ML and AI 2

Shubhang Periwal 19201104

3/29/2020

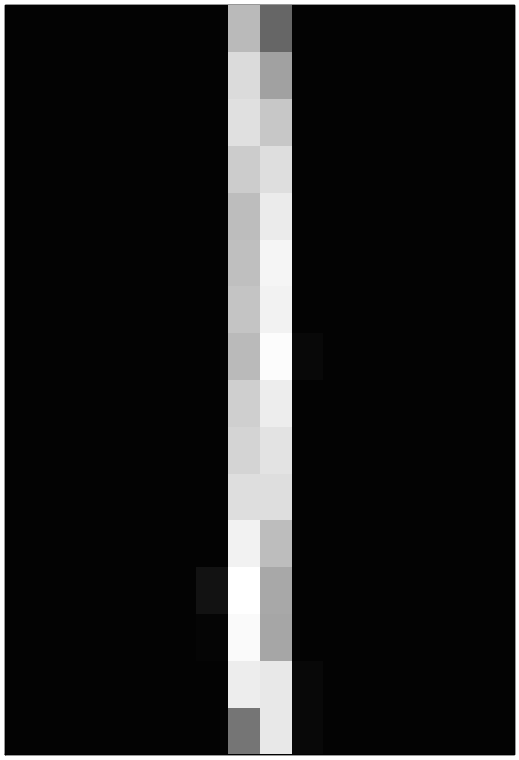
```
library(keras)
```

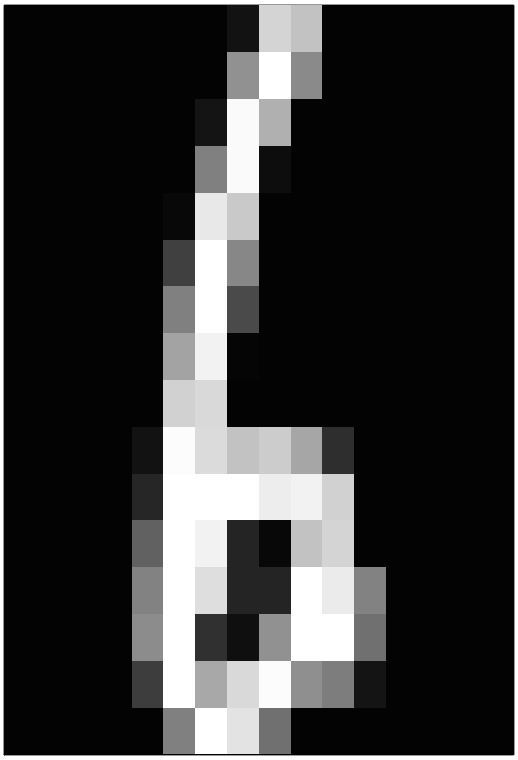
```
## Warning: package 'keras' was built under R version 3.6.3
```

```
tensorflow::tf$random$set_seed(0)
load("D:/data_usps_digits.RData")
```

```
# helper function
# to plot the digit
plot_digit <- function(index, data) {
tmp <- (-data + 1) / 2 # to convert back to original
z <- matrix( data = as.numeric(data[index, 256:1]), 16, 16 )
image(z[16:1,1:16], col = gray((1:100)/100),
xaxt = "n", yaxt = "n")
}

# plot few example digits
par(mfrow = c(1,2), mar = rep(1.5, 4))
plot_digit(14, x_train)
plot_digit(900, x_train)
```





```
#preprocessing
# Convert y to categorical using one-hot encoding
y_train <- to_categorical(y_train, num_classes = 10)
y_test <- to_categorical(y_test , num_classes = 10)
# convert x_train and x_test from data frame to matrix for valid network input
x_train <- as.matrix(x_train)
x_test <- as.matrix(x_test)

# normalizen x(input) to 0-1
range_norm <- function(x, a = 0, b = 1) {
( (x - min(x)) / (max(x) - min(x)) )*(b - a) + a }

x_train <- apply(x_train, 2, range_norm)
x_test <- apply(x_test, 2, range_norm)
range(x_train)
```

```
## [1] 0 1
```

```
range(x_test)
```

```
## [1] 0 1
```

