MACS 30200 HW1

Ling Dai

```
#import libraries
library(keras)
library(tensorflow)
use_python("/Users/lingdai/anaconda3/bin/python3")

#import data
fmnist <- dataset_fashion_mnist()

#set seed
set.seed(1234)
```

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

train_labels <- to_categorical(train_labels)
test_labels <- to_categorical(test_labels)

#Preprocess the data by converting the data to a 2D tensor with individual values between 0 and 1
img_rows <- img_cols <- 28
train_images <- array_reshape(train_images, c(60000, 28*28))
train_images <- train_images / 255
str(train_images)
```

```
##  num [1:60000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
```

```
test_images <- array_reshape(test_images, c(10000, 28*28))
test_images <- test_images / 255
str(test_images)
```

```
##  num [1:10000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
```

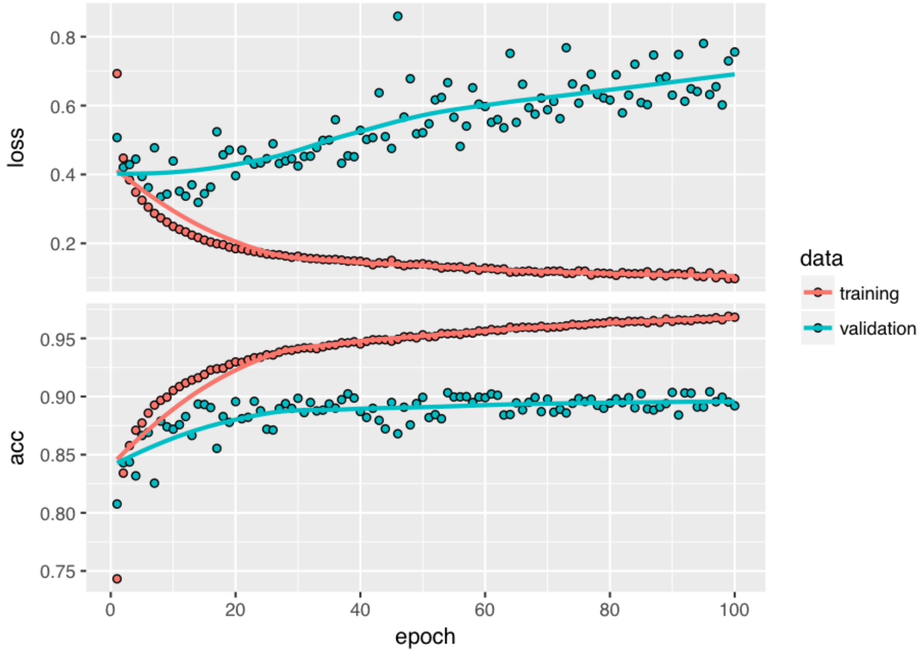
```
#Randomly split the training data into 50,000 training observations and 10,000 validation observations
training_ind <- sample(60000, size = 50000)
training_images <- train_images[training_ind, ]
training_labels <- train_labels[training_ind, ]
validation_images <- train_images[-training_ind, ]
validation_labels <- train_labels[-training_ind, ]
```

Alternative Models

```
#lowest 0.33 @ 14
network <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history5 <- network %>% fit(training_images, training_labels,
                 epochs = 100, batch_size = 256,
                 validation_data = list(validation_images, validation_labels))
plot(history5)
```



