Machine Learning and Adversaries

Dan Bruce, David Clark and Hector D. Menendez

Department of Computer Science
University College London

December 4, 2017

Machine Learning

Statistical process which learns from a specific discrimination related to a set of objects. There are several fields inside Machine Learning, the most relevant are:

Clustering: The ability of dividing objects into groups blindly, based on similarities.

Classification: A supervised identification of patterns into objects to separate them.

Learning Example

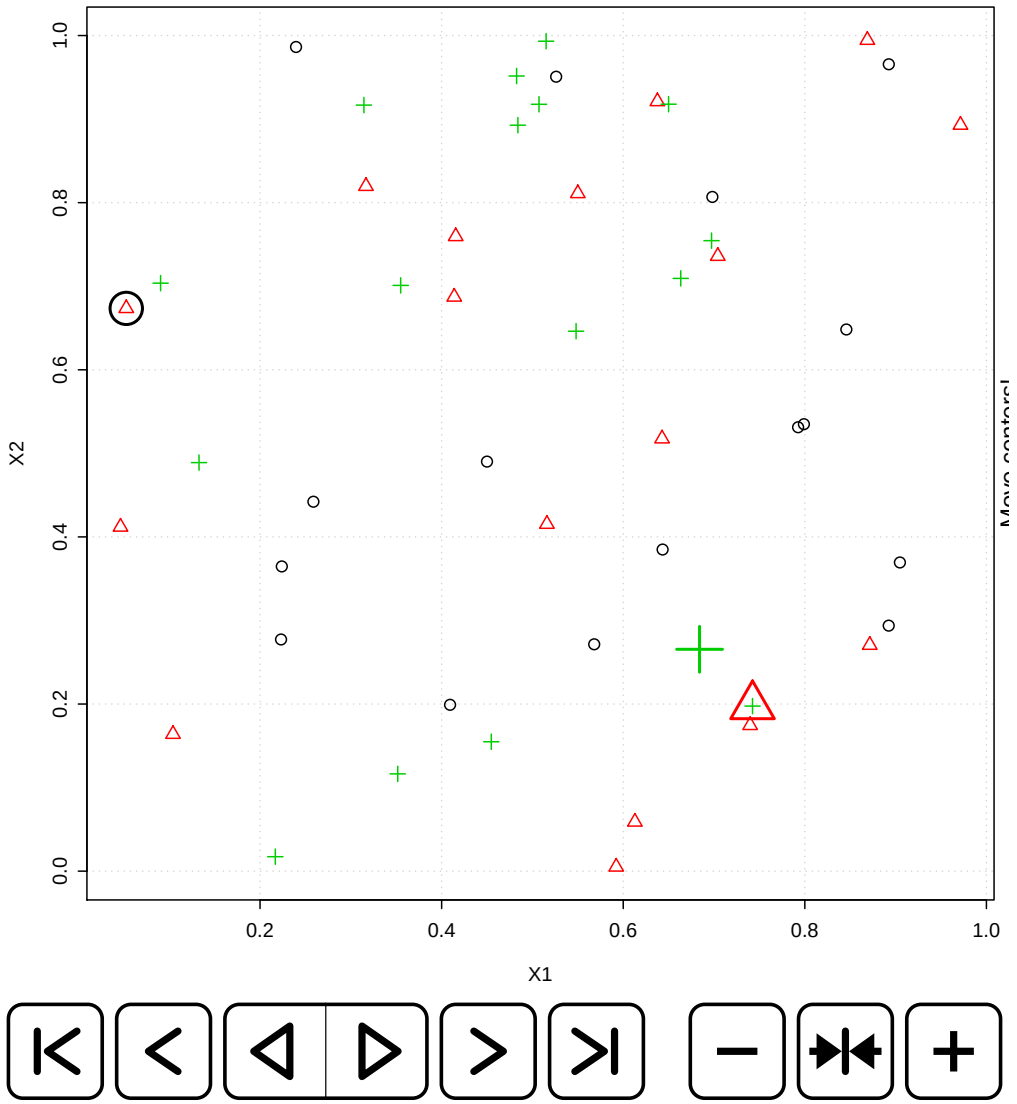
Imagine you have a set of objects and you want to group them when they are similar.

First, you extract information about the objects or **features** (preferably numerical).

