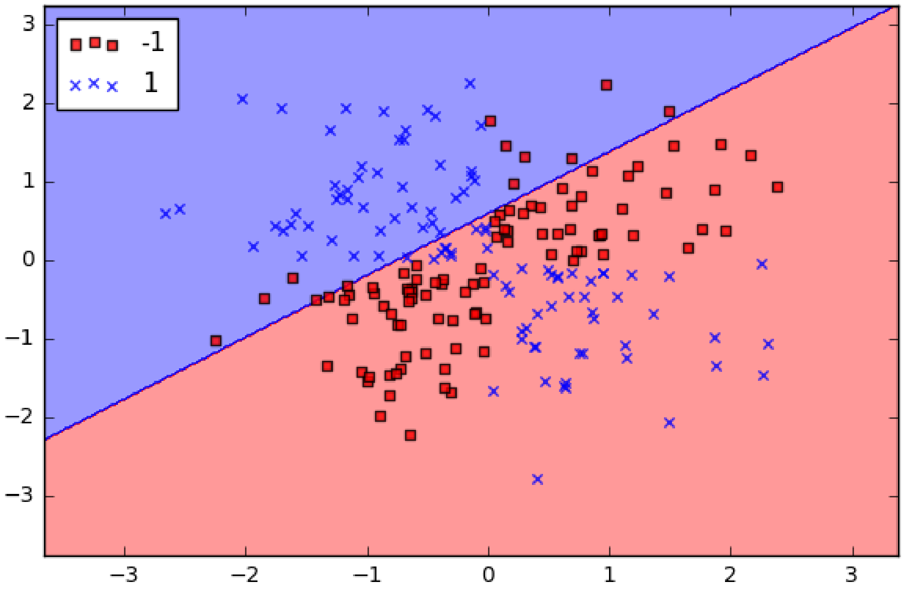
Support Vector Machine

By: Alexy Skoutnev

Mentor: Milad Eghtedari Naeini

Support Vector Machine Basics

- Supervised learning technique used for classification and regression analysis
  - Creates a decision boundary between data points
  - Separates data into distinct sets
  - Classifies new data points into a distinct set with high accuracy



Support Vector Machine Hyperplanes

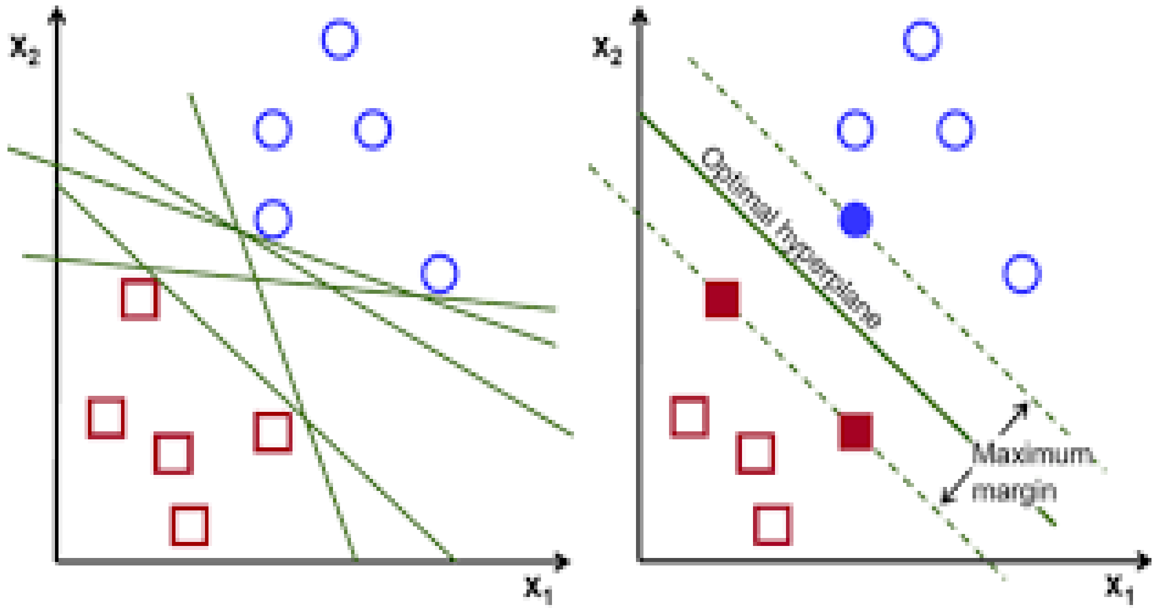
- A hyperplane is a subspace whose dimension is n-1 than the number of variables in the dataset
- In 3-dimensional datasets, 2-dimensional planes are used to separate data into two distinct groups
- There are many different hyperplanes that can classify data; however we want to find a hyperplane with the maximum margin

