Implementation of Support Vector Machine Classification using R package - Iris Dataset.

Expt No: 7 (a) May 2,2019
Author: Subalakshmi Shanthosi S (186001008)

Aim

Implementation of Support Vector Machine(SVM) using R package- Classification and Regression Training(CARET) for Iris dataset classification.

Description

1. Support Vector Machine:
    - Support Vector Machine is a Supervised Learning Model.
    - SVM can be applied for both classification and regression algorithms but predominantly used for classification problems.
    - Support Vector Algorithm Working:
        - Input : Data points from the dataset (iris).
        - Output : Hyperplane - The line which best separates the tags.
        - Careful choice of Kernal function which decides the accuracy of the model.
    - Advantages of using SVM for classification:
        - High Dimensionality.
        - Memory Efficiency.
        - Versatility.
    - Disadvantages of using SVM:
        - Kernel Parameters Selection : SVM shows poor performance on higher dimensional data.
        - Non-Probabilistic : Effectiveness is less evident as the algorithm places few data points above and below the decision boundry which might lead to misclassification if the between class varients among points is less.
2. Classification hyperplane based on the data point's distribution is presented below:

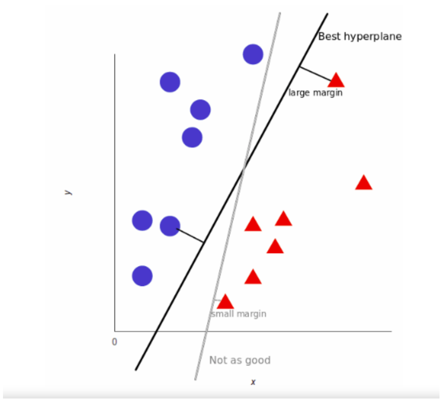


Figure 1: SVM Linear Model.

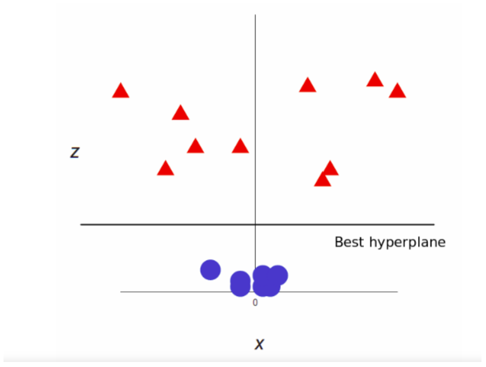


Figure 2: SVM Non Linear model by 3-D projection.

Tools and Packages

1. Tools
   - RStudio.
   - R Version 1.1.463
2. Support Vector Machine and Visualisation Packages
   - caret
   - ggplot2 (Visualisation)
   - GGally (Visualisation)

Dataset Description - Iris

- This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.
- Iris is a data frame with :
  - 150 cases (rows).
  - 5 features/variables(columns):
    - Sepal Length.
    - Sepal Width.
    - Petal Length.
    - Petal Width.
    - Species.
  - Iris *setosa* is linearly separable from *Iris versicolor and virginica*

Procedure

1. Split the data set as:
   - Training dataset.
   - Testing dataset.
2. Exploratory data Visualisation : To decide on the model to fit for a better precision in classification.
3. Feature Scaling and Model Fitting.
4. Calculate prediction and evaluate the SVM model/kernal accuracy.
5. Display the confusion matrix.

Support Vector Machine

- Support Vector Machine is a machine learning algorithm which:
  1. Solves classification problems.
  2. Uses flexible representation of decision boundary.
  3. Implements automatic complexity control to reduce overfitting.
  4. A single global minimum which can be found in polynomial time.
- Pseudocode :

