Homework 1

Adam Shelton

5/8/2019

Getting Data

```
fashion_mnist = keras::dataset_fashion_mnist()

x_train = fashion_mnist$train$x
y_train = fashion_mnist$train$y

x_test = fashion_mnist$test$x
y_test = fashion_mnist$test$y

rm(fashion_mnist)

x_train = array_reshape(x_train, c(nrow(x_train), 28 * 28)) / 255
x_test = array_reshape(x_test, c(nrow(x_test), 28 * 28)) / 255

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Initial test

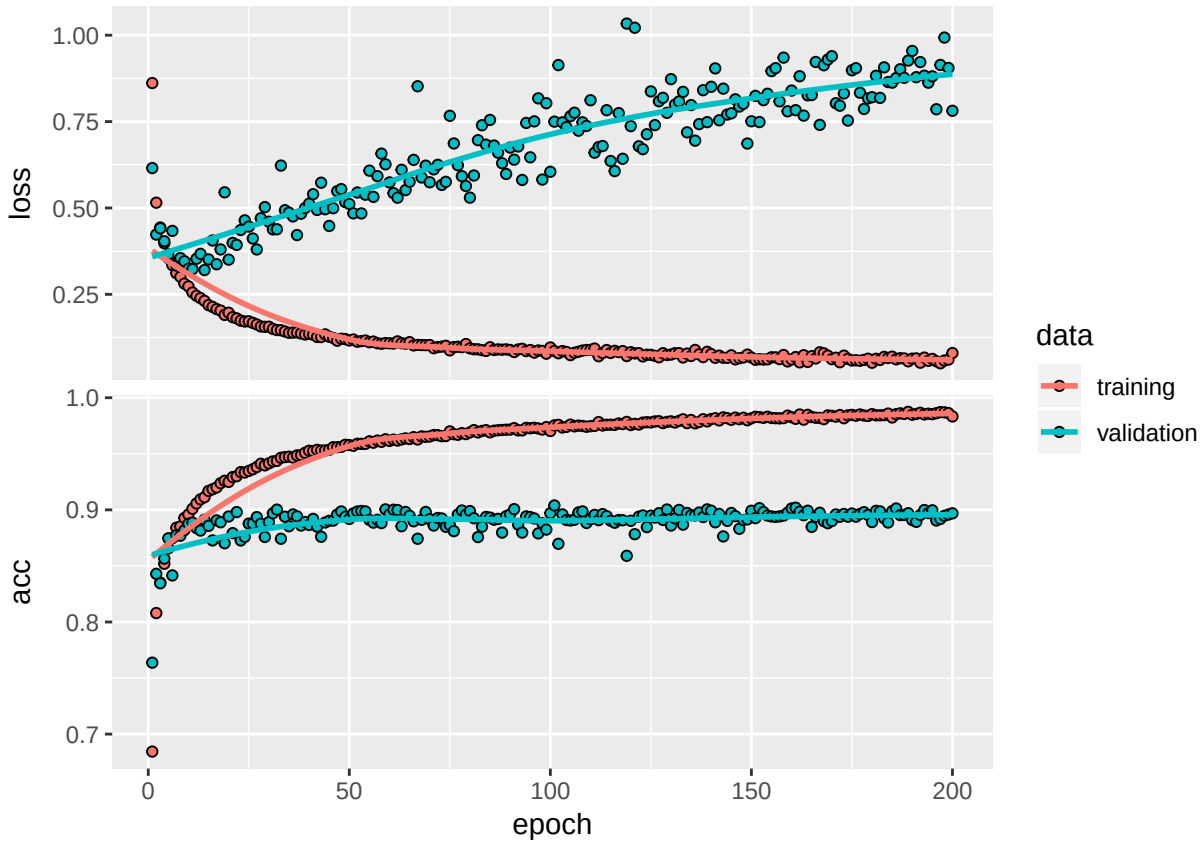
```
model = keras_model_sequential()

model %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(784)) %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

history_init = model %>% fit(
  x_train, y_train,
  epochs = 200, batch_size = 512,
  validation_split = 0.1666666666666667
)

