HW2

Shinjo Hiroki

2024-01-12

1.

(a)

Exists.

$$(\gamma_0, \gamma_1, \gamma_2) = (-2, 1, 1)$$

(b)

No.

(c)

Exists.

$$\begin{aligned}(a_0, a_1, a_2) &= (0, -1, -1)\\ (b_{10}, b_{11}, b_{12}) &= (-2, 1, 1)\\ (b_{20}, b_{21}, b_{22}) &= (0, -1, -1)\end{aligned}$$

2.

```r
pacman::p_load(ISLR2,
               tidyverse,
               dplyr,
               keras,
               tensorflow)
use_condaenv("r-tensorflow",required = T)
set.seed(123)
data(Smarket)
```

(a)

```r
Smarket$log_Volumne <- log(Smarket$Volume)

# Split data set for the tranning and test#
train <- filter(Smarket,Year != 2005)
test <- filter(Smarket,Year == 2005)

# Data Frama
y_train <- train$Today
x_train<- train %>%
  select(Lag1,Lag2,Lag3,Lag4,Lag5,log_Volumne)

y_test <- test$Today
x_test<- test %>%
  select(Lag1,Lag2,Lag3,Lag4,Lag5,log_Volumne)

# Input Matrixs
x_train_mat <- array_reshape(as.matrix(x_train), c(nrow(as.matrix(x_train)), 6, 1))
x_test_mat <- array_reshape(as.matrix(x_test), c(nrow(as.matrix(x_test)), 6, 1))
y_train_mat <- as.matrix(y_train)
y_test_mat <- as.matrix(y_test)
```

(b)

```r
LR <- lm(Today ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + log_Volumne,data = train)
yhat.LR = predict(LR, newdata = test)
mse.b <- mean((yhat.LR - y_test)^2)
mse.b
```

```
## [1] 0.4170463
```

