IG.3510-Machine Learning

Lectures 3: Classification (Part II)

Dr. Patricia CONDE-CESPEDES

patricia.conde-cespedes@isep.fr

September 30th, 2024

Plan

Outline

Introduction to Decision Trees

- Decision trees can be used for **classification** or for **regression**.
- Decision trees approaches involve **stratifying** or **segmenting** the predictor space into a number of simple regions.

Introduction to Decision Trees

- Decision trees can be used for **classification** or for **regression**.
- Decision trees approaches involve **stratifying** or **segmenting** the predictor space into a number of simple regions.
- The splitting rules used can be represented using a tree diagram, that's where the name **decision tree** comes from.

Advantages and disadvantages

Advantages

+ Decision trees are easy to **interpret**.

Disadvantages:

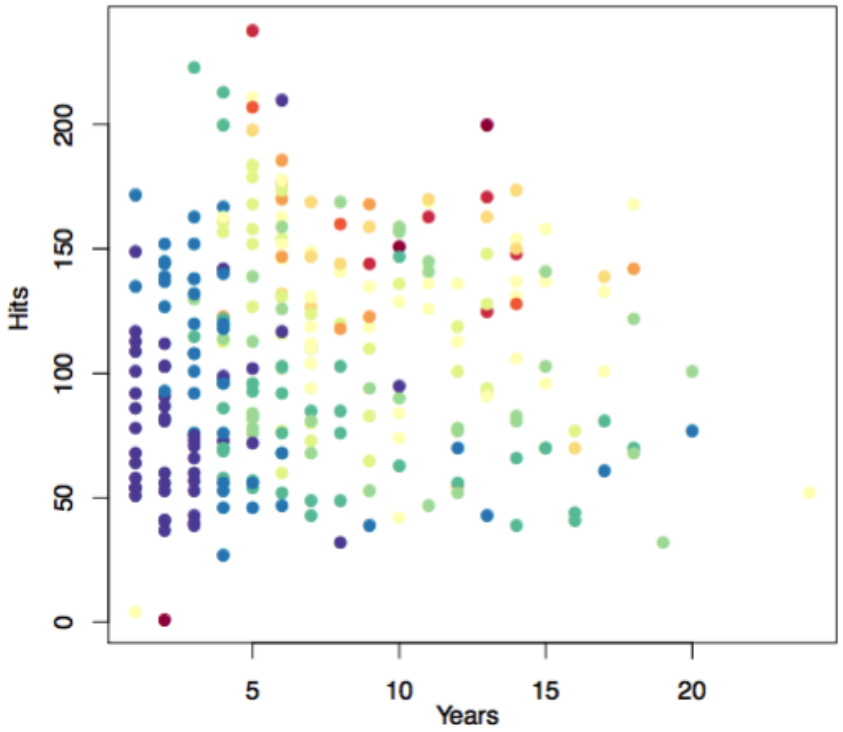
- Usually Decision trees are **not competitive** with other supervised learning approaches **in terms of prediction accuracy**.

Outline

Introductory example with the `Hitters` dataset (1/3)

Example : Predict a baseball player's `salary` based on :

- `Years` (the number of years that he has played in the major leagues)
- `Hits` (the number of hits the player made in the previous year)

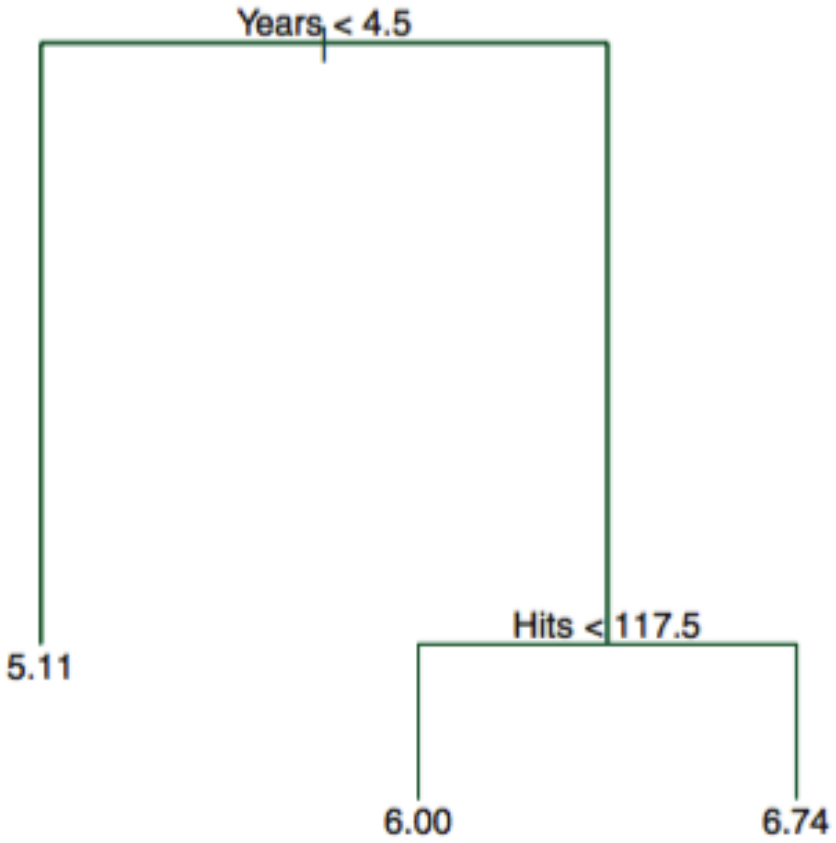


Raw data:
Salary is color-coded from low (blue), medium (green) to high (yellow,red).

How to stratify the predictors space?

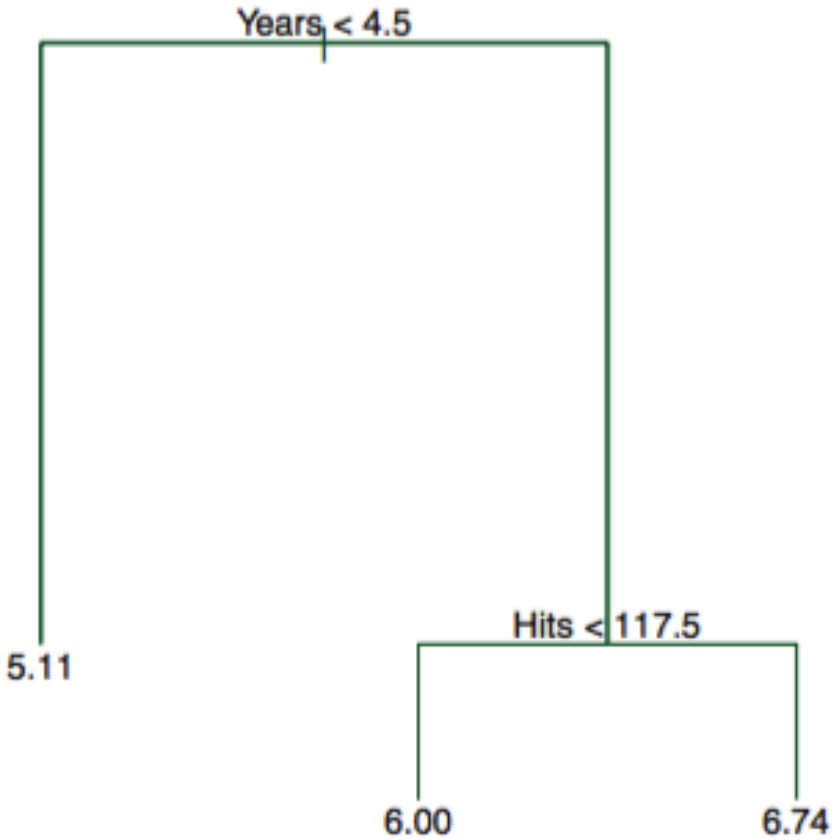
Regression tree for the `Hitters` data (2/3)

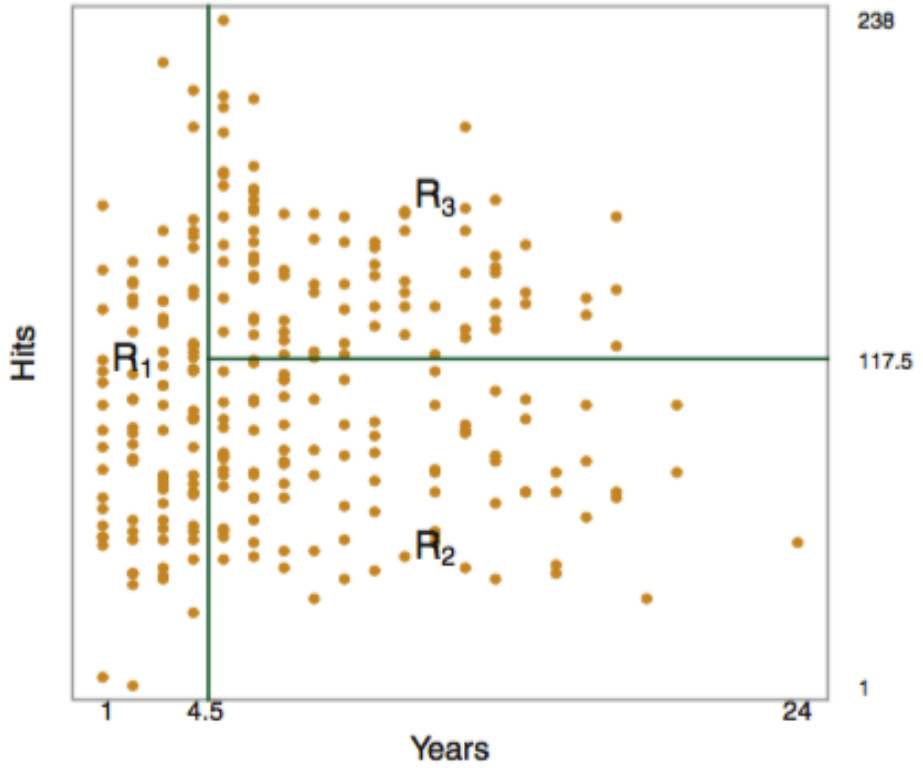
Overall, the tree stratifies the predictor' space into three regions:



Regression tree for the Hitters data (2/3)

Overall, the tree stratifies the predictor' space into three regions:



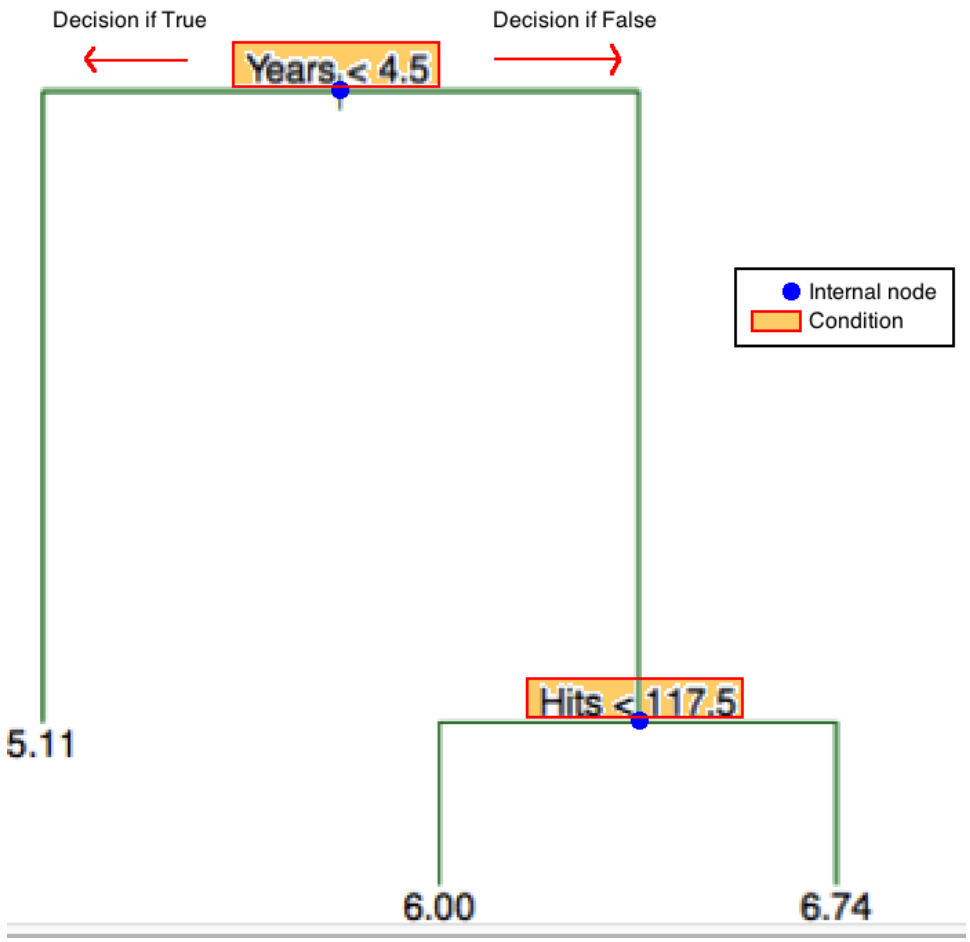


$R_1 = \{X | \texttt{Years} < 4.5\}$, $R_2 = \{X | \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$, and $R_3 = \{X | \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$.

How to interpret the regression tree for the `Hitters` example (3/3)

At a given internal node, the **condition** $X_j < t$ indicates the **rule** to split the predictor's space:

- If condition *True*, consider the left-hand branch
- else, consider the right-hand branch (which corresponds to $X_j \geq t$).



Terminology for Trees

Characteristics of trees:

- Nodes at the bottom with no branches are called **terminal nodes** or **leaves**

Terminology for Trees

Characteristics of trees:

- Nodes at the bottom with no branches are called **terminal nodes** or **leaves**
- Each **terminal node** represents a region R_j.

Terminology for Trees

Characteristics of trees:

- Nodes at the bottom with no branches are called **terminal nodes** or **leaves**
- Each **terminal node** represents a region R_j.
- The nodes in the tree where the predictor space is split are referred to as **internal nodes**.
  - For our example, the tree has two internal nodes and three terminal nodes, or leaves.

Terminology for Trees

Characteristics of trees:

- Nodes at the bottom with no branches are called **terminal nodes** or **leaves**
- Each **terminal node** represents a region R_j.
- The nodes in the tree where the predictor space is split are referred to as **internal nodes**.
  - For our example, the tree has two internal nodes and three terminal nodes, or leaves.
- The segments of the tree outgoing an internal node are called **branches**.

Terminology for Trees

Characteristics of trees:

- Nodes at the bottom with no branches are called **terminal nodes** or **leaves**
- Each **terminal node** represents a region R_j.
- The nodes in the tree where the predictor space is split are referred to as **internal nodes**.
  - For our example, the tree has two internal nodes and three terminal nodes, or leaves.
- The segments of the tree outgoing an internal node are called **branches**.

Predictions: The number in each leaf is the mean of the response variable for the observations that fall in the corresponding region.

Illustrative exercise

Given the following training observations:

X_1	X_2	Y
1	2	3
2	1	2
2	2	4
2	4	8
3	1	3
3	5	9
4	4	11
5	1	5
6	2	7
6	5	12

and the following rules:

- if $(X_2 > 3)$ then: R_1
- else:
  - If $(X_1 < 4)$ then R_2
  - else R_3

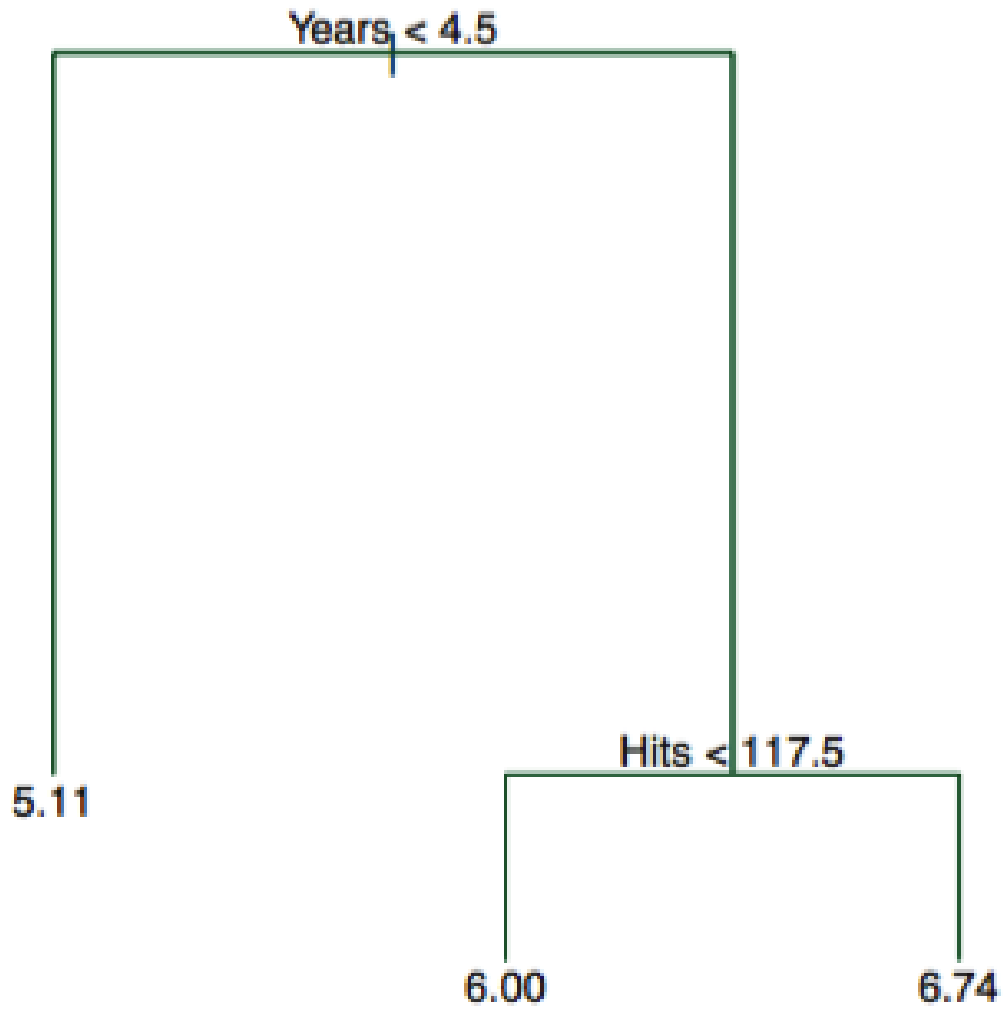
Questions:

1) Build the regression tree.
2) Make predictions for the following test observations:

   - $X_1 = 1$, $X_2 = 4$
   - $X_1 = 7$, $X_2 = 2$

Interpretation of Results for the `Hitters` data

- `Years of experience` is the most important factor to determine `Salary`
- For a more experienced player (more than 5 years), the number of hits made in the previous year is important to determine the salary.



The process of building a regression tree

There are roughly two steps:

Step 1: **Stratification:** Divide the predictor Space -the set of possible values for $X_1, X_2, \ldots, X_p$- into J disjoint regions: $R_1, R_2, \ldots, R_J$.

The process of building a regression tree

There are roughly two steps:

Step 1: **Stratification:** Divide the predictor Space -the set of possible values for $X_1, X_2, \ldots, X_p$- into J disjoint regions: $R_1, R_2, \ldots, R_J$.

Step 2: **Prediction:** Given an observation $(x_1, x_2, \ldots, x_p)$ that falls into the R_j region, its predicted value $\hat{y}$ is the mean of the response variable among all the training observations falling in R_j.

The process of building a regression tree

There are roughly two steps:

Step 1: **Stratification:** Divide the predictor Space -the set of possible values for $X_1, X_2, \ldots, X_p$- into J disjoint regions: $R_1, R_2, \ldots, R_J$.

Step 2: **Prediction:** Given an observation $(x_1, x_2, \ldots, x_p)$ that falls into the R_j region, its predicted value $\hat{y}$ is the mean of the response variable among all the training observations falling in R_j.

We will now focus on step 1. In theory, the regions could have any shape. However, to simplify we divide the predictor space into **high-dimensional rectangles**, or boxes.

How to stratify the feature space?

The goal is to find regions $R_1, R_2, \ldots, R_J$ that minimize the RSS (Residual Sum of Squares):

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean of the target Y for the training observations belonging to the jth region.

How to stratify the feature space?

The goal is to find regions $R_1, R_2, \ldots, R_J$ that minimize the RSS (Residual Sum of Squares):

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean of the target Y for the training observations belonging to the jth region.

Unfortunately, it is **computationally infeasible** to consider every possible partition of the feature space into J boxes!

Recursive Binary Splitting for stratification

Recursive Binary Splitting is a **top-down, greedy** approach.

Recursive Binary Splitting for stratification

Recursive Binary Splitting is a **top-down, greedy** approach.

- *top down:* it begins at the top of the tree (when all observations belong to a single region) and then **successively** splits the predictor space; each split is indicated via two new branches further down on the tree.

Recursive Binary Splitting for stratification

Recursive Binary Splitting is a **top-down, greedy** approach.

- *top down:* it begins at the top of the tree (when all observations belong to a single region) and then **successively** splits the predictor space; each split is indicated via two new branches further down on the tree.
- *greedy:* It is greedy because at each step, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Recursive Binary splitting process (1/2)

Step 1: Select the predictor j and the cutpoint s such that splitting the predictor space in two regions: $\{X|Xj < s\}$ and $\{X|Xj \geq s\}$ leads to the greatest decrease of *RSS*.

