# R implementation of the Support Vector Mahine Recursive Feature Extraction (SVM-RFE) Algorithm

## Introduction

Two functions were wrote in R to return a ranked list of the features, one for binary classification problems and another for multiclass classification problems. These functions implements the SVM-RFE algorithm.

## The SVM-RFE algorithm

The SVM-RFE algorithm proposed by Guyon returns a ranking of the features of a classification problem by training a SVM with a linear kernel and removing the feature with smallest ranking criterion. This criterion is the w value of the desicion hiperplane given by the SVM.
For more detailed information review the original paper.
*Gene Selection for Cancer Classification using Support Vector Machines (2002)*
*Isabelle Guyon, Jason Weston, Stephen Barnhill, Vladimir Vapnik*
*http://citeseer.ist.psu.edu/guyon02gene.html*

## R implementation of the SVM-RFE algorithm for binary classification problems

The next function implements the SVM-RFE algorithm as it is described in by Guyon. The R implementation of the LIBSVM library, *library(e1071),* was used to for this implementation.

```
svmrfeFeatureRanking = function(x,y){
    n = ncol(x)

    survivingFeaturesIndexes = seq(1:n)
    featureRankedList = vector(length=n)
    rankedFeatureIndex = n

    while(length(survivingFeaturesIndexes)>0){
        #train the support vector machine
        svmModel = svm(x[, survivingFeaturesIndexes], y, cost = 10, cachesize=500,
scale=F, type="C-classification", kernel="linear" )

        #compute the weight vector
        w = t(svmModel$coefs)%*%svmModel$SV

        #compute ranking criteria
        rankingCriteria = w * w

        #rank the features
        ranking = sort(rankingCriteria, index.return = TRUE)$ix

        #update feature ranked list
        featureRankedList[rankedFeatureIndex] = survivingFeaturesIndexes[ranking[1]]
        rankedFeatureIndex = rankedFeatureIndex - 1

        #eliminate the feature with smallest ranking criterion
        (survivingFeaturesIndexes = survivingFeaturesIndexes[-ranking[1]])

    }

    return (featureRankedList)
}
```

The function has two inputs:
x : a matrix where each column represents a feature and each row represents a sample
y : a vector of the labels corresponding to each sample

The function returns a vector of the features in x ordered by relevance. The first item of the vector has the index of the feature which is more relevant to perform the classification and the last item of the vector has the feature which is less relevant.

Example:
The following code shows how to use this funtion to train a SVM with the 20 most relevant features:

```
featureRankedList = svmrfeFeatureRanking(x,y)
svmModel = svm(x[, featureRankedList[1:20]], y, cost = 10, kernel="linear" )
```

**R implementation of the SVM-RFE algorithm for multiclass classification problems**

For the multiclass classification problem the same SVM-RFE algorithm is implemented with minor tweaks. The multiclass classification problem is reduced to several binary classification problems and the ranking criterion is now the average value of the w values of the hiperplanes defined by each SVM of these different binary classification problems.

```
svmrfeFeatureRankingForMulticlass = function(x,y){
    n = ncol(x)

    survivingFeaturesIndexes = seq(1:n)
    featureRankedList = vector(length=n)
    rankedFeatureIndex = n

    while(length(survivingFeaturesIndexes)>0){
        #train the support vector machine
        svmModel = svm(x[, survivingFeaturesIndexes], y, cost = 10, cachesize=500,
scale=F, type="C-classification", kernel="linear" )

        #compute the weight vector
        multiclassWeights = svm.weights(svmModel)

        #compute ranking criteria
        multiclassWeights = multiclassWeights * multiclassWeights
        rankingCriteria = 0
        for(i in 1:ncol(multiclassWeights))rankingCriteria[i] =
mean(multiclassWeights[,i])

        #rank the features
        (ranking = sort(rankingCriteria, index.return = TRUE)$ix)

        #update feature ranked list
        (featureRankedList[rankedFeatureIndex] = survivingFeaturesIndexes[ranking[1]])
        rankedFeatureIndex = rankedFeatureIndex - 1

        #eliminate the feature with smallest ranking criterion
        (survivingFeaturesIndexes = survivingFeaturesIndexes[-ranking[1]])
        cat(length(survivingFeaturesIndexes),"\n")
    }

}

svm.weights<-function(model){
w=0
  if(model$nclasses==2){
       w=t(model$coefs)%*%model$SV
  }else{    #when we deal with OVO svm classification
      ## compute start-index
```

```
      start <- c(1, cumsum(model$nSV)+1)
      start <- start[-length(start)]

      calcw <- function (i,j) {
        ## ranges for class i and j:
        ri <- start[i] : (start[i] + model$nSV[i] - 1)
        rj <- start[j] : (start[j] + model$nSV[j] - 1)

      ## coefs for (i,j):
        coef1 <- model$coefs[ri, j-1]
        coef2 <- model$coefs[rj, i]
        ## return w values:
        w=t(coef1)%*%model$SV[ri,]+t(coef2)%*%model$SV[rj,]
        return(w)
      }

      W=NULL
      for (i in 1 : (model$nclasses - 1)){
        for (j in (i + 1) : model$nclasses){
          wi=calcw(i,j)
          W=rbind(W,wi)
        }
      }
      w=W
  }
  return(w)
}
```

The inputs and output of this function are the same as the ones described for the SVM-RFE for binary classification.

**Reproducing Guyon results**

All this code can be found at
http://www.uccor.edu.ar/paginas/seminarios/Software/SVM-RFE.zip
as well as a reproduction of the Guyon tests on the leukemia and colon cancer databases.