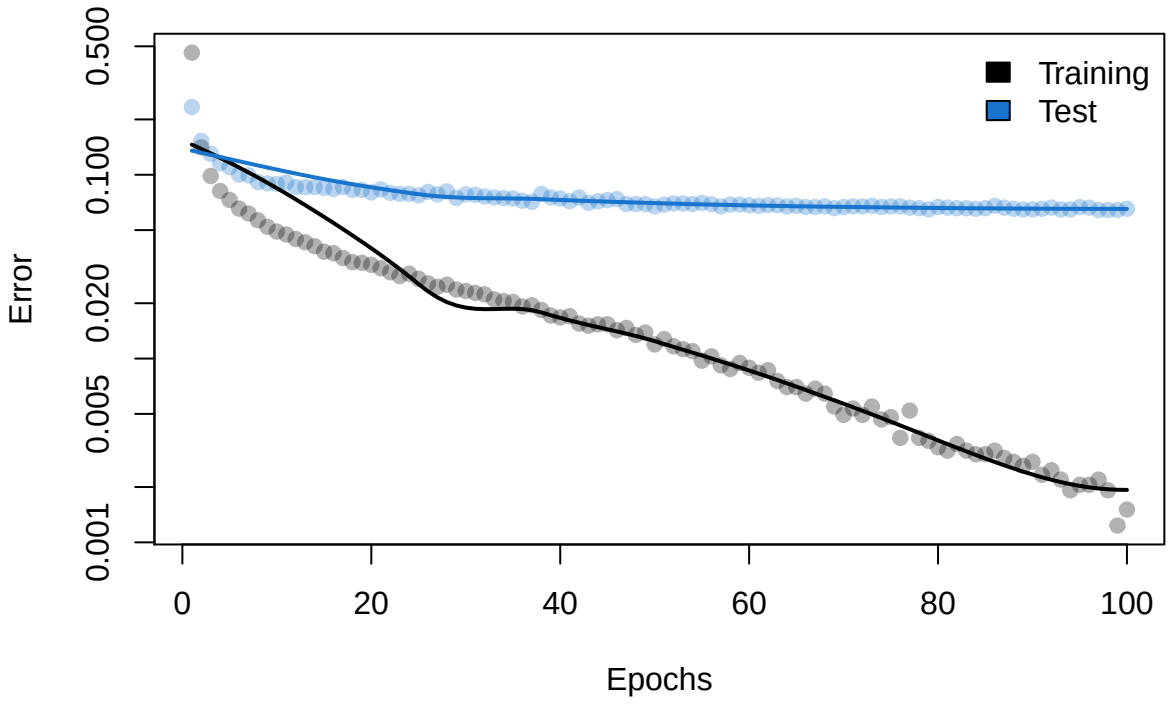
```
V <- ncol(x_train) # 256
model_2h <- keras_model_sequential() %>%
```

```
layer_dense(units = 256, activation = "relu", input_shape = V,) %>%#input layer

  #hidden layer 1
layer_dense(units = 64, activation = "relu") %>%
  #hidden layer 2
layer_dense(units = 10, activation = "softmax") %>%
  #output layer
compile(
loss = "categorical_crossentropy", metrics = "accuracy",
optimizer = optimizer_sgd(),
)
# count parameters
count_params(model_2h)
```

```
## [1] 82890
```

```
# fit the model on the training data
# and evaluate on test data at each epoch
fit <- model_2h %>% fit(
x = x_train, y = y_train,
validation_data = list(x_test, y_test),
epochs = 100,
verbose = 0
)
```