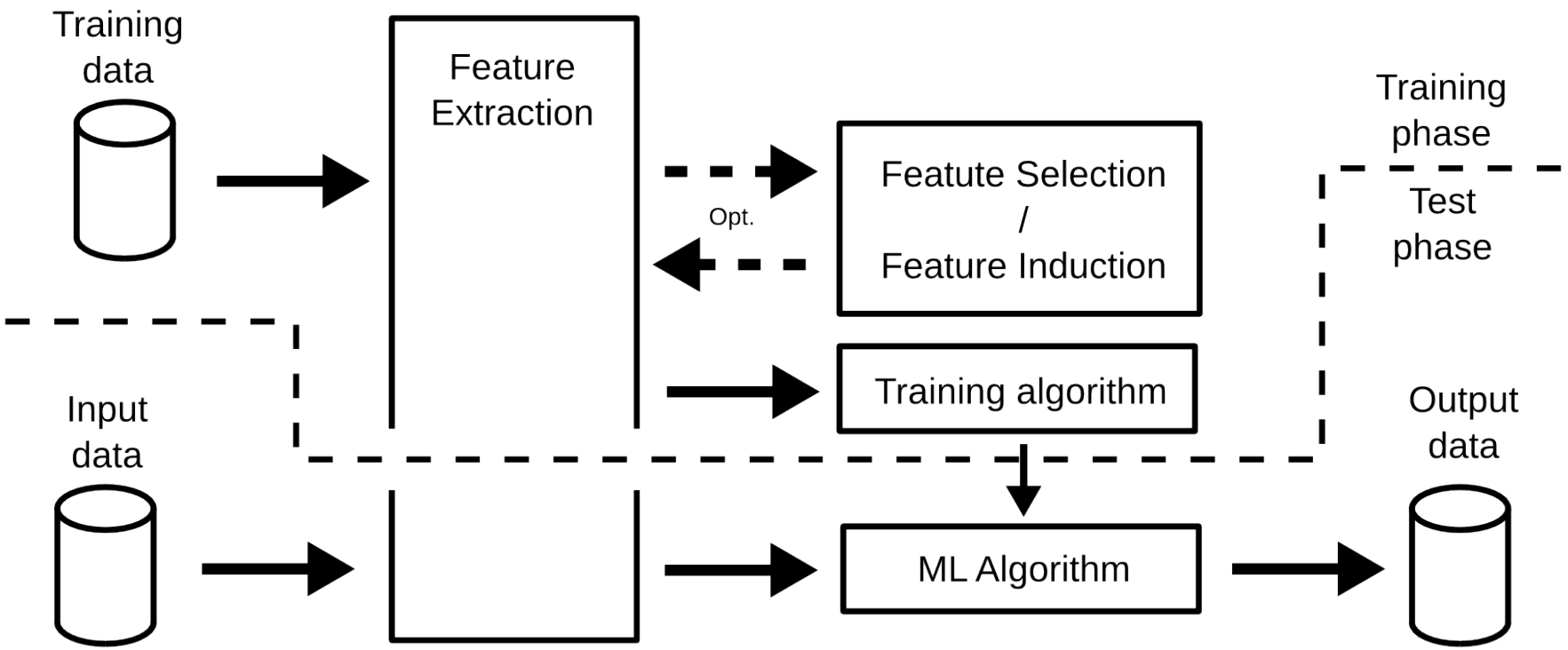
Second, you define your notion of **similarity**.

Third, you set your separability criteria and learning process, i.e., your **algorithm**, and run it.

K-means



General Structure



Malware/Benign-ware classification

Researchers normally aim to create a methodology to distinguish malware and bening-ware.

Several works apply **classification** algorithms for this aim.

These algorithms learns from program features and aim to identify patterns on them.

Program features

Static analysis: Information from the disassemble version of the program, from the control flow graph, etc.

Dynamic analysis: information from the traces, network messages, etc.

Binary-based analysis: information from the entropy or n-gram distribution of files.

Similarity/distance measures

The discrimination normally depends on the similarity measures, if the features are discriminative, these similarities can be general. Some examples are:

- Euclidean distance.
- Cosine distance (text).
- Tree distances.
- Levenshtein distance.
- Kernels.

Cleaning Missing Values

Missing values produce abnormal behaviours int he classifiers, therefore it is better to approximate them or remove them:

```
1 library(mlbench)
2 data(BreastCancer)
3 dataset <- BreastCancer[complete.cases(BreastCancer),]
```

Correlations

Some variables are correlated with another variables.

Correlations produce redundancy in the information and affects the models.