```
min(history5$metrics$val_loss)
```

```
## [1] 0.3188581
```

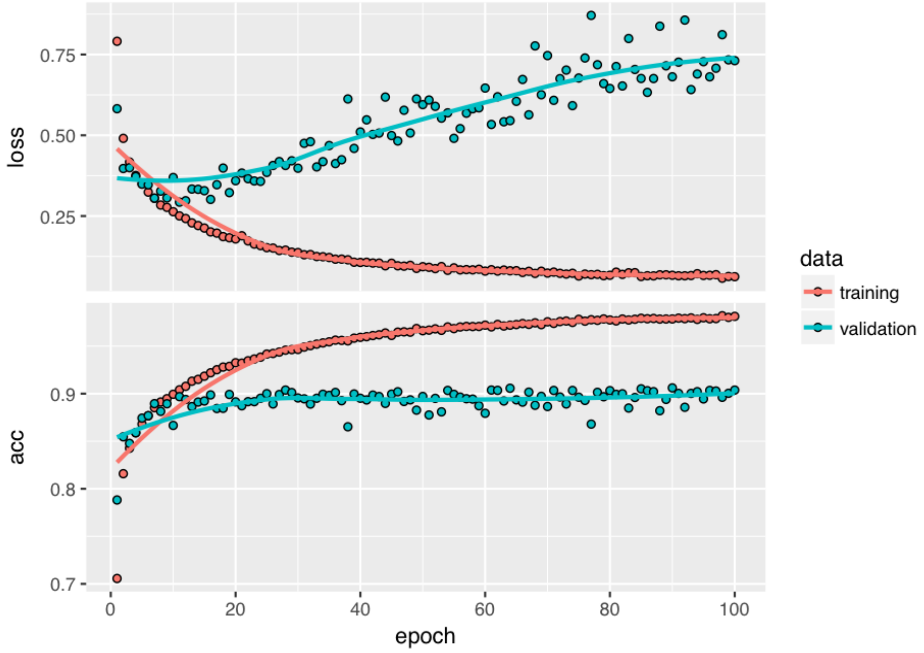
```
which(history5$metrics$val_loss==min(history5$metrics$val_loss))
```

```
## [1] 14
```

```
#lowest 0.31 @ 9
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")
```

```r
network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history6 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 512,
                validation_data = list(validation_images, validation_labels))
plot(history6)
```



```r
min(history6$metrics$val_loss)
```

```
## [1] 0.2930373
```

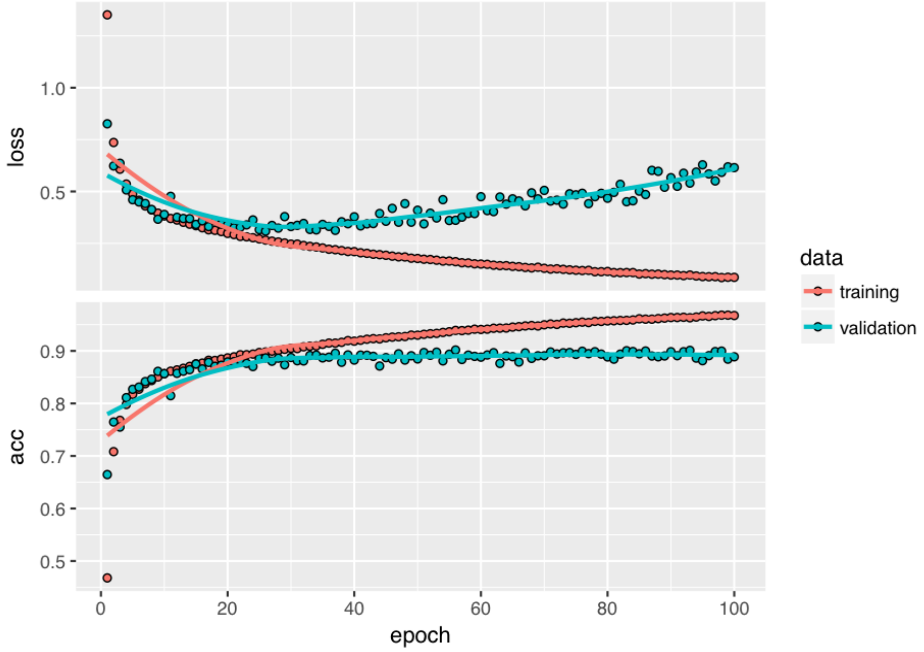
```r
which(history6$metrics$val_loss==min(history6$metrics$val_loss))
```

```
## [1] 11
```

```r
#lowest 0.31 @ 33
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history7 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 512,
                validation_data = list(validation_images, validation_labels))
plot(history7)
```



