# Stats 503 Homework 5

Xingwen Wei, Zhao Wei, Tianshi Wang

4/1/2021

### QUestion 1

First, we can re-write the constraints from the bottom objective function.

$$y_i(\beta_0 + \beta^T x_i) \geq 1 - \zeta_i \tag{1}$$
$$y_i f(x_i) \geq 1 - \zeta_i \tag{2}$$
$$\zeta_i \geq 1 - y_i f(x_i) \tag{3}$$
$$\zeta_i \geq 0 \tag{4}$$
$$\zeta_i = [1 - y_i f(x_i)]_+ \tag{5}$$

Then, we can plug in this form of constraint of $\zeta_i$ back to objective function and apply a constant multiplication $\frac{1}{C}$ which does not affect the result. Finally, realizing the identity that $\lambda = \frac{1}{C}$, we establish the equivalence.

$$min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n} \zeta_i \tag{6}$$
$$min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n} [1 - y_i f(x_i)]_+ \tag{7}$$
$$min_{\beta_0,\beta} \frac{\lambda}{2}||\beta||^2 + \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ \tag{8}$$
$$\tag{9}$$

### Question 2

a) The test accuracy is 0.94 with the best model selected from 5 fold cross validation. According to the tuning summary, the model chosen has cost = 100 and radial kernel. We find that it is easy to confuse 3 and 5, and 7 and 9 based on the confusion table.

```
set.seed(2021)

# Data Preprocessing
load("C:/Users/xingw/Desktop/503/stats503/hw5/mnist.RData")
x_train <- x_train/255
x_test <- x_test/255

train_img <- matrix(x_train, dim(x_train)[1], prod(dim(x_train)[2:3]))
test_img <- matrix(x_test, dim(x_test)[1], prod(dim(x_test)[2:3]))

train_lab <- as.factor(y_train)
```

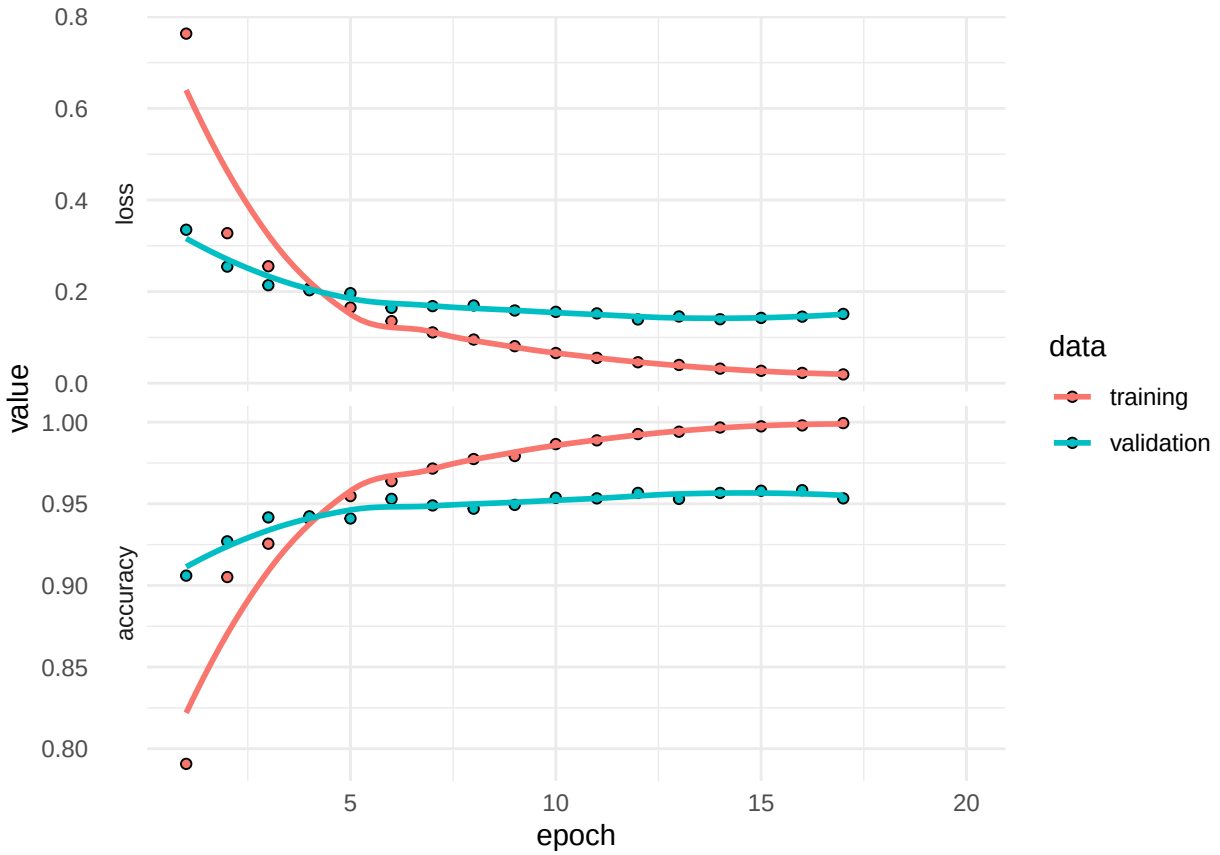
```
test_lab <- as.factor(y_test)

train_data <- data.frame(train_lab, train_img)
colnames(train_data)[1] = 'lab'
test_data <- data.frame(test_lab, test_img)
colnames(test_data)[1] = 'lab'
```

```
# SVM

library(e1071)
t =tune(svm, lab~., data=train_data, ranges=list(cost=c(0.1,1,10,100), kernel=c("polynomial", "radial"))
pred = predict(t$best.model, newdata=test_data)
summary(t)
table(pred, test_data$lab)
sum((pred==test_data$lab))/dim(test_data)[1]
```

