

# Assignment 1

September 16, 2020

## Group 6:

- Chen Siyi A0194556R
- Gao Haochun A0194525Y
- He Yuan A0211297H
- Wang Pei A0194486M
- Wang Zi A0194504E

## 0.1 Run the R code

```
[1]: library(e1071)
rice <- read.delim("RiceDataset.txt", header = TRUE, stringsAsFactors = TRUE)

index <- 1:nrow(rice)
set.seed(123)
testindex <- sample(index, trunc(length(index)/3))
testset <- rice[testindex,]
trainset <- rice[-testindex,]
svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification",
                   kernel="linear")

results_train <- predict(svm.model, trainset[,-8])
results_test <- predict(svm.model, testset[,-8])
cf.train <- as.matrix(table(pred=results_train, actual=trainset$CLASS))
cf.test <- as.matrix(table(pred=results_test, actual=testset$CLASS))
```

## 0.2 Interpretation of R code

`library(e1071)` imports the package `e1071`.

`rice <- read.delim("RiceDataset.txt", header = TRUE, stringsAsFactors = TRUE)`  
reads data from source file.

`index <- 1:nrow(rice)` creates a vector of row index of examples in the data set.

`set.seed(123)` sets the random seed as 123.

`testindex <- sample(index, trunc(length(index)/3))` samples 1/3 of indexes randomly as test set indexes.

`testset <- rice[testindex,]` extracts the test set by index.  
`trainset <- rice[-testindex,]` extracts the rest of examples as training set.  
`svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification", kernel="linear")` creates a SVM model object with linear kernel, using CLASS attribute as label and other attributes as input.  
`results_train <- predict(svm.model, trainset[,-8])` makes prediction on training set using the SVM model.  
`results_test <- predict(svm.model, testset[,-8])` makes prediction on test set using the SVM model.  
`cf.train <- as.matrix(table(pred=results_train, actual=trainset$CLASS))` converts prediction results on training set to a confusion matrix.  
`cf.test <- as.matrix(table(pred=results_test, actual=testset$CLASS))` converts prediction results on test set to a confusion matrix.

### 0.3 Accuracy of linear SVM on training and test data sets

```
[2]: accuracy_train = (cf.train[1,1] + cf.train[2,2])/sum(cf.train)
accuracy_test = (cf.test[1,1] + cf.test[2,2])/sum(cf.test)
accuracy_train
accuracy_test
```

0.930314960629921

0.926771653543307

- Accuracy over training set: 93.03%
- Accuracy over test set: 92.68%

### 0.4 Using different kernels

#### 0.4.1 Linear kernel

```
[10]: iteration = 100
train_total = 0
test_total = 0
for(rand in 1:iteration) {
  set.seed(rand)
  testindex <- sample(index, trunc(length(index)/3))
  testset <- rice[testindex,]
  trainset <- rice[-testindex,]
  svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification", kernel="linear")

  results_train <- predict(svm.model, trainset[,-8])
  results_test <- predict(svm.model, testset[,-8])
```

```

cf.train <- as.matrix(table(pred=results_train,actual=trainset$CLASS))
cf.test <- as.matrix(table(pred=results_test,actual=testset$CLASS))

accuracy_train = (cf.train[1,1] + cf.train[2,2])/sum(cf.train)
accuracy_test = (cf.test[1,1] + cf.test[2,2])/sum(cf.test)
train_total = train_total + accuracy_train
test_total = test_total + accuracy_test
}

train_total/iteration
test_total/iteration

```

0.930948818897638

0.929653543307087

#### 0.4.2 Polynomial kernel

```

[5]: iteration = 100
train_total = 0
test_total = 0
for(rand in 1:iteration) {
  set.seed(rand)
  testindex <- sample(index, trunc(length(index)/3))
  testset <- rice[testindex,]
  trainset <- rice[-testindex,]
  svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification",
  ↴kernel="polynomial")

  results_train <- predict(svm.model,trainset[,-8])
  results_test <- predict(svm.model, testset[,-8])
  cf.train <- as.matrix(table(pred=results_train,actual=trainset$CLASS))
  cf.test <- as.matrix(table(pred=results_test,actual=testset$CLASS))

  accuracy_train = (cf.train[1,1] + cf.train[2,2])/sum(cf.train)
  accuracy_test = (cf.test[1,1] + cf.test[2,2])/sum(cf.test)
  train_total = train_total + accuracy_train
  test_total = test_total + accuracy_test
}

train_total/iteration
test_total/iteration