Recursive Binary splitting process (1/2)

Step 1: Select the predictor j and the cutpoint s such that splitting the predictor space in two regions: $\{X|Xj < s\}$ and $\{X|Xj \geq s\}$ leads to the greatest decrease of RSS. In other words:
For any pair (j, s) we define the pair of half-planes:

$$R_1(j, s) = \{X|X_j < s\} \text{ and } R_2(j, s) = \{X|Xj \geq s\}$$

and seek the values of j and s that minimize:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

where $\hat{y}_{R_1}$ is the mean response for training observations in $R_1(j, s)$, and $\hat{y}_{R_2}$ is the mean response for the training observations in $R_2(j, s)$.

This step can be done quite quickly, especially if p is small!

Recursive Binary splitting process (2/2)

Next steps consist in repeating step 1 to recursively split the previously created regions:

Step 2: Repeat step 1, look for the best predictor and best cutpoint in order to split the data further so as to minimize the *RSS*. However, instead of splitting the entire predictor space, split one of the two previously identified regions. So, at the end of this step, there are three regions.

Recursive Binary splitting process (2/2)

Next steps consist in repeating step 1 to recursively split the previously created regions:

Step 2: Repeat step 1, look for the best predictor and best cutpoint in order to split the data further so as to minimize the *RSS*. However, instead of splitting the entire predictor space, split one of the two previously identified regions. So, at the end of this step, there are three regions.

Step 3: Repeat in order to split one of these three regions further, so as to minimize the *RSS*.

Recursive Binary splitting process (2/2)

Next steps consist in repeating step 1 to recursively split the previously created regions:

Step 2: Repeat step 1, look for the best predictor and best cutpoint in order to split the data further so as to minimize the *RSS*. However, instead of splitting the entire predictor space, split one of the two previously identified regions. So, at the end of this step, there are three regions.

Step 3: Repeat in order to split one of these three regions further, so as to minimize the *RSS*.

Step 4: Repeat the process until a **stopping criterion** is reached. *For instance, until no region contains more than five observations*.

Recursive Binary splitting process (2/2)

Next steps consist in repeating step 1 to recursively split the previously created regions:

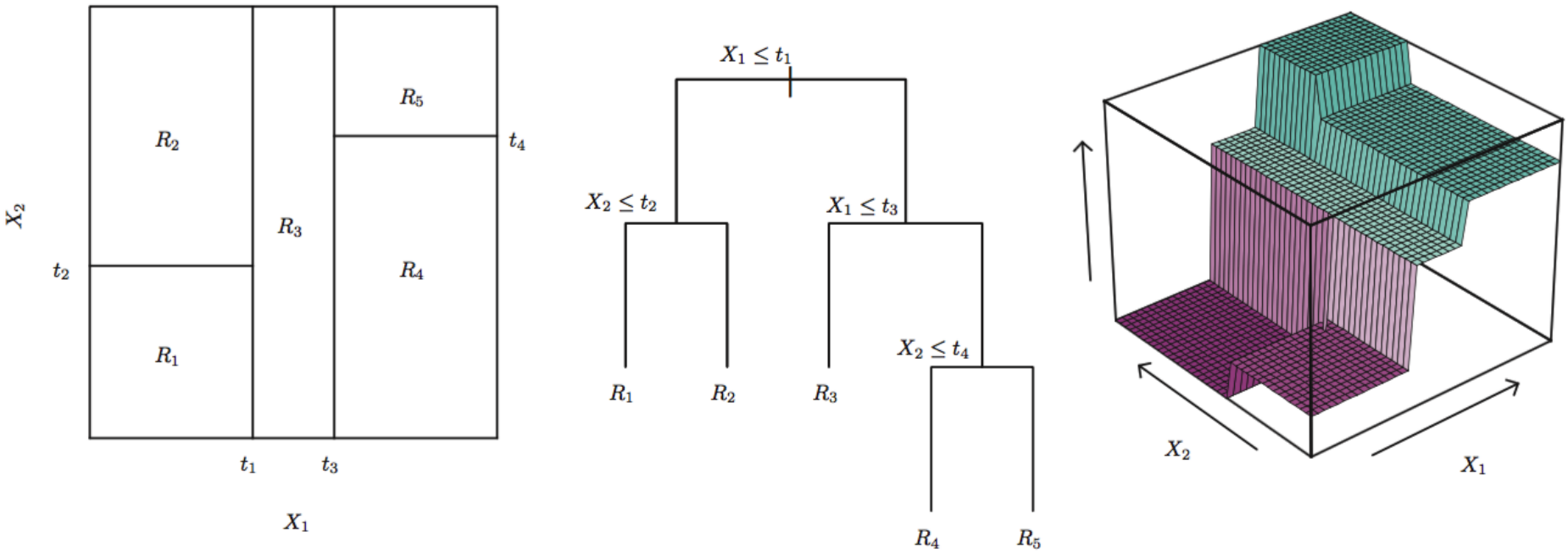
Step 2: Repeat step 1, look for the best predictor and best cutpoint in order to split the data further so as to minimize the *RSS*. However, instead of splitting the entire predictor space, split one of the two previously identified regions. So, at the end of this step, there are three regions.

Step 3: Repeat in order to split one of these three regions further, so as to minimize the *RSS*.

Step 4: Repeat the process until a **stopping criterion** is reached. *For instance, until no region contains more than five observations*.

Predictions: once the regions created $R_1, \ldots, R_J$ make predictions by taking the mean value of the observations in each region.

Example of the *Recursive Binary splitting* process result

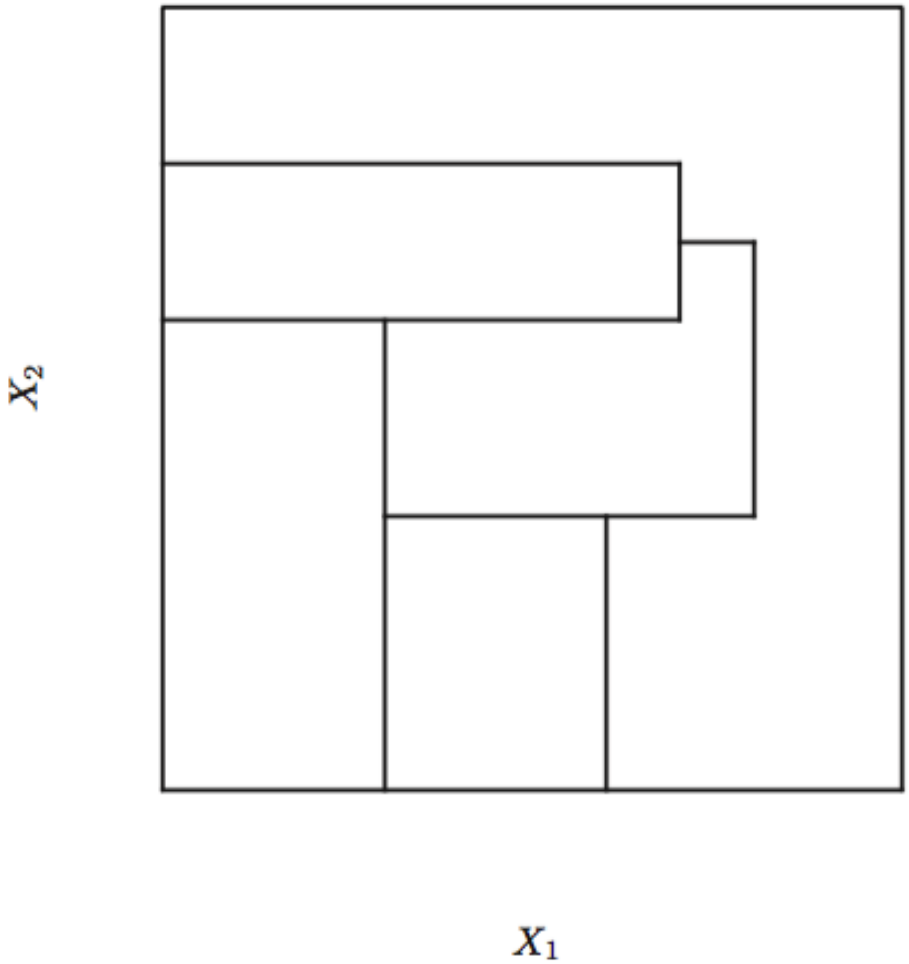


Left: The output of *Recursive Binary Splitting* on a two-dimensional example.
Center: A tree corresponding to the partition in the left panel.

Right: A perspective plot of the prediction surface corresponding to that tree.

counter example of the *Recursive binary splitting*

A partition of two-dimensional feature space that **could not** result from recursive binary splitting.



Tree Pruning

How many leaves must the tree have?

Tree Pruning

How many leaves must the tree have?

A tree with so many terminal nodes might cause **overfitting** leading to good performance in the training set but poor performance in the test set.

- For instance, consider a tree having as many terminal nodes as observations in such a way that each observation has its own region. So, the training error is zero.

Tree Pruning

How many leaves must the tree have?

A tree with so many terminal nodes might cause **overfitting** leading to good performance in the training set but poor performance in the test set.

- For instance, consider a tree having as many terminal nodes as observations in such a way that each observation has its own region. So, the training error is zero.

SOLUTION:

A good strategy is to grow a very large tree T_0 (with many leaves), and then **prune** it back in order to obtain a **subtree**.

This approach is called **Cost complexity pruning** also known as **weakest link pruning**.

Cost complexity pruning / weakest link pruning

Intuition: the goal is to select a subtree that leads to the lowest test error.

Cost complexity pruning / weakest link pruning

Intuition: the goal is to select a subtree that leads to the lowest test error.

Approach: For each value of α there is a subtree $T \subset T_0$ such that:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \text{ is minimal.}$$

where $|T|$ is the number of terminal nodes of the tree T, R_m is the region corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response associated to R_m.

Cost complexity pruning / weakest link pruning

Intuition: the goal is to select a subtree that leads to the lowest test error.

Approach: For each value of α there is a subtree $T \subset T_0$ such that:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \text{ is minimal.}$$

where $|T|$ is the number of terminal nodes of the tree T, R_m is the region corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response associated to R_m.

Which value of α to select?

Select the optimal value α^* using cross-validation to estimate the error test.

Then, use the full dataset to obtain the subtree that corresponds to α^*.

About the parameter α

The choice of the parameter α is crucial!

- The tuning parameter α controls a trade-off between the *RSS* (fit to the training data) and the subtree's complexity.
- When $\alpha = 0$, then we get T_0.

About the parameter α

The choice of the parameter α is crucial!

- The tuning parameter α controls a trade-off between the *RSS* (fit to the training data) and the subtree's complexity.
- When $\alpha = 0$, then we get T_0.
- As α increases, there is a price to pay for having a tree with many terminal nodes, and so a smaller subtree will be preferable.

About the parameter α

The choice of the parameter α is crucial!

- The tuning parameter α controls a trade-off between the *RSS* (fit to the training data) and the subtree's complexity.
- When $\alpha = 0$, then we get T_0.
- As α increases, there is a price to pay for having a tree with many terminal nodes, and so a smaller subtree will be preferable.
- As α increases, branches get pruned from the tree in a nested and predictable fashion, so obtaining the whole sequence of subtrees as a function of α is easy!

About the parameter α

The choice of the parameter α is crucial!

- The tuning parameter α controls a trade-off between the *RSS* (fit to the training data) and the subtree's complexity.
- When $\alpha = 0$, then we get T_0.
- As α increases, there is a price to pay for having a tree with many terminal nodes, and so a smaller subtree will be preferable.
- As α increases, branches get pruned from the tree in a nested and predictable fashion, so obtaining the whole sequence of subtrees as a function of α is easy!

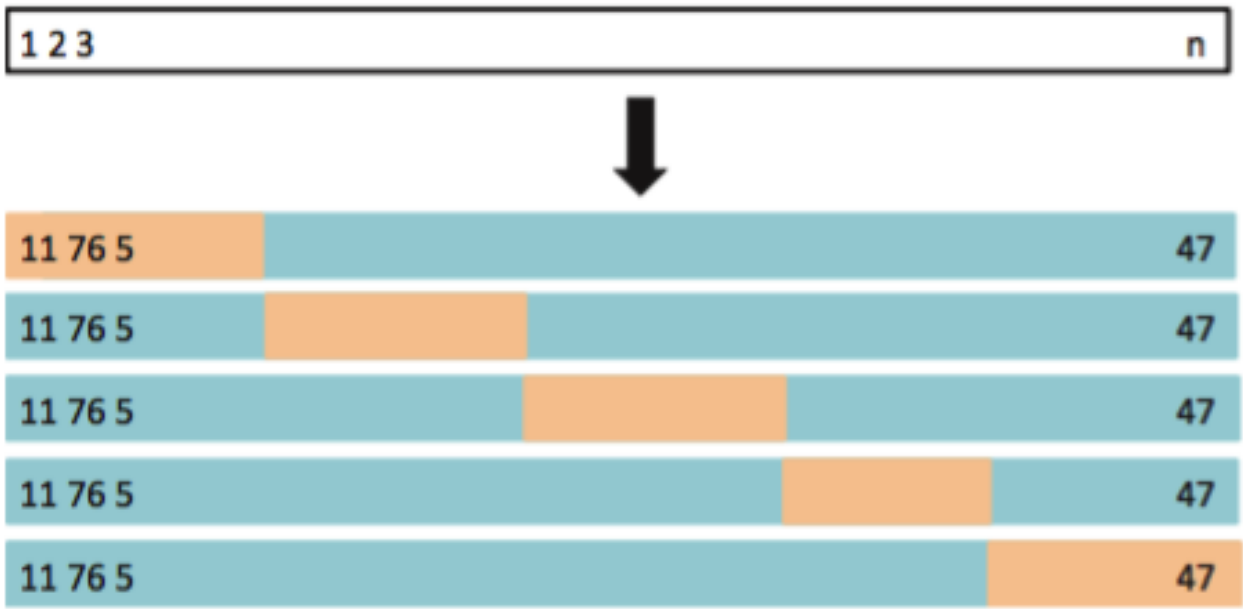
When performing cross-validation α plays the role of a **penalty** for a very big tree that barely contributes to decrease the test error.

Reminder : K-fold Cross-validation

Idea : randomly split the data into K equal-sized groups or *folds*. Then, leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part.

Repeat for each fold $k = 1, 2, ...K$ fold and estimate the test error. Finally the estimated overall test error is the average of the K estimates.

A schematic display of 5-fold CV



A set of observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set. The test error is estimated by averaging the five estimates.

Summary of the Building of a regression tree

Step 1: Use *Recursive binary splitting* to build a large tree T_0 on the training data.

Step 2: Apply *cost complexity pruning* to T_0 in order to obtain a sequence of best subtrees, as a function of α.

Step 3: Use K-fold cross-validation to choose α. For $k = 1, \ldots, K$ do:

1. Repeat Steps 1 and 2 on the $\frac{K-1}{K}$*th* fraction of the training data, excluding the *kth* fold.
2. Estimate the test error on the data in the left-out *kth* fold, as a function of α.

Average the results, and choose α that minimizes the average estimate error test.

Step 4: Return the subtree from Step 2 that corresponds to the chosen value of α.

Example with the `Hitters` data set (1/3)

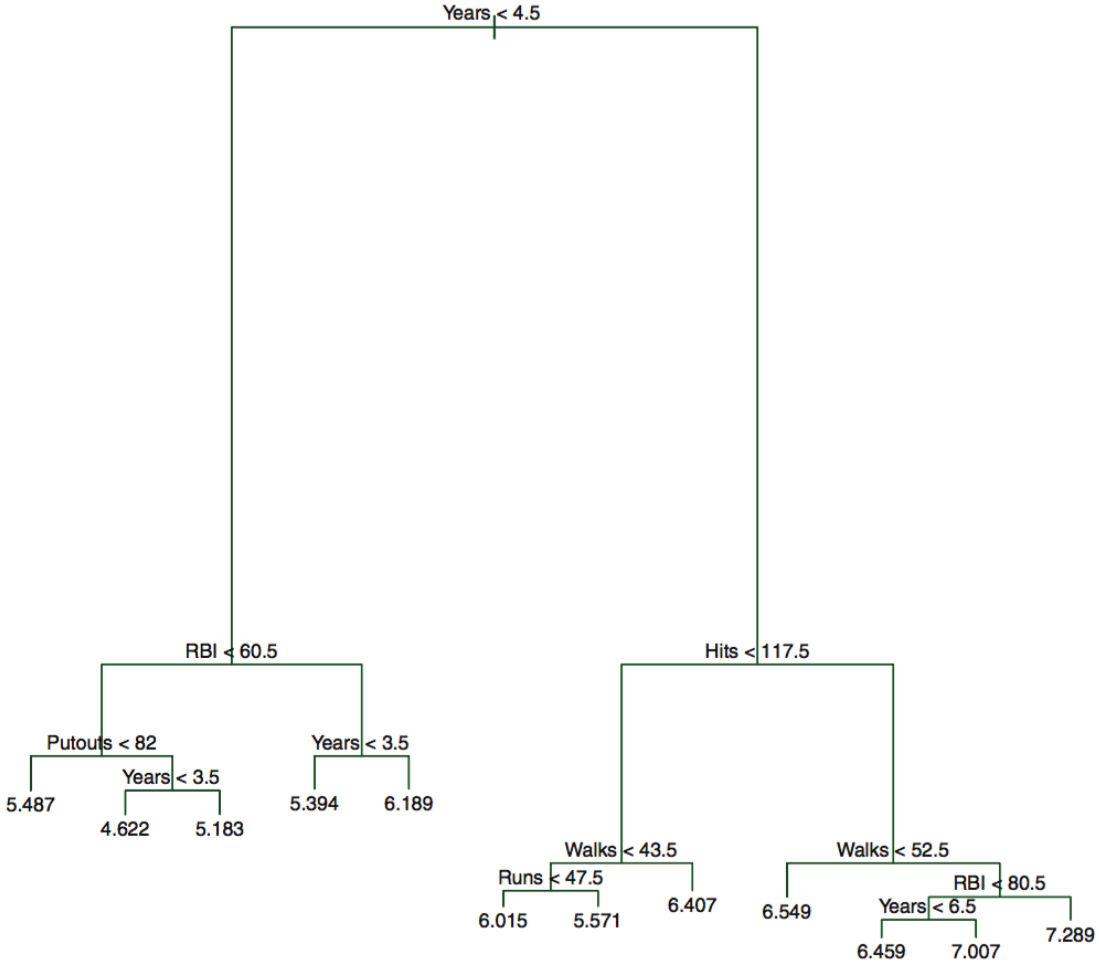
Consider the `Hitters` dataset:

- First, randomly divide the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We build a large regression tree T_0 on the training data and vary α in order to create subtrees with different numbers of terminal nodes.
- Finally, perform six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α.

Notice there is a ONE to ONE correspondance between α and the number of leaves $|T|$.

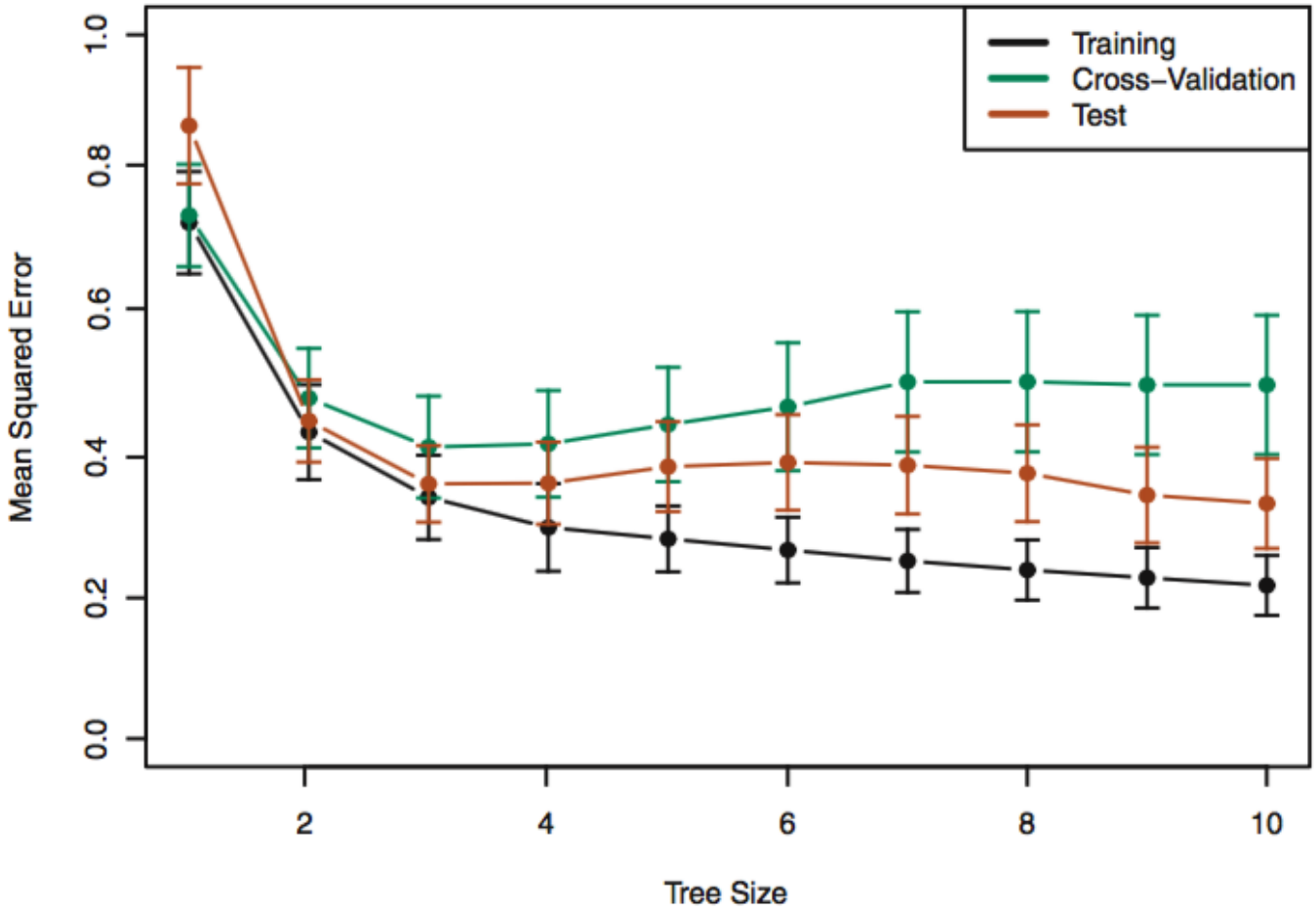
Large regression tree T_0, `Hitters` example (2/3)

Unpruned tree resulting from the recursive binary splitting on the `Hitters` data with 9 predictors:



Cross-validation, `Hitters` example (3/3)

CV error as a function of the number of leaves.