<u>p-dimensional hyperplane</u>

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

Maximum-margin Classifier

- Creates separating hyperplane that is farthest from the training observation
- There is a boundary between the sides of the hyperplane and the width is known as the "margin", which is maximized



Creating the Maximum Margin Hyperplane

- An optimization problem that has a constraint,

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{p1}) \geq M \ \forall i = 1, 2, \ldots$$

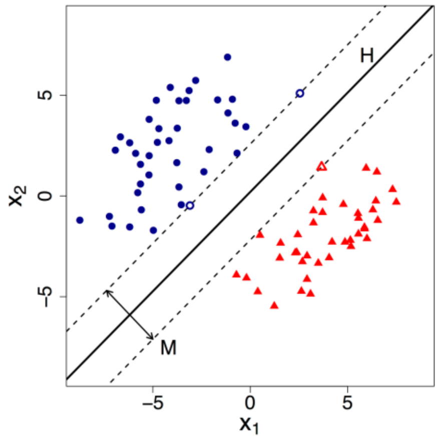
- Where we want to maximize M subject to,

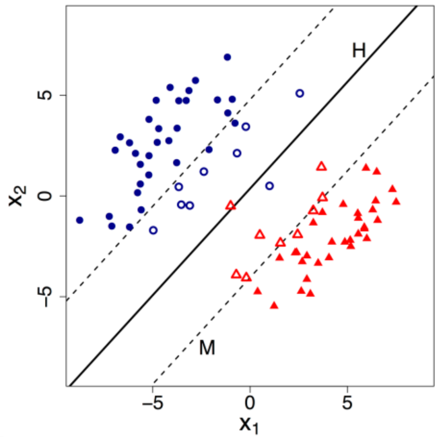
$$\sum_{j=1}^{p} \beta^2{}_j = 1$$

- The optimization of the Maximum Margin Hyperplane is handled by a software like R

Support Vector Classifiers

- Support Vector Classifier (soft margin classifier) is more robust and tends to be a better classifier than maximum margin classifier
- Sometimes a perfect separation of data is not possible, therefore some observations could be on the incorrect side of the margin
- We introduce a new slack variable ϵ_i that allows individual data points to be in the wrong side of the margin.





Support Vector Classifiers Optimization

- An optimization problem that has a constraint,

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{p1}) \geq M(1 - \varepsilon_i)\ \forall i = 1, 2, \ldots$$

- Subject to,

$$\sum_{j=1}^{p} \beta^2{}_j = 1$$

- And a turning parameter C,

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i < C$$

- C bounds the sums of ϵ_i and determines the number and severity of the violations in the margin region.

