STAT 588: Data Mining
Xin Yang
NetID: xy213
RUID: 182006660

Report of Midterm Part II

Question 1:

Code:

```
data=load(file = '~/Desktop/Midterm/geno.R', envir = parent.frame(), verbose = FALSE)
set.seed(182006660)
x0=t(x0)
index <-  sort(sample(nrow(x0), nrow(x0)*.8))
train_x0 <- x0[index,]
test_x0 <-  x0[-index,]
train_y <- y[index]
test_y <-  y[-index]
```

Question 2:

Code:

```
design_matrix=cbind(x0==0, x0==1, x0==2)*1
train_design_x0 <- design_matrix[index,]
test_design_x0 <- design_matrix[-index,]
```

Question 3:

1. SVM:

   Code:

   ```
   library(e1071)
   #svmfit <- svm(x=train_design_x0, y=as.factor(train_y), kernel='linear')
   svm_model <- svm(x=train_design_x0, y=as.factor(train_y))
   print(svm_model)
   summary(svm_model)

   svm_predict <- predict(svm_model, test_design_x0)
   table(svm_predict,as.factor(test_y))
   ```

   Results:

   ```
     0 1 2
   0 0 0 0
   1 5 6 4
   2 0 0 0
   ```

2. Random Forest:

   Code:

   ```
   library(randomForest)
   rf_model <- randomForest(x=train_design_x0, y=as.factor(train_y))
   ```

```
rf_predict <- predict(rf_model, test_design_x0)
table(rf_predict,as.factor(test_y))
```

Results:

```
  0 1 2
0 0 0 0
1 4 6 3
2 1 0 1
```

3. Naïve Bayes

Code:

```
nb_model <- naiveBayes(x=train_design_x0, y=as.factor(train_y))
nb_predict <- predict(nb_model, test_design_x0)
table(nb_predict, as.factor(test_y))
```

Results:

```
  0 1 2
0 5 6 4
1 0 0 0
2 0 0 0
```

Can you use Naïve Bayes here?

Naïve Bayes is not suitable here because the data dimension is too high. Fitting a Naïve Bayes model could be very slow.

Question 4:

1. GLM net:

Code:

```
library(glmnet)
glm_model <- glmnet(x=train_design_x0, y=as.factor(train_y), family='multinomial')
glm_predict <- predict(glm_model, test_design_x0, type="response")[,,1]
glm_labels <- colnames(glm_predict)[apply(glm_predict,1,which.max)]
glm_factors<-factor(c(glm_labels), levels = colnames(glm_predict))
table(glm_factors, as.factor(test_y))
```

Results:

```
  0 1 2
0 0 0 0
1 5 6 4
2 0 0 0
```

2. PENALIZED SVM using elastic net

According to the documents of penalizedSVM: https://cran.r-project.org/web/packages/penalizedSVM/penalizedSVM.pdf.

The penalizedSVM package only supports binary cases (-1 and 1). I tried to categorize two classes as one class, labeled as 1, and the rest one class as -1. Repeat 3 times to get the results. But the process is extremely slow. Here I switch to the sparseSVM packet, and by setting alpha=0.5, I could get the elastic net.

Code I tried for penalizedSVM:

```
library(penalizedSVM)
y_0_train = sign(train_y - 0.5)
y_0_test = sign(test_y - 0.5)
svmfs_model = svmfs(train_design_x0, as.numeric(y_0_train), fs.method =
c("DrHSVM"),verbose = FALSE)
svmfs_predict = predict.penSVM(svmfs_model, test_design_x0, as.factor(y_0_test))
print(pl_predict$tab)
```

Code for sparseSVM:

```
library(sparseSVM)
# 0 as class 1, 1,2 as class 0
y_0_train <- train_y
for(i in 1:length(train_y)){
  y_0_train[i]<-0
  if(train_y[i] == 0){
    y_0_train[i]<-1
  }
}
y_0_test <- test_y
for(i in 1:length(test_y)){
  y_0_test[i]<-0
  if(test_y[i] == 0){
    y_0_test[i]<-1
  }
}
pen_svm_model = sparseSVM(X=train_design_x0, y = y_0_train, alpha = 0.5)
pen_svm_pred = predict(pen_svm_model, test_design_x0)
table(pen_svm_pred[,1], y_0_test)

# 1 as class 1, 0,2 as class 0
y_1_train <- train_y
for(i in 1:length(train_y)){
  y_1_train[i]<-0
  if(train_y[i] == 1){
    y_1_train[i]<-1
  }
}
y_1_test <- test_y
for(i in 1:length(test_y)){
```

```
  y_1_test[i]<-0
  if(test_y[i] == 1){
    y_1_test[i]<-1
  }
}
pen_svm_model = sparseSVM(X=train_design_x0, y = y_1_train, alpha = 0.5)
pen_svm_pred = predict(pen_svm_model, test_design_x0)
table(pen_svm_pred[,1], y_1_test)


# 2 as class 1, 0,1 as class 0
y_2_train <- train_y
for(i in 1:length(train_y)){
  y_2_train[i]<-0
  if(train_y[i] == 2){
    y_2_train[i]<-1
  }
}
y_2_test <- test_y
for(i in 1:length(test_y)){
  y_2_test[i]<-0
  if(test_y[i] == 2){
    y_2_test[i]<-1
  }
}
pen_svm_model = sparseSVM(X=train_design_x0, y = y_2_train, alpha = 0.5)
pen_svm_pred = predict(pen_svm_model, test_design_x0)
summary(pen_svm_model)
table(pen_svm_pred[,1], y_2_test)
```

Results:

	0	1	2
0	5	0	0
1	0	6	0
2	0	0	4

Question 5:

For all the above classifiers, all hyper parameters remain default. The selection of hyper parameters, such as the kernel function in the SVM would largely affect the performance. Besides, the random selection of the dataset would also affect the classification results because the training samples are not equal for all three classes. Based on my results, the SVM shows 6/15 classification accuracy, the random forest shows 7/15 accuracy, the naïve bayes shows 5/15 accuracy. The GLM net shows 6/15 classification accuracy, and the Penalized SVM shows

15/15 accuracy. Among all the classifiers, the penalized SVM with elastic net gives the best classification performance.