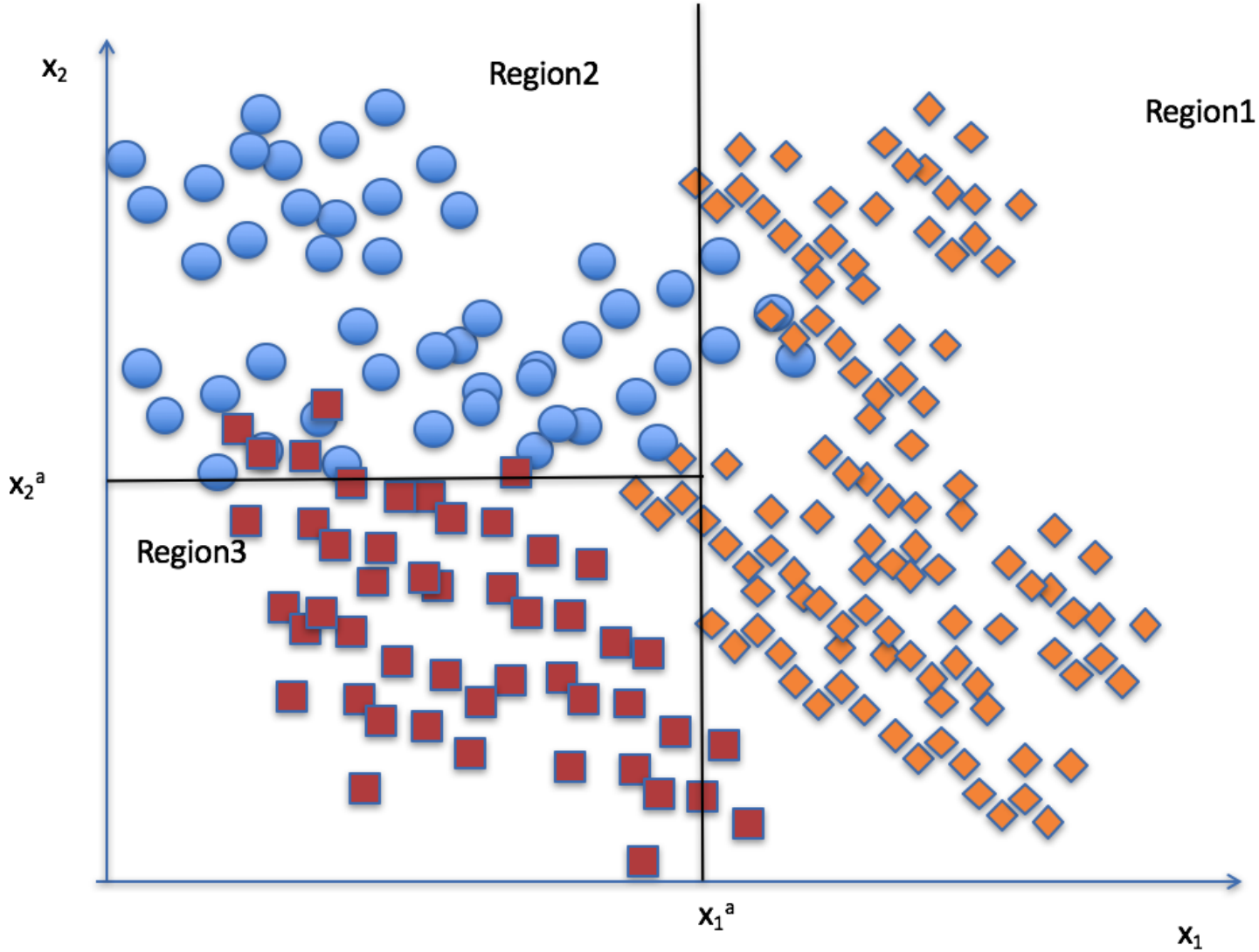


Orange: test error; black: training error curve; Green: CV error. Also shown are standard error bars around the estimated errors.

Outline

Classification trees

Let us suppose the response variable Y has 3 categories:



How to build a Classification tree?

- The procedure is very similar to that of regression trees, except that the predicted variable Y is **qualitative**.

How to build a Classification tree?

- The procedure is very similar to that of regression trees, except that the predicted variable Y is **qualitative**.
- In classification, given a new observation we predict that it belongs to the **most commonly occurring class** in the region to which it belongs to.
- Likewise in regression, start by building a large classification tree *recursive binary splitting*.

How to build a Classification tree?

- The procedure is very similar to that of regression trees, except that the predicted variable Y is **qualitative**.
- In classification, given a new observation we predict that it belongs to the **most commonly occurring class** in the region to which it belongs to.
- Likewise in regression, start by building a large classification tree *recursive binary splitting*. However, in classification we can not use the *RSS* as a criterion for making binary splits.
- Instead we minimize two other criteria which measure the **purity** of the node. These are the **Gini index** and the **cross-entropy**.

The classification error rate

The classification error rate of a region m is simply the fraction of the training observations in that region that do not belong to the most common class:

$$\text{Error}_{m,\text{Train}} = 1 - \max_k(\hat{p}_{mk}).$$

Here $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that belong to class k.

The classification error rate is preferable if prediction accuracy of the final pruned tree is the goal.

Gini index G

For a given region m the Gini index is defined by:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training observations in region m that belong to class k.

Gini index G

For a given region m the Gini index is defined by:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training observations in region m that belong to class k.

Intuition: Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are either close to 0 or 1. For this reason the *Gini index* is referred to as a measure of **node purity** - a small value indicates that a node contains predominantly observations from a single class.

Cross-entropy or Deviance D

An alternative to the Gini index is **cross-entropy**. For a given region m this index is given by:

$$D_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

It turns out that the Gini index and the cross-entropy are very similar numerically.

Cross-entropy or Deviance D

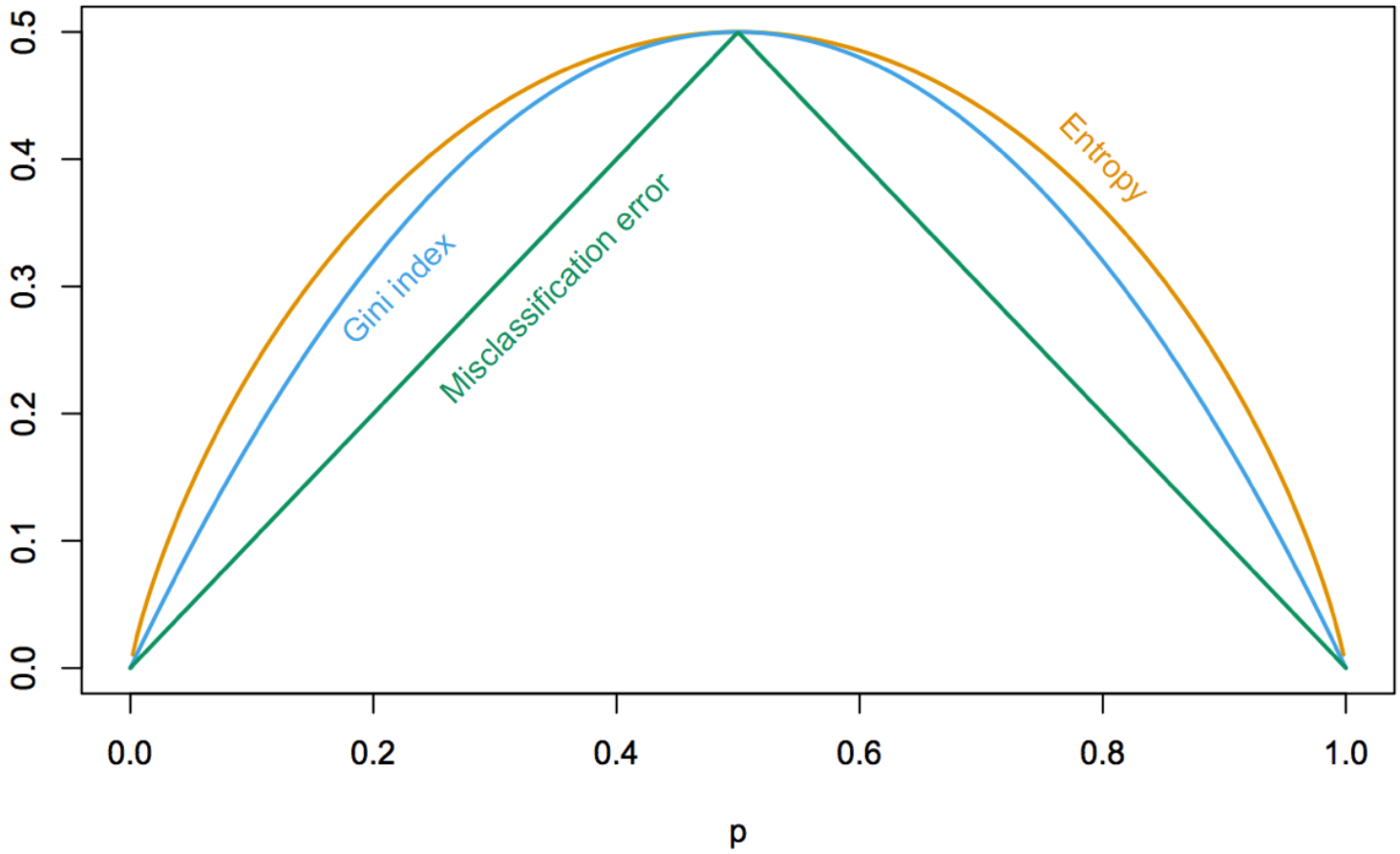
An alternative to the Gini index is **cross-entropy**. For a given region m this index is given by:

$$D_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

It turns out that the Gini index and the cross-entropy are very similar numerically.

Since $0 \leq \hat{p}_{mk} \leq 1$, it follows that $-\hat{p}_{mk} \log \hat{p}_{mk} \geq 0$. One can deduce that the cross-entropy will take on a value near zero if the $\hat{p}_{mk}$'s are all near 0 or near 1. Therefore, the cross-entropy will take on a small value if the *mth* node is pure.

Comparison classification error rate, Gini index and entropy



Cross-entropy and the Gini index are differentiable, and hence more practical to numerical optimization. However, the classification error rate is preferable if prediction accuracy is the goal.

Example of Classification tree with the `heart` data

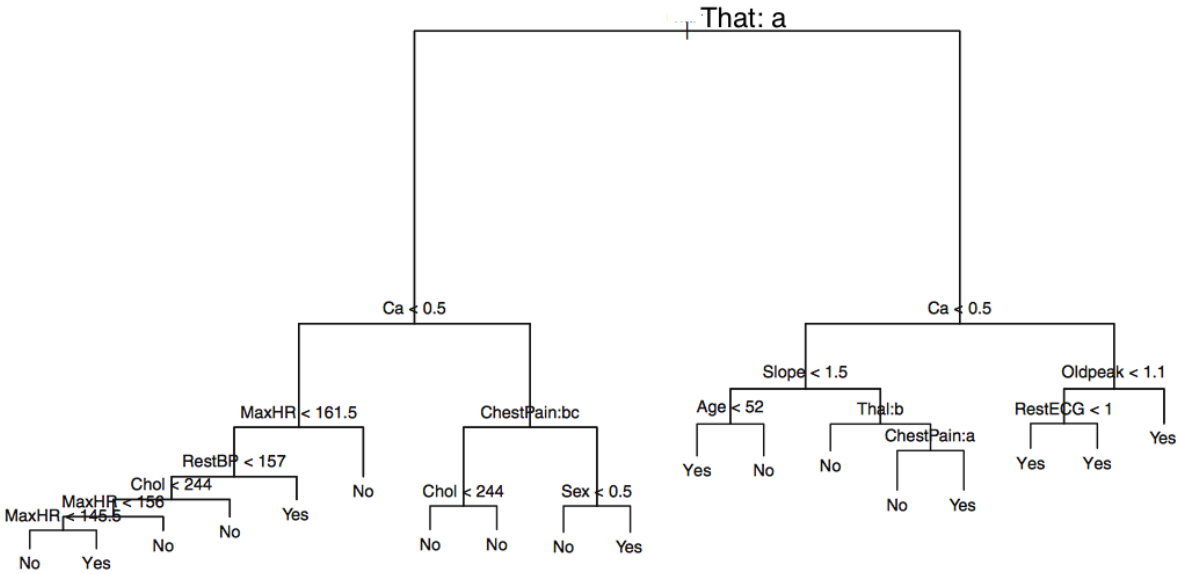
Consider the `Heart` data set:

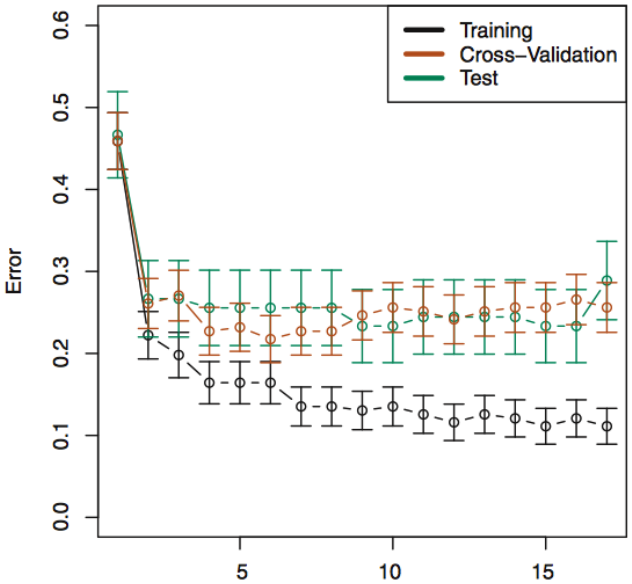
- These data contain a binary variable `HD` for 303 patients who presented with chest pain.

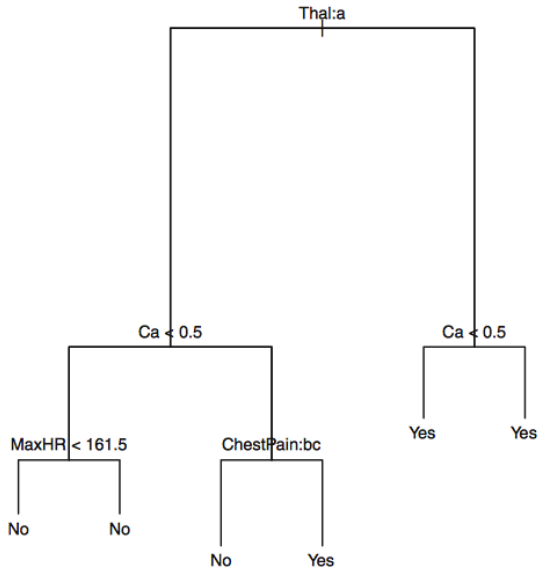
$$HD = \begin{cases} \text{Yes: presence of heart disease.} \\ \text{No: No heart disease.} \end{cases}$$

- 13 predictors including `Age`, `Sex`, `Chol` (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with 6 terminal nodes (see next slide).
- Decision trees can be constructed with qualitative predictors as well, that is the case for this example. Consider the top node `Thal` (3 categories: *normal*, *fixed* and *reversible defects*).

Classification tree, Heart dataset







Some remarks:

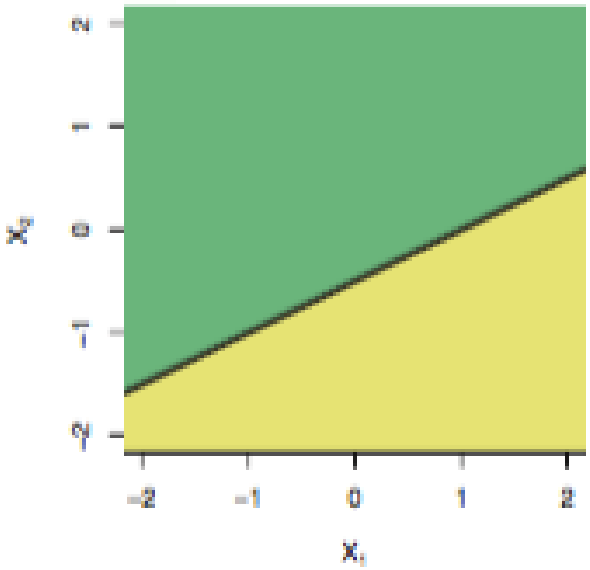
- It is possible to include qualitative predictors.
- Some splits yield to two terminal nodes that have the same predicted value