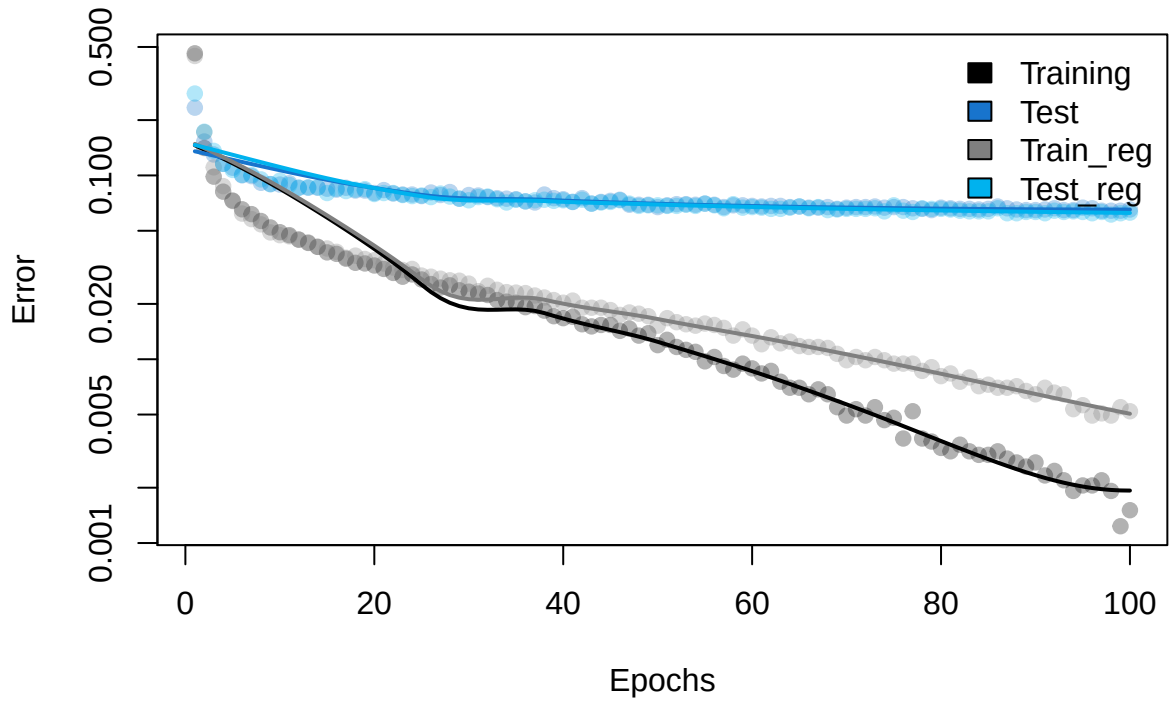


```
## [1] 82890
```

We compare the performance of the regularized model against the performance of the unregularized one.

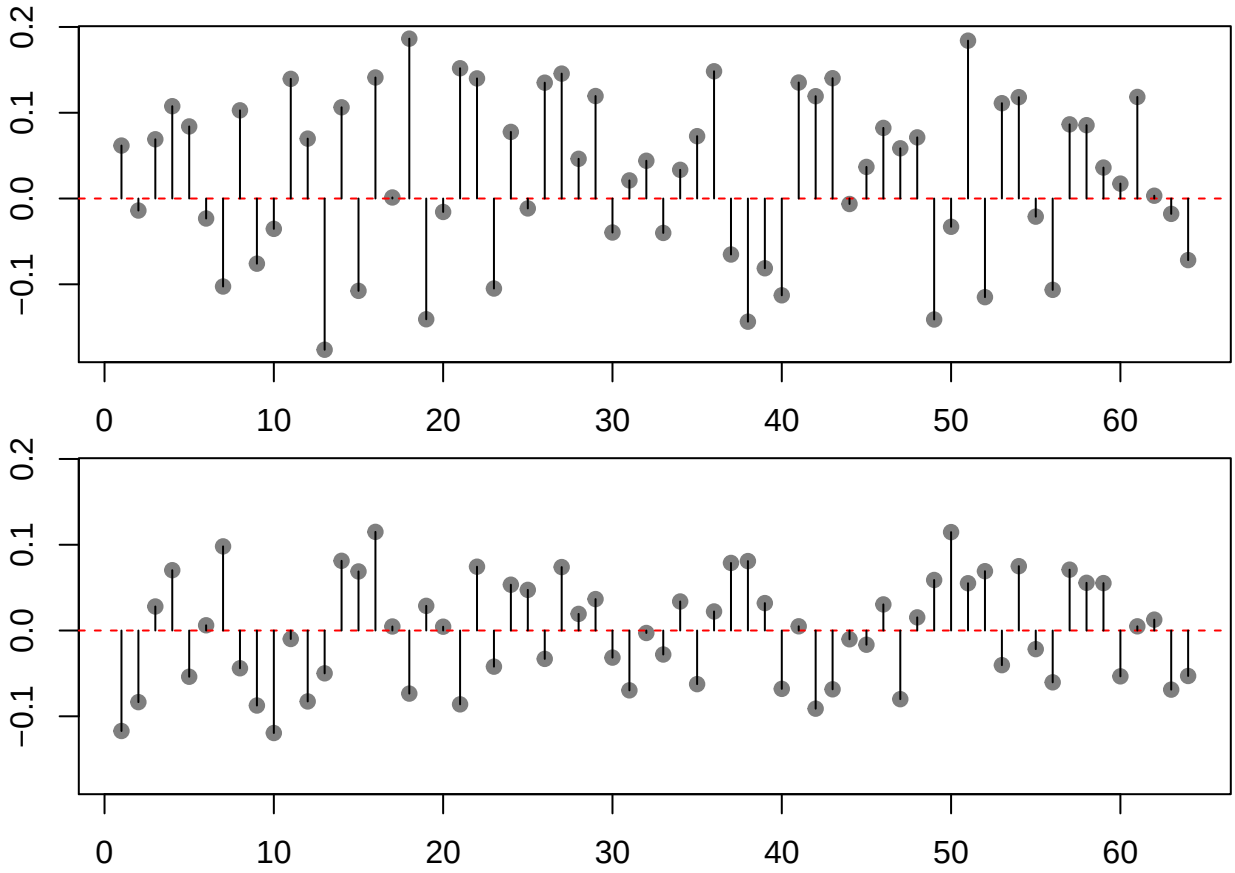
```
out <- 1 - cbind(fit$metrics$accuracy,
fit$metrics$val_accuracy,
fit_reg$metrics$accuracy,
fit_reg$metrics$val_accuracy)
# check performance
matplot(out, pch = 19, ylab = "Error", xlab = "Epochs",
col = adjustcolor(cols, 0.3),
log = "y")

matlines(apply(out, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("topright", legend = c("Training", "Test", "Train_reg", "Test_reg"),
fill = cols, bty = "n")
```




```
apply(out, 2, min)
```

```
## [1] 0.001234412 0.064275026 0.004937589 0.061285496
```




```
## [1] 84650
```

```
model_reg_3h <- keras_model_sequential() %>%
layer_dense(units = 256, activation = "relu", input_shape = V,#hidden layer 1
kernel_regularizer = regularizer_l2(l = 0.001)) %>%
layer_dense(units = 64, activation = "relu",#hidden layer 2
kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 32, activation = "relu",#hidden layer 3
kernel_regularizer = regularizer_l2(l = 0.001)) %>%
layer_dense(units = 10, activation = "softmax") %>%
compile(
loss = "categorical_crossentropy",
optimizer = optimizer_sgd(),
metrics = "accuracy"
)

# count parameters
count_params(model_reg_3h)
```