plot(history_init)
```




```
model %>% evaluate(x_test, y_test)
```

```
## $loss
## [1] 0.8245576
##
## $acc
## [1] 0.8918
```

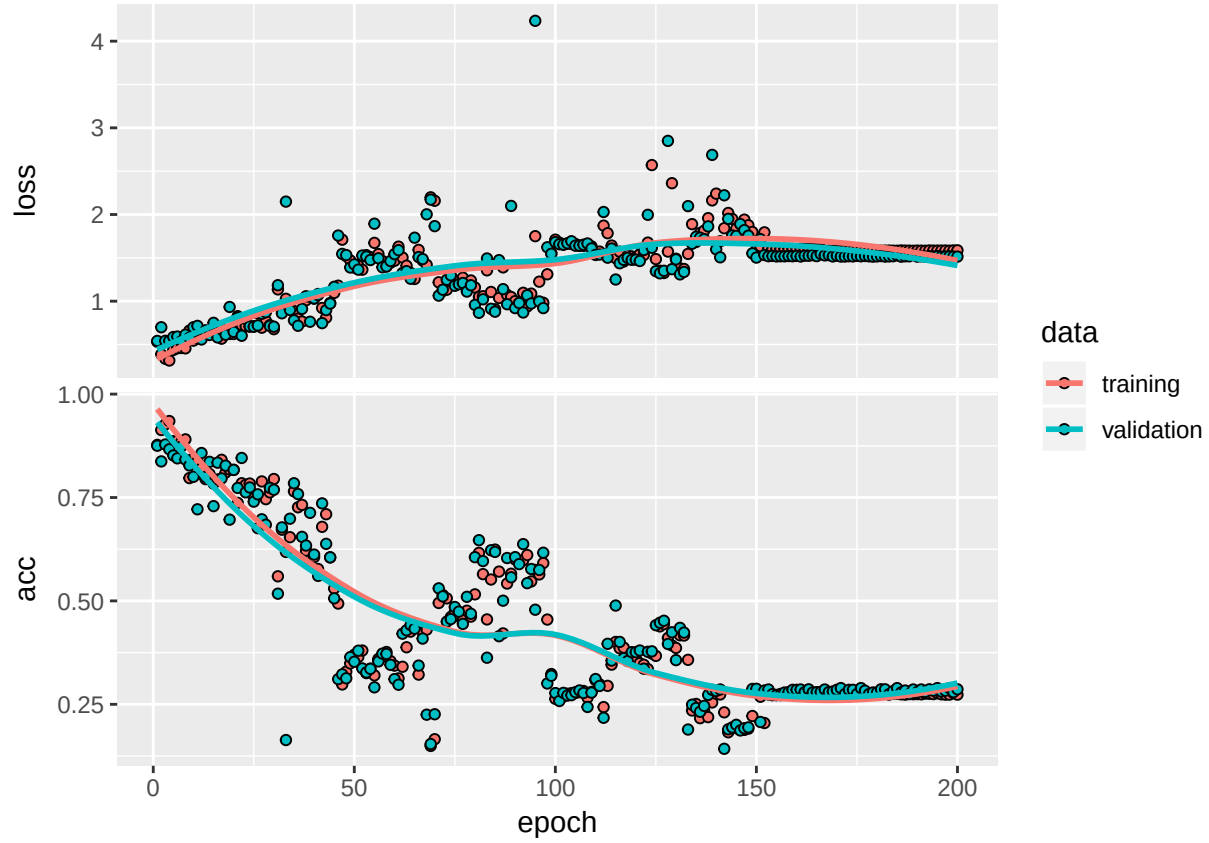
Implementing Dropout

```
model %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = 'softmax')

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

```
history_do = model %>% fit(
  x_train, y_train,
  epochs = 200, batch_size = 512,
  validation_split = 0.1666666666666667
)

plot(history_do)
```




```
model %>% evaluate(x_test, y_test)
```

```
## $loss
## [1] 1.516909
##
## $acc
## [1] 0.2806
```

