STAT 427 Notes

Daniel Polites

Setup

```
# Loads the MNIST dataset, saves as an .RData file if not in WD
if (!(file.exists("mnist_data.RData"))) {

  # ## installs older python version
  # reticulate::install_python("3.10:latest")
  # keras::install_keras(python_version = "3.10")
  # ## re-loads keras
  # library(keras)

  ## get MNIST data
  mnist <- dataset_mnist()
  ## save to WD as .RData
  save(mnist, file = "mnist_data.RData")

} else {
  ## read-in MNIST data
  load(file = "mnist_data.RData")
}

# Access the training and testing sets
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y

rm(mnist)
```

```
## plot function, from OG data
plot_mnist <- function(plt) {
  ## create image
  image(x = 1:28,
        y = 1:28,
        ## image is oriented incorrectly, this fixes it
        z = t(apply(plt, 2, rev)),
        ## 255:0 puts black on white canvas,
        ## changing to 0:255 puts white on black canvas
        col = gray((255:0)/255),
        axes = FALSE)

  ## create plot border
  rect(xleft = 0.5,
       ybottom = 0.5,
       xright = 28 + 0.5,
       ytop = 28 + 0.5,
       border = "black",
       lwd = 1)
}
```

```
## train data

# initialize matrix
x_train_2 <- matrix(nrow = nrow(x_train),
                    ncol = 28*28)

## likely a faster way to do this in the future
for (i in 1:nrow(x_train)) {
  ## get each layer's matrix image, stretch to 28^2 x 1
  x_train_2[i, ] <- matrix(x_train[i, , ], 1, 28*28)
}

x_train_2 <- x_train_2 %>%
  as.data.frame()

## test data
x_test_2 <- matrix(nrow = nrow(x_test),
                   ncol = 28*28)

for (i in 1:nrow(x_test)) {
  x_test_2[i, ] <- matrix(x_test[i, , ], 1, 28*28)
}

x_test_2 <- x_test_2 %>%
  as.data.frame()


## re-scale data
x_train_2 <- x_train_2 / 256
x_test_2 <- x_test_2 / 256

## response
# x_test_2$y <- y_test
# x_train_2$y <- y_train
```

Model

```
## for speed
n <- 2000

## init data
x_glm <- x_train_2[1:n, ]
y_glm <- y_train[1:n]
```

```r
train_pred <- list()

## drop cols with all 0s
x_glm <- x_glm[, (colSums(x_glm) > 0)]
```

```r
## 10 model method
for (i in 0:9) {
print(i)

y_glm_i = (y_glm == i)

init_model <- cv.glmnet(x = x_glm %>% as.matrix,
                        y = y_glm_i,
                        family = binomial,
                        alpha = 1)

train_pred[[i + 1]] <- predict(init_model,
                               x_glm %>% as.matrix,
                               s = init_model$lambda.min,
                               type = "response")
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```
## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge
## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge

## Warning: glmnet.fit: algorithm did not converge
```

```r
## format results
predictions <- data.frame(train_pred)
names(predictions) <- c("zero",
                        "one",
                        "two",
                        "three",
                        "four",
                        "five",
                        "six",
                        "seven",
                        "eight",
                        "nine")

#write.csv(predictions, "pred.csv", row.names = FALSE)

## convert to numeric
max_col <- apply(X = predictions,
                 MARGIN = 1,
                 FUN = function(x) names(x)[which.max(x)])

word_to_number <- c("zero" = 0,
                    "one" = 1,
                    "two" = 2,
                    "three" = 3,
                    "four" = 4,
                    "five" = 5,
                    "six" = 6,
                    "seven" = 7,
                    "eight" = 8,
                    "nine" = 9)

preds <- word_to_number[max_col] %>% as.numeric

## confusion matrix
table(y_glm, preds)
```

```
##      preds
## y_glm   0   1   2   3   4   5   6   7   8   9
##     0 191   0   0   0   0   0   0   0   0   0
##     1   0 217   0   0   0   2   0   0   1   0
##     2   0   2 188   0   1   1   0   2   3   1
##     3   0   0   2 179   0   5   0   1   1   3
##     4   0   1   0   0 208   0   1   0   1   3
##     5   1   1   1   1   1 170   3   0   1   1
##     6   0   0   0   0   0   2 198   0   0   0
##     7   0   1   1   1   1   0   0 218   0   2
##     8   0   5   1   2   1   2   0   0 160   1
##     9   3   0   0   4   6   1   1   4   0 191
```

```
## misclassification rate
mean(!(y_glm == preds))
```

```
## [1] 0.04
```