```

0.913169291338582

0.911417322834646

#### 0.4.3 Sigmoid kernel

```
[6]: iteration = 100
train_total = 0
test_total = 0
for(rand in 1:iteration) {
    set.seed(rand)
    testindex <- sample(index, trunc(length(index)/3))
    testset <- rice[testindex,]
    trainset <- rice[-testindex,]
    svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification",kernel="sigmoid")

    results_train <- predict(svm.model,trainset[,-8])
    results_test <- predict(svm.model, testset[,-8])
    cf.train <- as.matrix(table(pred=results_train,actual=trainset$CLASS))
    cf.test <- as.matrix(table(pred=results_test,actual=testset$CLASS))

    accuracy_train = (cf.train[1,1] + cf.train[2,2])/sum(cf.train)
    accuracy_test = (cf.test[1,1] + cf.test[2,2])/sum(cf.test)
    train_total = train_total + accuracy_train
    test_total = test_total + accuracy_test
}

train_total/iteration
test_total/iteration
```

0.866145669291339

0.865488188976379

#### 0.4.4 Radial basis kernel

```
[11]: iteration = 100
train_total = 0
test_total = 0
for(rand in 1:iteration) {
    set.seed(rand)
    testindex <- sample(index, trunc(length(index)/3))
    testset <- rice[testindex,]
    trainset <- rice[-testindex,]
    svm.model <- svm(CLASS ~ ., data=trainset, type="C-classification",kernel="radial")

    results_train <- predict(svm.model,trainset[,-8])
    results_test <- predict(svm.model, testset[,-8])
    cf.train <- as.matrix(table(pred=results_train,actual=trainset$CLASS))
    cf.test <- as.matrix(table(pred=results_test,actual=testset$CLASS))
```

```

accuracy_train = (cf.train[1,1] + cf.train[2,2])/sum(cf.train)
accuracy_test = (cf.test[1,1] + cf.test[2,2])/sum(cf.test)
train_total = train_total + accuracy_train
test_total = test_total + accuracy_test
}

train_total/iteration
test_total/iteration

```

0.930740157480315

0.92744094488189

#### 0.4.5 Summary

##### Accuracy of different kernels on test sets:

- Linear: 92.97%
- Polynomial: 91.14%
- Sigmoid: 86.55%
- Radial basis: 92.74%

##### Conclusion:

- Linear kernel achieves the best accuracy and we are not able to achieve better accuracy using other kernels.
- Because of the negligible difference between the accuracy of training and test sets, we can conclude that there is no overfitting problem with non-linear kernels.
- Though radial basis kernel almost achieves the same accuracy as linear kernel, it's better to use linear kernel because according to Occam's Razor, linear kernel, which is simpler, is better.

##### Explanations:

- Linear kernel

As the SVM model with linear kernel has the best accuracy among the four kernels, linear kernel is the best. This result indicates that our Rice Dataset is close to linearly separable because linear kernel usually works well on linearly separable dataset.

- Polynomial kernel

Polyonmial kernel maps features to higher orders to fit polynomial hypersurface. It drastically increases computation as more parameters are to be fitted to training data. Fitting a polynomial boundary may cause overfitting which decreases prediction accuracy, but since polynomial hypersurface can degenerate to nearly-linear hypersurface with linear separable training data, the accuracy did not decrease much.

- Sigmoid kernel

Sigmoid kernel is a complex non-linear kernel which uses hyperbolic tangent hence it's accuracy is not as high as linear kernel on dataset that is close to linear separable.

- Radius Basis kernel

The Gaussian kernel maps data to a higher dimensional space, making it more linear separable. If the parameter Sigma is overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. Therefore, in this case, the train accracy of RBF is very similar with that of linear.