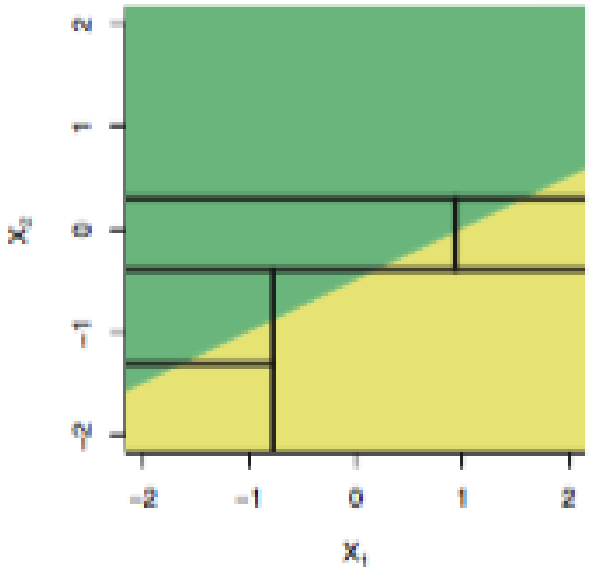
Trees vs. Linear Models

Linear model

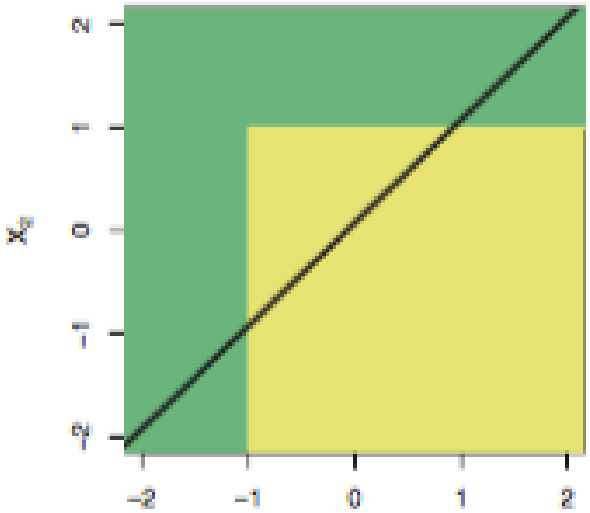
Tree-based model

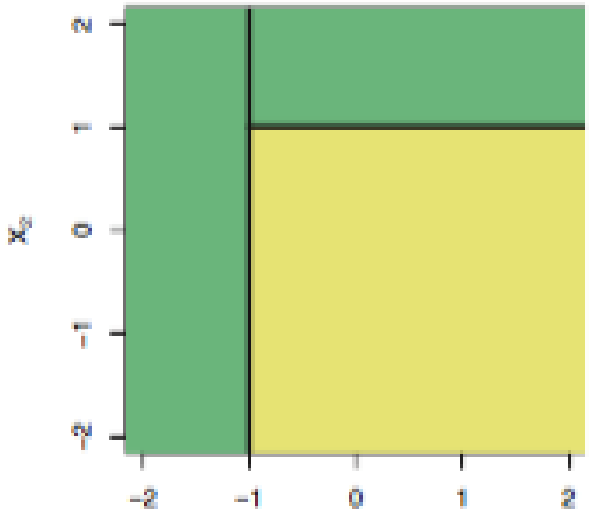
True Linear boundary





Non-linear boundary

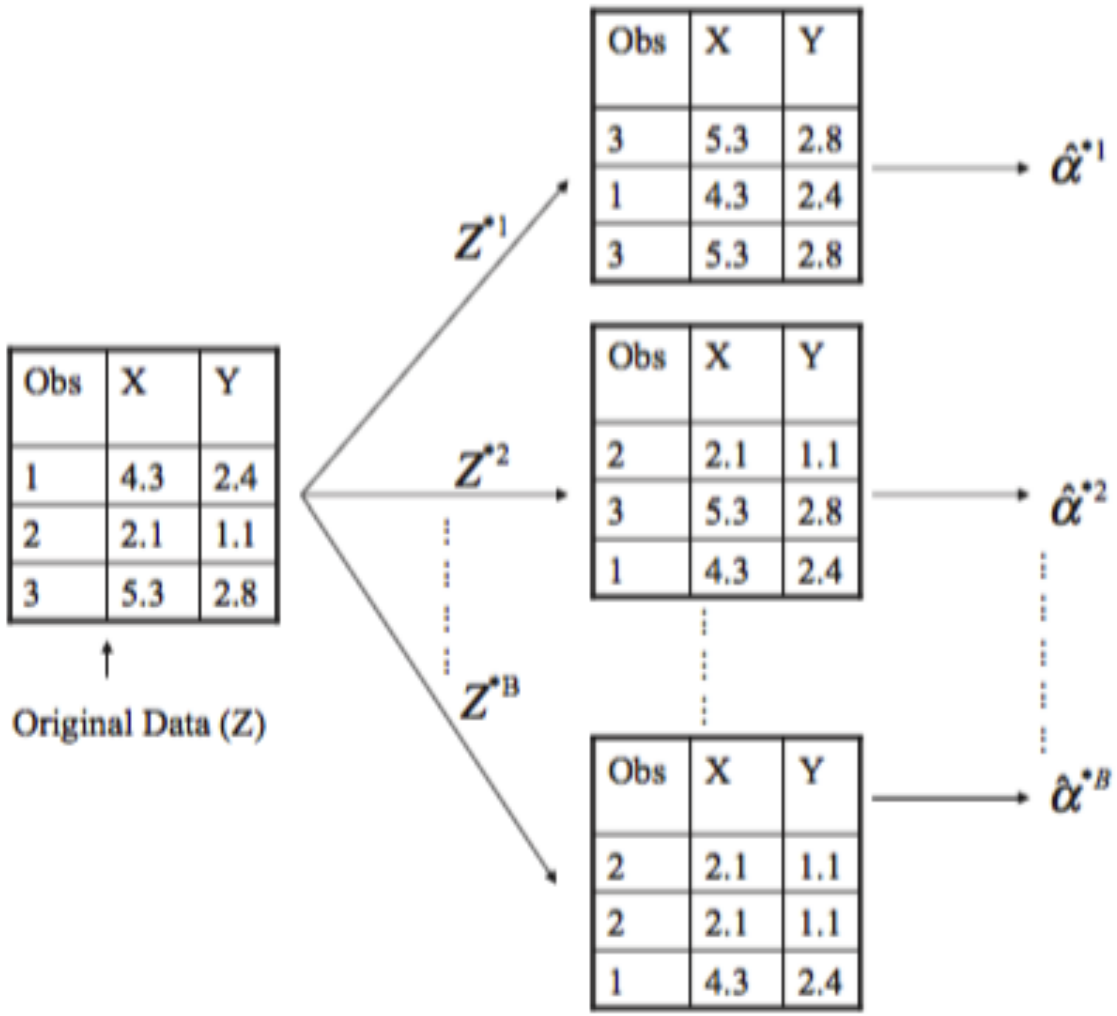




Outline

Bagging or bootstrap aggregation

We obtain distinct data sets by **repeatedly sampling** observations from the original data set with **replacement**.



What is Bagging? - Introduction

Decision trees can suffer from *high variance* (before prunning).

- **Bootstrap aggregation**, or **bagging**, is a general procedure for reducing the variance frequently used in the context of decision trees.

What is Bagging? - Introduction

Decision trees can suffer from *high variance* (before prunning).

- **Bootstrap aggregation**, or **bagging**, is a general procedure for reducing the variance frequently used in the context of decision trees.

Reminder: Given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the empirical mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$.

So, Averaging a set of observations reduces variance!

What is Bagging? - Introduction

Decision trees can suffer from *high variance* (before prunning).

- **Bootstrap aggregation**, or **bagging**, is a general procedure for reducing the variance frequently used in the context of decision trees.

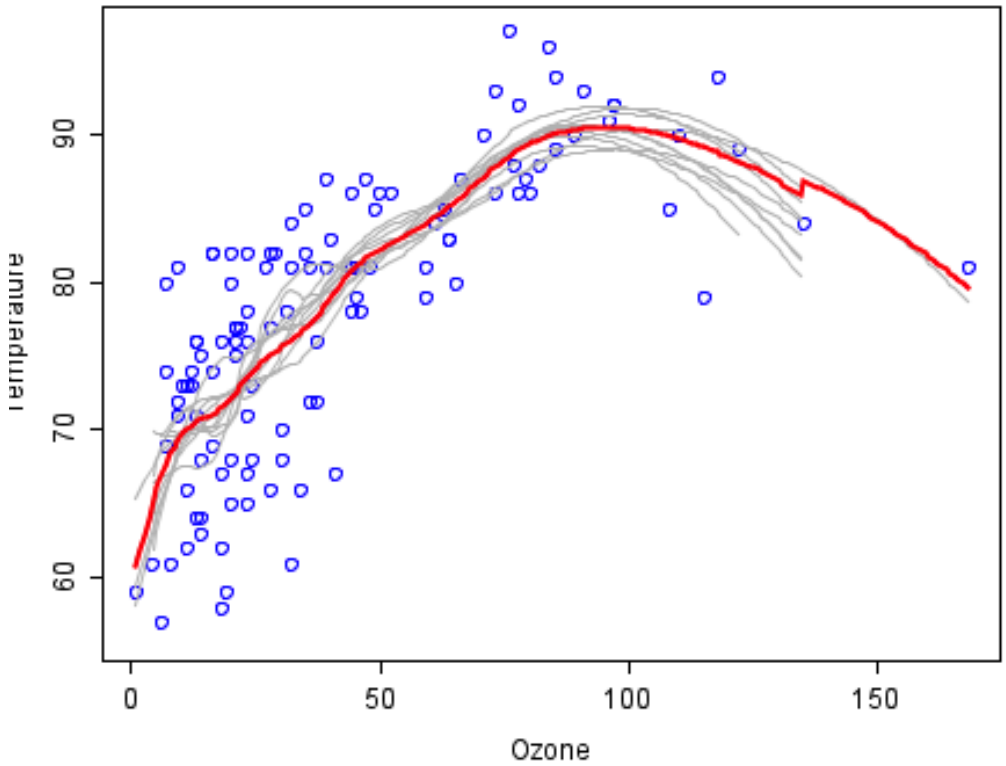
Reminder: Given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the empirical mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$.

So, Averaging a set of observations reduces variance!

However, this is not practical because we generally do not have access to multiple training sets.

Bagging Illustration

Example: relationship between ozone and temperature:
$B = 100$ models were fitted on bootstrap samples. (Gray) Predictions from 10 fitted models, (red) Average of the 100 fitted models.



Clearly average is more stable and there is less overfit!

Source: https://en.wikipedia.org/wiki/Bootstrap_aggregating

How Bagging proceeds?

Bagging proceeds as follows:

Step 1: Generate B different bootstrapped training data sets by taking samples from the original dataset.

How Bagging proceeds?

Bagging proceeds as follows:

Step 1: Generate B different bootstrapped training data sets by taking samples from the original dataset.

Step 2: Fit the method on the *bth* bootstrapped training set in order to get the prediction $\hat{f}^{*b}(x)$ for a given observation x.

How Bagging proceeds?

Bagging proceeds as follows:

Step 1: Generate B different bootstrapped training data sets by taking samples from the original dataset.

Step 2: Fit the method on the *bth* bootstrapped training set in order to get the prediction $\hat{f}^{*b}(x)$ for a given observation x.

remark: at this point each individual tree has high variance!

How Bagging proceeds?

Bagging proceeds as follows:

Step 1: Generate B different bootstrapped training data sets by taking samples from the original dataset.

Step 2: Fit the method on the bth bootstrapped training set in order to get the prediction $\hat{f}^{*b}(x)$ for a given observation x.

remark: at this point each individual tree has high variance!

Step 3: Then,

- for regression, **average** the B predictions.
- for classification, take a **majority vote**, the most commonly occurring class among the B predictions.

How Bagging proceeds?

Bagging proceeds as follows:

Step 1: Generate B different bootstrapped training data sets by taking samples from the original dataset.

Step 2: Fit the method on the bth bootstrapped training set in order to get the prediction $\hat{f}^{*b}(x)$ for a given observation x.

remark: at this point each individual tree has high variance!

Step 3: Then,

- for regression, **average** the B predictions.
- for classification, take a **majority vote**, the most commonly occurring class among the B predictions.

Averaging these B trees reduces the variance.

Out-of-Bag (OOB) Error Estimation

An straightforward way to estimate the test error in bagging.

- Trees are fit to bootstrapped subsets of the observations. So, on average, each bagged tree makes use of around $\frac{2}{3}$ of the observations. (we will see the proof in the tutorial course).
- The remaining $\frac{1}{3}$ of the observations are referred to as the **out-of-bag (OOB)** observations.

Out-of-Bag (OOB) Error Estimation

An straightforward way to estimate the test error in bagging.

- Trees are fit to bootstrapped subsets of the observations. So, on average, each bagged tree makes use of around $\frac{2}{3}$ of the observations. (we will see the proof in the tutorial course).
- The remaining $\frac{1}{3}$ of the observations are referred to as the **out-of-bag (OOB)** observations.
- It is possible to predict the response for the *ith* observation using each of the trees in which that observation was OOB. This will yield around $\frac{B}{3}$ predictions for each observation.
- Then, we can estimate the overall OOB MSE for each of the n observations. This will be the estimated test error!

Outline

Introduction to Random Forests (RF)

Random forests provides an improvement over bagged trees that **decorrelates the trees** by making a slightly modification. This reduces the variance when averaging the estimates.

Introduction to Random Forests (RF)

Random forests provides an improvement over bagged trees that **decorrelates the trees** by making a slightly modification. This reduces the variance when averaging the estimates.

- As in bagging, a number of decision trees are built on bootstrapped training samples.

Introduction to Random Forests (RF)

Random forests provides an improvement over bagged trees that **decorrelates the trees** by making a slightly modification. This reduces the variance when averaging the estimates.

- As in bagging, a number of decision trees are built on bootstrapped training samples.
- Modification: when building these decision trees, each time a split in a tree is considered, a **random selection of m predictors** is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

Introduction to Random Forests (RF)

Random forests provides an improvement over bagged trees that **decorrelates the trees** by making a slightly modification. This reduces the variance when averaging the estimates.

- As in bagging, a number of decision trees are built on bootstrapped training samples.
- Modification: when building these decision trees, each time a split in a tree is considered, a **random selection of m predictors** is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
  - Typical values for m are $\frac{p}{2}$ or $\sqrt{p}$ (for instance, if $p = 100$, choose among only 10 predictors).

Why Random Forests reduces variance?

At each split, the algorithm must consider only **a minority** of the predictors.

- *Idea*: suppose that there is one very strong predictor in the data. Then, most of the bagged trees will use this predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be **highly correlated**.

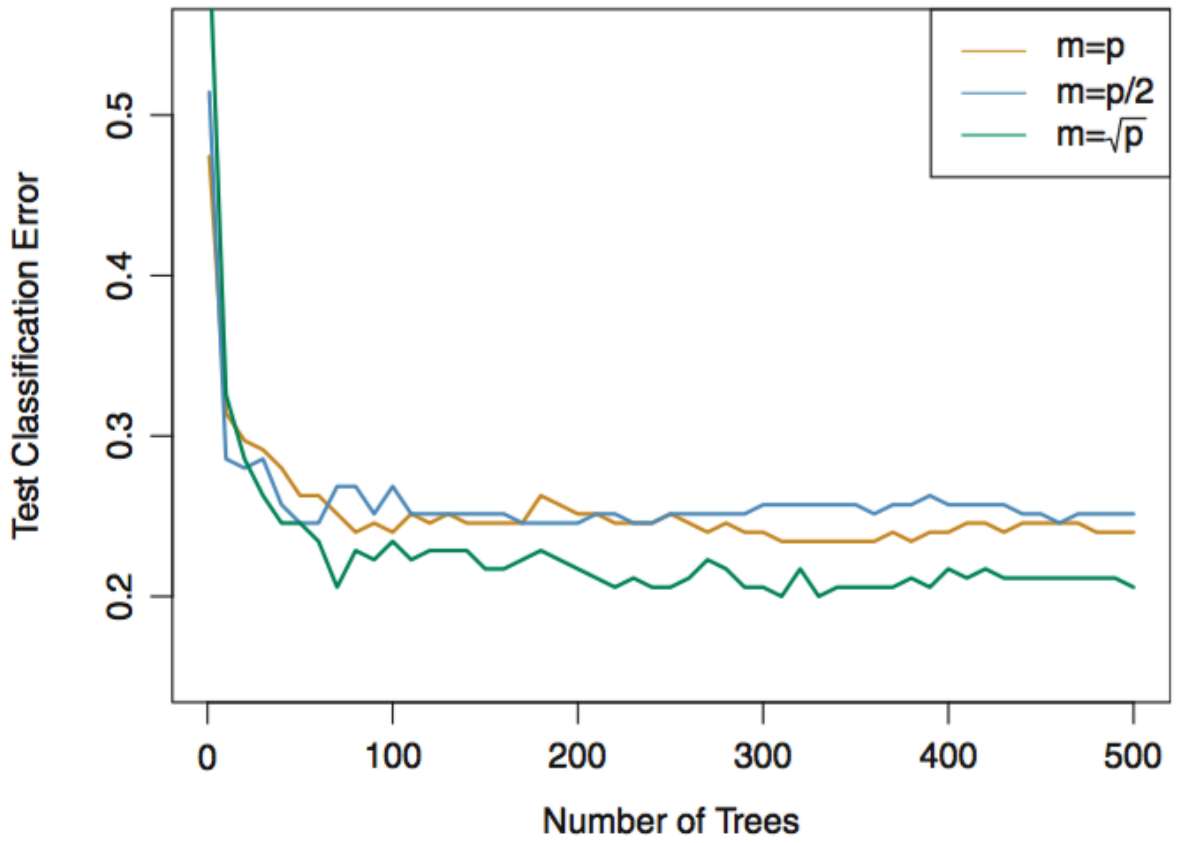
Unfortunately, averaging highly correlated quantities does not lead to as substantial reduction in variance as averaging uncorrelated quantities.

$$\textbf{Reminder: } V(\bar{Z}) = V\left(\frac{\sum_i Z_i}{n}\right) = \frac{\sigma_Z^2}{n} + 2\sum_{i \neq j} cov(Z_i, Z_j)$$

The term $cov(Z_i, Z_j) = 0$ only if the Z_i's are not correlated (independent).

The choice of m in Random Forest

Gene expression data: Performace of Random forests for different values of m.



Goal: predict cancer type based on 500 genes with high variance.
If $m = p$, this amounts simply to bagging.

Outline

Introduction to Boosting

Like bagging, **boosting** is a **ensemble learning** method.

Introduction to Boosting

Like bagging, **boosting** is a **ensemble learning** method.

- **Boosting** combines a set of weak learners into strong learners. A weak learner refers to a learning algorithm that only predicts slightly better than randomly.

There are different types of boosting algorithms :

Introduction to Boosting

Like bagging, **boosting** is a **ensemble learning** method.

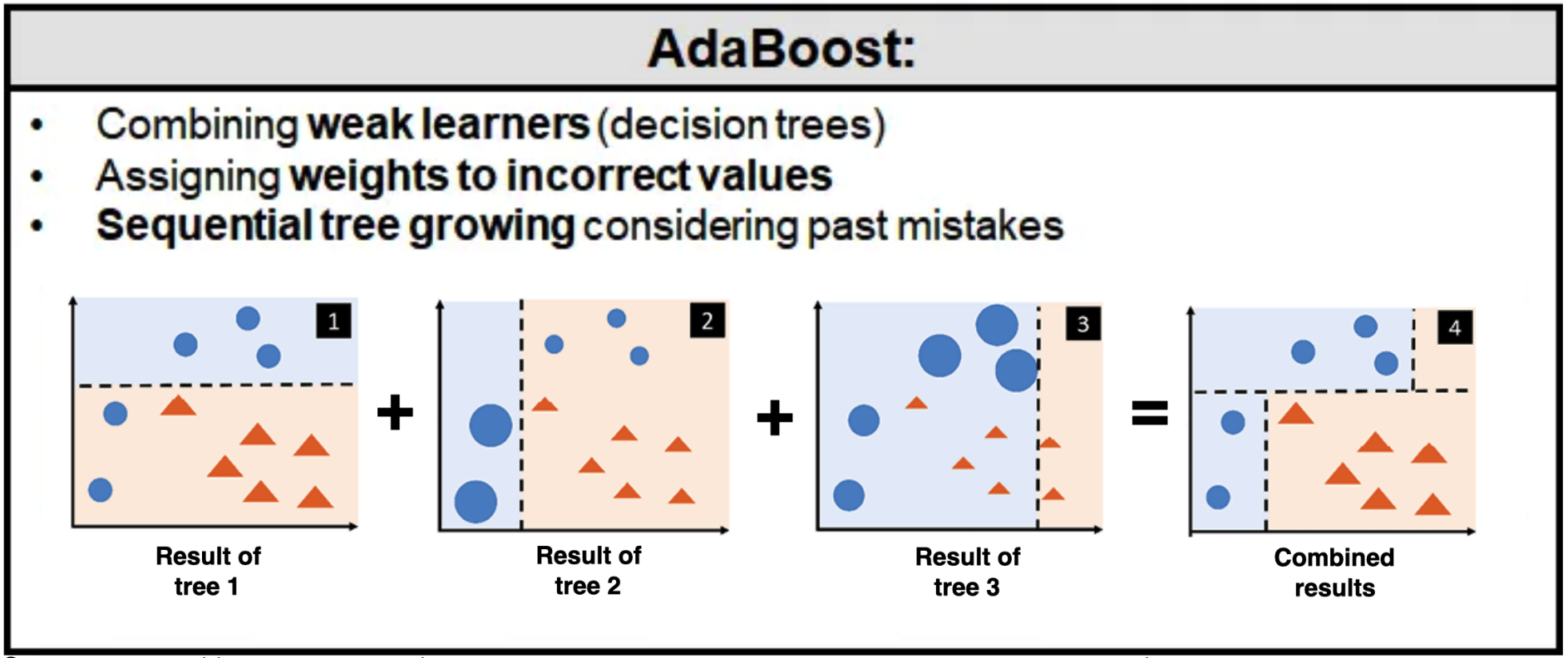
- **Boosting** combines a set of weak learners into strong learners. A weak learner refers to a learning algorithm that only predicts slightly better than randomly.

There are different types of boosting algorithms :

- AdaBoost (Adaptive Boosting)
- Gradient Boosting
- XGBoost (Extreme Gradient Boosting)

What is the idea behind boosting?

Intuition: Adaboost (Adaptative boosting)



Source: https://vitalflux.com/adaboost-algorithm-explained-with-python-example/, https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

The final prediction is the weighted majority vote of all weak learners

Gradient boosting and XGboost

- Gradient boosting trains learners based upon minimizing a loss function (i.e., training on the residuals of the model).

Gradient boosting and XGboost

- Gradient boosting trains learners based upon minimizing a loss function (i.e., training on the residuals of the model).
- Unlike fitting a single large decision tree to the data, in gradient boosting the learners are small decision trees grown slowly.

Gradient boosting and XGboost

- Gradient boosting trains learners based upon minimizing a loss function (i.e., training on the residuals of the model).
- Unlike fitting a single large decision tree to the data, in gradient boosting the learners are small decision trees grown slowly.
- At each step, a decision tree is fit to the **residuals** or **errors** of the current tree. Then, the residuals are updated. This implies to slowly improve in areas where the classifier does not perform well.

Gradient boosting and XGboost

- Gradient boosting trains learners based upon minimizing a loss function (i.e., training on the residuals of the model).
- Unlike fitting a single large decision tree to the data, in gradient boosting the learners are small decision trees grown slowly.
- At each step, a decision tree is fit to the **residuals** or **errors** of the current tree. Then, the residuals are updated. This implies to slowly improve in areas where the classifier does not perform well.