They can be easily removed:

```
data(BreastCancer)
dataset <- BreastCancer[complete.cases(BreastCancer),]
dataset<-as.matrix(sapply(dataset[-1],as.numeric))
correlations <- cor(dataset[,-10])
hcorrelations <- findCorrelation(correlations, cutoff=0.85)
dataset <- dataset[,-hcorrelations]
```

Learning Algorithms

Once the data is set and divided in training and test, we apply the classification algorithm. Some well-known algorithms are:

- Naïve Bayes.
- Decision Tress.
- Random Forest.
- Support Vector Machines.
- Neural Networks.

Our First classifier: SVM

```
1 library(e1071)
2 trainIndex<-sample(1:length(iris[,1]),100)
3 trainData<-iris[trainIndex,]
4 testData<-iris[-trainIndex,]
5 svm_model <- svm(Species ~ ., data=trainData,probability=TRUE)
6 pred <- predict(svm_model,testData[,-5])
7 probs_svm <- predict(svm_model,type="prob", testData[,-5],
      probability=TRUE)
8 table(pred,testData[,5])
```

Output:

```
1 pred          setosa versicolor virginica
2   setosa          14          0         0
3   versicolor       0         16         2
4   virginica        0          2        16
```

Our First classifier

This example uses the **Iris dataset**, one of the most famous datasets in classification composed of 150 data instances and 3 classes: *setosa*, *versicolor* and *virginica*.

We divide the dataset into up to 100 instances of **training** and the rest for **testing**.

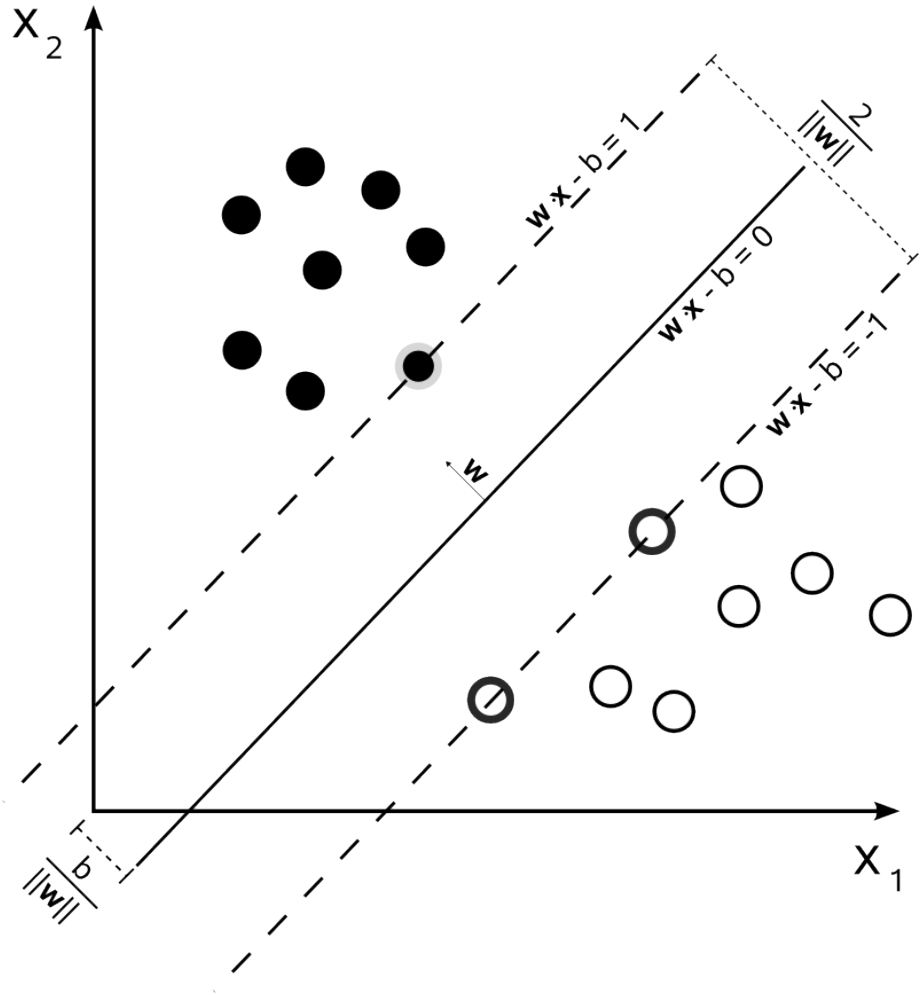
We train a **Support Vector Machine** classifier, specifying the feature Species as the class.

We **test** it with the rest of the data and get the class and the probability of belonging to that class.

Finally, we get the **prediction table**.

