 2D MAX_POOL with pool size=2*2,
- layer 3: 2D CONV 64 filters size 3*3, with RELU activation,

- layer 4: 2D MAX_POOL with pool size=2*2,
- layer 5: 2D CONV 64 filters size 3*3,
- layer 6: layer_flatten,
- layer 7: dense with 64 units and RELU activation,
- layer 8: you determine. (20 points)

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
                input_shape =  c(28, 28, 1)) %>%         # 1
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%           # 2
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%  # 3
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%                       # 4
  layer_conv_2d(filters = 64, kernel_size = c(3, 3)) %>%  # 5
  layer_flatten() %>%                                     # 6
  layer_dense(units = 64, activation = "relu") %>%    # 7
  layer_dense(units = 10, activation = "softmax")
```

6. Compile the model with rmsprop as the optimizer and accuracy as the metric. You decide the loss function (5 points).

```
model %>% compile(
  loss = "binary_crossentropy",
  optimizer = "rmsprop",
  metrics = c("acc")
)
```

7. Fit the model with 5 epochs, 0.3 validation split, and batch size of 64. (5 points)

```
num_epochs <- 5

train_history <- model %>% fit(
  train_images_reshaped,
  train_labels_categorical,
  validation_data = list(test_images_reshaped, test_labels_categorical),
  epochs = num_epochs, batch_size = 128,
  validation_split = 0.3
)
```

8. Evaluate the results using the test images. (5 points)

```
validation_history <- model %>% evaluate(
  test_images_reshaped, test_labels_categorical
)
validation_history
```

```
## $loss
## [1] 0.05348513
##
## $acc
## [1] 0.97887
```

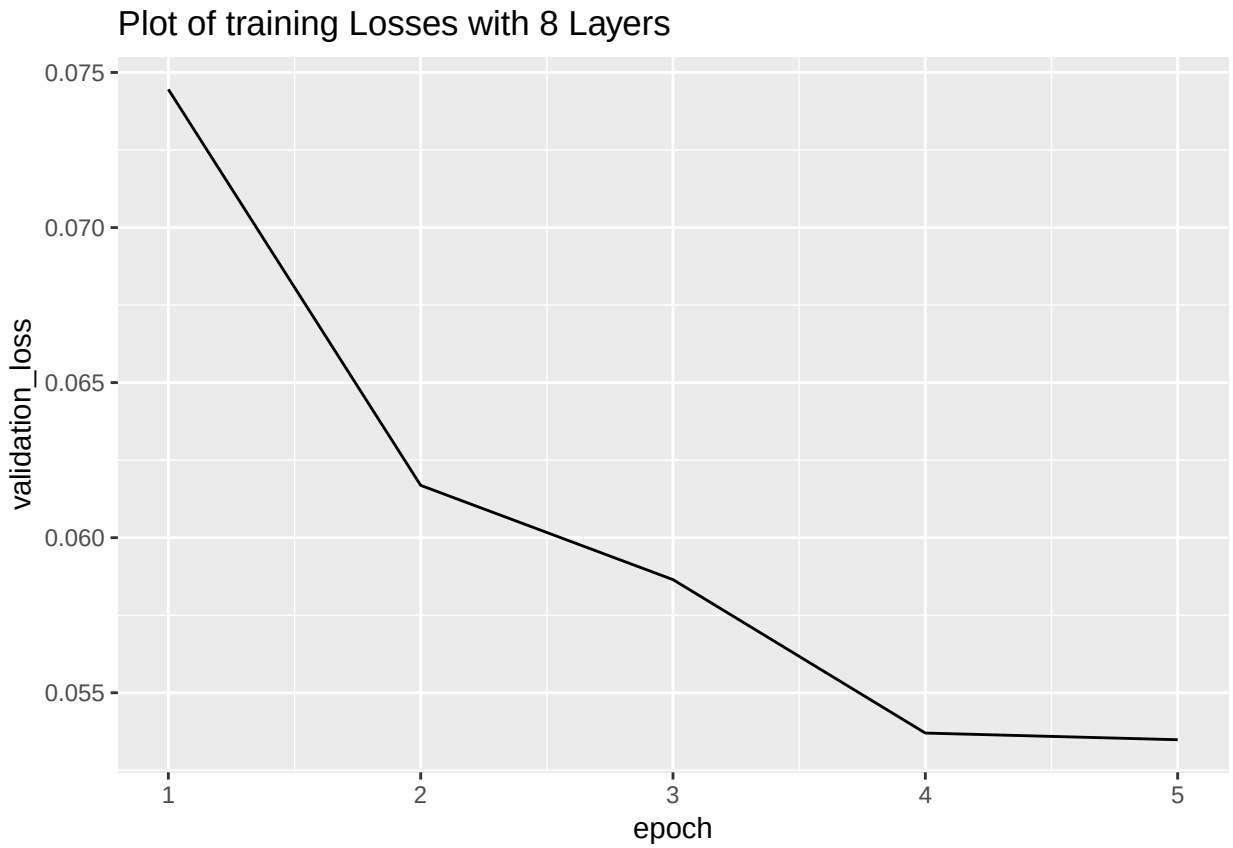
9. Create a new CNN model to include a dropout layer with 50% dropout rate (25 points).

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
                input_shape =  c(28, 28, 1)) %>%         # 1
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%           # 2
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%  # 3
```

```
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%                          # 4
  layer_conv_2d(filters = 64, kernel_size = c(3, 3)) %>%  # 5
  layer_flatten() %>%       # 6
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 64, activation = "relu") %>%    # 7
  layer_dense(units = 10, activation = "softmax")

model %>% compile(
  loss = "binary_crossentropy",
  optimizer = optimizer_rmsprop(lr = 1e-4),
  metrics = c("acc")
)

train_history2 <- model %>% fit(
  train_images_reshaped,
  train_labels_categorical,
  validation_data = list(test_images_reshaped, test_labels_categorical),
  epochs = num_epochs, batch_size = 128,
  validation_split = 0.3
)
```

10. Plot the two model performances. (10 points)

```
library(ggplot2)
train_loss_obj <- data.frame(
  epoch = seq(1:length(train_history$metrics$val_loss)),
  validation_loss = train_history$metrics$val_loss
)

ggplot(train_loss_obj,
       aes(x = epoch, y = validation_loss)
       ) + geom_line() + ggtitle("Plot of training Losses with 8 Layers")
```




```
train_loss_obj2 <- data.frame(
  epoch = seq(1:length(train_history2$metrics$val_loss)),
  validation_loss = train_history2$metrics$val_loss
)

ggplot(train_loss_obj2,
       aes(x = epoch, y = validation_loss)
       ) + geom_line() + ggtitle("Plot of training Losses with 8 Layers and dropout")
```

11. Explain the results. (10 points) The accuracy of the neural net without dropout is a full percentage point better than with dropout. It seems that the model was not overfitting and so the dropout did not help.