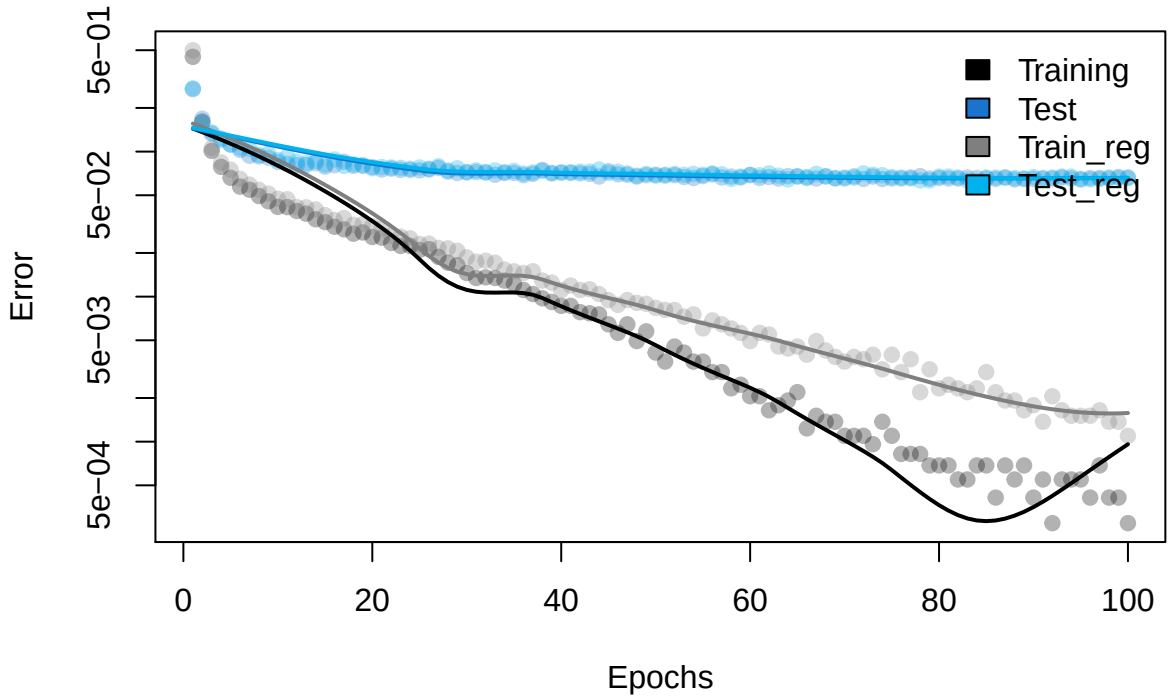
```
## [1] 84650
```

```
# fit the model on the training data
# and evaluate on test data at each epoch
fit_3hreg <- model_reg_3h %>% fit(
x = x_train, y = y_train,
```

```
validation_data = list(x_test, y_test),
epochs = 100,
verbose = 0
)
```

