HW5

Wenjing Li

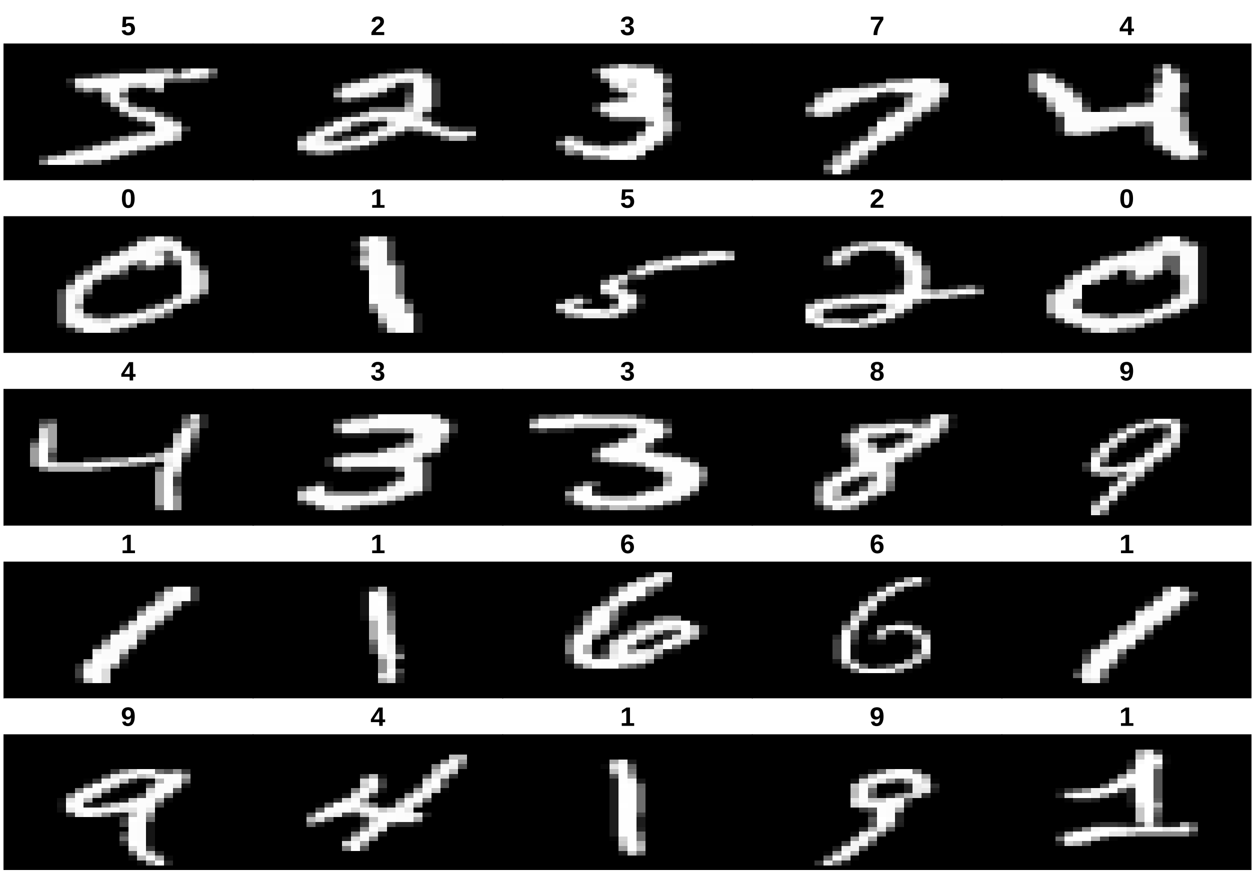
3/29/2020

Question 2

The dataset is from **MNIST database**. It contains handwritten digits. We will use the training dataset to train the model and get models using **SVM** and **Neural Network** methods respectively.