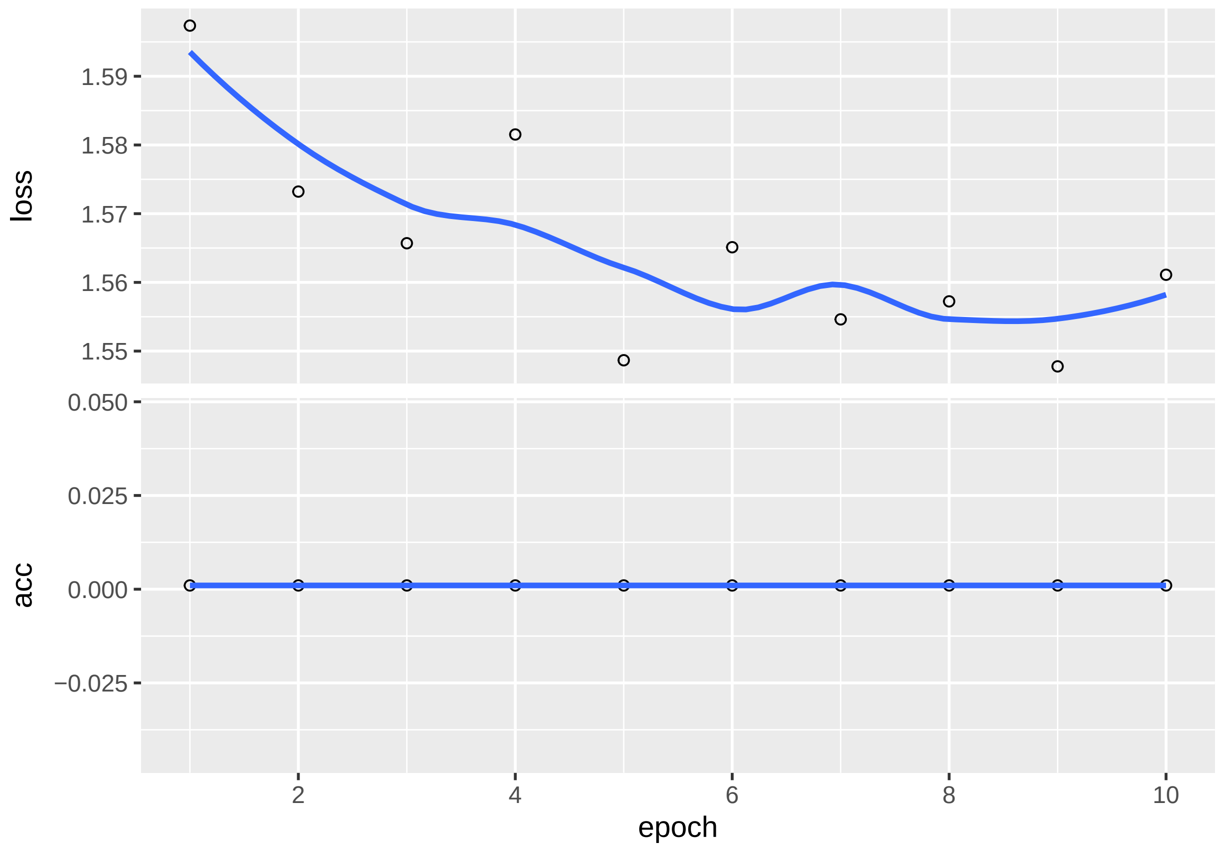
(c)

```r
model_RNN <- keras_model_sequential()
model_RNN %>%
  layer_simple_rnn(units = 50, input_shape = c(6, 1),activation = "linear") %>%
  layer_dense(units = 1, activation = "linear")

model_RNN %>% compile(optimizer = "rmsprop",
                  loss = "mean_squared_error", metrics = c("acc"))
history <- model_RNN %>% fit(
  x_train_mat, y_train_mat,
  epochs = 10,batch_size = 32)
```

```
## Epoch 1/10
## 32/32 - 1s - loss: 1.5974 - acc: 0.0010 - 952ms/epoch - 30ms/step
## Epoch 2/10
## 32/32 - 0s - loss: 1.5732 - acc: 0.0010 - 132ms/epoch - 4ms/step
## Epoch 3/10
## 32/32 - 0s - loss: 1.5657 - acc: 0.0010 - 102ms/epoch - 3ms/step
## Epoch 4/10
## 32/32 - 0s - loss: 1.5815 - acc: 0.0010 - 84ms/epoch - 3ms/step
## Epoch 5/10
## 32/32 - 0s - loss: 1.5487 - acc: 0.0010 - 86ms/epoch - 3ms/step
## Epoch 6/10
## 32/32 - 0s - loss: 1.5651 - acc: 0.0010 - 82ms/epoch - 3ms/step
## Epoch 7/10
## 32/32 - 0s - loss: 1.5546 - acc: 0.0010 - 71ms/epoch - 2ms/step
## Epoch 8/10
## 32/32 - 0s - loss: 1.5572 - acc: 0.0010 - 83ms/epoch - 3ms/step
## Epoch 9/10
## 32/32 - 0s - loss: 1.5478 - acc: 0.0010 - 83ms/epoch - 3ms/step
## Epoch 10/10
## 32/32 - 0s - loss: 1.5611 - acc: 0.0010 - 84ms/epoch - 3ms/step
```

```r
plot(history)
```



```r
yhat.RNN <- predict(model_RNN, x_test_mat)
```

```
## 8/8 - 0s - 149ms/epoch - 19ms/step
```

```r
mse.c <- mean((yhat.RNN - y_test_mat)^2)
mse.c
```

```
## [1] 0.4215221
```