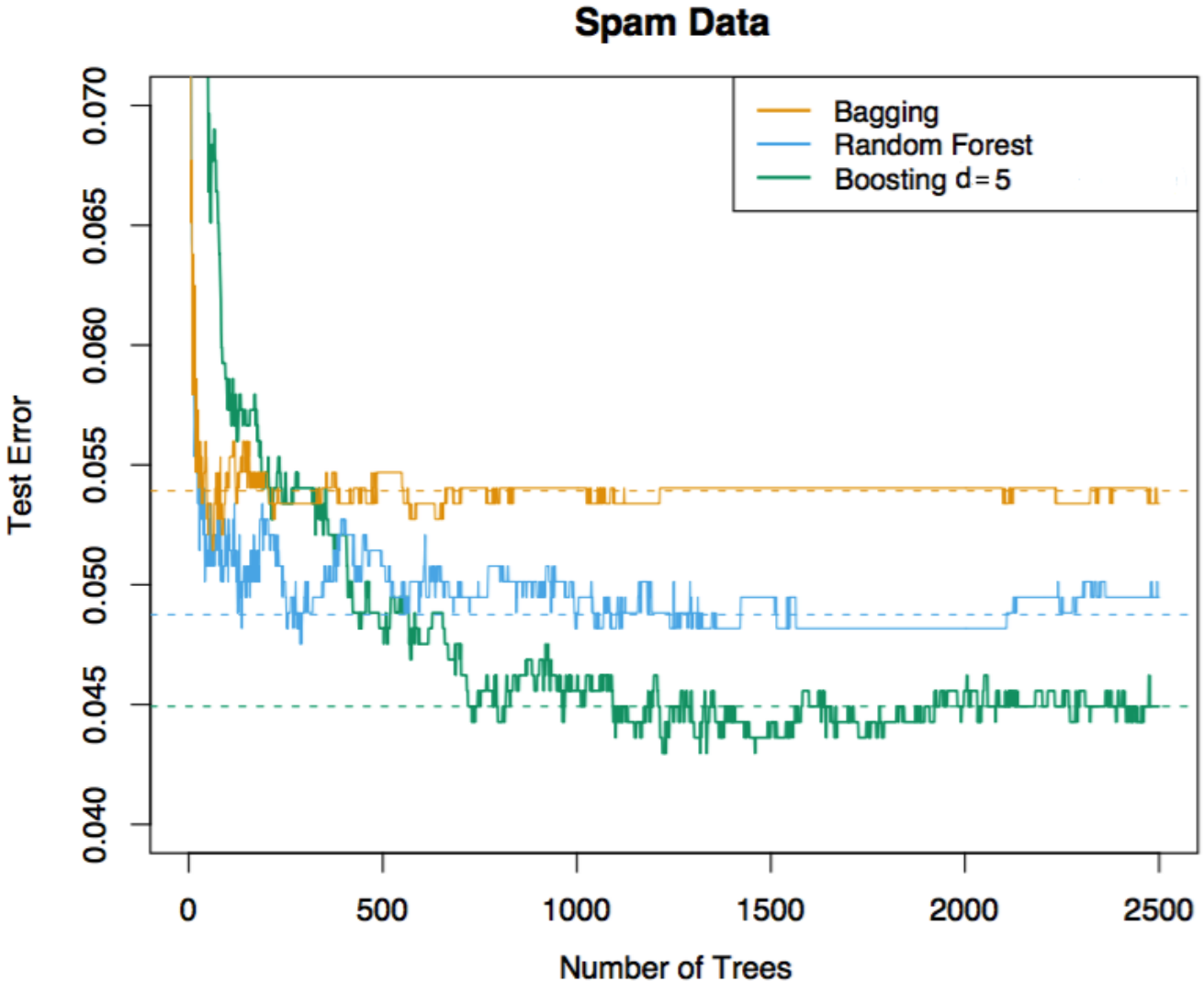
Tuning parameters:

- The **number of trees** B.
- The **depth** d or number of terminal nodes in each tree.
- A **shrinkage parameter** $\lambda > 0$ which controls the rate at which boosting learns and scales the contribution of each weak learner. Typical values are 0.01 or 0.001.

Outline

Comparison example of classification trees

Spam data contains 2-class target variable and 50 predictors.



Summary on Decision trees

- Decision trees are simple and interpretable models for regression and classification.
- However they are often not competitive with other methods in terms of prediction accuracy.

Summary on Decision trees

- Decision trees are simple and interpretable models for regression and classification.
- However they are often not competitive with other methods in terms of prediction accuracy.
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees at the expense of interpretability. They work by growing many trees on the training data and then **combining** the predictions of the resulting ensemble of trees.

Summary on Decision trees

- Decision trees are simple and interpretable models for regression and classification.
- However they are often not competitive with other methods in terms of prediction accuracy.
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees at the expense of interpretability. They work by growing many trees on the training data and then **combining** the predictions of the resulting ensemble of trees.
- The latter two methods - random forests and boosting - are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Outline

Introduction

A little of history:

Support Vector Machines, usually called simple **SVM**, was developed in the 1990s by Vladimir Vapnik.

Since the, *SVM*s have been shown to perform well in a variety of settings, and are often considered one of the best *out of the box* classifiers.

Introduction

A little of history:

Support Vector Machines, usually called simple **SVM**, was developed in the 1990s by Vladimir Vapnik.

Since the, *SVM*s have been shown to perform well in a variety of settings, and are often considered one of the best *out of the box* classifiers.

SVM principle for the two-class classification problem:

SVM principle

Finding a hyperplane that separates the classes in feature space.

What is a Hyperplane?

- A hyperplane in p dimensions is a flat affine subspace of dimension $p - 1$.

What is a Hyperplane?

- A hyperplane in p dimensions is a flat affine subspace of dimension $p-1$.
- In general the equation for a hyperplane has the form:

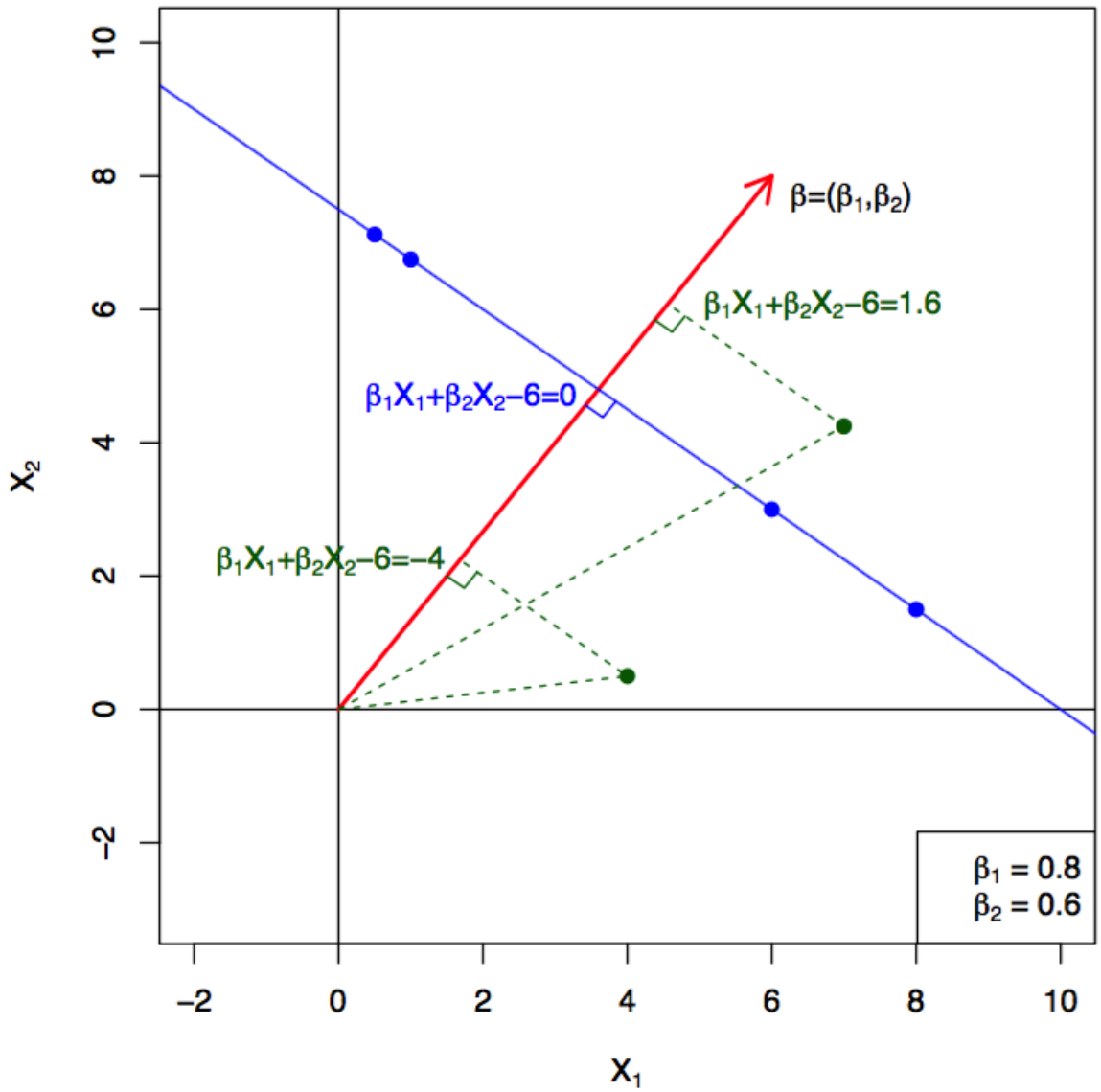
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

- If $p = 2$ dimensions a hyperplane is a line of equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- If $\beta_0 = 0$, the hyperplane goes through the origin.
- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is called the **normal vector** and it is orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions



A separating hyperplane

For any point $\mathbf{X} = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ in $p-$dimensional space, there are 3 possibilities:

1. $\mathbf{X}$ lies on the hyperplane then it satisfies:

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

A separating hyperplane

For any point $\mathbf{X} = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ in $p-$dimensional space, there are 3 possibilities:

1. $\mathbf{X}$ lies on the hyperplane then it satisfies:
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$
2. $\mathbf{X}$ does not satisfy this equation and rather,
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$
Then $\mathbf{X}$ lies on one side of the hyperplane.

A separating hyperplane

For any point $\mathbf{X} = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ in $p-$dimensional space, there are 3 possibilities:

1. $\mathbf{X}$ lies on the hyperplane then it satisfies:
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$
2. $\mathbf{X}$ does not satisfy this equation and rather,
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$
Then $\mathbf{X}$ lies on one side of the hyperplane.
3. On the other hand, if
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$
then $\mathbf{X}$ lies on the other side of the hyperplane.

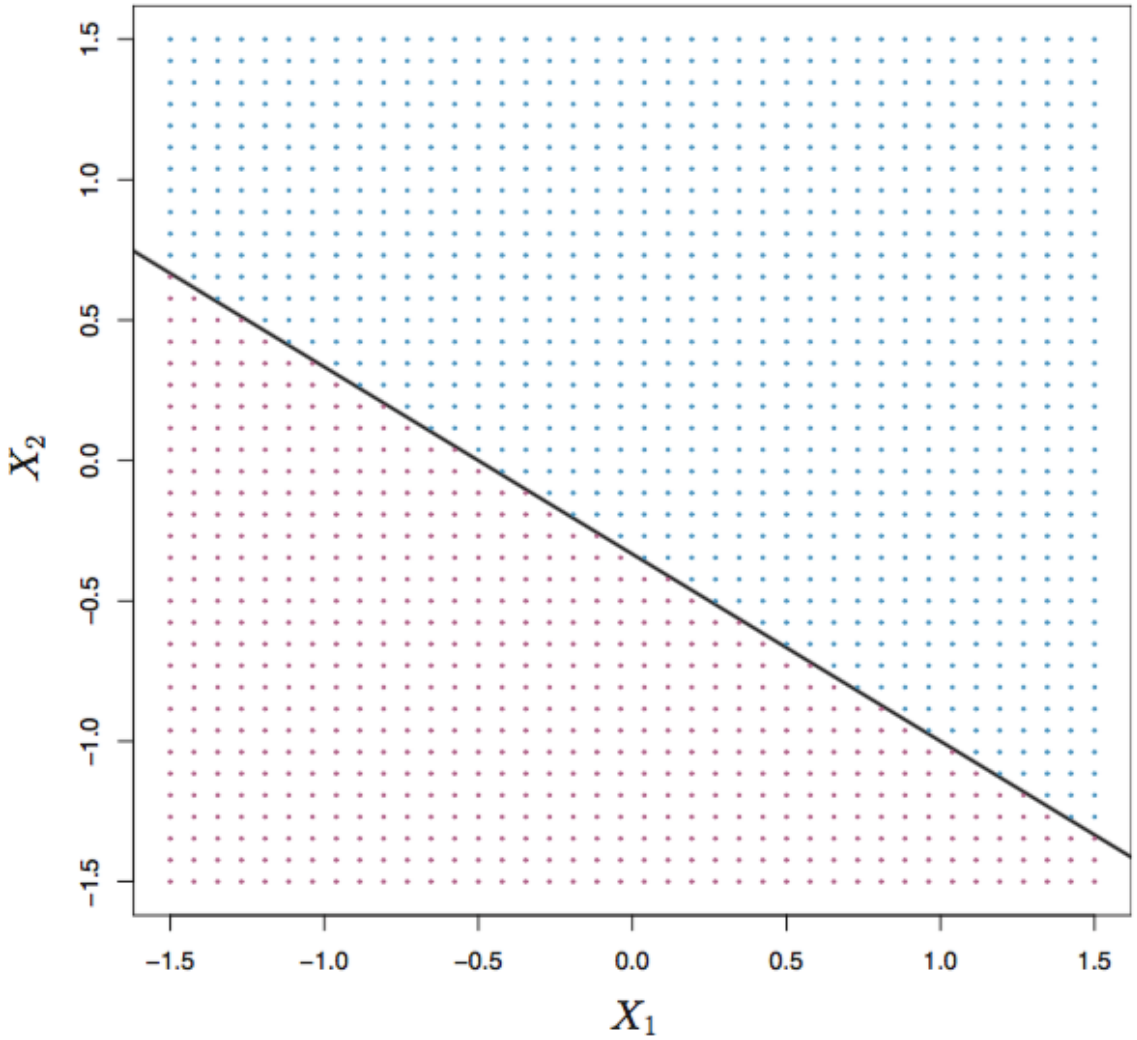
A separating hyperplane

For any point $\mathbf{X} = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ in $p-$dimensional space, there are 3 possibilities:

1. $\mathbf{X}$ lies on the hyperplane then it satisfies:
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$
2. $\mathbf{X}$ does not satisfy this equation and rather,
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$
Then $\mathbf{X}$ lies on one side of the hyperplane.
3. On the other hand, if
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$
then $\mathbf{X}$ lies on the other side of the hyperplane.

So, a hyperplane divides a p-dimensional space into **two halves**.

An example of a separating hyperplane in $\mathbb{R}^2$



The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown.

- Blue region: set of points for which $1 + 2X_1 + 3X_2 > 0$,
- Red region: set of points for which $1 + 2X_1 + 3X_2 < 0$.

Classification Using a Separating Hyperplane

Now suppose a $n \times p$ data matrix that consists of n training observations in $p-$dimensional space

- 1st observation: $x_1 = (x_{11}, \ldots x_{1p})$
- 2nd observation: $x_2 = (x_{21}, \ldots x_{2p})$

$$\vdots$$

- nth observation: $x_n = (x_{n1}, \ldots x_{np})$

These observations fall into two classes, $y_1, \ldots, y_n \in \{-1, 1\}$

Classification Using a Separating Hyperplane

Now suppose a $n \times p$ data matrix that consists of n training observations in $p-$dimensional space

- 1st observation: $x_1 = (x_{11}, \ldots x_{1p})$
- 2nd observation: $x_2 = (x_{21}, \ldots x_{2p})$

$$\vdots$$

- nth observation: $x_n = (x_{n1}, \ldots x_{np})$

These observations fall into two classes, $y_1, \ldots, y_n \in \{-1, 1\}$

Consider a test observation, $x^* = \left(x_1^*, \ldots x_p^*\right)$.

Classification Using a Separating Hyperplane

Now suppose a $n \times p$ data matrix that consists of n training observations in $p-$dimensional space

- 1st observation: $x_1 = (x_{11}, \ldots x_{1p})$
- 2nd observation: $x_2 = (x_{21}, \ldots x_{2p})$

$$\vdots$$

- nth observation: $x_n = (x_{n1}, \ldots x_{np})$

These observations fall into two classes, $y_1, \ldots, y_n \in \{-1, 1\}$

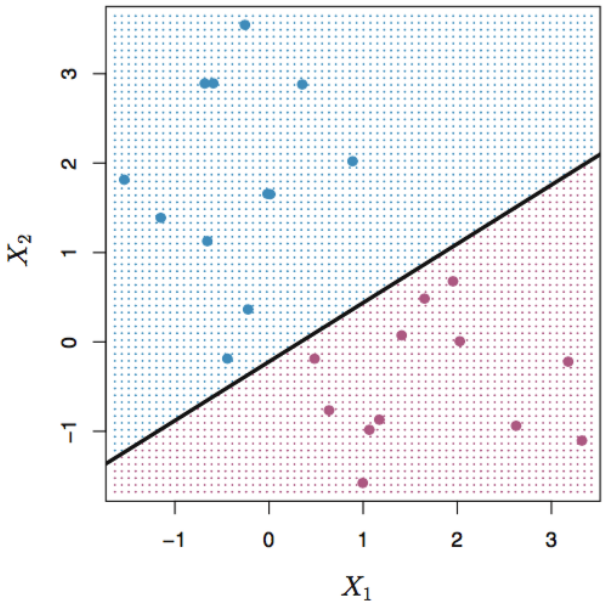
Consider a test observation, $x^* = \left(x_1^*, \ldots x_p^*\right)$.

Goal: develop a classifier based on the training data that correctly classifies the test observation.

idea: build a **separating hyperplane**.

How to classify using a separating hyperplan?

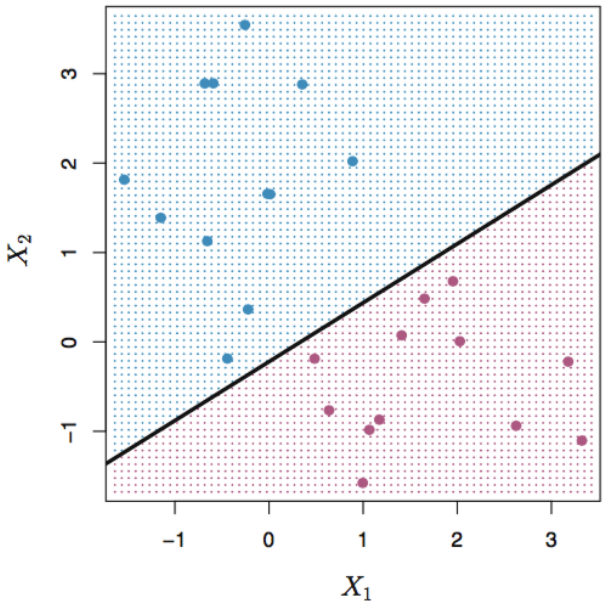
Suppose there exists a hyperplane that perfectly separates the two classes in the training observations:



By coding:
$y_i = +1$ for blue and $y_i = -1$ for red class.
Then, a separating hyperplane has the property:
$y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_p x_{ip}) > 0 \, \forall i = 1, \ldots, n$

How to classify using a separating hyperplan?

Suppose there exists a hyperplane that perfectly separates the two classes in the training observations:



By coding:
$y_i = +1$ for blue and $y_i = -1$ for red class.
Then, a separating hyperplane has the property:
$y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_p x_{ip}) > 0 \, \forall i = 1, \ldots, n$

Given a test observation x^*, classify it based on the **sign** of:
$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^*$
if $f(x^*) > 0$ then blue , if $f(x^*) < 0$ then red.

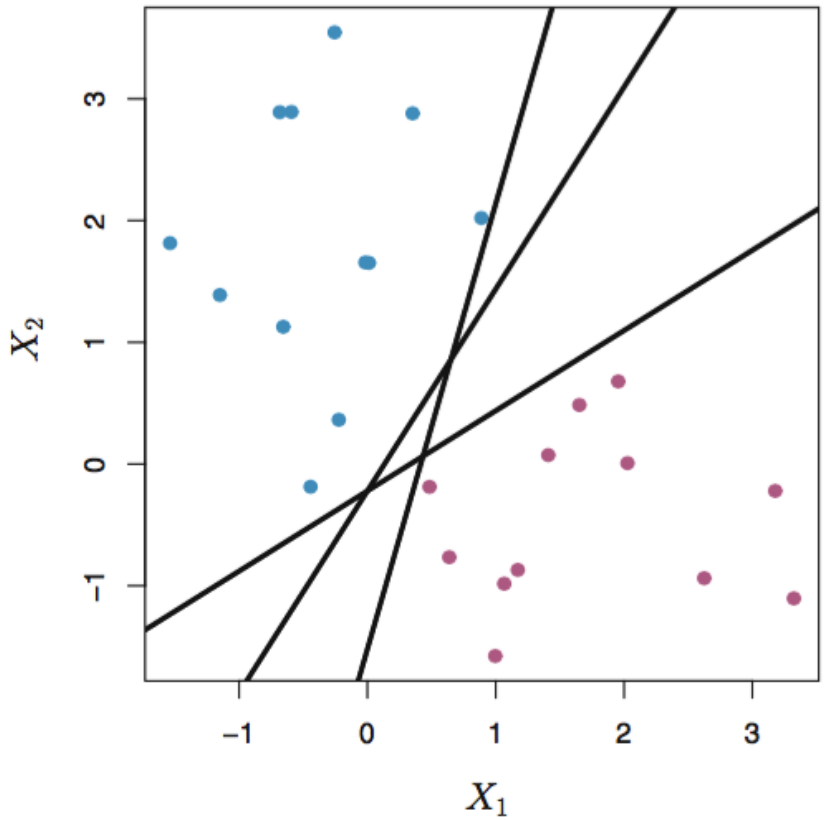
$f(x^*)$ can be interpreted as **magnitude of confidence**.

- If $f(x^*)$ is far from zero $\Rightarrow$ confident about its class assignment.
- if $f(x^*)$ is close to zero $\Rightarrow$ less confident about its class assignment.

Outline

What separating hyperplan to choose?

If a perfect separating hyperplan exists, then there exist an infinite number of such hyperplanes.



Which one to choose?