Get some basic description of the dataset.

```
par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  train <- x_train[i, , ]
  img <- t(apply(train, 2, rev))
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste(y_train[i]))
}
```



Preprocess the data.

```
x_train <- x_train / 255
x_test <- x_test / 255

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

train_labels = as.factor(y_train)
test_labels = as.factor(y_test)

train_dat = data.frame(train_labels[1:1000], x_train.1[1:1000,])
colnames(train_dat)[1] = "labels"

test_dat = data.frame(test_labels, x_test.1)
colnames(test_dat)[1] = "labels"
```

Train the SVM model and get the best model as: gamma = 0.1, cost = 0.1, degree = 1, kernel = polynomial.

```
library(e1071)
set.seed(123)
tc = tune.control(cross=5)
tune.out =tune(svm, labels~., data=train_dat,ranges=list(gamma = c(0.1,0.5),degree=c(1,2,3),cost=c(0.1,
#tune.out =tune(svm, labels~., data=train_dat,  kernel="radial")
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  gamma degree cost     kernel
##    0.1      2  0.1 polynomial
##
## - best performance: 0.086
##
## - Detailed performance results:
##    gamma degree cost     kernel error dispersion
## 1    0.1      1  0.1 polynomial 0.112 0.01151086
## 2    0.5      1  0.1 polynomial 0.106 0.01193734
## 3    0.1      2  0.1 polynomial 0.086 0.01294218
## 4    0.5      2  0.1 polynomial 0.088 0.01440486
## 5    0.1      3  0.1 polynomial 0.111 0.01635543
## 6    0.5      3  0.1 polynomial 0.111 0.01635543
## 7    0.1      1  1.0 polynomial 0.105 0.01837117
## 8    0.5      1  1.0 polynomial 0.108 0.02079663
## 9    0.1      2  1.0 polynomial 0.088 0.01440486
## 10   0.5      2  1.0 polynomial 0.088 0.01440486
## 11   0.1      3  1.0 polynomial 0.111 0.01635543
## 12   0.5      3  1.0 polynomial 0.111 0.01635543
## 13   0.1      1  0.1     radial 0.838 0.06467225
## 14   0.5      1  0.1     radial 0.901 0.01635543
```

```
## 15   0.1      2  0.1     radial 0.838 0.06467225
## 16   0.5      2  0.1     radial 0.901 0.01635543
## 17   0.1      3  0.1     radial 0.838 0.06467225
## 18   0.5      3  0.1     radial 0.901 0.01635543
## 19   0.1      1  1.0     radial 0.291 0.07948270
## 20   0.5      1  1.0     radial 0.833 0.03271085
## 21   0.1      2  1.0     radial 0.291 0.07948270
## 22   0.5      2  1.0     radial 0.833 0.03271085
## 23   0.1      3  1.0     radial 0.291 0.07948270
## 24   0.5      3  1.0     radial 0.833 0.03271085
```

```
test_pred = predict(tune.out$best.model, newdata = test_dat)
table(test_pred, test_dat$labels)
```

```
##
## test_pred    0    1    2    3    4    5    6    7    8    9
##         0  940    0   13    1    1    4    9    0   21    8
##         1    1 1104    4    8    3   12    5   21    5    6
##         2    4    2  943   23    3    4   14   18   11    4
##         3    0    1    5  824    0   18    0    9   19    6
##         4    2    1   15    0  887    7   11   12   14   35
##         5   18    2    5   77    0  802   24    1   35    8
##         6   10    4   12    3   14    9  894    0   11    0
##         7    3    2   22   15    2   10    1  937   12   28
##         8    1   19   11   46    1   17    0    1  815    3
##         9    1    0    2   13   71    9    0   29   31  911
```

```
(acc = 1-sum((test_pred != test_labels)) / dim(test_dat)[1])
```

```
## [1] 0.9057
```

So the accuracy for the best svm selected form the above informmation is:

2(b)

Build MLP (**Multi Layer Perception**)

