MSiA400_Lab2

Data Import

```
dat <- read_csv("gradAdmit.csv")
```

```
## Rows: 400 Columns: 4

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): admit, gre, gpa, rank

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Problem 1a

```
set.seed(123)
n <- nrow(dat)
train_i <- sample.int(n = n, size = floor(n * 0.8), replace = FALSE)
train <- dat[train_i, ]
test <- dat[-train_i, ]
```

Problem 1b

Kernel Selection and Degree, gamma, and coef0 search

```
set.seed(123)
folds <- createFolds(1:nrow(train), k=5)
kernels <- c("linear", "polynomial", "radial", "sigmoid")
train_acc_by_fold <- numeric(5)
val_acc_by_fold <- numeric(5)
avg_train_acc <- numeric(4)
avg_val_acc <- numeric(4)

degrees <- c(2,3,4)
gammas <- c(0.01, 0.1, 1)
coef0s <- c(0.01, 0.1, 1, 10)
costs <- c(0.01, 0.01, 1, 10, 100)
acc_df <- tibble(
```

```r
  degree = numeric(),
  gamma = numeric(),
  coef0 = numeric(),
  cost = numeric(),
  kernel = character(),
  avg_training_acc = numeric(),
  avg_validation_acc = numeric()
)


for (k in seq_len(length(degrees))){
  for (m in seq_len(length(gammas))){
    for (n in seq_len(length(coef0s))){
      for (p in seq_len(length(costs))){
        for(i in seq_len(4)){
          for (j in seq_len(5)){
            curr_val_set <- train[folds[[j]], ]
            curr_train_set <- train[-folds[[j]], ]
            curr_svm <- suppressWarnings(svm(factor(admit) ~ ., data = curr_train_set,
                              kernel = kernels[i],
                              degree = degrees[k],
                              gamma = gammas[m],
                              coef0 = coef0s[n],
                              cost = costs[p]))

            curr_train_pred <- predict(curr_svm, newdata = curr_train_set, type = 'response')
            curr_train_acc <- sum(curr_train_set$admit == curr_train_pred) / nrow(curr_train_set)

            curr_pred <- predict(curr_svm, newdata = curr_val_set, type = 'response')
            curr_val_acc <- sum(curr_val_set$admit == curr_pred) / nrow(curr_val_set)

            train_acc_by_fold[j] <- curr_train_acc
            val_acc_by_fold[j] <- curr_val_acc
          }
          acc_df <- acc_df %>% add_row(degree = degrees[k], gamma = gammas[m], coef0 = coef0s[n], cost =
        }
      }
    }
  }
}


acc_df
```

```
## # A tibble: 720 x 7
##    degree gamma coef0  cost kernel     avg_training_acc avg_validation_acc
##     <dbl> <dbl> <dbl> <dbl> <chr>                 <dbl>              <dbl>
##  1      2  0.01  0.01  0.01 linear                0.675              0.675
##  2      2  0.01  0.01  0.01 polynomial            0.675              0.675
##  3      2  0.01  0.01  0.01 radial                0.675              0.675
##  4      2  0.01  0.01  0.01 sigmoid               0.675              0.675
##  5      2  0.01  0.01  0.01 linear                0.675              0.675
##  6      2  0.01  0.01  0.01 polynomial            0.675              0.675
```

```
##  7      2  0.01  0.01  0.01 radial               0.675              0.675
##  8      2  0.01  0.01  0.01 sigmoid              0.675              0.675
##  9      2  0.01  0.01  1    linear               0.685              0.669
## 10      2  0.01  0.01  1    polynomial           0.675              0.675
## # ... with 710 more rows
```

```
newdf <- acc_df[order(-acc_df$avg_validation_acc, -acc_df$avg_training_acc),]
newdf
```

```
## # A tibble: 720 x 7
##    degree gamma coef0  cost kernel     avg_training_acc avg_validation_acc
##     <dbl> <dbl> <dbl> <dbl> <chr>                 <dbl>              <dbl>
##  1      4  0.01 10      100 polynomial            0.725              0.694
##  2      4  0.01 10       10 polynomial            0.710              0.694
##  3      3  0.1   0.01   100 polynomial            0.718              0.691
##  4      3  1     0.01     1 polynomial            0.718              0.691
##  5      2  0.1   0.01   100 polynomial            0.710              0.691
##  6      2  0.1   1       10 polynomial            0.710              0.691
##  7      2  1     0.01    10 polynomial            0.710              0.691
##  8      2  1     0.1      1 polynomial            0.710              0.691
##  9      3  0.01 10      100 polynomial            0.710              0.691
## 10      4  0.01 10        1 polynomial            0.710              0.691
## # ... with 710 more rows
```

As seen from the dataframe above, polynomial kernel with degree 4, gamma 0.01, coef0 = 10, cost = 10 is most optimal. Both training and validation accuracy is high, and the difference between them is small, which indicates less overfitting.

problem 1c

```
best_m <- svm(factor(admit) ~ ., data = train,
                            kernel = "polynomial",
                            degree = 4,
                            gamma = 0.01,
                            coef0 = 10,
                            cost = 10)
pred <- predict(best_m, newdata = test, type = 'response')
sum(test$admit == pred) / nrow(test)
```

```
## [1] 0.7375
```