(d)

```r
model_FNN <- keras_model_sequential()
model_FNN %>%
  layer_dense(units = 50, activation = "linear",input_shape = c(6,1)) %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 25, activation = "linear") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = "linear")

model_FNN %>% compile(optimizer = "rmsprop",
                  loss = "mean_squared_error", metrics = c("acc"))

history_FNN <- model_FNN %>% fit(
  x_train_mat, y_train_mat,
  epochs = 10, batch_size = 32)
```

```
## Epoch 1/10
## 32/32 - 1s - loss: 1.5933 - acc: 0.0010 - 702ms/epoch - 22ms/step
## Epoch 2/10
## 32/32 - 0s - loss: 1.5473 - acc: 6.6800e-04 - 99ms/epoch - 3ms/step
## Epoch 3/10
## 32/32 - 0s - loss: 1.5503 - acc: 0.0010 - 84ms/epoch - 3ms/step
## Epoch 4/10
## 32/32 - 0s - loss: 1.5343 - acc: 0.0010 - 76ms/epoch - 2ms/step
## Epoch 5/10
## 32/32 - 0s - loss: 1.5285 - acc: 0.0010 - 100ms/epoch - 3ms/step
## Epoch 6/10
## 32/32 - 0s - loss: 1.5358 - acc: 0.0010 - 97ms/epoch - 3ms/step
## Epoch 7/10
## 32/32 - 0s - loss: 1.5320 - acc: 0.0010 - 86ms/epoch - 3ms/step
## Epoch 8/10
## 32/32 - 0s - loss: 1.5269 - acc: 0.0010 - 121ms/epoch - 4ms/step
## Epoch 9/10
## 32/32 - 0s - loss: 1.5213 - acc: 0.0010 - 118ms/epoch - 4ms/step
## Epoch 10/10
## 32/32 - 0s - loss: 1.5233 - acc: 0.0010 - 106ms/epoch - 3ms/step
```

```r
yhat.FNN <- model_FNN %>% predict(x_test_mat)
```

```
## 8/8 - 0s - 158ms/epoch - 20ms/step
```

```r
mse.d <- mean((yhat.FNN-y_test)^2)
mse.d
```

```
## [1] 0.4179631
```

