Data 609 Module 8 HW

Bryan Persaud

5/23/2021

Ex.1

Use the nnet package to analyze the iris data set. Use 80% of the 150 samples as the training data and the rest for validation. Discuss the results.

Solution:

```
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 4.0.5
```

```
set.seed(123)
iris_split <- sort(sample(nrow(iris), nrow(iris) * 0.8))
train <- iris[iris_split, -5]
validation <- iris[-iris_split, -5]
iris_network <- nnet(Species ~ ., data = iris, subset = iris_split, size = 2)
```

```
## # weights:  19
## initial  value 135.090792
## iter  10 value 58.046705
## iter  20 value 16.784114
## iter  30 value 3.034567
## iter  40 value 0.346292
## iter  50 value 0.000433
## iter  60 value 0.000159
## final  value 0.000100
## converged
```

```
summary(iris_network)
```

```
## a 4-2-3 network with 19 weights
## options were - softmax modelling
##    b->h1   i1->h1   i2->h1   i3->h1   i4->h1
##    -0.21    -7.16    -0.71   -12.55    -5.18
##    b->h2   i1->h2   i2->h2   i3->h2   i4->h2
##     1.43     1.20     1.14    -2.19    -1.80
##    b->o1   h1->o1   h2->o1
##  -617.25     8.59  1157.98
##    b->o2   h1->o2   h2->o2
```

```
##   175.23    -5.15   251.89
##    b->o3   h1->o3   h2->o3
##   441.92    -3.53 -1409.57
```

Using the nnet package we can see a neural network created for the iris dataset. There are 19 weights used for the iterations with a final value of 0.000100.

Ex.2

As a mini project, install the keras package and learn how to use it. Then, carry out various tasks that may be useful to your project and studies.

Solution:

Using the keras package also involves using tensorflow. This can be installed with keras using the install_keras() function. I will be using the imdb dataset included in the keras package to learn how to use the keras package.

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.0.5
```

```
imdb <- dataset_imdb(num_words = 20000)
x_train <- imdb$train$x %>%
  pad_sequences(maxlen = 100)
x_test <- imdb$test$x %>%
  pad_sequences(maxlen = 100)
y_train <- imdb$train$y
y_test <- imdb$test$y
```

We load in the imdb dataset and prepare the data by taking the train and test data from the imdb dataset.

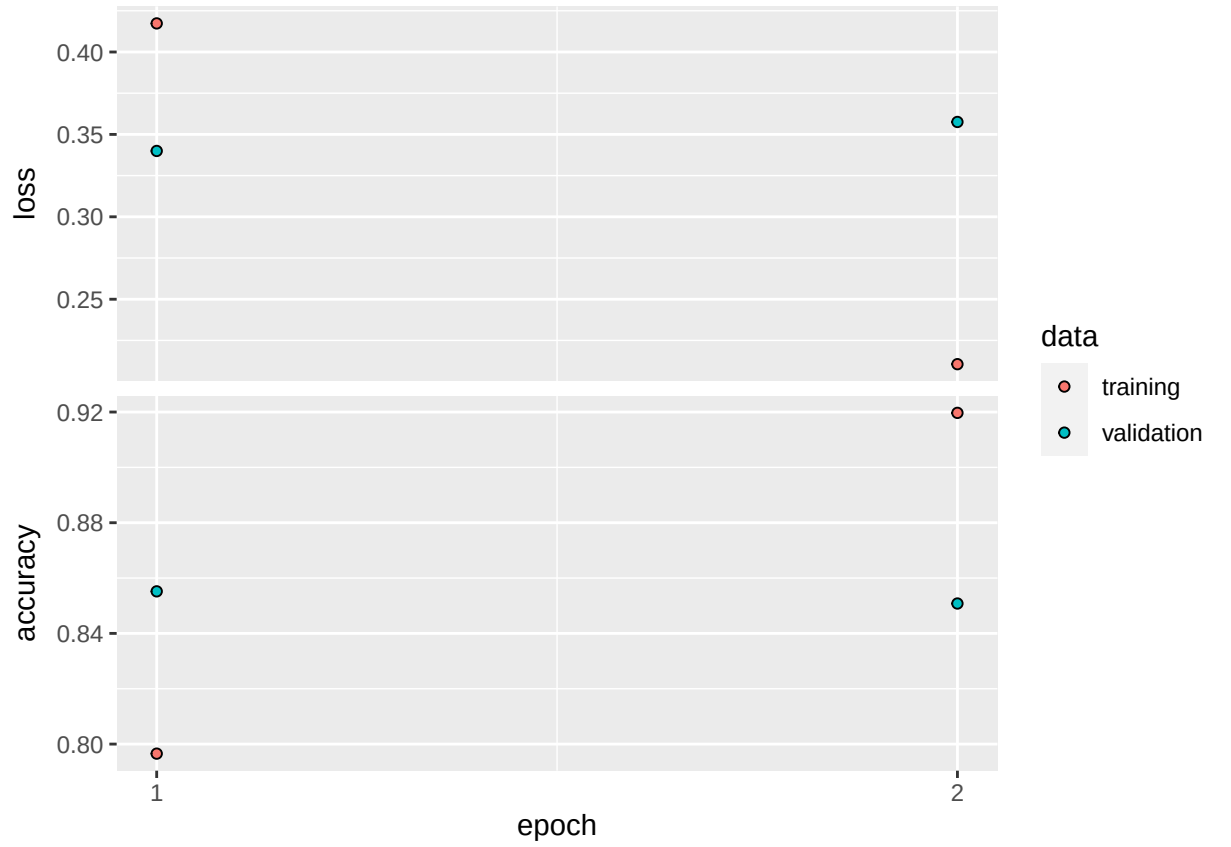
```
model <- keras_model_sequential()
model %>%
  layer_embedding(20000, 128, input_length = 100) %>%
  layer_dropout(0.25) %>%
  layer_conv_1d(64, 4, padding = "valid", activation = "relu", strides = 1) %>%
  layer_max_pooling_1d(4) %>%
  layer_lstm(70) %>%
  layer_dense(1) %>%
  layer_activation("sigmoid")
```

We create our model to be used. For keras the %>% operator is used.

```
model %>%
  compile(loss = "binary_crossentropy", optimizer = "adam", metrics = c("accuracy"))
model_fit <- model %>%
  fit(x_train, y_train, epochs = 2, batch_size = 30, validation_split = 0.2)
```

We add in a compiler and fit in the train data for x and y into the model.

```
plot(model_fit)
```



Here is a plot of the fitted data.

```
model %>%
  evaluate(x_test, y_test, verbose = 0)
```

```
##     loss accuracy
## 0.385035 0.841120
```

Here the test data for x and y are being used to evaluate the performance of the model. The model has an accuracy of 0.8433600 or 84.34%.

Using keras package I learned how to prepare the data with train and test data and create a model to use these data. There are a lot of built in functions to help build the model and even to add in a compiler to be used. Once you build the model you want you can then fit in the train and test data you are using. From here you can evaluate your model and plot the data to see how accurate your model is as well as make predictions. Overall the keras package was a good tool to learn and I see it being useful in future project and studies.

References

https://blog.rstudio.com/2017/09/05/keras-for-r/