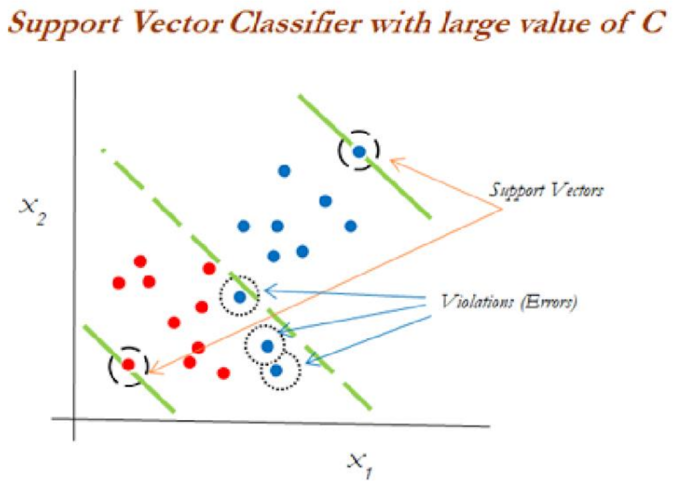
Support Vector Classifiers Optimization

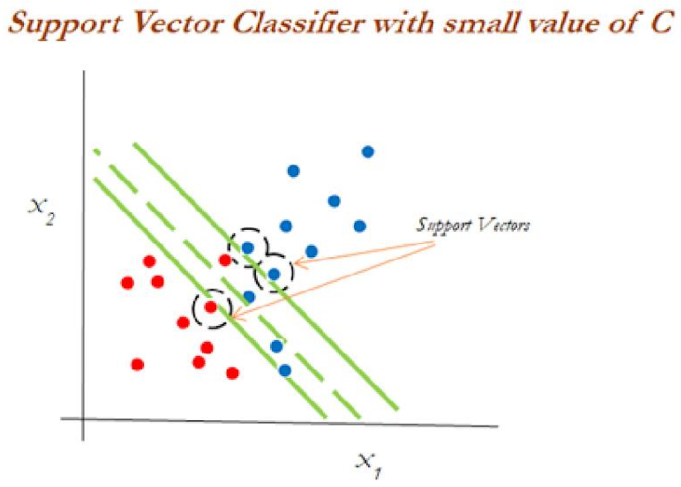
- C is treated as a tuning parameter that is usually chosen by cross-validation
- C balances the bias-variance trade-off seen in the dataset

Small Tuning Parameter	Large Tuning Parameter
- Narrow Margins - Few Violations - More Biased - Less Variance	- Wider Margins - More Violations - Less Biased - High Variance

Support Vectors

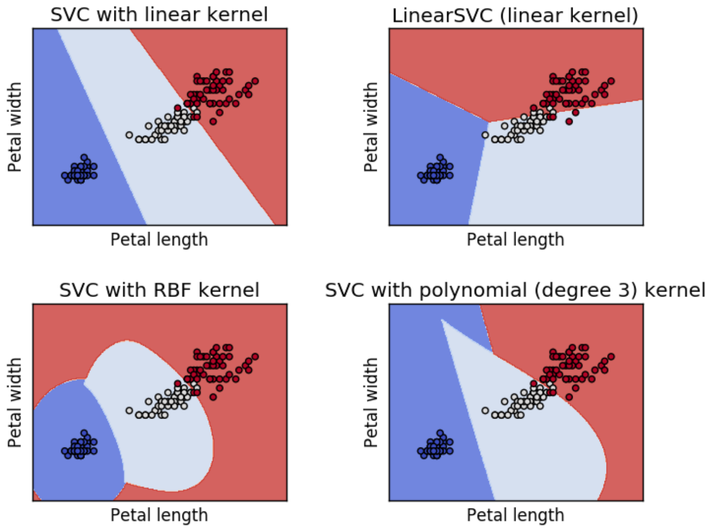
- We find that observations that lie on the margin, or those who violate the margin, are the only observations that affect the hyperplane
- Observations that lie on the correct side of the margin do not influence the hyperplane at all
- These observations on the wrong side of the margin are called “Support Vectors”





Support Vector Machine

- Support Vector Machine is an extension of support vector classifier
- Support Vector Machine expands the feature space by introducing kernels
- Kernels removes the computational requirements for higher dimension vector spaces and allows us to deal with non-linear data
- There are three common types of kernels: linear, polynomial, radial



Support Vector Machine Optimization

- The linear support classifier has a solution function of,

$$f(x) = \beta_0 + \sum_{j=1}^{n} \alpha_j \langle x, x_j \rangle$$

- Where α_j and β_0 are parameters measured by all the pairs of the inner products of the training data
- α_j is nonzero only for support vectors and α_j is equal to zero if not
- Thus, our solution function can be rewritten as,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j \langle x, x_j \rangle$$

- Where S is the collection of support vectors which results in fewer computations

Generalization of Kernels

- We can generalize our inner product by using a kernel seen as,

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- The kernel function can be transformed into a polynomial kernel of degree d,

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} (x_{ij} x_{i'j}))^d$$

- The solution function has the form,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j \, K(x_i, x_{i'})$$

Generalization of Kernels

- The kernel function can also be transformed into a radical kernel,

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

- Where the solution function has the form,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j K(x_i, x_{i'})$$