Weight Regularization

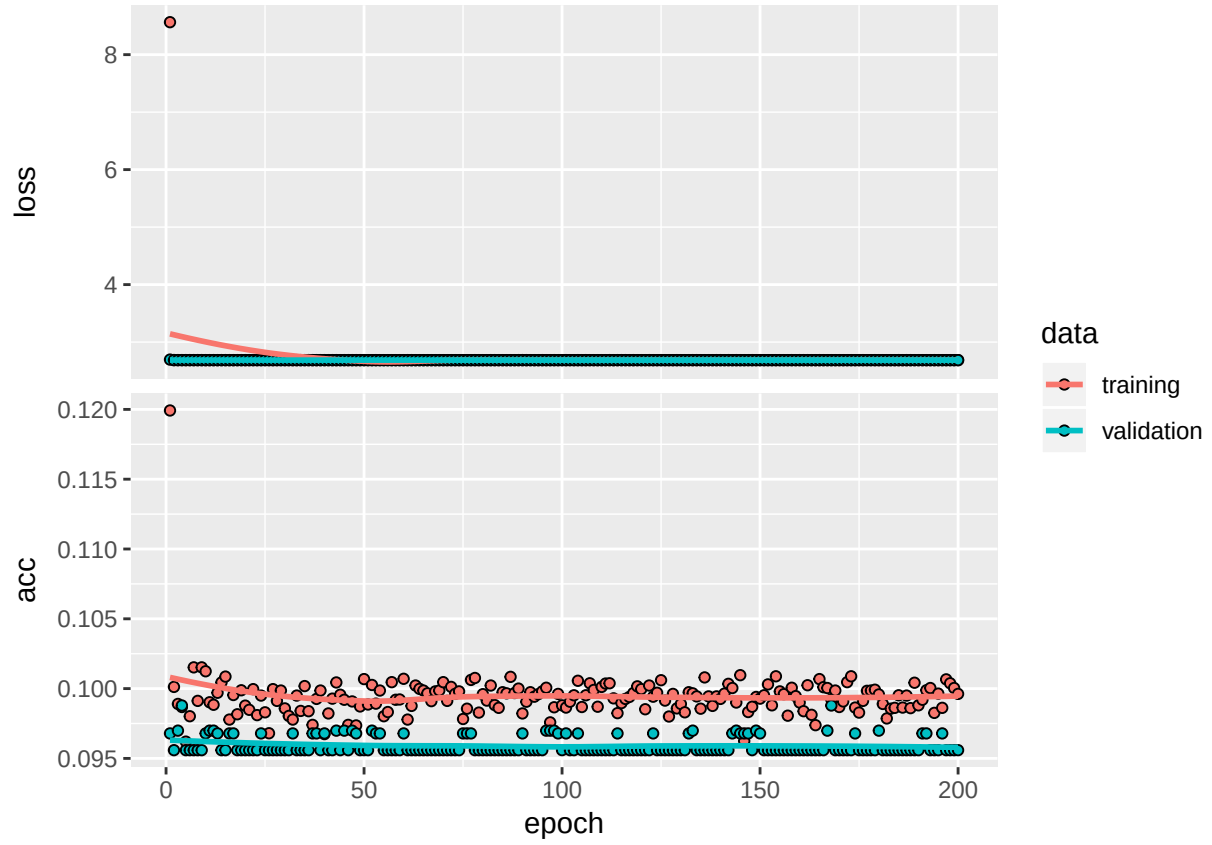
```
model %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(784), kernel_regularizer = regularizer_l
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
```

```
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

history_l1 = model %>% fit(
  x_train, y_train,
  epochs = 200, batch_size = 512,
  validation_split = 0.1666666666666667
)

plot(history_l1)
```




```
model %>% evaluate(x_test, y_test)
```