```
min(history7$metrics$val_loss)
```

```
## [1] 0.3089006
```

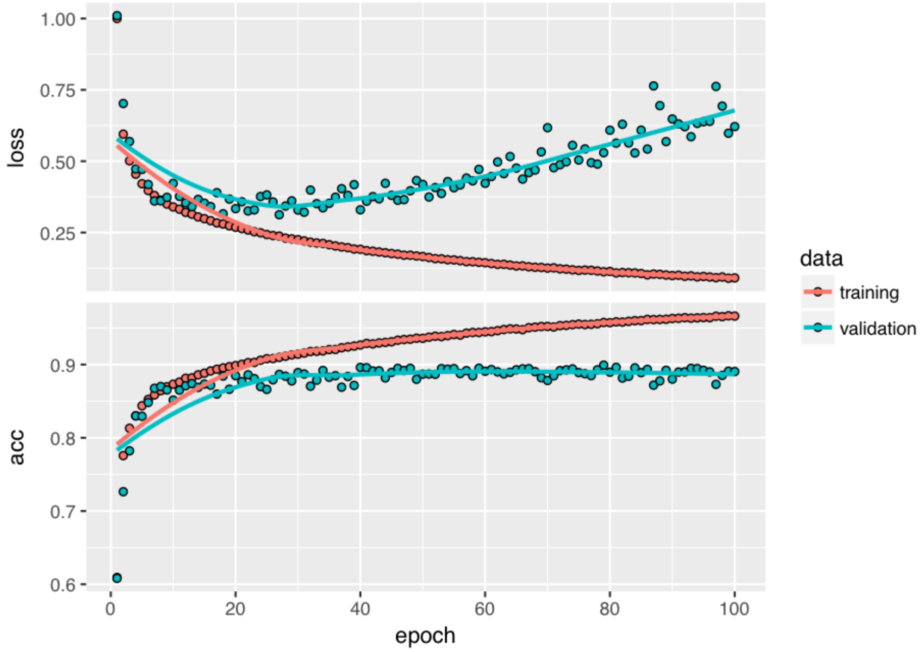
```
which(history7$metrics$val_loss==min(history7$metrics$val_loss))
```

```
## [1] 26
```

```
#lowest 0.31 @ 13
network <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history8 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 256,
                validation_data = list(validation_images, validation_labels))
plot(history8)
```



```
min(history8$metrics$val_loss)
```

```
## [1] 0.3125098
```

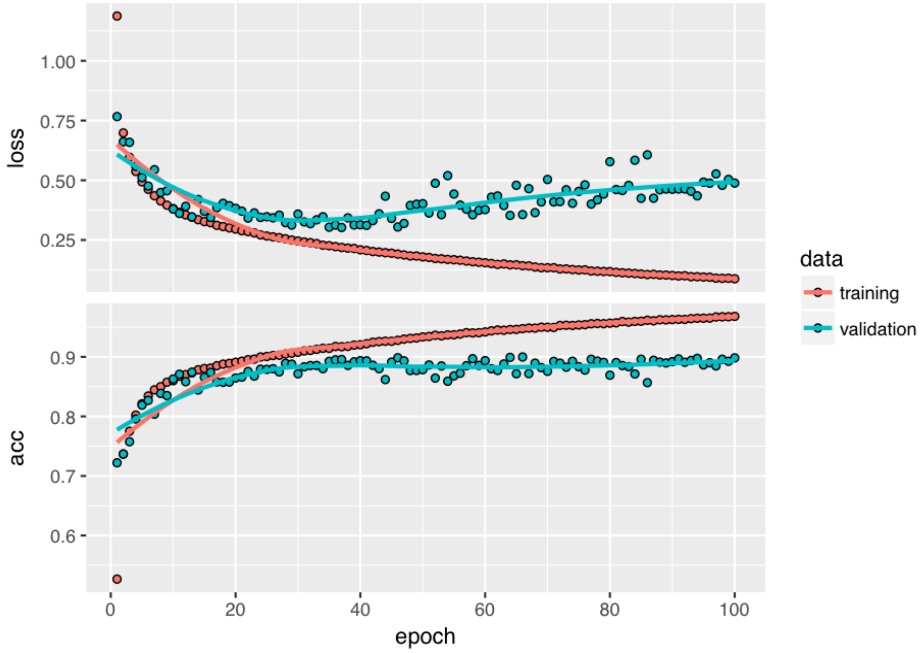
```
which(history8$metrics$val_loss==min(history8$metrics$val_loss))
```