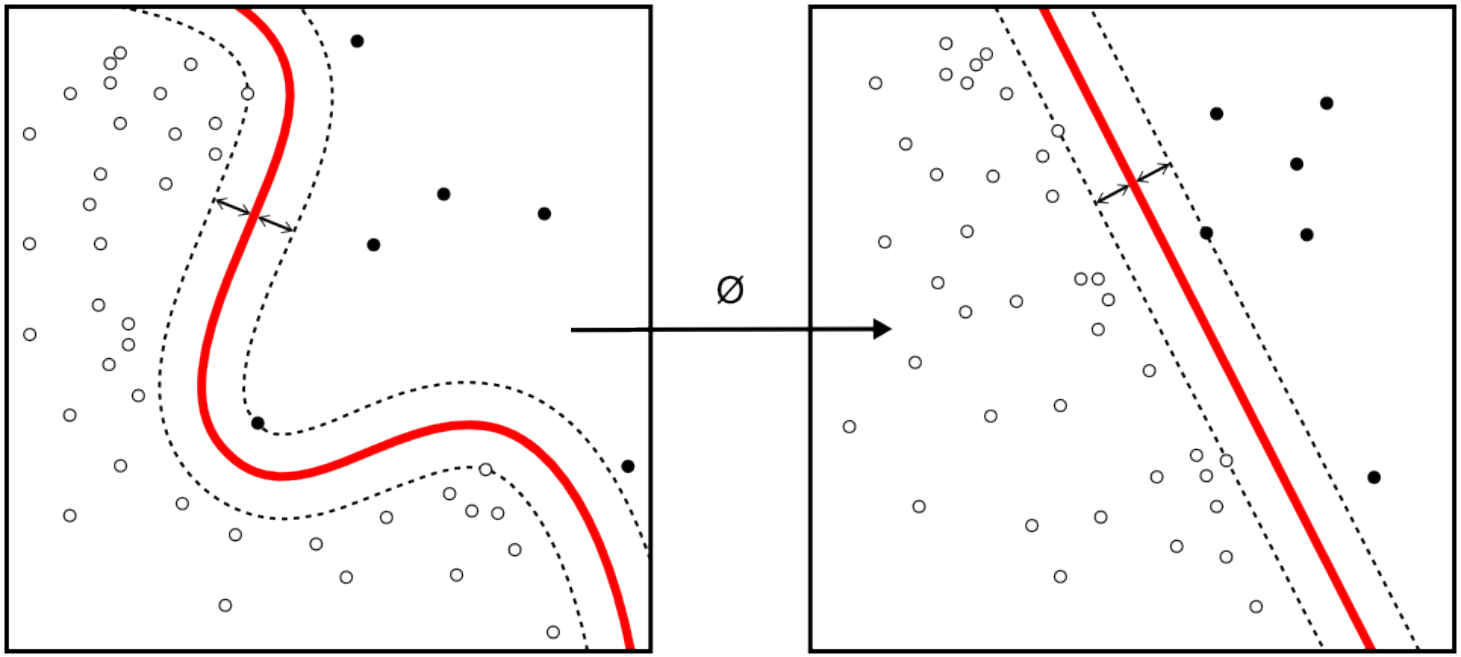
Support Vector Machine

Finds the maximum hyperplane between the two classes:



Support Vector Machine

Using kernels you can perform non-linear separations:



Naïve Bayes Classifier

Assuming that all the features (x_i) are independent, we want to create a classification method to predict the class (C_k), maximizing the probability of each feature inside each class $P(x_i|C_k)$.

We model the features by class, using, for example, Gaussian, and we create the conditional probabilities.

After, our model aims to maximize, given a new instance $E = (x_1, \ldots, x_n)$:

$$y = \underset{k \in 1,\ldots,K}{\arg\max} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Naïve Bayes