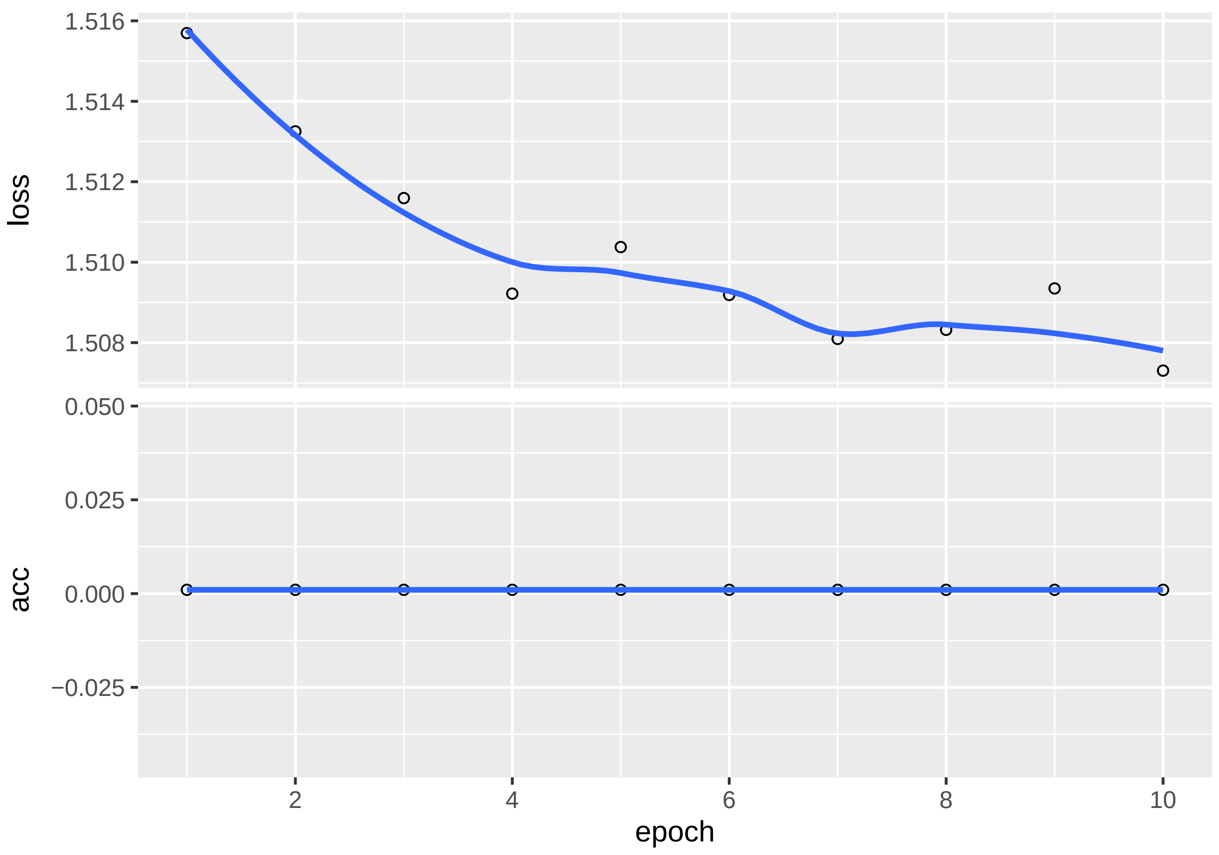
(e)

```r
model_LSTM <- keras_model_sequential()
model_LSTM %>%
  layer_lstm(units = 50, input_shape = c(6, 1),activation = "linear") %>%
  layer_dense(units = 1, activation = "linear")

model_LSTM %>% compile(optimizer = "rmsprop",
                  loss = "mean_squared_error", metrics = c("acc"))
history <- model_LSTM %>% fit(
  x_train_mat, y_train_mat,
  epochs = 10,batch_size = 32)
```

```
## Epoch 1/10
## 32/32 - 1s - loss: 1.5157 - acc: 0.0010 - 1s/epoch - 42ms/step
## Epoch 2/10
## 32/32 - 0s - loss: 1.5133 - acc: 0.0010 - 133ms/epoch - 4ms/step
## Epoch 3/10
## 32/32 - 0s - loss: 1.5116 - acc: 0.0010 - 120ms/epoch - 4ms/step
## Epoch 4/10
## 32/32 - 0s - loss: 1.5092 - acc: 0.0010 - 133ms/epoch - 4ms/step
## Epoch 5/10
## 32/32 - 0s - loss: 1.5104 - acc: 0.0010 - 117ms/epoch - 4ms/step
## Epoch 6/10
## 32/32 - 0s - loss: 1.5092 - acc: 0.0010 - 148ms/epoch - 5ms/step
## Epoch 7/10
## 32/32 - 0s - loss: 1.5081 - acc: 0.0010 - 130ms/epoch - 4ms/step
## Epoch 8/10
## 32/32 - 0s - loss: 1.5083 - acc: 0.0010 - 134ms/epoch - 4ms/step
## Epoch 9/10
## 32/32 - 0s - loss: 1.5093 - acc: 0.0010 - 129ms/epoch - 4ms/step
## Epoch 10/10
## 32/32 - 0s - loss: 1.5073 - acc: 0.0010 - 151ms/epoch - 5ms/step
```

```r
plot(history)
```



```r
yhat.LSTM <- predict(model_LSTM, x_test_mat)
```

```
## 8/8 - 0s - 173ms/epoch - 22ms/step
```

```r
mse.e <- mean((yhat.LSTM - y_test_mat)^2)
mse.e
```

```
## [1] 0.4204832
```

(f)

```r
mse <- c(mse.b,mse.c,mse.d,mse.e)
which(mse == min(mse))
```

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

Hence a linear regression achieves the smallest MSE.