```
## [1] 27
```

```
#lowest 0.3 @ 28
network <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history9 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 256,
                validation_data = list(validation_images, validation_labels))
plot(history9)
```



```
min(history9$metrics$val_loss)
```

```
## [1] 0.3023292
```

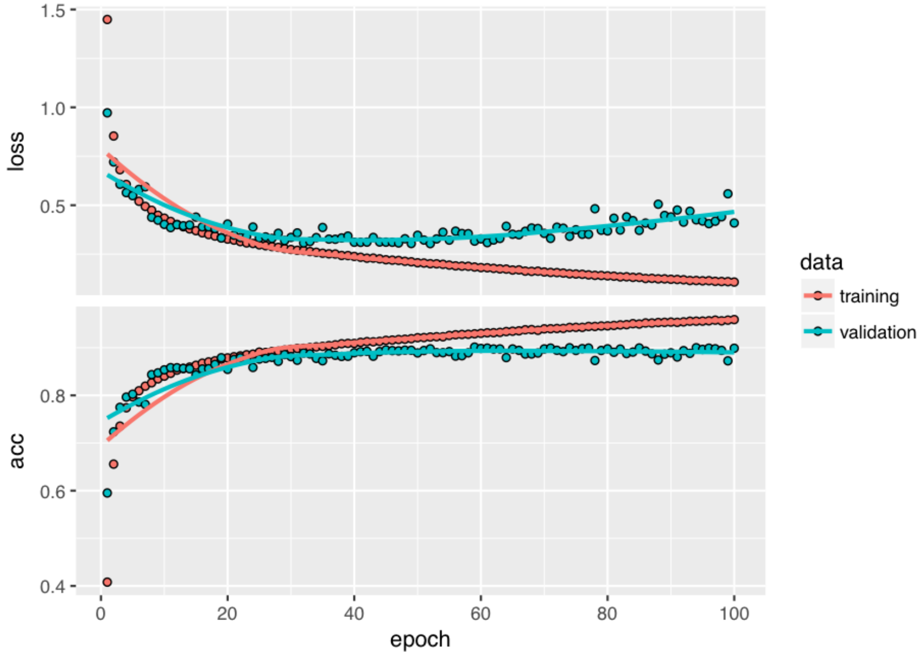
```
which(history9$metrics$val_loss==min(history9$metrics$val_loss))
```

```
## [1] 37
```

```
#lowest 0.3 @ 51
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "sigmoid", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history10 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 512,
                validation_data = list(validation_images, validation_labels))
plot(history10)
```



```
min(history10$metrics$val_loss)
```

```
## [1] 0.3038093
```

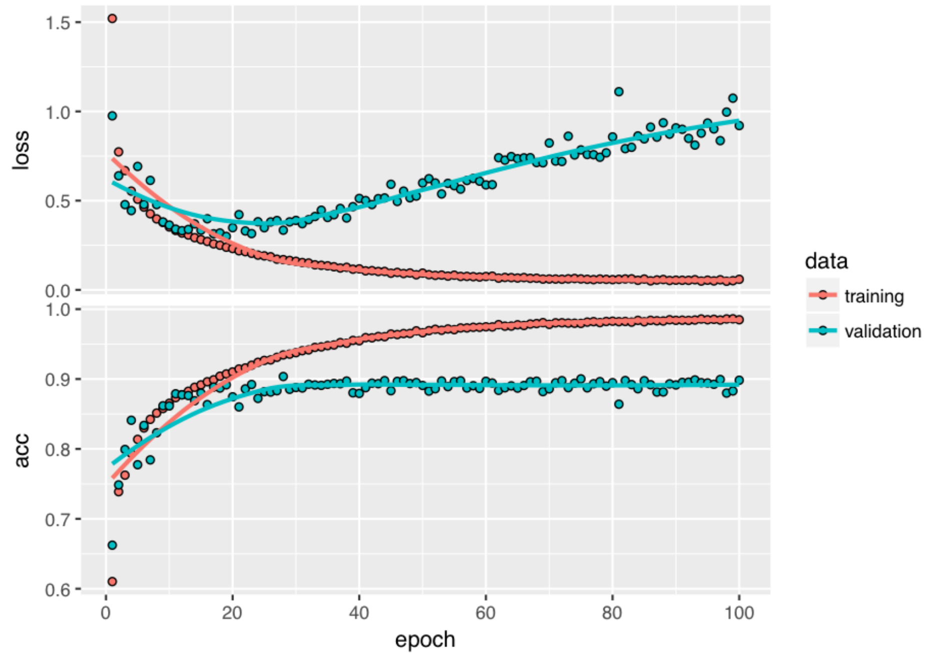
```
which(history10$metrics$val_loss==min(history10$metrics$val_loss))
```