The maximal margin hyperplane

A natural choice is the **maximal margin hyperplane**, also known as the **optimal separating hyperplane**, which is the separating hyperplane that is farthest from the training observations.

The maximal margin hyperplane

A natural choice is the **maximal margin hyperplane**, also known as the **optimal separating hyperplane**, which is the separating hyperplane that is farthest from the training observations.

Compute the distance from each training observation to a given separating hyperplane. The smallest such distance is the minimal distance among all the observations to the hyperplane, and is known as the **margin**.

The maximal margin hyperplane

A natural choice is the **maximal margin hyperplane**, also known as the **optimal separating hyperplane**, which is the separating hyperplane that is farthest from the training observations.

Compute the distance from each training observation to a given separating hyperplane. The smallest such distance is the minimal distance among all the observations to the hyperplane, and is known as the **margin**.

The **maximal margin hyperplane** is the separating hyperplane for which the margin is the largest

The maximal margin hyperplane

A natural choice is the **maximal margin hyperplane**, also known as the **optimal separating hyperplane**, which is the separating hyperplane that is farthest from the training observations.

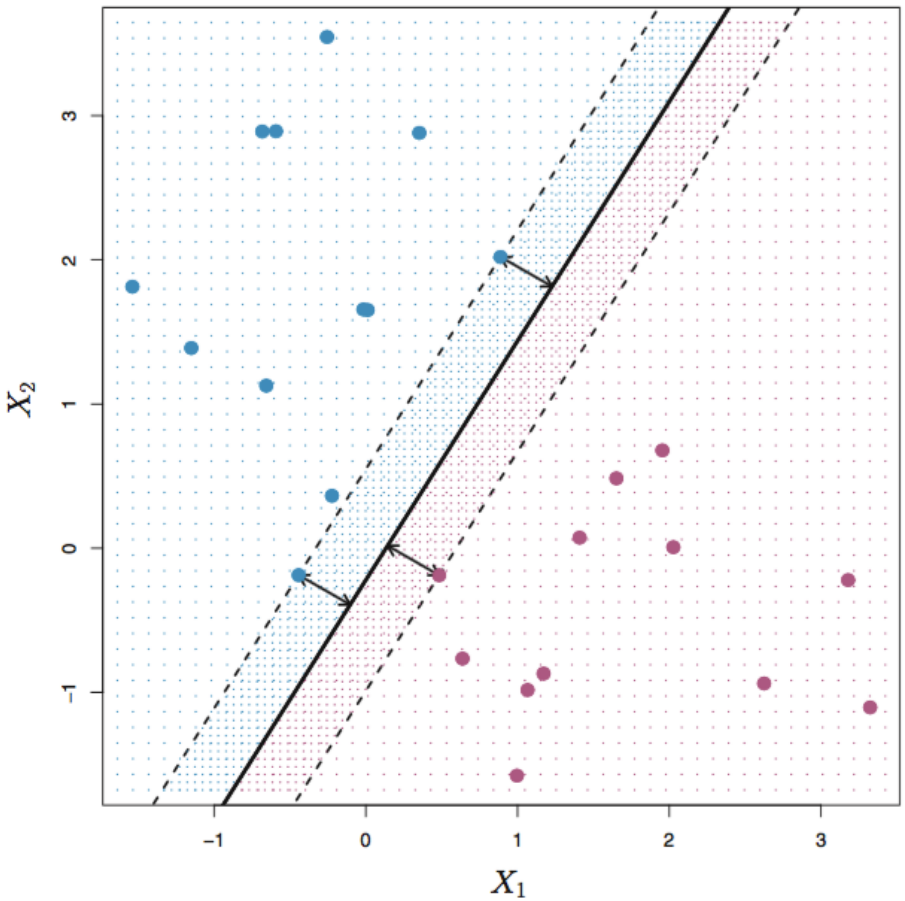
Compute the distance from each training observation to a given separating hyperplane. The smallest such distance is the minimal distance among all the observations to the hyperplane, and is known as the **margin**.

The **maximal margin hyperplane** is the separating hyperplane for which the margin is the largest

Then, classify a test observation based on which side of the maximal margin hyperplane it lies. This is known as the **maximal margin classifier**.

Maximal Margin Classifier

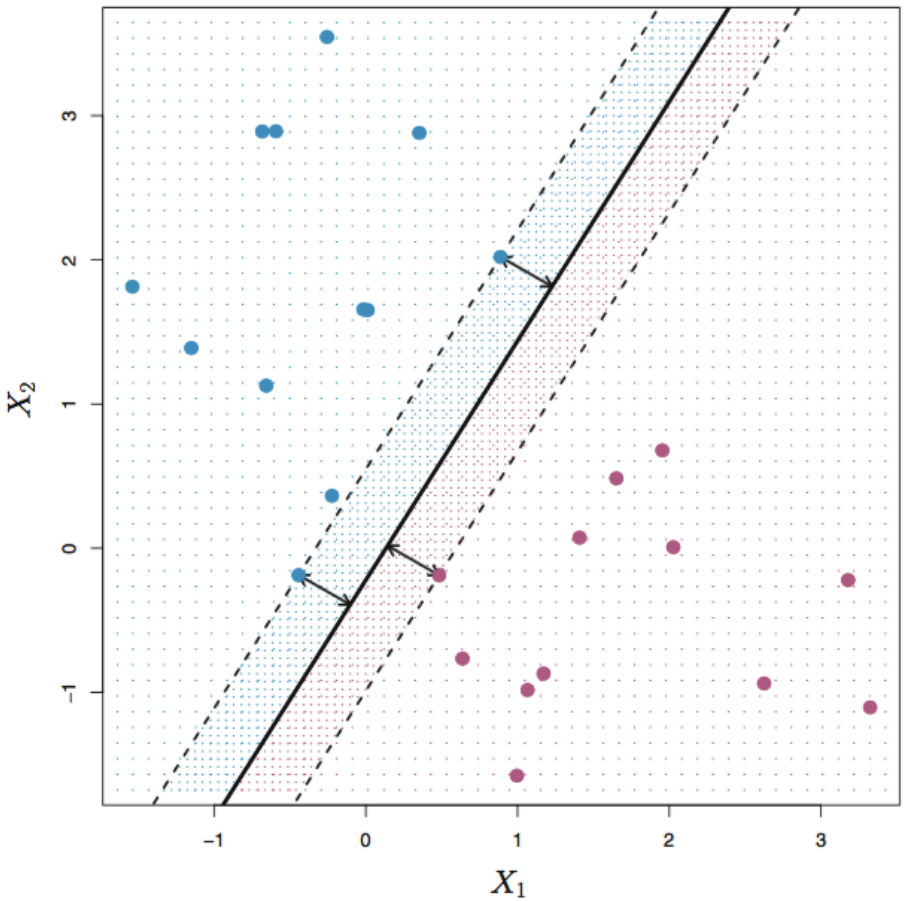
Example of Maximal margin hyperplane



- the maximal margin hyperplane represents the midline of the widest **slab** that can be inserted between the two classes.

Maximal Margin Classifier

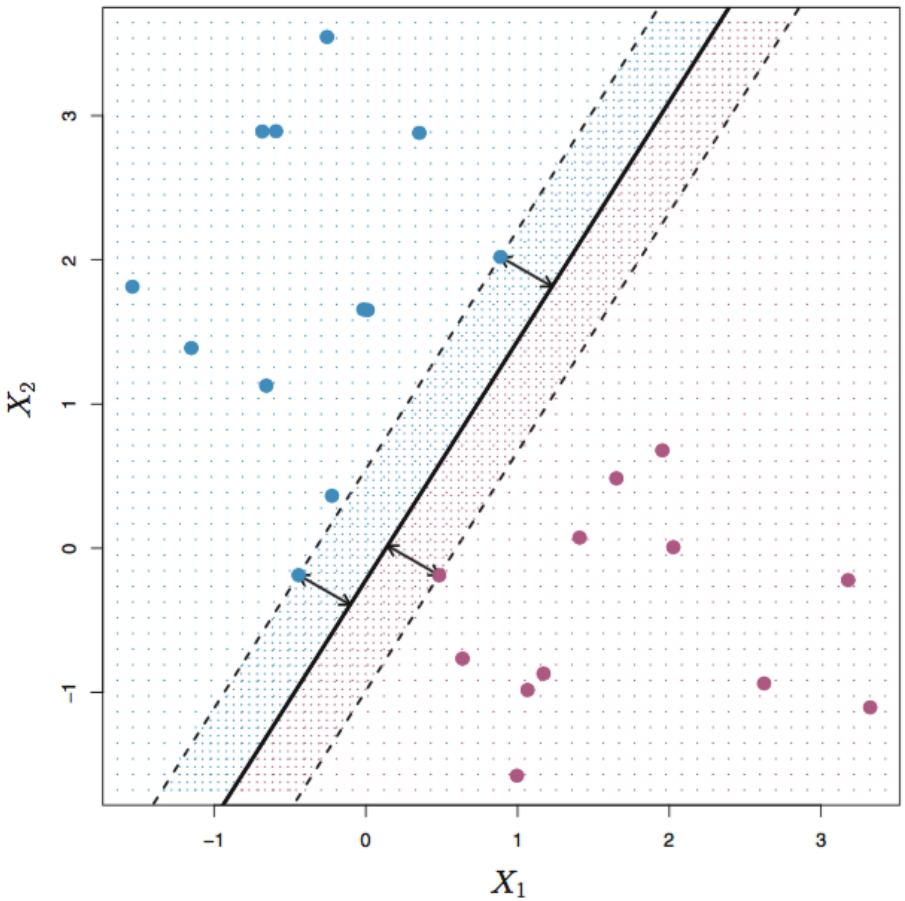
Example of Maximal margin hyperplane



- the maximal margin hyperplane represents the midline of the widest **slab** that can be inserted between the two classes.
- The dashed lines indicate the width of the margin.

Maximal Margin Classifier

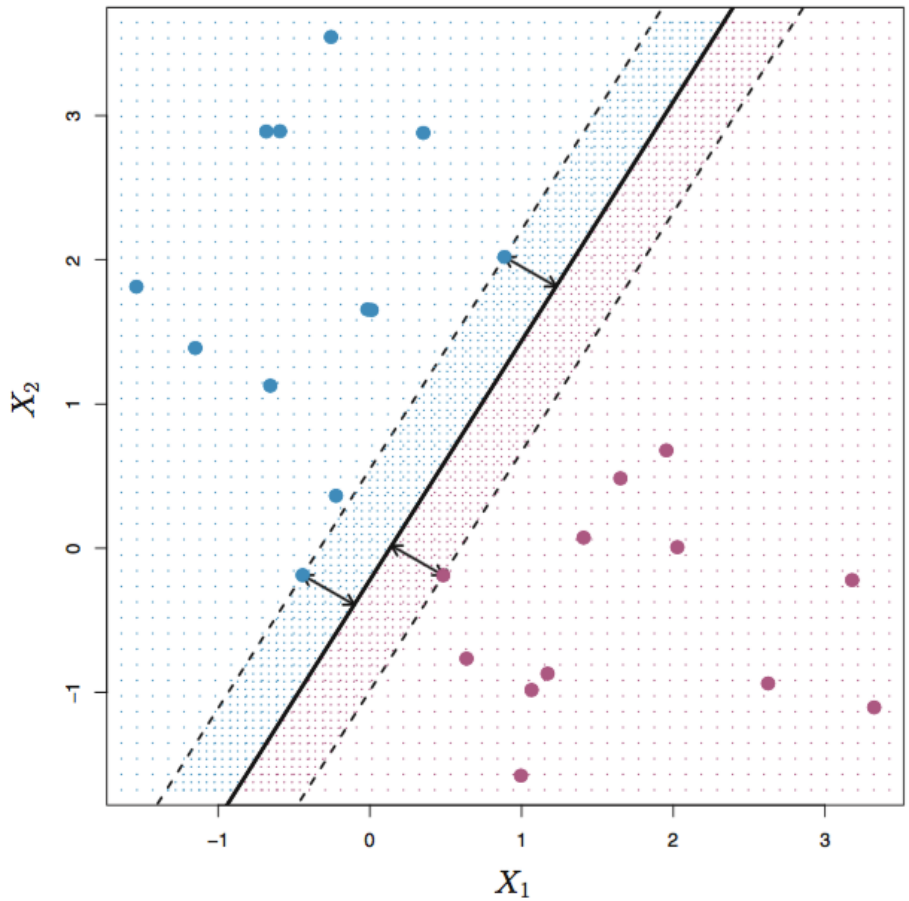
Example of Maximal margin hyperplane



- the maximal margin hyperplane represents the midline of the widest **slab** that can be inserted between the two classes.
- The dashed lines indicate the width of the margin.
- the three training observations equidistant from the maximal margin hyperplane that lie along the dashed lines are known as **support vectors**.

Maximal Margin Classifier

Example of Maximal margin hyperplane



- the maximal margin hyperplane represents the midline of the widest **slab** that can be inserted between the two classes.
- The dashed lines indicate the width of the margin.
- the three training observations equidistant from the maximal margin hyperplane that lie along the dashed lines are known as **support vectors**.

The maximal margin hyperplane depends only on the support vectors!

Construction of the Maximal Margin Classifier

Given n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. The maximal margin hyperplane is the solution to the optimization problem:

$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{maximize}} \; M$$

$$\text{subject to} : \sum_{j=1}^{p} \beta_j^2 = 1 \qquad \text{and}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.$$

- The second constrant guarantees that each observation will be on the correct side of the hyperplane (provided that M is positive.).

Construction of the Maximal Margin Classifier

Given n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. The maximal margin hyperplane is the solution to the optimization problem:

$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{maximize}} \; M$$

$$\text{subject to} : \sum_{j=1}^{p} \beta_j^2 = 1 \qquad \text{and}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.$$

- The second constrant guarantees that each observation will be on the correct side of the hyperplane (provided that M is positive.).
- M represents the margin of the hyperplane.

Construction of the Maximal Margin Classifier

Given n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. The maximal margin hyperplane is the solution to the optimization problem:

$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{maximize}} \; M$$

$$\text{subject to} : \sum_{j=1}^{p} \beta_j^2 = 1 \qquad \text{and}$$

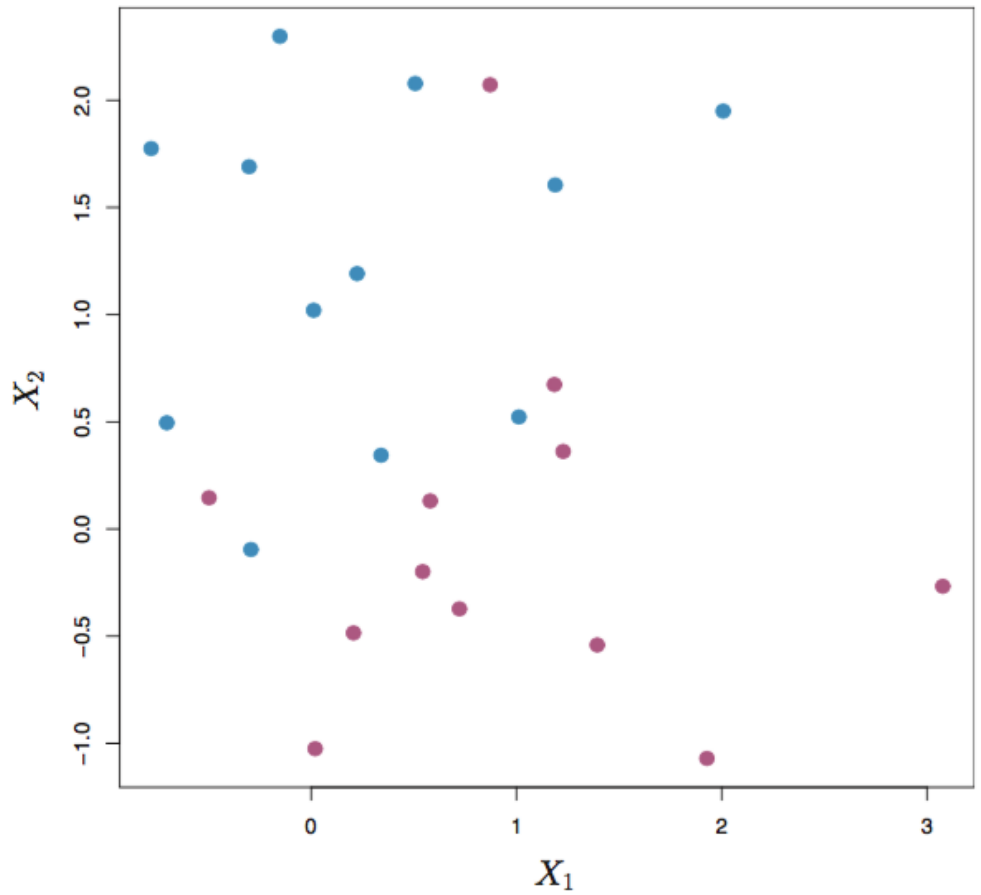
$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.$$

- The second constrant guarantees that each observation will be on the correct side of the hyperplane (provided that M is positive.).
- M represents the margin of the hyperplane.
- If the first constrant holds, the distance from the *ith* observation to the hyperplane is: $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$.

Situations when the Maximal Margin classifier fails (1/2)

The Non-separable Case

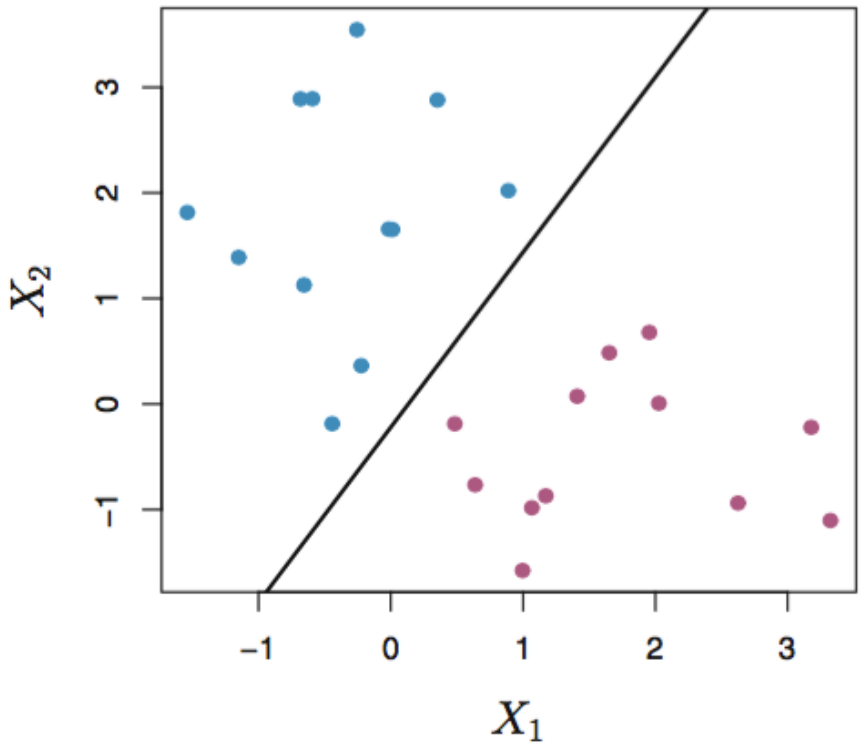
In many real life situations the two classes are unseparable

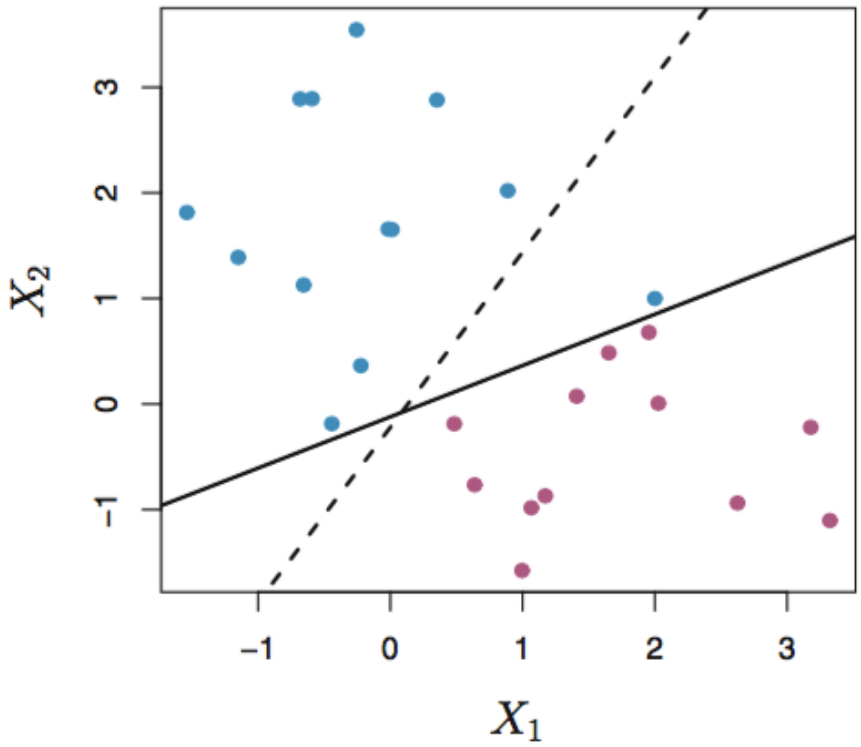


- The maximal margin hyperplane does not exist.
- In this case, the optimization problem has no solution with $M > 0$.