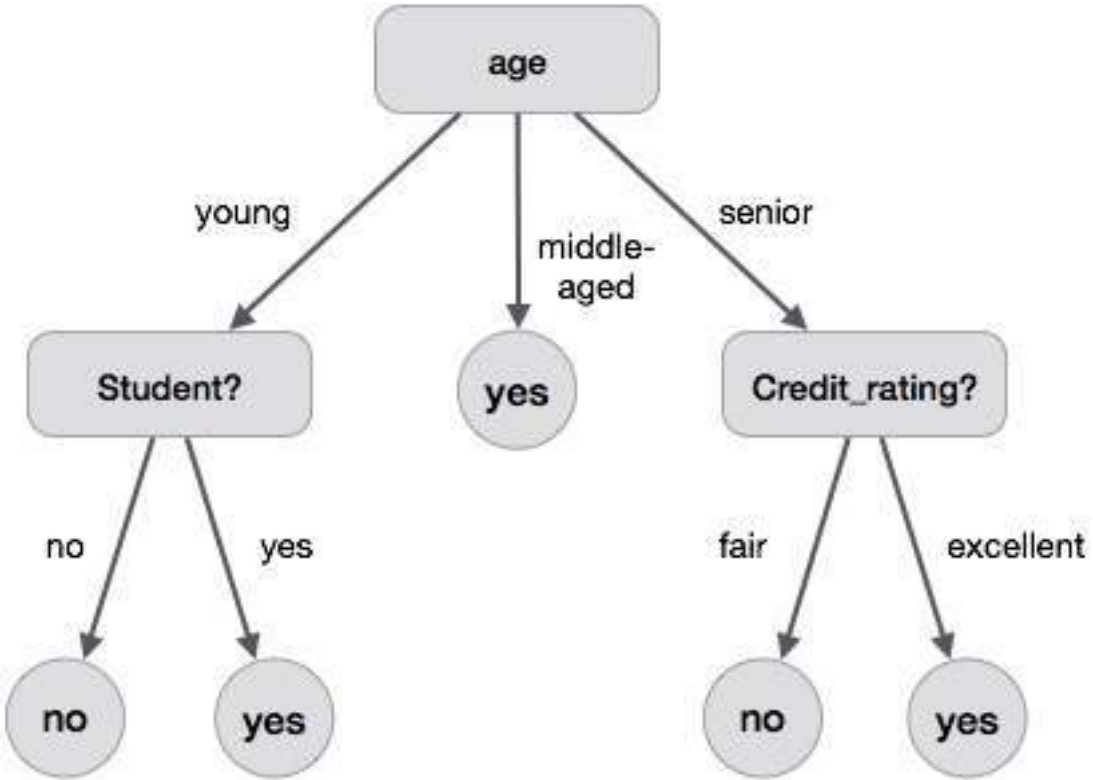
```
1 library(e1071)
2 trainIndex<-sample(1:length(iris[,1]),100)
3 trainData<-iris[trainIndex,]
4 testData<-iris[-trainIndex,]
5 nb_model <- naiveBayes(Species ~ ., data=trainData,probability=
      TRUE)
6 pred <- predict(nb_model,testData[,-5])
7 table(pred,testData[,5])
```

Output:

```
1 pred          setosa versicolor virginica
2   setosa          19          0         0
3   versicolor       0         15         1
4   virginica        0          1        14
```

Trees

Divides the space in potential decisions



Tree

```
1 library(party)
2 trainIndex<-sample(1:length(iris[,1]),100)
3 trainData<-iris[trainIndex,]
4 testData<-iris[-trainIndex,]
5 tree_model <- ctree(Species ~ ., data=trainData)
6 pred <- predict(tree_model,testData[,-5])
7 table(pred,testData[,5])
```

Output:

```
1 pred           setosa  versicolor  virginica
2   setosa           18           0          0
3   versicolor        0          12          2
4   virginica         0           1         17
```