```
## [1] 52
```

```
#lowest 0.31 @ 17
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "elu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "elu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "elu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "elu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)
```

```
history11 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 512,
                validation_data = list(validation_images, validation_labels))
plot(history11)
```



```
min(history11$metrics$val_loss)
```

```
## [1] 0.2995572
```

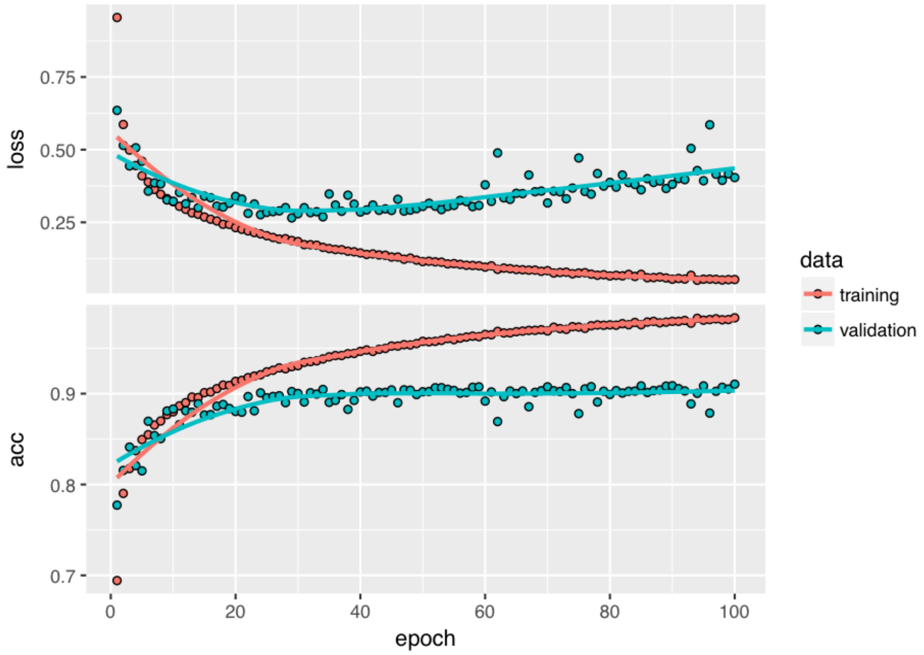
```
which(history11$metrics$val_loss==min(history11$metrics$val_loss))
```

```
## [1] 19
```

```
# 0.26 @ 33
network <- keras_model_sequential() %>%
  layer_dense(units = 1024, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history12 <- network %>% fit(training_images, training_labels,
                epochs = 100, batch_size = 1024,
                validation_data = list(validation_images, validation_labels))
plot(history12)
```