Situations when the Maximal Margin classifier fails: (2/2)

Noisy data and sensitivity to individual observations





The addition of a single observation leads to a dramatic change in the maximal margin hyperplane.
This extremely sensitive suggests overfitting. The margin is smaller!

Outline

Support Vector Classifier - Introduction

Solution: Consider a hyperplane that does **not** perfectly separate the two classes, in the interest of

- Greater robustness to individual observations, and
- Better classification of *most* of the training observations.

Such a classifier is called **support vector classifier** or **soft margin classifier**.

Support Vector Classifier - Introduction

Solution: Consider a hyperplane that does **not** perfectly separate the two classes, in the interest of

- Greater robustness to individual observations, and
- Better classification of *most* of the training observations.

Such a classifier is called **support vector classifier** or **soft margin classifier**. This classifier allows some observations:

- to be not only on the wrong side of the margin but also
- to be on the wrong side of the hyperplane.

Support Vector Classifier - Introduction

Solution: Consider a hyperplane that does **not** perfectly separate the two classes, in the interest of

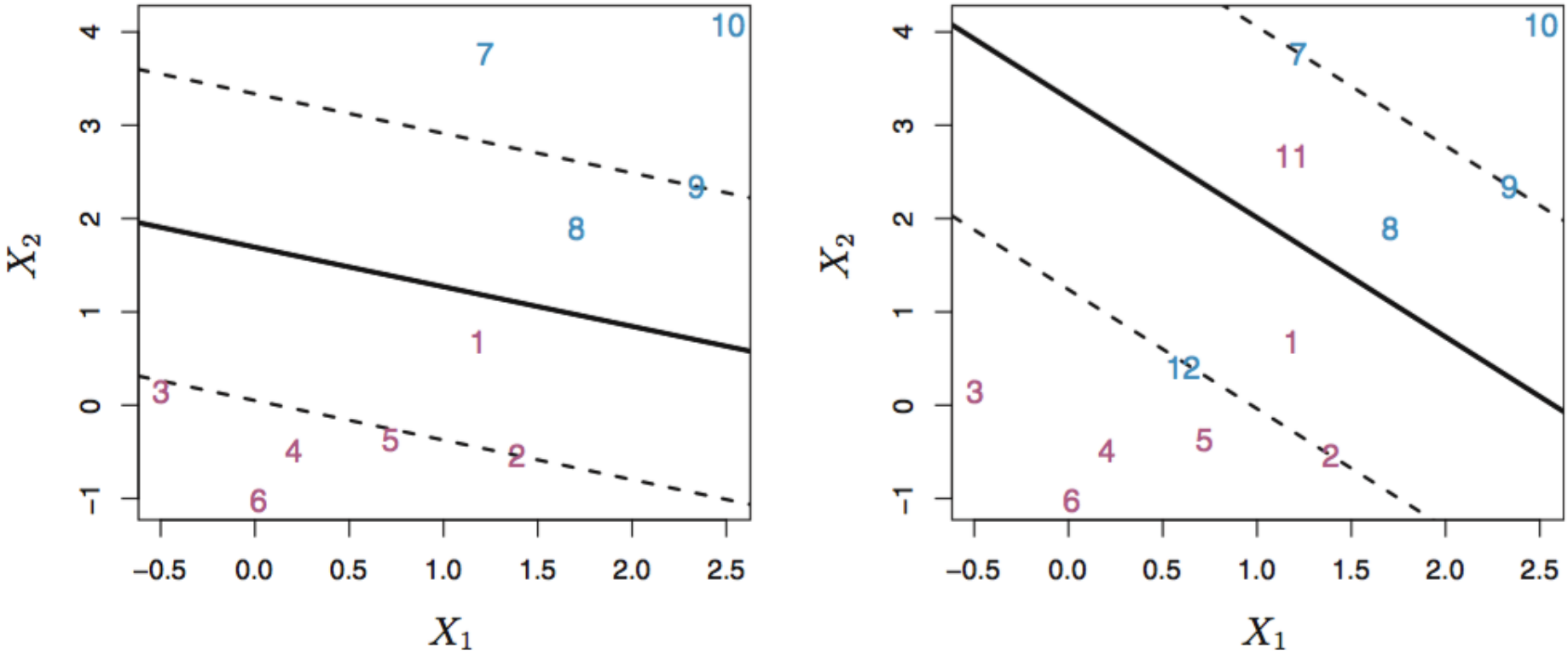
- Greater robustness to individual observations, and
- Better classification of *most* of the training observations.

Such a classifier is called **support vector classifier** or **soft margin classifier**. This classifier allows some observations:

- to be not only on the wrong side of the margin but also
- to be on the wrong side of the hyperplane.

Observations on the wrong side of the hyperplane correspond to misclassified training observations.

Support vector classifier relaxation



Left: Red class: 1 is on the wrong side of the margin. Blue class: observation 8 is on the wrong side of the margin.
Right: Same as left panel with two additional points, 11 and 12. Both are on the wrong side of the hyperplane and the wrong side of the margin.

Construction of the Support Vector Classifier

The **Support Vector Classifier** is the solution to the problem:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} : \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C \qquad \forall i = 1, \ldots, n.$$

- C is a nonnegative tuning parameter, M is the width of the margin,
- $\epsilon_1, \ldots, \epsilon_n$ are **slack variables** that allow individual observations to be on the wrong side of the margin or of the hyperplane.

Parameters' interpretation

- The **slack variable** ϵ_i tells where the *ith* observation is located:
  - If $\epsilon_i = 0$ then observation i is on the correct side of the margin.
  - If $0 < \epsilon_i < 1$ then observation i is on the wrong side of the margin,
  - If $\epsilon_i > 1$, then observation i is on the wrong side of the hyperplane.

Parameters' interpretation

- The **slack variable** ϵ_i tells where the *ith* observation is located:
  - If $\epsilon_i = 0$ then observation i is on the correct side of the margin.
  - If $0 < \epsilon_i < 1$ then observation i is on the wrong side of the margin,
  - If $\epsilon_i > 1$, then observation i is on the wrong side of the hyperplane.

- The **tuning parameter** C bounds $\sum_{i=1}^{n} \epsilon_i$, it represents a **budget** for the amount that the margin can be violated by the n observations.
  - If $C = 0$ then $\epsilon_1 = \ldots = \epsilon_n = 0$, then we obtain the maximal margin classifier problem.

Parameters' interpretation

- The **slack variable** ϵ_i tells where the *ith* observation is located:
  - If $\epsilon_i = 0$ then observation i is on the correct side of the margin.
  - If $0 < \epsilon_i < 1$ then observation i is on the wrong side of the margin,
  - If $\epsilon_i > 1$, then observation i is on the wrong side of the hyperplane.

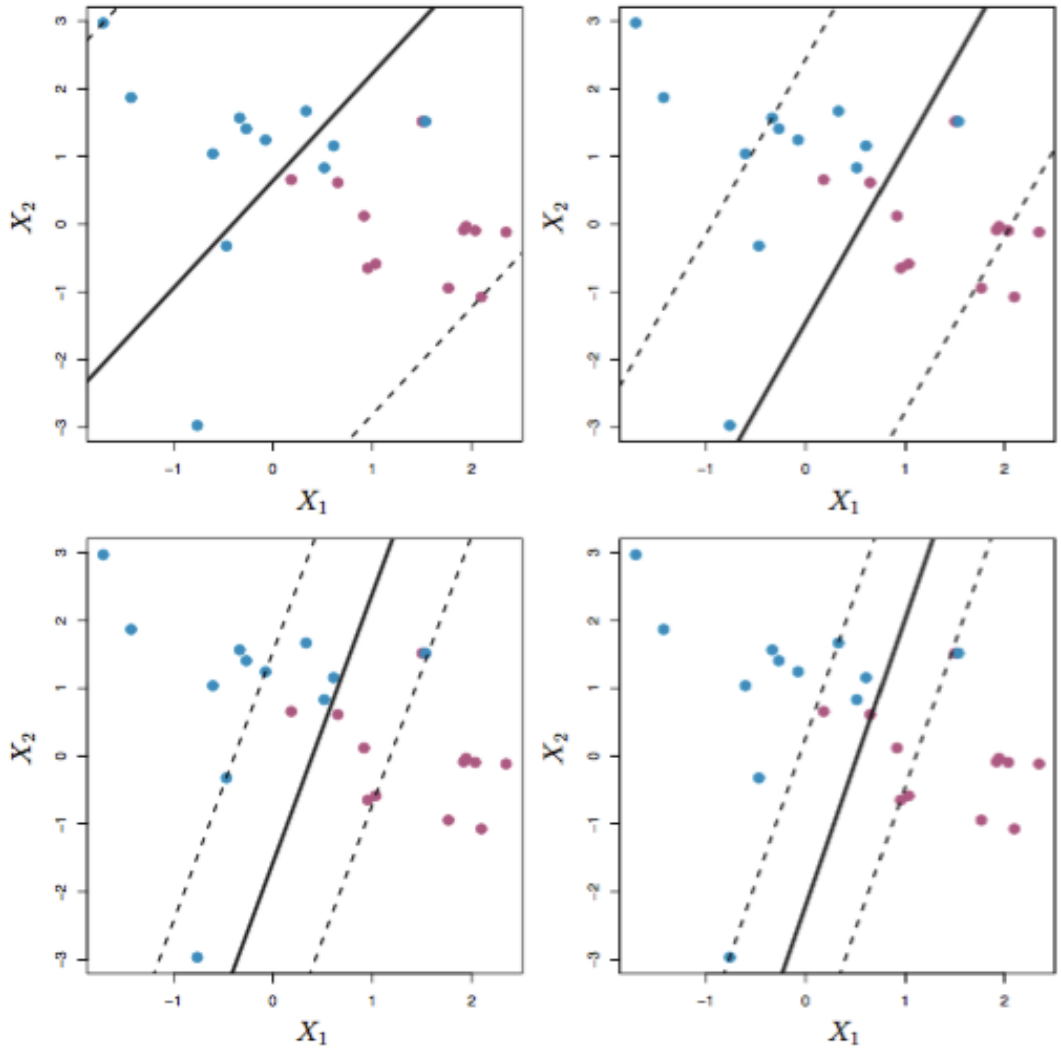
- The **tuning parameter** C bounds $\sum_{i=1}^{n} \epsilon_i$, it represents a **budget** for the amount that the margin can be violated by the n observations.
  - If $C = 0$ then $\epsilon_1 = \ldots = \epsilon_n = 0$, then we obtain the maximal margin classifier problem.
  - If $C > 0$ no more than C observations can be on the wrong side of the hyperplane,

Parameters' interpretation

- The **slack variable** ϵ_i tells where the *ith* observation is located:
  - If $\epsilon_i = 0$ then observation i is on the correct side of the margin.
  - If $0 < \epsilon_i < 1$ then observation i is on the wrong side of the margin,
  - If $\epsilon_i > 1$, then observation i is on the wrong side of the hyperplane.

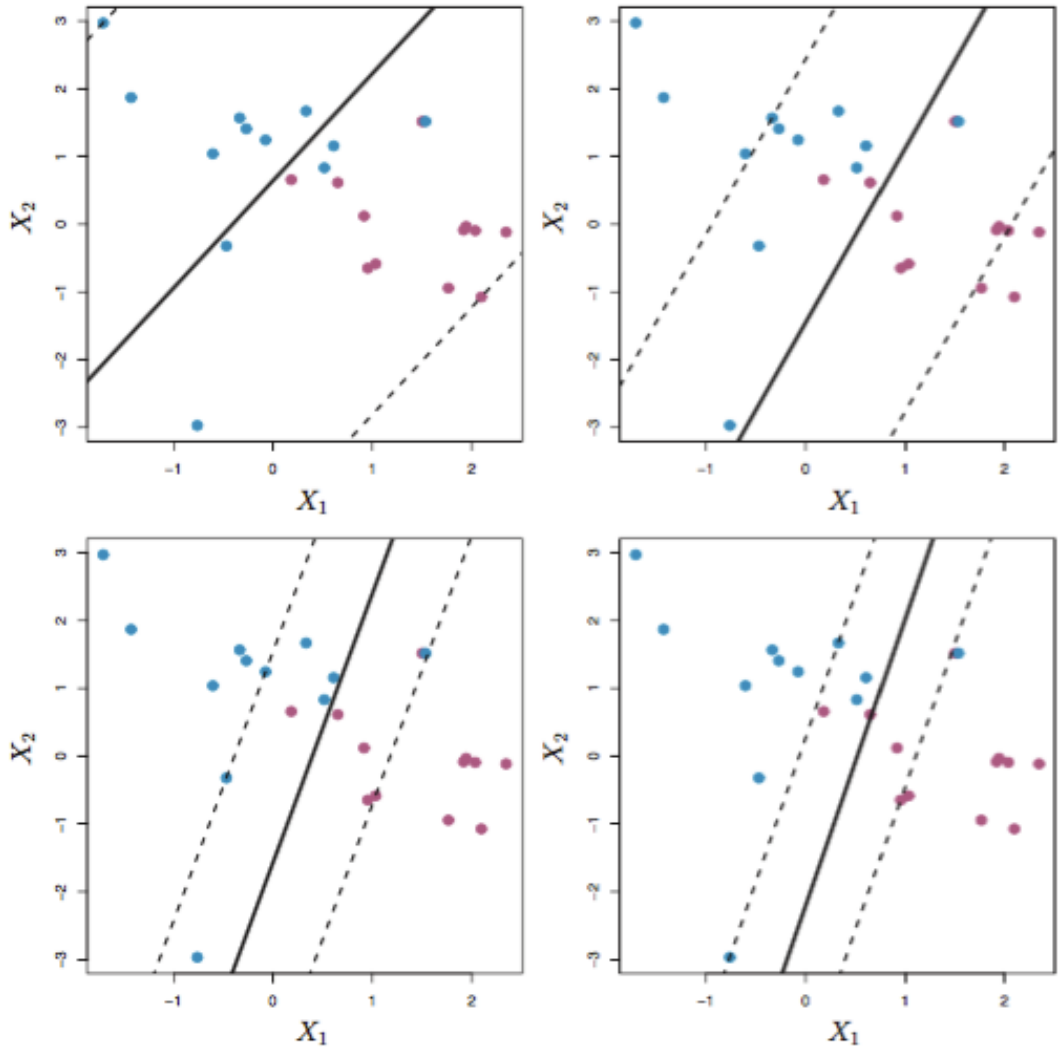
- The **tuning parameter** C bounds $\sum_{i=1}^{n} \epsilon_i$, it represents a **budget** for the amount that the margin can be violated by the n observations.
  - If $C = 0$ then $\epsilon_1 = \ldots = \epsilon_n = 0$, then we obtain the maximal margin classifier problem.
  - If $C > 0$ no more than C observations can be on the wrong side of the hyperplane,
  - As C increases, tolerance increases and so the margin will widen.
  - Conversely, as C decreases and the margin will narrow.

The regularization parameter C



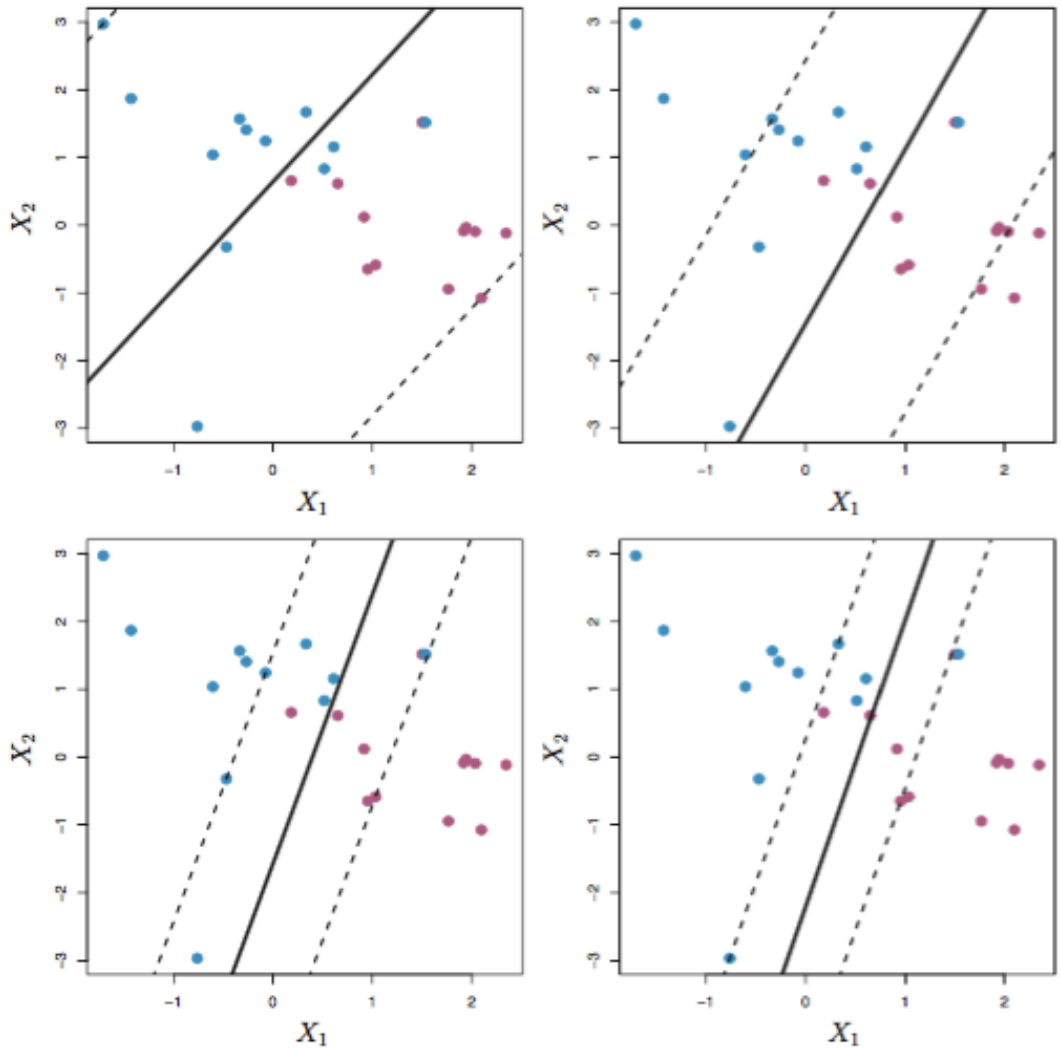
- C is generally chosen via cross-validation.
- C controls the bias-variance trade-off.
- If C small, then narrow margins rarely violated $\Rightarrow$ low bias but high variance.

The regularization parameter C



- C is generally chosen via cross-validation.
- C controls the bias-variance trade-off.
- If C small, then narrow margins rarely violated $\Rightarrow$ low bias but high variance.
- If C large, then wide margins and more violations are allowed $\Rightarrow$ classifier more biased but may have lower variance.

The regularization parameter C

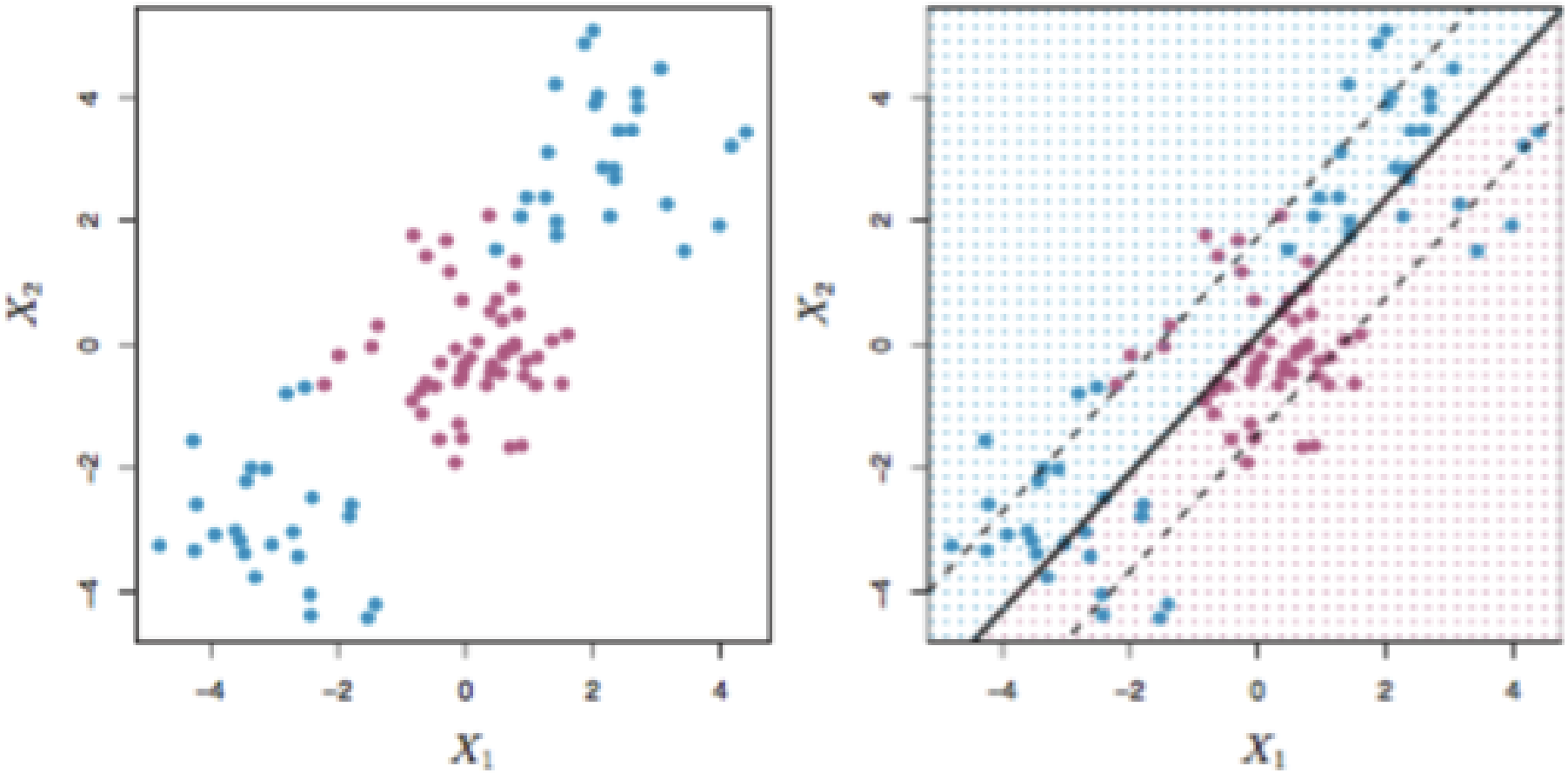


- C is generally chosen via cross-validation.
- C controls the bias-variance trade-off.
- If C small, then narrow margins rarely violated $\Rightarrow$ low bias but high variance.
- If C large, then wide margins and more violations are allowed $\Rightarrow$ classifier more biased but may have lower variance.