Recursive Partition

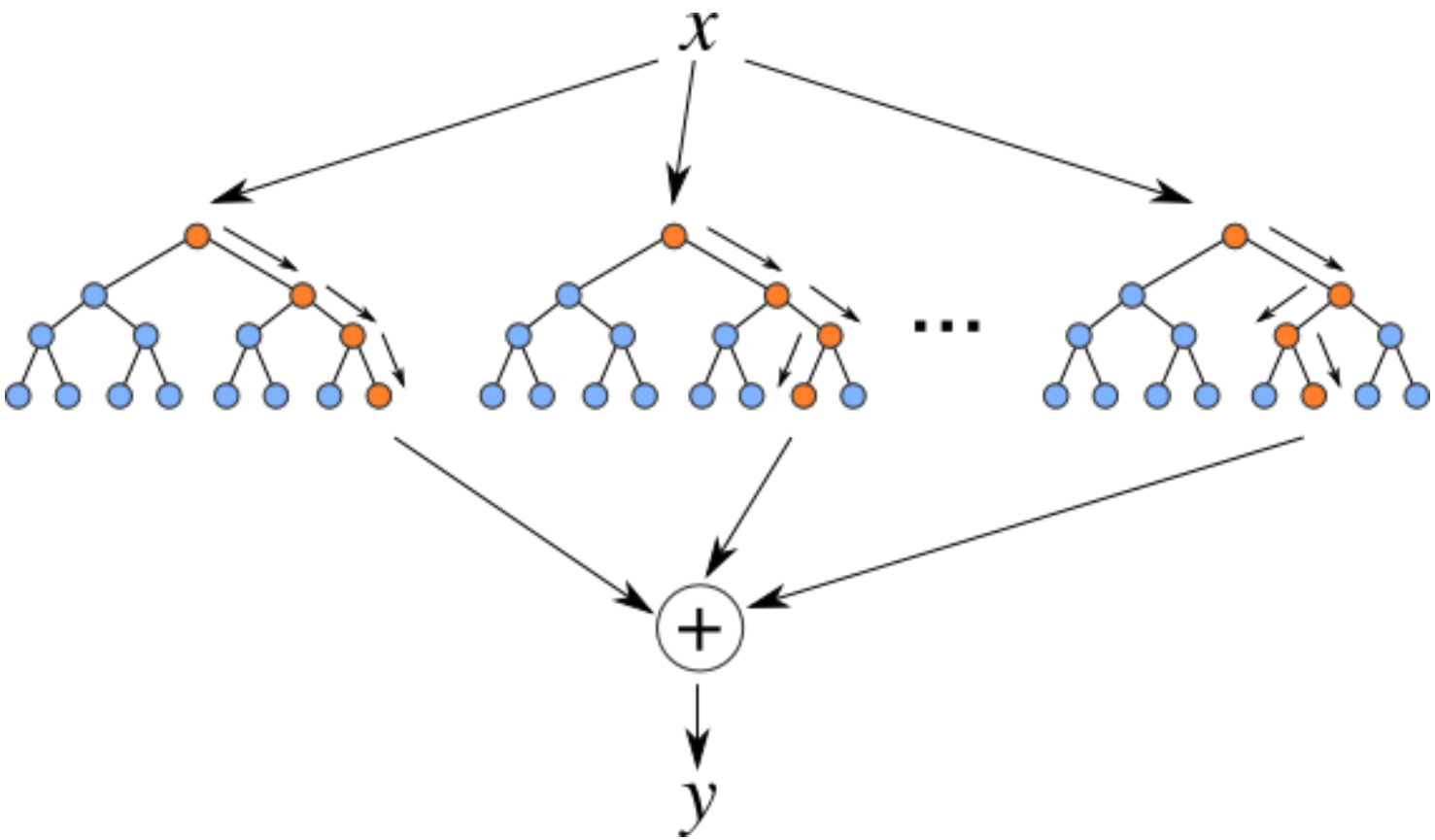
```
1 library(rpart)
2 trainIndex<-sample(1:length(iris[,1]),100)
3 trainData<-iris[trainIndex,]
4 testData<-iris[-trainIndex,]
5 rp_model <- rpart(Species ~ ., data=trainData)
6 pred <- predict(rp_model,testData[,-5],type="class")
7 table(pred,testData[,5])
```

Output:

```
1 pred          setosa versicolor virginica
2   setosa          17          0         0
3   versicolor       0         14         3
4   virginica        0          2        14
```

Forest

Combined several trees, trained with different features, into a voting system:



Forest

```
library(party)
trainIndex<-sample(1:length(iris[,1]),100)
trainData<-iris[trainIndex,]
testData<-iris[-trainIndex,]
rf_model <- cforest(Species ~ ., data=trainData)
pred <- predict(rf_model, newdata=testData[,-5],)
table(pred,testData[,5])
```

Output:

```
pred          setosa versicolor virginica
  setosa          21          0         0
  versicolor       0         17         0
  virginica        0          1        11
```

Validation

After training the classifier, normally there is an optimization step, named validation, focused on improving its parameters.

One of the most relevant validation processes is **n-fold cross-validation**.

This divides the train set in n parts, trains n-1 of them and tests with the separated one.

This process is repeated with different sets.