```
min(history12$metrics$val_loss)
```

```
## [1] 0.2652172
```

```
which(history12$metrics$val_loss==min(history12$metrics$val_loss))
```

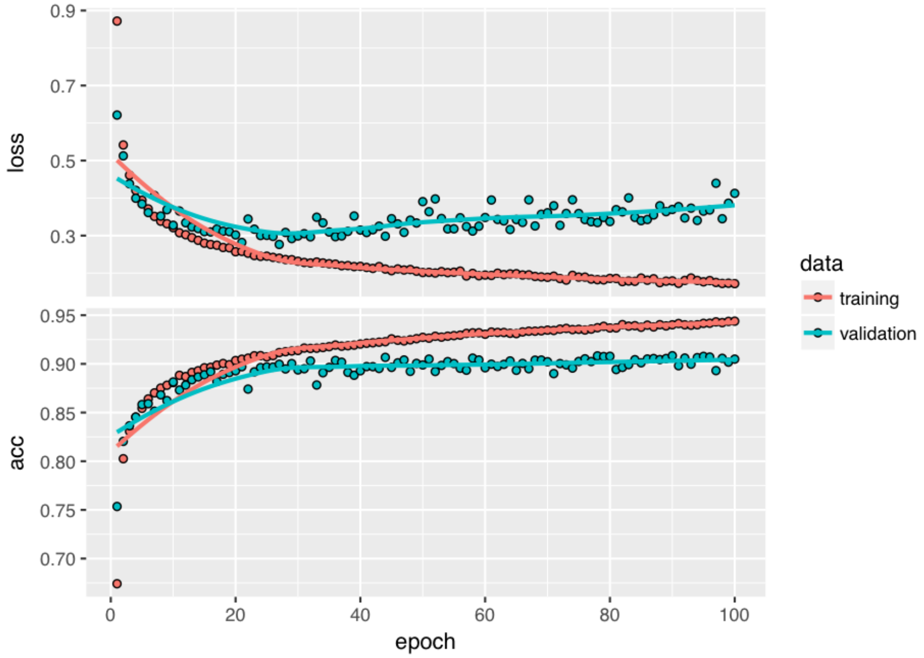
```
## [1] 29
```

```
#0.28 @ 26
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history13 <- network %>% fit(training_images, training_labels,
                             epochs = 100, batch_size = 512,
                              validation_data = list(validation_images, validation_labels))

plot(history13)
```



```
min(history13$metrics$val_loss)
```

```
## [1] 0.2763669
```

```
which(history13$metrics$val_loss==min(history13$metrics$val_loss))
```

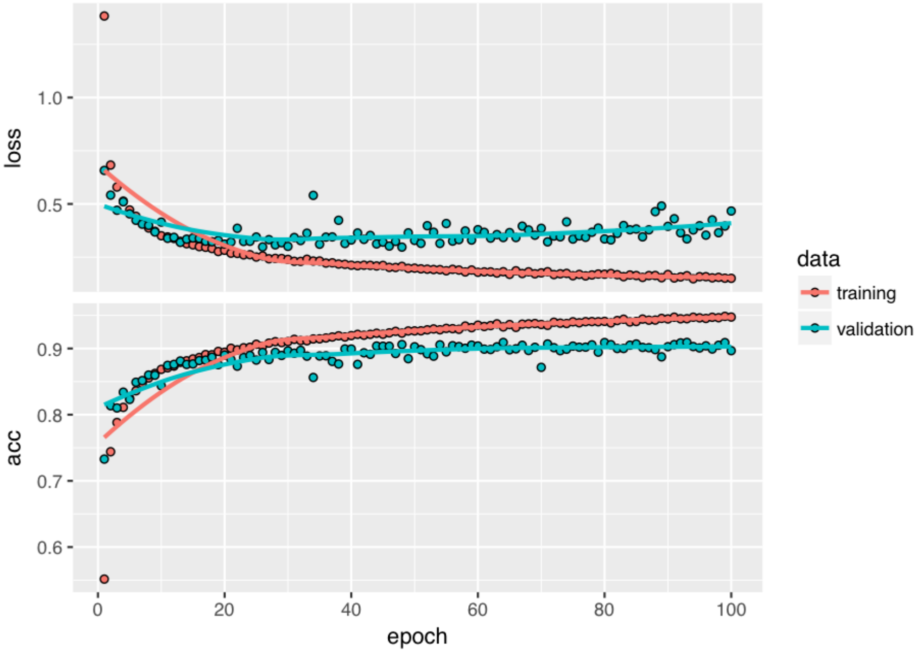
```
## [1] 27
```

```
#0.3 @ 32
network <- keras_model_sequential() %>%
  layer_dense(units = 1024, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 1024, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 1024, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 1024, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history13 <- network %>% fit(training_images, training_labels,
                            epochs = 100, batch_size = 1024,
                            validation_data = list(validation_images, validation_labels))

plot(history13)
```



```
min(history13$metrics$val_loss)
```

```
## [1] 0.2970895
```

```
which(history13$metrics$val_loss==min(history13$metrics$val_loss))
```

```
## [1] 48
```