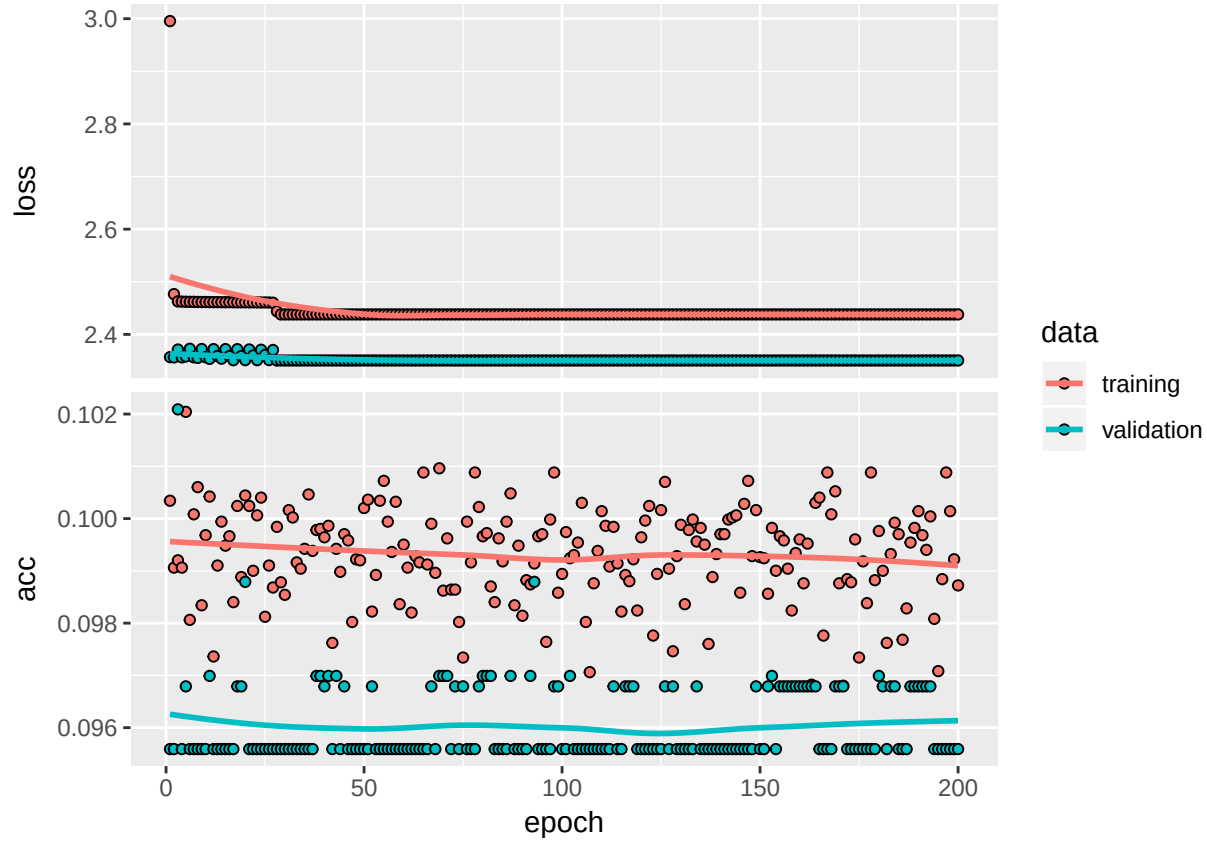
```
## $loss
## [1] 2.685974
##
## $acc
## [1] 0.1
```

```
model %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(784), kernel_regularizer = regularizer_
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l2(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l2(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = 'relu', kernel_regularizer = regularizer_l2(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

history_l2 = model %>% fit(
  x_train, y_train,
  epochs = 200, batch_size = 512,
  validation_split = 0.1666666666666667
)

plot(history_l2)
```



```
model %>% evaluate(x_test, y_test)
```

```
## $loss
## [1] 2.350299
##
## $acc
## [1] 0.1
```

Other Options

```
tune_grid = expand.grid(num_units = c(256, 512), bat_size = c( 512), epochs = c(50, 200), do_ratio = c(

hyper_param_tune = function(index, t_grid, model) {
  params = t_grid[index,]

  if (params$weight_method == 2) {
    model %>%
  layer_dense(units = params$num_units, activation = 'relu', input_shape = c(784), kernel_regularizer =
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l2(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l2(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l2(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = 10, activation = 'softmax')
  } else{
    model %>%
  layer_dense(units = params$num_units, activation = 'relu', input_shape = c(784), kernel_regularizer =
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l1(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l1(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = params$num_units, activation = 'relu', kernel_regularizer = regularizer_l1(params
  layer_dropout(rate = params$do_ratio) %>%
  layer_dense(units = 10, activation = 'softmax')
  }


model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

history_l2 = model %>% fit(
  x_train, y_train,
  epochs = params$epochs, batch_size = params$bat_size,
  validation_split = 0.1666666666666667
)

model %>% evaluate(x_test, y_test) %>% as_tibble() %>% bind_cols(params) %>% return()
```

```
}

results = sapply(1:length(tune_grid$num_units), hyper_param_tune, tune_grid, model)
```

Final Model