b) We used cross validation to select the best model within a handful of candidates, including one layer with 128 neurons and two layers with 36 neurons each and etc. According to the cross validation result, we chose the model with one layer and 128 neurons. Then we use early stopping to prevent overfitting.

```
set.seed(123)
#install.packages('devtools')
#devtools::install_github("rstudio/keras")
#devtools::install_github("rstudio/tensorflow")
#library(tensorflow)
#install_tensorflow()
#library(keras)
#install_keras()
#reticulate::use_condaenv("anaconda3", required = TRUE)
library(keras)
library(tensorflow)

# MLP
digit_mlp <- keras_model_sequential()
digit_mlp %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation='relu') %>%
  layer_dense(units = 10, activation = 'softmax')

digit_mlp %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)

train_label = as.matrix(y_train)
train_label = as.integer(train_label)
test_label = as.matrix(y_test)
test_label = as.integer(test_label)
mlp_history = digit_mlp %>% fit(x_train, train_label, epochs = 20, validation_split = 0.3, batch_size =

library(ggplot2)
plot(mlp_history) + theme_minimal()
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
mlp_test <- digit_mlp %>% evaluate(x_test, test_label)
cat('MLP test accuracy: ', mlp_test[2])
```

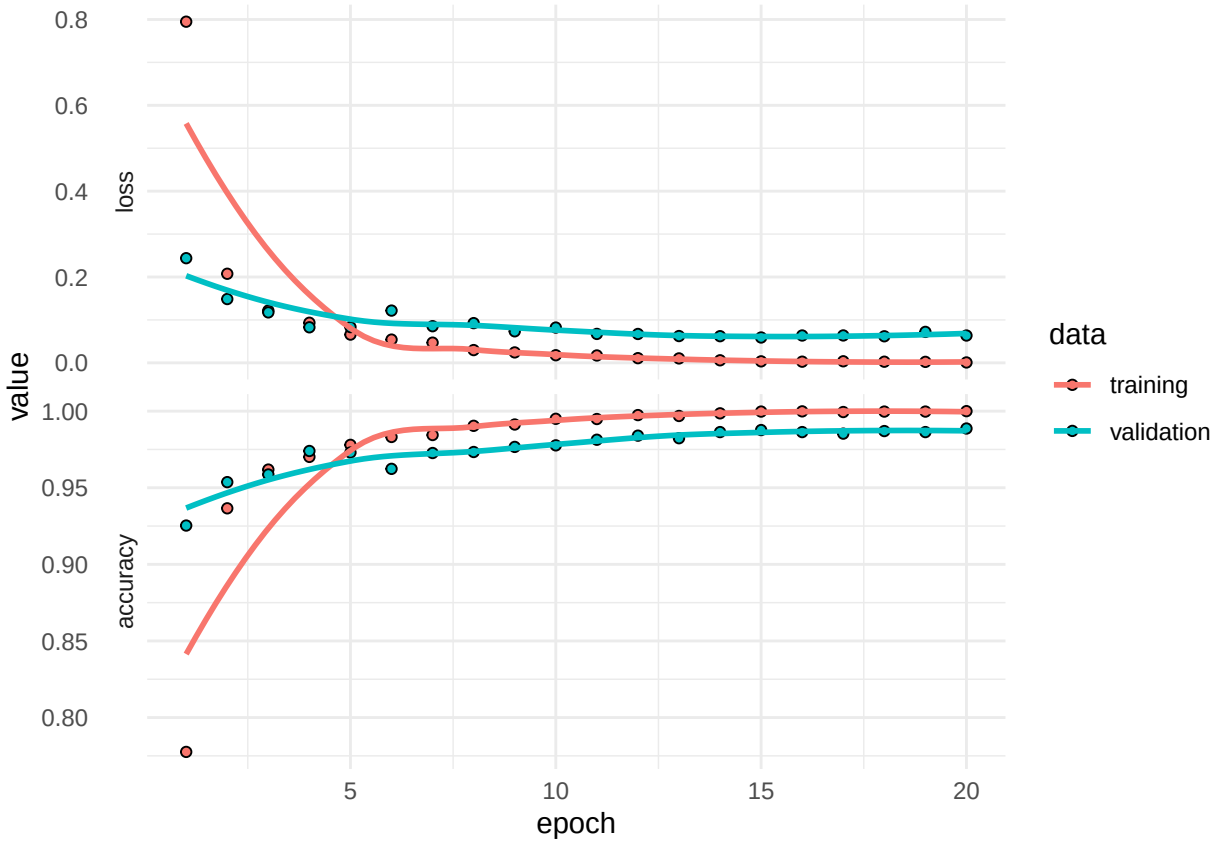
```
## MLP test accuracy:  0.9389
```

We tried 1, 2, and 3 convolutional and pooling layers pairs with 1 and 2 fully connected layers. The model with best validation error is the model with two convolutional and pooling layer pairs and 2 fully connected layers.