Evaluation

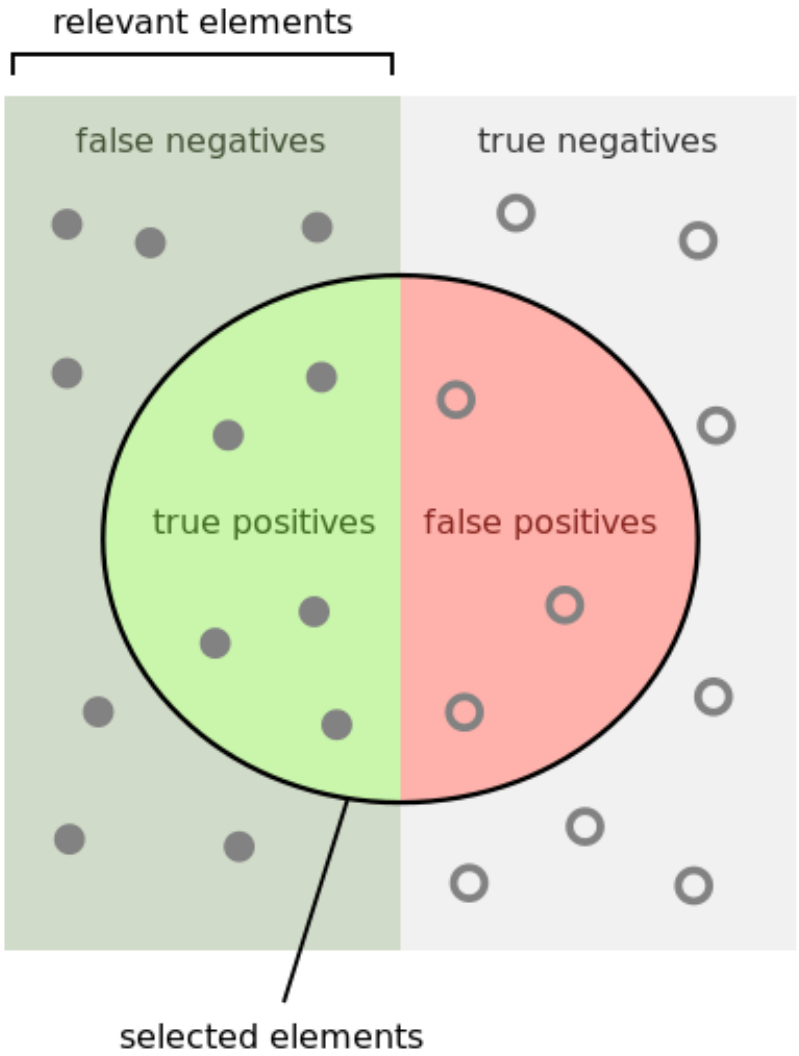
The quality of the classifier depends on the target, the most common one is the **accuracy**.

Other famous metrics are **precision** and **recall**.

Other targets might be **false positives or negatives**.

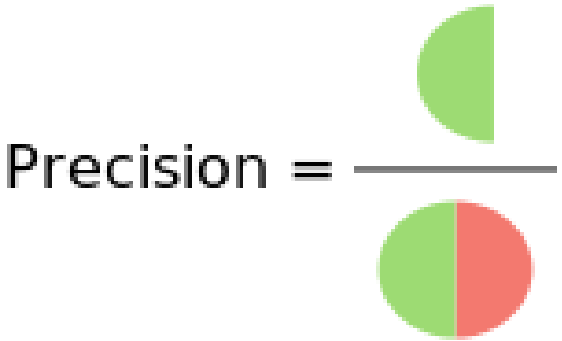
To control the trade off between false positives and negatives, we can use the **ROC curve**.

Evaluation

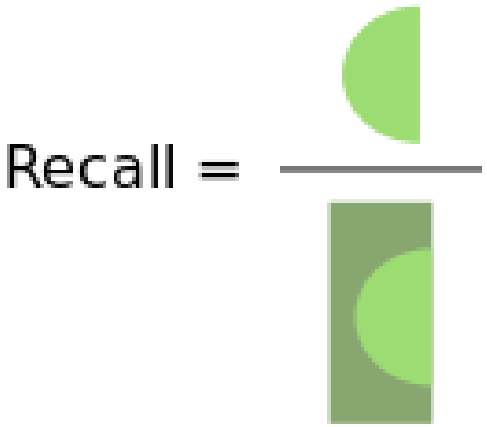

relevant elements
false negatives
true negatives
true positives
false positives
selected elements

Evaluation

How many selected items are relevant?