Only observations that either lie directly on the margin or that violate the margin will affect the hyperplane. These are the **support vectors**.

Outline

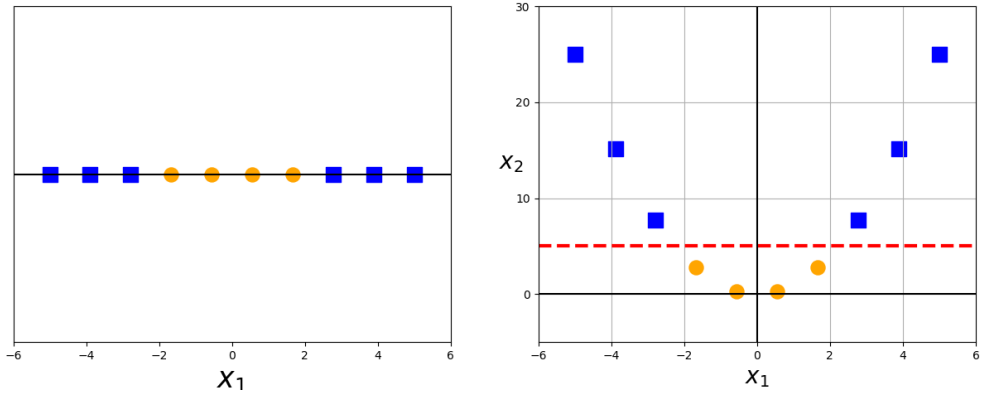
Classification with Non-linear Decision Boundaries



If the decision boundary is not linear, the Support Vector Classifier performs poorly!

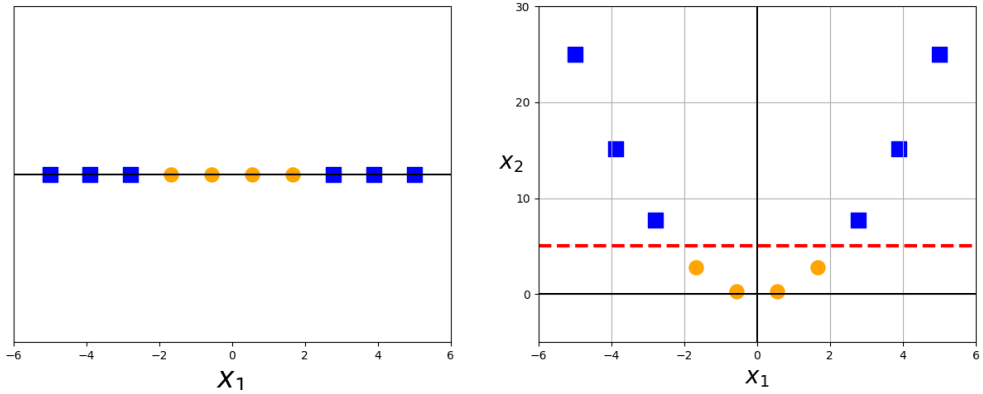
Feature expansion

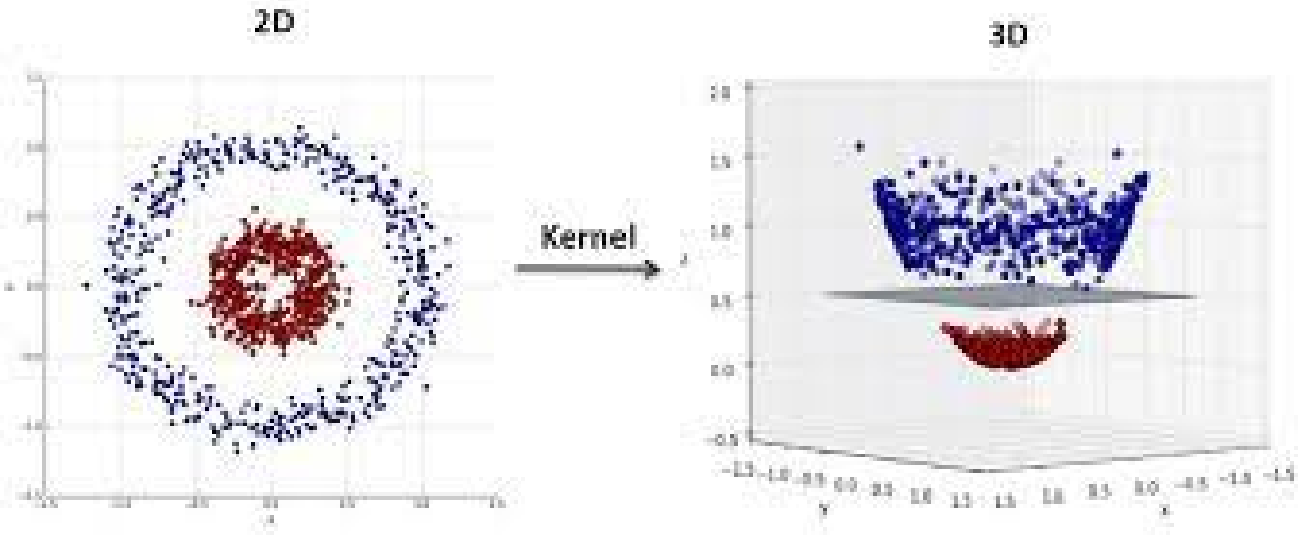
In a higher dimensional space the data becomes linearly separable.



Feature expansion

In a higher dimensional space the data becomes linearly separable.





Source: `https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f`

Feature Expansion

- In order to address non-linearity, consider enlarging the feature space by including transformations of the predictors e.g. X_1^2, X_1^3, $X_1 X_2$, $X_1 X_2^2$ (quadratic, cubic or even higher-order polynomial terms).

Feature Expansion

- In order to address non-linearity, consider enlarging the feature space by including transformations of the predictors e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$ (quadratic, cubic or even higher-order polynomial terms).
- This implies to go from a p-dimensional space to a $P > p$ dimensional space.
- Then, we can fit a support-vector classifier in the enlarged space and get non-linear decision boundaries in the original space.

Feature Expansion

- In order to address non-linearity, consider enlarging the feature space by including transformations of the predictors e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$ (quadratic, cubic or even higher-order polynomial terms).
- This implies to go from a p-dimensional space to a $P > p$ dimensional space.
- Then, we can fit a support-vector classifier in the enlarged space and get non-linear decision boundaries in the original space.
- Example: Suppose we consider $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2). Then the decision boundary would be of the form:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

As we enlarge the feature space computations become unmanageable!
Support vector machine allows to enlarge the feature space while keeping efficient computations.

Inner products and Support Vectors

It can be shown that the linear support vector classifier can be written as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

where $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{j=1}^{p} a_j b_j$ is the **inner product** between vectors **a** and **b** $\in \mathbb{R}^p$.

There are n parameter α_i, one per training obsevation.

Inner products and Support Vectors

It can be shown that the linear support vector classifier can be written as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

where $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{j=1}^{p} a_j b_j$ is the **inner product** between vectors **a** and **b** $\in \mathbb{R}^p$.
There are n parameter α_i, one per training obsevation.

To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0, we need $\binom{n}{2}$ inner products between all pairs of training observations.
However, it turns out that α_i is nonzero only for the **support vectors**:

$$f(X) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

where $\mathcal{S}$ is the the collection of indices of these support points.

Kernels and Support Vector Machines

Now, consider replacing the inner product in the support vector classifier optimization function with a **generalization** of the form: $K(x_i, x_{i'})$.

where K is refered to as a **kernel**. A kernel is a function that quantifies the similarity of two observations. For instance:

- **Linear kernel**: $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$, that is the kernel for support vector classifier.

Kernels and Support Vector Machines

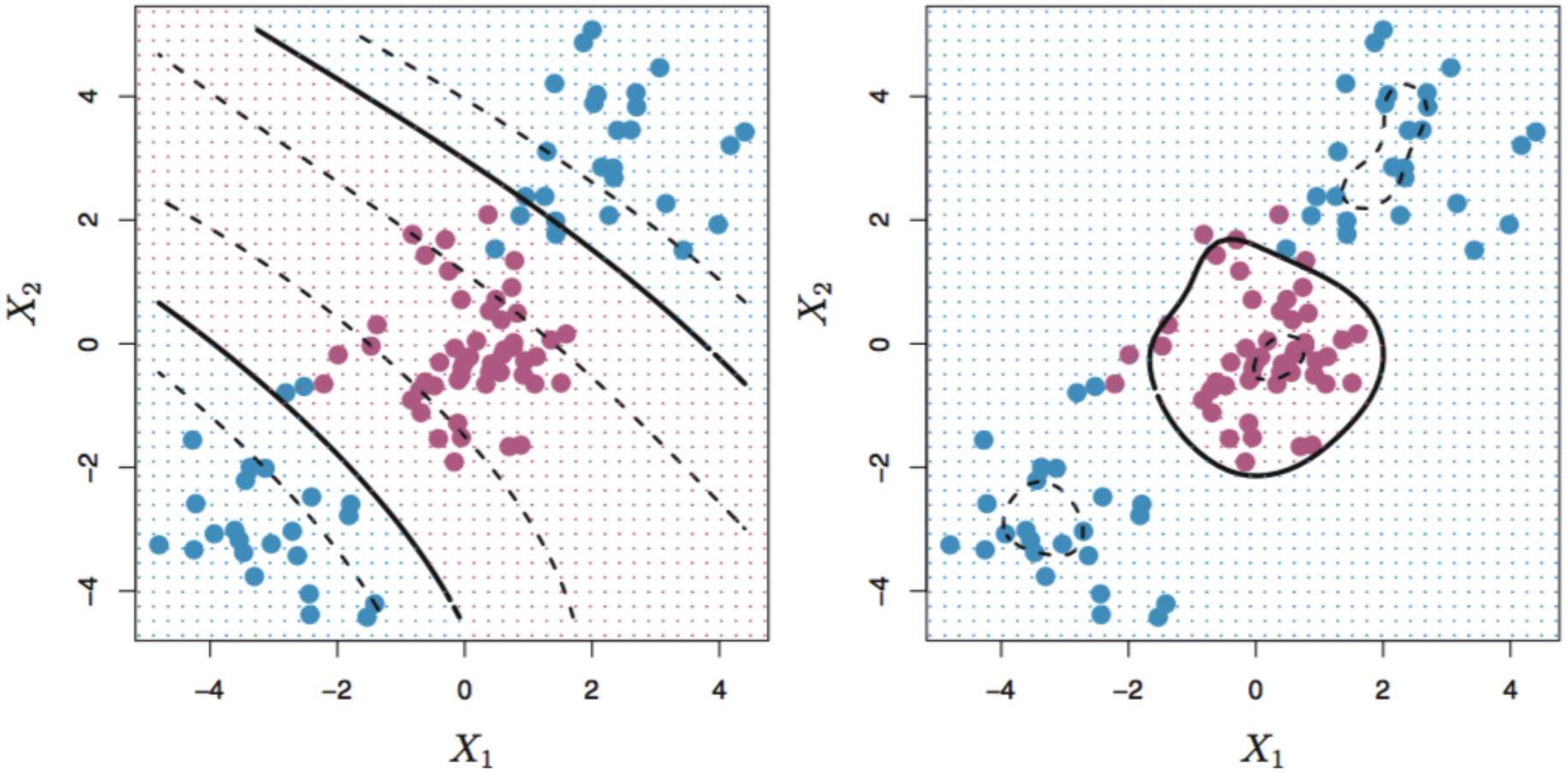
Now, consider replacing the inner product in the support vector classifier optimization function with a **generalization** of the form: $K(x_i, x_{i'})$.

where K is refered to as a **kernel**. A kernel is a function that quantifies the similarity of two observations. For instance:

- **Linear kernel**: $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$, that is the kernel for support vector classifier.
- **polynomial kernel**: $K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$ with $d > 0$ integer.
- **Radial kernel**: $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$ where γ is a positive constant.

The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space using **kernels**.

Example of *SVM* with polynomial and radial kernels



Left: SVM with a polynomial kernel; Right: SVM with a radial kernel.

What is the advantage of using a kernel?

The most important advantage is **computational**.

What is the advantage of using a kernel?

The most important advantage is **computational**.

Indeed, using kernels, one needs only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs i, i'. This allows to operate in the original feature space without computing the coordinates of the data in a higher dimensional space. This is known as the **kernel trick**.

What is the advantage of using a kernel?

The most important advantage is **computational**.

Indeed, using kernels, one needs only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs i, i'. This allows to operate in the original feature space without computing the coordinates of the data in a higher dimensional space. This is known as the **kernel trick**.

In many applications, the enlarged feature space is so large that computations are intractable.

What is the advantage of using a kernel?

The most important advantage is **computational**.

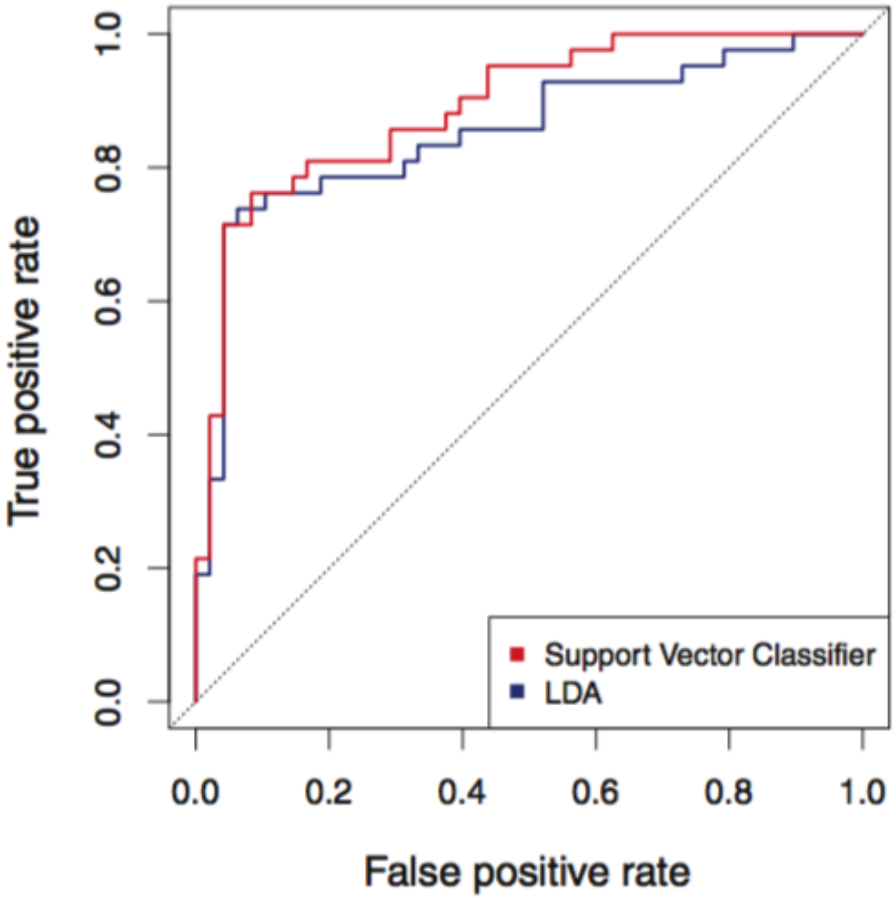
Indeed, using kernels, one needs only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs i, i'. This allows to operate in the original feature space without computing the coordinates of the data in a higher dimensional space. This is known as the **kernel trick**.

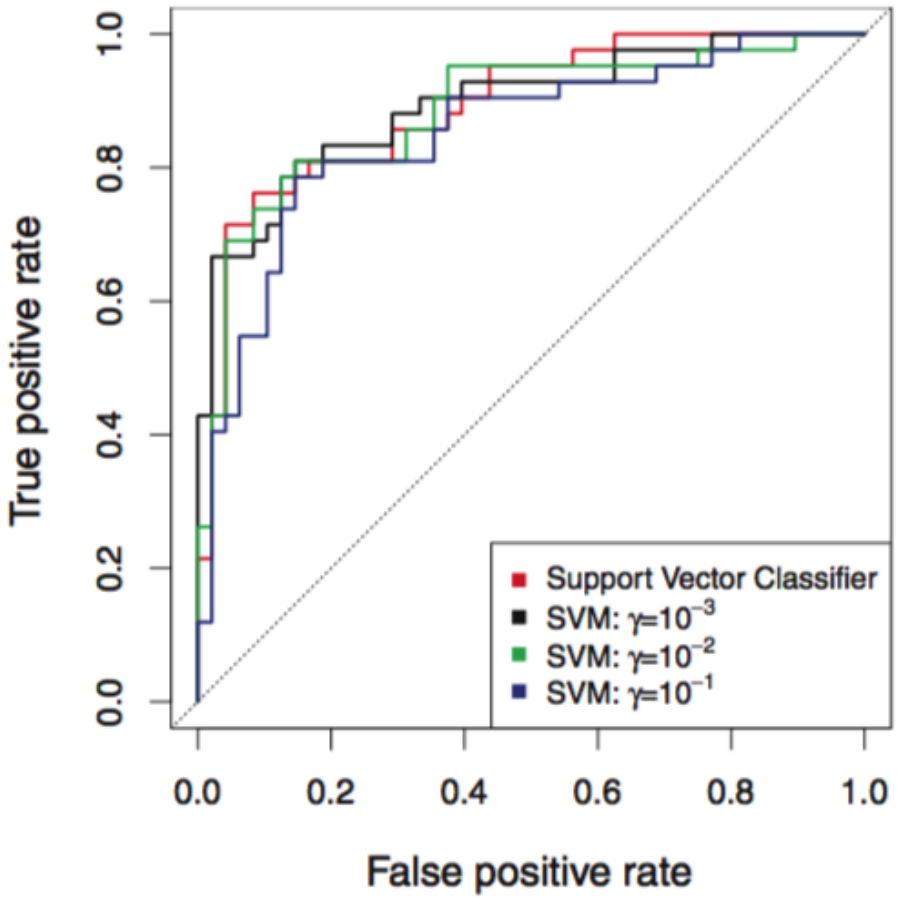
In many applications, the enlarged feature space is so large that computations are intractable.

For some kernels, such as the radial kernel, the feature space is implicit and infinite-dimensional (Taylor series of the exponential function) , so we could never do the computations there anyway!

Application to the Heart Disease Data, test data

13 predictors such as Age, Sex in order to predict whether an individual has heart disease, 297 subjects, randomly split into 207 training and 90 test observations. *ROC* curves*:





* Probability scores are calculated using Platt scaling.

SVM with More than Two Classes

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?.

SVM with More than Two Classes

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?. Two approaches:

1. **OVA (One versus All)** : Fit K different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

SVM with More than Two Classes

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?. Two approaches:

1. **OVA (One versus All)** : Fit K different 2-class *SVM* classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

2. **OVO (One versus One)** : Fit all $\binom{n}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions.

Which to choose? If K is not too large, use *OVO*.

Which to use: SVM or Logistic Regression (LR) or LDA?

- *SVM* became very popular since the introduction of kernels.
- When classes are (nearly) separable, *SVM* does better than LR. So does LDA.
- If the goal is to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. It is possible to use kernels with LR and LDA as well, but computations are more expensive.

Summary

- The support vector machine is a generalization of a simple and intuitive classifier called the maximal margin classifier.
- The support vector classifier, an extension of the maximal margin classifier that can be applied in a broader range of cases.
- The support vector machine, which is a further extension of the support vector classifier in order to accommodate non-linear class boundaries.

People often loosely refer to the maximal margin classifier, the support vector classifier, and the support vector machine as "support vector machines".

References

- James, Gareth; Witten, Daniela; Hastie, Trevor and Tibshirani, Robert. "An Introduction to Statistical Learning with Applications in R", 2nd edition, New York : "Springer texts in statistics", 2021. Site web: `https://hastie.su.domains/ISLR2/ISLRv2_website.pdf`
- Hastie, Trevor; Tibshirani, Robert and Friedman, Jerome (2009). "The Elements of Statistical Learning (Data Mining, Inference, and Prediction), 2nd edition". New York: "Springer texts in statistics". Site web : `http://statweb.stanford.edu/~tibs/ElemStatLearn/`