```
set.seed(520)

digit_cnn <- keras_model_sequential()
digit_cnn %>%
  layer_conv_2d(filter=32, kernel_size = c(3, 3), padding='same', input_shape=c(28,28,1)) %>%
  layer_activation('relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_conv_2d(filter=32, kernel_size = c(3, 3)) %>%
  layer_activation('relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
 # layer_dropout(0.25) %>%
  layer_flatten() %>%
  layer_dense(64) %>%
  layer_activation('relu') %>%
  layer_dense(10) %>%
  layer_activation('softmax')

cnn_train_x = array(x_train, dim = c(dim(x_train)[1], dim(x_train)[2], dim(x_train)[3], 1))
```

```
cnn_test_x = array(x_test, dim=c(dim(x_test)[1], dim(x_test)[2], dim(x_test)[3], 1))

digit_cnn %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)

cnn_history = digit_cnn %>% fit(cnn_train_x, train_label, epochs=20, validation_split=0.3, batch_size=64
                                callbacks = list(callback_early_stopping(monitor='val_loss', mode='min'
plot(cnn_history) + theme_minimal()
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
cnn_test <- digit_cnn %>% evaluate(cnn_test_x, test_label)
cat('CNN test accuracy: ', cnn_test[2])
```

```
## CNN test accuracy:  0.9729667
```

**Summary** Our testing error for SVM is 0.06 and our best testing error for MLP is 0.06 as well. The best testing error for CNN is 0.03. We believe the superior result from CNN is a result of utilizing the image spatial structure.