Using Support Vector Machine in R

- I used a dataset named spam7 that contains data whether an email is spam or not
- The 6 predictors of the dataset were
  - crl.tot: total length of words in capitals
  - dollar: number of occurrences of the \$ symbol
  - bang: number of occurrences of the ! symbol
  - money: number of occurrences of the word ‘money’
  - n000: number of occurrences of the string ‘000’
  - make: number of occurrences of the word ‘make’
- The response variable “yesno” was the indication whether it is spam or not
  - yesno: outcome variable, a factor with levels n not spam, y spam

Sorting Train and Test data

```
#packages
library(DAAG) #Spam7 Data Set
library(e1071)

#Error Function - confusion matrix
compute_error = function(a,b)
{
  tmp = table(a,b)
  out = ( sum(tmp) - sum(diag(tmp)) ) / sum(tmp)
  out
}

#Test if Factor
class(train)
?spam7
set.seed(667)
ind = sample(1:5, size = nrow(spam7), replace = TRUE)
#test dataset is 1/5 size of total dataset
#train data set is 4/5 size of total dataset

test_index =  which(ind == 1)
train      =  spam7[ -test_index, ]
test       =  spam7[  test_index, ]

dim(train)
dim(test)
train
test
```

First Prediction of Support Vector Machine Code

```
#SVM Model
svm.spam = svm(yesno ~ . , data = train, kernel = 'linear', gamma = 1, cost = 1e5)
summary(svm.spam)
svm.spam$index
ypredict_1 = predict(svm.spam, test)
table(predict = ypredict_1, truth = test$yesno)
#Error of prediction
svm_svm_error = compute_error(test$y, ypredict_1)
svm_svm_error
```

Result

- 1399 Support Vectors
- Linear Kernel
- Test error: 18.65%

```
Console   Terminal ×   Jobs ×
~/

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1e+05

Number of Support Vectors:  1399

 ( 653 746 )


Number of Classes:  2

Levels:
 n y
```

```
> ypredict_1 = predict(svm.spam, test)
> table(predict = ypredict_1, truth = test$yesno)
       truth
predict   n   y
      n 542 153
      y  19 208
> #Error of prediction
> svm_bestmodel_error = compute_error(test$y, ypredict_1)
> svm_bestmodel_error
[1] 0.186551
```

Tuned Support Vector Machine Code

```
#Tuned model through cross validation
tune.out=tune(svm ,yesno ~ . ,data=train ,kernel = c('linear', 'polynomail'),
              ranges=list(cost=c(.0001, 0.001, 0.01, 0.1, 1,5,10,100), gamma = c(0.001, 0.01, .1, 1) ))
tune.out
summary(tune.out)
bestmod = tune.out$best.model
summary(bestmod)
#Prediction through test data set
ypredict = predict(bestmod, test)
table(predict = ypredict, truth = test$yesno)
#Error of prediction
svm_bestmodel_error = compute_error(test$y, ypredict )
svm_bestmodel_error
```

Result

- 1531 Support Vectors
- Polynomial Kernel
- Test error: 14.52%

```
Console   Terminal ×   Jobs ×
~/
> tune.out=tune(svm ,yesno ~ . ,data=train ,kernel = c('linear', 'polynomail'), ranges=list(cost=c(.0001, 0.001, 0.01, 0.1, 1,5,10,100), gamma = c
(0.001, 0.01, .1, 1) ))
There were 50 or more warnings (use warnings() to see the first 50)
> tune.out

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100 0.001

- best performance: 0.163624

> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100 0.001

- best performance: 0.163624
```

Result

```
> bestmod = tune.out$best.model
> summary(bestmod)

Call:
best.tune(method = svm, train.x = yesno ~ ., data = train, ranges = list(cost = c(1e-04, 0.001, 0.01, 0.1, 1, 5, 10, 
    100), gamma = c(0.001, 0.01, 0.1, 1)), kernel = c("linear", "polynomail"))


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  linear NA 
       cost:  100 

Number of Support Vectors:  1531

 ( 766 765 )


Number of Classes:  2 

Levels: 
 n y



Warning message:
In if (x$kernel == 1) cat("     degree: ", x$degree, "\n") :
  the condition has length > 1 and only the first element will be used
> #Prediction through test data set
> ypredict = predict(bestmod, test)
> table(predict = ypredict, truth = test$yesno)
       truth
predict   n   y
      n 541 114
      y  20 247
> #Error of prediction
> svm_bestmodel_error = compute_error(test$y, ypredict )
> svm_bestmodel_error
[1] 0.1453362
```