Homework10

Steven Black

4/24/2019

IS 677: Introduction to Data Science Spring 2019

Homework Assignment 10 (Due: April 28, 2019, midnight EST)

Use the CIFAR10 dataset that comes with Keras. It has 50,000 training and 10,000 testing images of 10 classes:

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck

The image size is 32*32*3. The 3 indicates that these are color images.

```
library(keras)
k_clear_session()
```

1. Clear the session and load the CIFAR10 data into a variable called cifar. (5 points)

```
cifar <- dataset_cifar10()
```

2. Create a small training dataset using the first 1000 training images (and the corresponding labels) from CIFAR10. Similarly, create a small test dataset (and the corresponding labels) using the first test 500 images from CIFAR10. (5 points).

```
num_training_images <- 1000
num_testing_images <- 500

train_images <- cifar$train$x[1:num_training_images, 1:32, 1:32, 1:3]
train_labels <- cifar$train$y[1:num_training_images]

test_images <- cifar$test$x[1:num_testing_images, 1:32, 1:32, 1:3]
test_labels <- cifar$test$y[1:num_testing_images]
```

3. Create one-hot encoding for the labels for both train and test labels. (5 points)

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

4. Instantiate a VGG16 convolutional base without the top layer. (5 points)

```
conv_base <- application_vgg16(
  weights = "imagenet",
  include_top = FALSE,
  input_shape = c(32, 32, 3)
```

```
)
conv_base
```

```
## Model
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## input_1 (InputLayer)         (None, 32, 32, 3)         0
## _________________________________________________________________
## block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
## _________________________________________________________________
## block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
## _________________________________________________________________
## block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
## _________________________________________________________________
## block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
## _________________________________________________________________
## block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
## _________________________________________________________________
## block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
## _________________________________________________________________
## block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
## _________________________________________________________________
## block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
## _________________________________________________________________
## block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
## _________________________________________________________________
## block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
## _________________________________________________________________
## block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
## _________________________________________________________________
## block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
## _________________________________________________________________
## block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
## _________________________________________________________________
## block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
## _________________________________________________________________
## block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
## _________________________________________________________________
## block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
## _________________________________________________________________
## block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
## _________________________________________________________________
## block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
## =================================================================
## Total params: 14,714,688
## Trainable params: 14,714,688
## Non-trainable params: 0
## _________________________________________________________________
```

5. Extract features from the CIFAR10 images so as to fit the conv_base. (40 points)

```
datagen <- image_data_generator(rescale = 1/255)
batch_size <- 10
```

```
train_generator <- flow_images_from_data(
    x = train_images,
    y = train_labels_categorical,
    generator = datagen,
    batch_size = batch_size
)
features_batch <- conv_base %>% predict_generator(train_generator,
                                                  steps = num_training_images / batch_size)

test_generator <- flow_images_from_data(
    x = test_images,
    y = test_labels_categorical,
    generator = datagen,
    batch_size = batch_size
)

test_features_batch <- conv_base %>% predict_generator(test_generator,
                                                  steps = num_testing_images / batch_size)
```

```
dim(features_batch)
```

```
## [1] 1000    1    1  512
```

6. Flatten the features in order to feed them to a densely connected classifier. (5 points)

```
reshape_features <- function(features) {
  array_reshape(features, dim = c(nrow(features), 1 * 1 * 512))
}

train_features <- reshape_features(features_batch)
test_features <- reshape_features(test_features_batch)
```

7. Build a model with one dense layer with 256 units and "relu" activation, one dropout layer with 50% dropout rate, and a dense output layer with appropriate parameters. (15 points)

```
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu",
              input_shape = 1 * 1 * 512) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = "sigmoid")

model
```

```
## Model
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense_1 (Dense)              (None, 256)               131328
## _________________________________________________________________________
## dropout_1 (Dropout)          (None, 256)               0
## _________________________________________________________________________
## dense_2 (Dense)              (None, 10)                2570
## =========================================================================
## Total params: 133,898
## Trainable params: 133,898
## Non-trainable params: 0
```

```
## ________________________________________________________________________
```