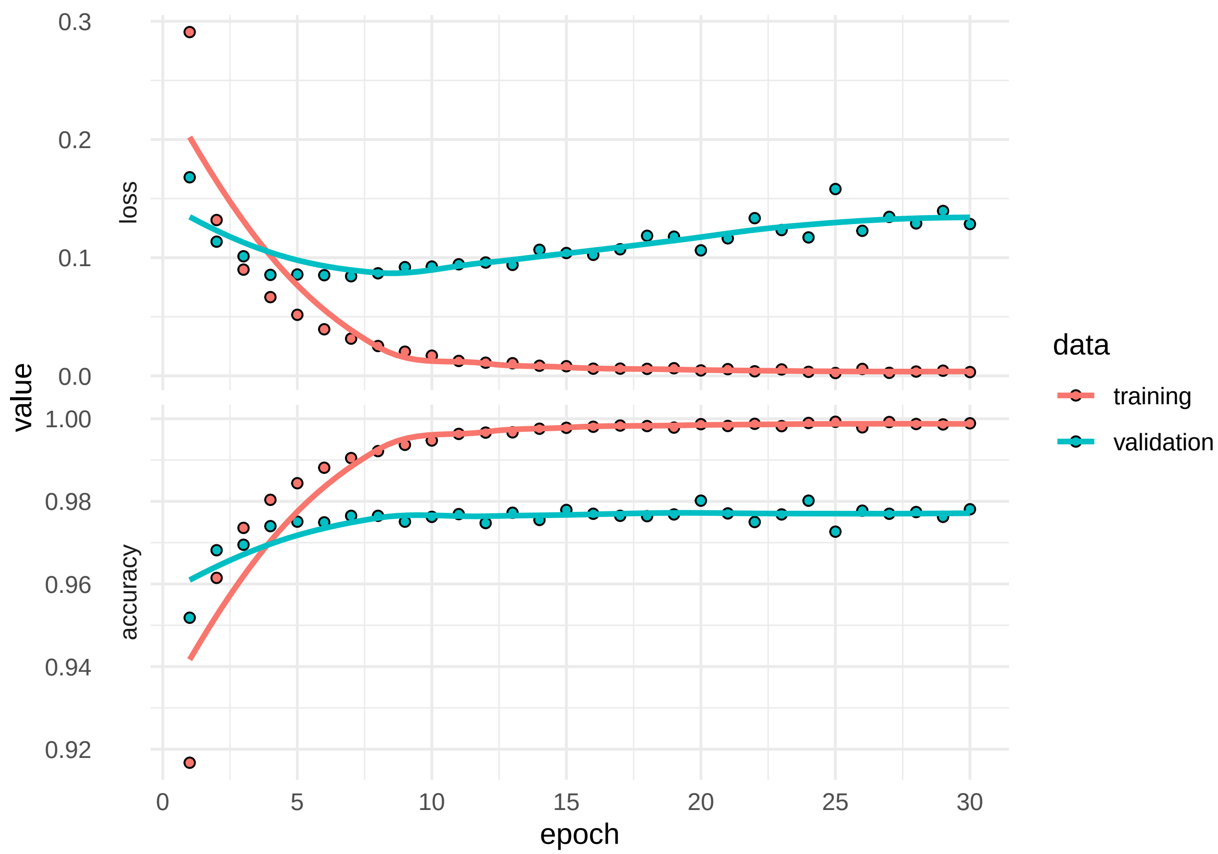
```
load("C:/Users/wenji/Downloads/STATS 503/HW5/mnist.Rdata")
x_train <- x_train / 255
x_test <- x_test / 255
model_fashion <- keras_model_sequential()
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:
## \Users\wenji\AppData\Local\Continuum\anaconda3\envs\rstudio/python.exe": The
## system cannot find the file specified
```

```
model_fashion %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
```

```r
model_fashion %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
MLP.history = model_fashion %>%
  fit(x_train, y_train, epochs = 30,
      validation_split = 0.2, batch_size = 32)
plot(MLP.history) + theme_minimal()
```

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



And we can get the test information as below.

```r
score.mlp <- model_fashion %>% evaluate(x_test, y_test)
cat('Test accuracy:', score.mlp$acc, "\n")
```

```
## Test accuracy: 0.9787
```

Build the structure of the **CNN** model:

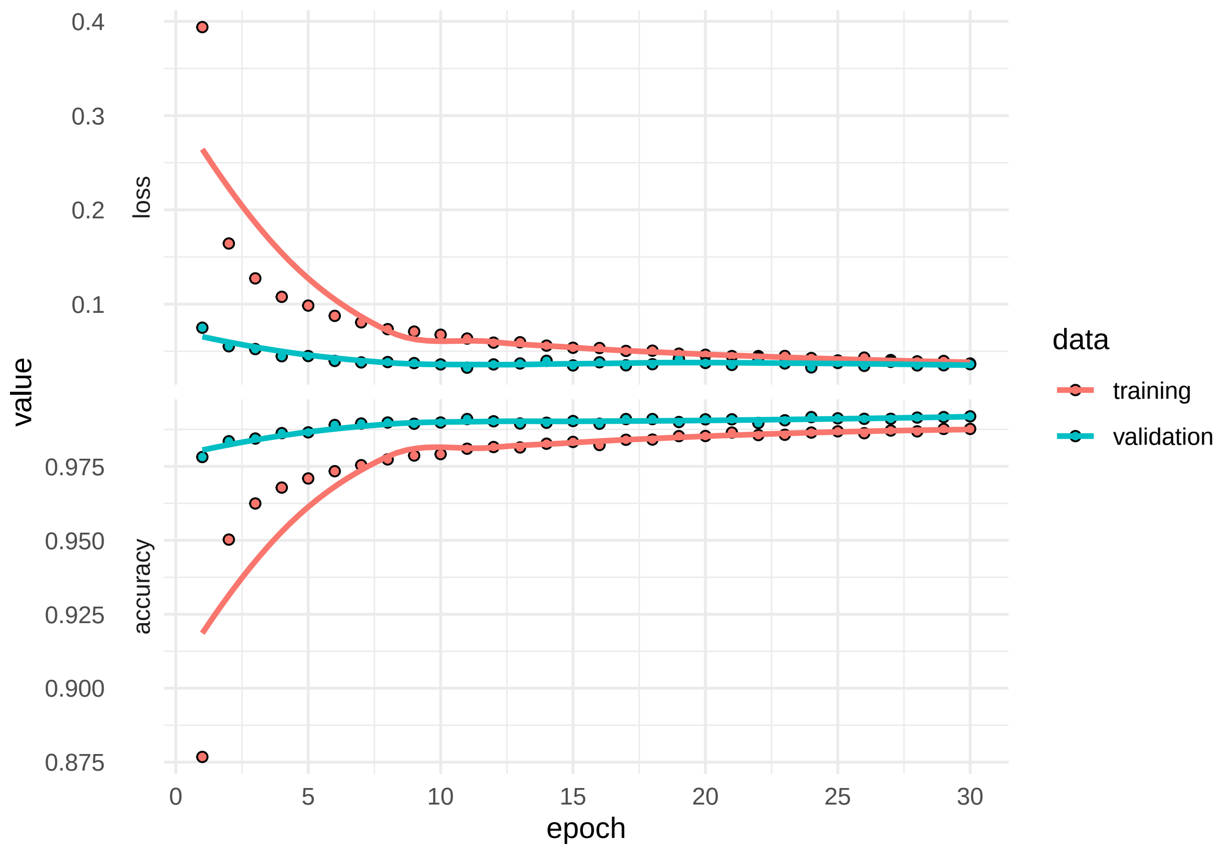
At first, create the structure of the CNN model.

```r
model_fashion.cnn <- keras_model_sequential()
#configuring the Model
model_fashion.cnn %>%
#defining a 2-D convolution layer
  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)) %>%
#another 2-D convolution layer
  layer_conv_2d(filter=32 ,kernel_size=c(3,3)) %>%
  layer_activation("relu") %>%
#Defining a Pooling layer which reduces the dimentions of the
#features map and reduces the computational complexity of the model
  layer_max_pooling_2d(pool_size=c(2,2)) %>%
#dropout layer to avoid overfitting
  layer_dropout(0.25) %>%
#flatten the input
  layer_flatten() %>%
  layer_dense(64) %>%
  layer_activation("relu") %>%
  layer_dropout(0.5) %>%
#output layer-10 classes-10 units
  layer_dense(10) %>%
#applying softmax nonlinear activation function to the output layer
#to calculate cross-entropy
  layer_activation("softmax")
```

Train the model and generate a history plot.

```r
train_images.cnn = array(x_train, dim = c(dim(x_train)[1],
                                          dim(x_train)[2], dim(x_train)[3], 1))
test_images.cnn = array(x_test, dim = c(dim(x_test)[1],
                                        dim(x_test)[2], dim(x_test)[3], 1))
model_fashion.cnn %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
CNN.history = model_fashion.cnn %>%
  fit(train_images.cnn, y_train, epochs = 30,
      validation_split = 0.2, batch_size = 32)
plot(CNN.history) + theme_minimal()
```

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



And then, we can get test accuracy as below.

```
score.cnn <- model_fashion.cnn %>% evaluate(test_images.cnn, y_test)
cat('Test accuracy:', score.cnn$acc, "\n")
```

```
## Test accuracy: 0.9922
```

To warp up, we can get a table.

```
table_error = c(acc,
                score.mlp$acc,
                score.cnn$acc)
dim(table_error) = c(1,3)
colnames(table_error) = c("SVM", "NN(MLP)", "NN(CNN)")
rownames(table_error) = c("Testing Accuracy")
cap = paste("*Training Accuracy for 3 Methods*")
knitr::kable(table_error, caption = cap)
```

Table 1: *Training Accuracy for 3 Methods*

	SVM	NN(MLP)	NN(CNN)
Testing Accuracy	0.9057	0.9787	0.9922