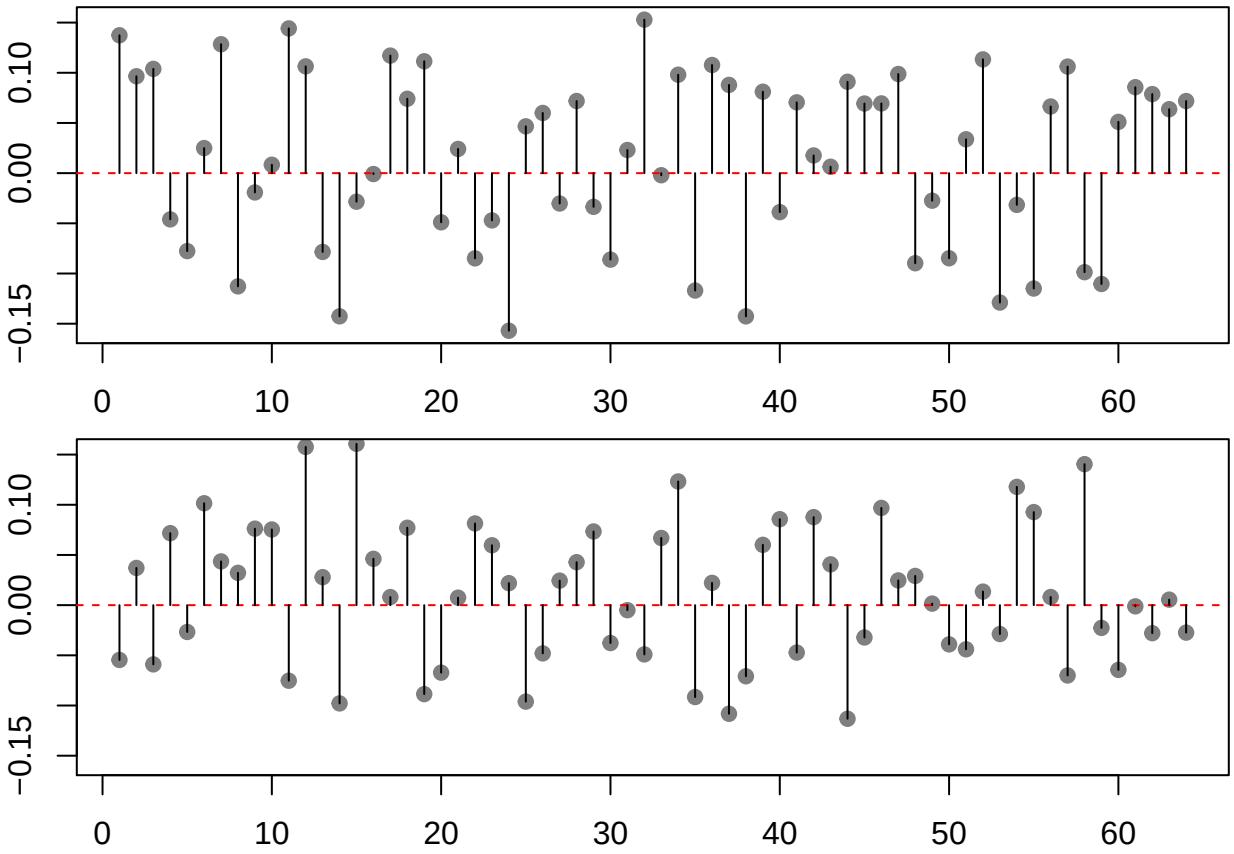
```
out <- 1 - cbind(fit_3h$metrics$accuracy,
fit_3h$metrics$val_accuracy,
fit_3hreg$metrics$accuracy,
fit_3hreg$metrics$val_accuracy)
# check performance
matplot(out, pch = 19, ylab = "Error", xlab = "Epochs",
col = adjustcolor(cols, 0.3),
log = "y")

matlines(apply(out, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("topright", legend = c("Training", "Test", "Train_reg", "Test_reg"),
fill = cols, bty = "n")
```




```
apply(out, 2, min)
```

```
## [1] 0.0002743006 0.0642750263 0.0010972619 0.0627802610
```


0.10
0.00
−0.15
0
10
20
30
40
50
60
0.10
0.00
−0.15
0
10
20
30
40
50
60