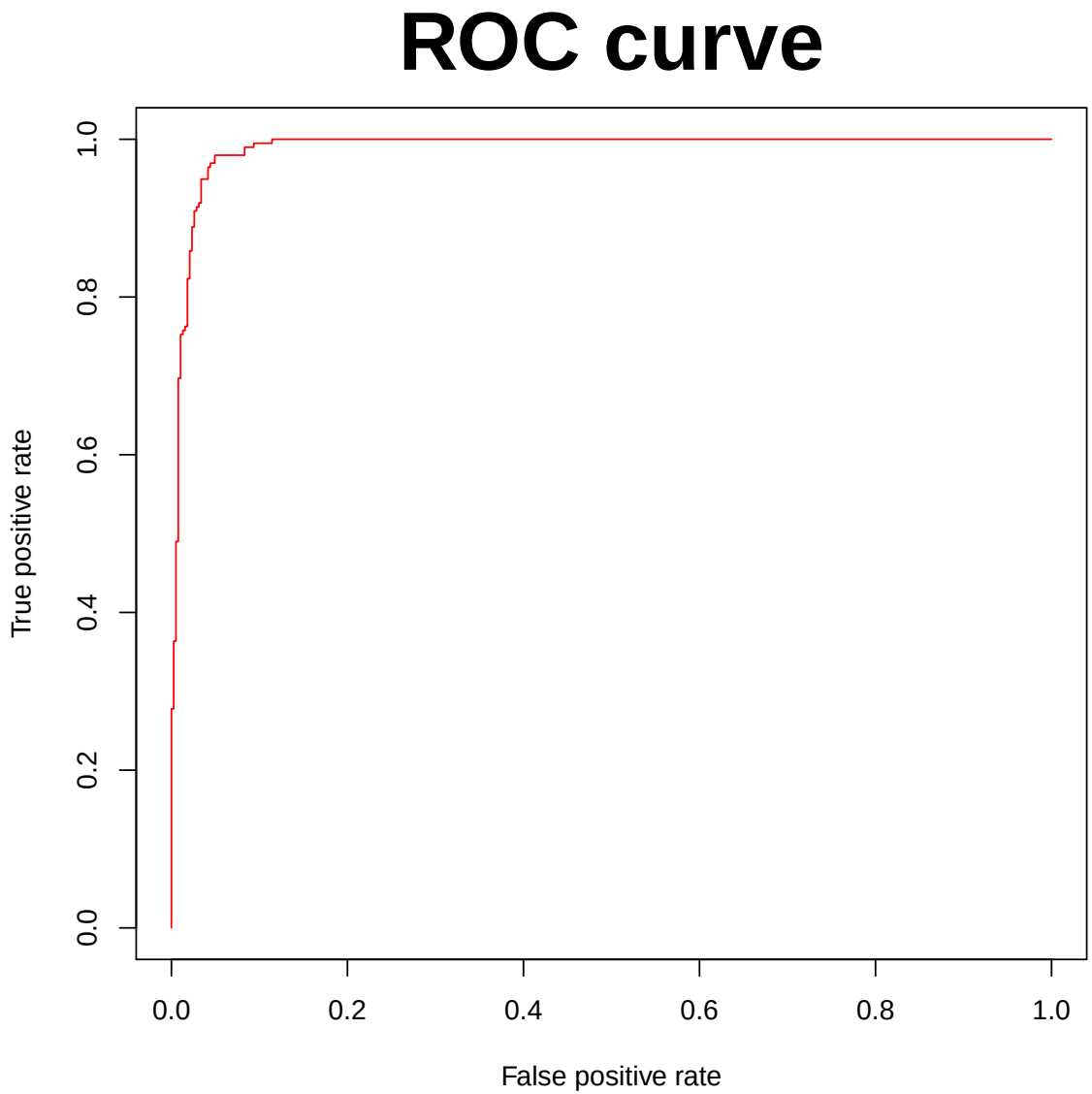
How many relevant items are selected?



The ROC curve

```
1  library(mlbench)
2  library(caret)
3  library(ROCR)
4  data(BreastCancer)
5  dataset <- BreastCancer[complete.cases(BreastCancer),]
6  dataset<-as.matrix(sapply(dataset[-1],as.numeric))
7  correlations <- cor(dataset[,-10])
8  hcorrelations <- findCorrelation(correlations, cutoff=0.85)
9  dataset <- as.data.frame(dataset[,-hcorrelations])
10 trainIndex<-sample(1:length(dataset[,1]),100)
11 trainData<-dataset[trainIndex,]
12 testData<-dataset[-trainIndex,]
13 rf_model <- cforest(Class ~ ., data=trainData)
14 pred <- predict(rf_model, newdata=as.data.frame(testData[,-9],))
15 probs <- pred-1
16 table(round(pred),testData[,9])
17 rocc<-prediction(probs,testData[,9]-1)
18 performance <- performance(rocc,"tpr","fpr")
```

ROC curve visualization



Exercises

Find the best classifier for the BreastCancer and Iris datasets using the above classifiers.

Run the classifier 10 times and compute the average accuracy for the evaluation.

Find the ROC curve of BreastCancer using 3 classifiers and visualize it.