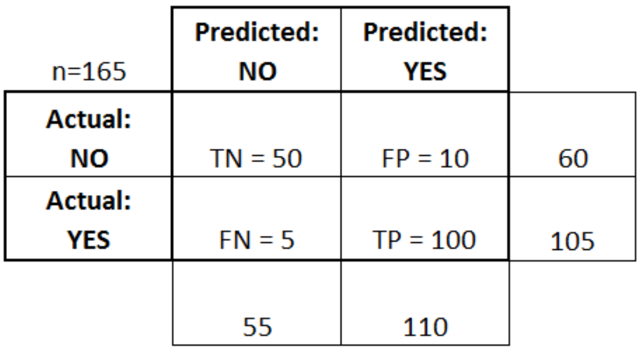
```
• Initialisation:
    * For the specified kernel, and kernel parameters, compute the kernel of distances between the datapoints.
    * The main work here is the computation K=XX$^{\mathrm{T}}$.
    * For the linear kernel, return K, for the polynomial of degree d return $\frac{1}{\sigma K^{\mathrm{d}}}$.
    * For the RBF kernel, compute K = $\exp(-\frac{(x-x^{'})^2}{2\sigma^2})$.
• Training
    * Assemble the constraint set as matrices to solve:
          $min_{\mathrm{x}} \frac{1}{2} x^{\mathrm{T}} t_{\mathrm{i}} t_{\mathrm{j}} K_{\mathrm{x}} + q^{\mathrm{T}} x.$          (1)
      subject to $G_{\mathrm{x}} <= h$
      $A_{\mathrm{x}} = b$
    * Pass these matrices to the solver.
    * Identify the support vectors as those that are within some specified distance of the closest point and
      dispose of the rest of the training data.
    * Calculate b$^*$ using equation:
          $b^* = \frac{1}{N_{\mathrm{s}}} \sum_{allsupportvectors} (t_{\mathrm{j}} - \sum_{i=1}^{n} \lambda_{\mathrm{i}} t_{\mathrm{i}} x_{\mathrm{i}}{}^{\mathrm{T}} x_{\mathrm{j}}).$          (2)
• Classification
    * For the given test data z, Use the support vector to classify the data for the relevant kernal by :
        · Compute the inner product of the test data and the support vectors.
        · Perform the classification as:
              $\sum_{i=1}^{n} \lambda_{\mathrm{i}} t_{\mathrm{i}} K(x_{\mathrm{i}}, z) + b^*.$          (3)
          returning -
          The label (Hard Classification)
          The value(Soft Classification)
```

Algorithm 1: The Support Vector Algorithm

Confusion Matrix

- A confusion matrix is a table that can be generated for a classifier on a Data Set

  True Positives(TP)- These are the cases where the predicted and actual both are yes.
  True Negatives(TN)- These are the cases where the predicted value is no and actual value is yes.
  False Positive(FP)- These are the cases where the predicted value is yes and actual value is no.
  False Negative(FN)- These are the cases where prediction is no and actual value is no.



n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- **Accuracy**: Overall, how often is the classifier correct? (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate**: Overall, how often is it wrong? (FP+FN)/total = (10+5)/165 = 0.09 equivalent to 1 minus Accuracy also known as ”Error Rate”
- **True Positive Rate**: When it’s actually yes, how often does it predict yes? TP/actual yes = 100/105 = 0.95 also known as ”Sensitivity” or ”Recall”
- **False Positive Rate**: When it’s actually no, how often does it predict yes? FP/actual no = 10/60 = 0.17
- **True Negative Rate**: When it’s actually no, how often does it predict no? TN/actual no = 50/60 = 0.83 equivalent to 1 minus False Positive Rate also known as ”Specificity”
- **Precision**: When it predicts yes, how often is it correct? TP/predicted yes = 100/110 = 0.91
- **Prevalence**: How often does the yes condition actually occur in our sample? actual yes/total = 105/165 = 0.64

Coding

```
# Use library e1071, you can install it using install.packages(  e1071  ). Load library

# Use library ggplot2 for visualisation
library(”e1071”)

library(ggplot2)

# Use library TeachingDemos to save output and commands

library(TeachingDemos)

txtStart(”svmOutput.txt”)
#Using Iris data

head(iris ,5)

#Attach the Data
```

```
attach(iris)

#Divide Iris data to x (containt the all features) and y only the classes

x <- subset(iris, select=-Species)
y <- Species

# Exploratory Data visualisation

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point() +
  labs(title = 'Petal Length vs Petal Width')
ggsave("svm.pdf")

#Create SVM Model and show summary

svm_model <- svm(Species ~ ., data=iris)
summary(svm_model)

#Or you can use command like this Create SVM Model and show summary

svm_model1 <- svm(x,y)
summary(svm_model1)

#Run Prediction and you can measuring the execution time in R

pred <- predict(svm_model1,x)
system.time(pred <- predict(svm_model1,x))

#See the confusion matrix result of prediction, using command table to compare the result

table(pred,y)

#Tuning SVM to find the best cost and gamma ..

svm_tune <- tune(svm, train.x=x, train.y=y, kernel="radial", ranges=list(cost=10^(-1:2), ga

print(svm_tune)

#After you find the best cost and gamma, you can create svm model again and try to run aga

svm_model_after_tune <- svm(Species ~ ., data=iris, kernel="radial", cost=1, gamma=0.5)
summary(svm_model_after_tune)

#Run Prediction again with new model

pred <- predict(svm_model_after_tune,x)
system.time(predict(svm_model_after_tune,x))

#See the confusion matrix result of prediction, using command table to compare the result

table(pred,y)

txtStop()
```