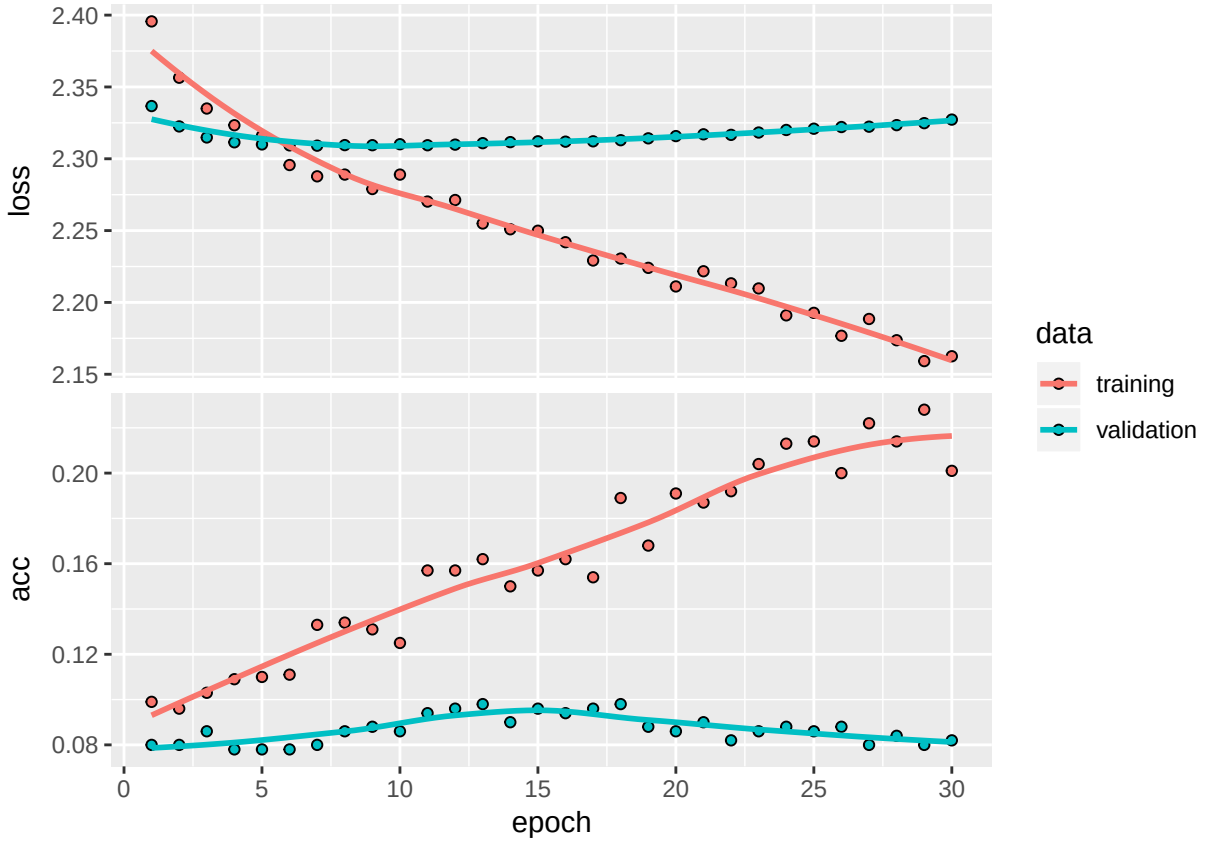
8. Compile the model with categorical_crossentropy as the loss function and optimizer_rmsprop with 0.01% learning rate (lr=0.0001). (5 points)

```
model %>% compile(
  optimizer = optimizer_rmsprop(lr=0.0001),
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)
```

9. Fit the model using 30 epochs. Plot the loss and accuracies. (5 points)

```
history <- model %>% fit(
  train_features, train_labels_categorical,
  epochs = 30,
  batch_size = 20,
  validation_data = list(test_features, test_labels_categorical)
)

plot(history)
```



10. Note that the model is likely to have low accuracy. Explain why. (10 points)

This model is overfitting almost from the start. The accuracy for the training set is drastically higher than that of the validation set. The overfitting is probably due to not using data augmentation, which is important for a small dataset like this.