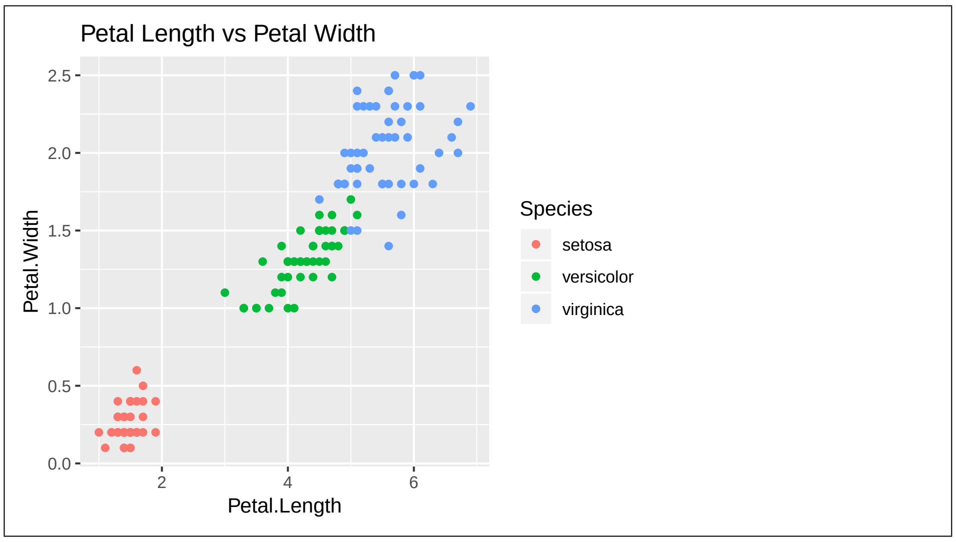
Output

```
> head(iris, 5)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
> attach(iris)
> x <- subset(iris, select = -Species)
> y <- Species
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
+ geom_point() + labs(title = "Petal Length vs Petal Width")
> ggsave("svm.pdf")
> svm_model <- svm(Species ~ ., data = iris)
> summary(svm_model)

Call:
svm(formula = Species ~ ., data = iris)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  51

 ( 8 22 21 )


Number of Classes:  3

Levels:
 setosa versicolor virginica



> svm_model1 <- svm(x, y)
> summary(svm_model1)

Call:
svm.default(x = x, y = y)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  51

 ( 8 22 21 )


Number of Classes:  3

Levels:
 setosa versicolor virginica
```

```
> pred <- predict(svm_model1, x)
> system.time(pred <- predict(svm_model1, x))
   user  system elapsed
  0.001   0.000   0.001
> table(pred, y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
> svm_tune <- tune(svm, train.x = x, train.y = y, kernel = "radial",
+ ranges = list(cost = 10^(-1:2), gamma = c(0.5, 1, 2)))
> print(svm_tune)

Parameter tuning of svm:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.04

> svm_model_after_tune <- svm(Species ~ ., data = iris, kernel = "radial",
+ cost = 1, gamma = 0.5)
> summary(svm_model_after_tune)

Call:
svm(formula = Species ~ ., data = iris, kernel = "radial",
    cost = 1, gamma = 0.5)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.5

Number of Support Vectors:  59

 ( 11 23 25 )


Number of Classes:  3

Levels:
 setosa versicolor virginica




> pred <- predict(svm_model_after_tune, x)
> system.time(predict(svm_model_after_tune, x))
   user  system elapsed
  0.001   0.001   0.002
```

```
> table(pred, y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
```



Result

Thus the implementation of Support Vector machine is executed successfully using R program.