Research questions
Tree-based methods: application to economics

Pierre Michel

2018–2019

Master in economics (2^{nd} year)

What is Machine Learning?

- **Machine Learning** (ML): field of computer science, statistics and AI

What is Machine Learning?

- **Machine Learning** (ML): field of computer science, statistics and AI
- Development, analysis and implementation of **computational methods**

What is Machine Learning?

- **Machine Learning** (ML): field of computer science, statistics and AI
- Development, analysis and implementation of **computational methods**
- Allows a machine to evoluates in a given environment through a **training process**

Applications of Machine Learning?

Use examples:

- Pattern recognition (objects, hand-written characters)
- Clustering of phone-calls
- Medical and clinical decision making
- Web mining, search engines
- Image indexation

The impact of machine learning on economics

"I believe that ML will have a dramatic impact on the field of economics within a short time frame."

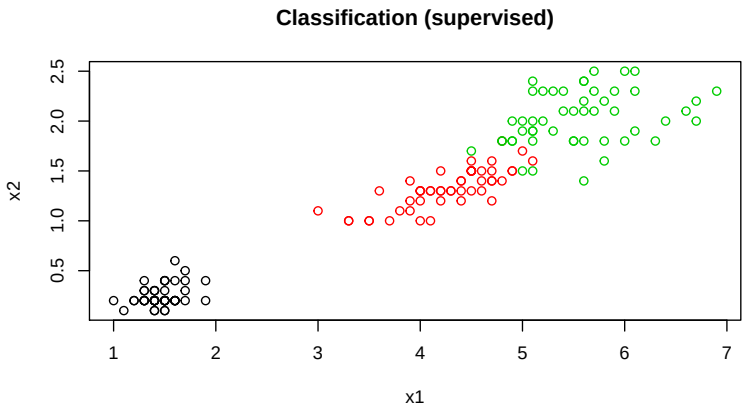
— Susan Athey, The Impact of Machine Learning on Economics, 2018.

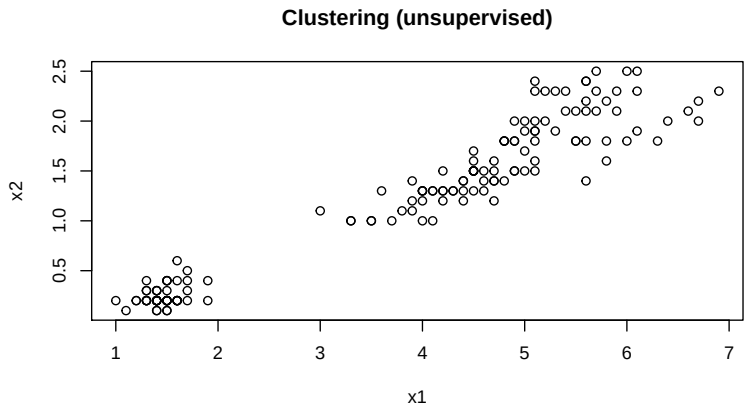
The impact of machine learning on economics

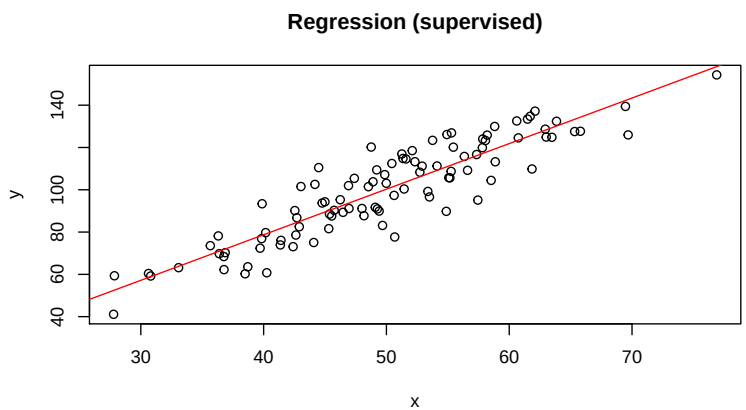
Applications of ML in many fields of economics:

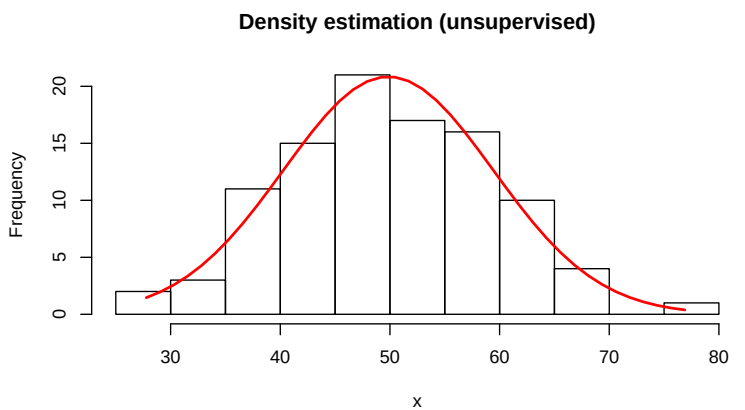
- Econometrics
- Health economics
- Labour economics
- Housing economics
- Economics of the internet

Supervised versus Unsupervised learning









2 Tree-based methods
- Binary trees
- Regression trees
- Classification trees

Decision trees

Tree-based methods (also called **decision trees**) aim to:

Decision trees

Tree-based methods (also called **decision trees**) aim to:

- **partition** the feature space into a **set of regions**.

Decision trees

Tree-based methods (also called **decision trees**) aim to:

- **partition** the feature space into a **set of regions**.
- **fit** a simple model in each region.

Decision trees

Tree-based methods (also called **decision trees**) aim to:

- **partition** the feature space into a **set of regions**.
- **fit** a simple model in each region.
- useful for regression, classification, clustering.

Decision trees

Tree-based methods (also called **decision trees**) aim to:

- **partition** the feature space into a **set of regions**.
- **fit** a simple model in each region.
- useful for regression, classification, clustering.

They are **simple** and **powerful**.

Decision trees

Tree-based methods (also called **decision trees**) aim to:

- **partition** the feature space into a **set of regions**.
- **fit** a simple model in each region.
- useful for regression, classification, clustering.

They are **simple** and **powerful**.

Most popular method: Classification and Regression Trees (CART), Breiman, 1984.

Building a regression tree

Suppose we have a set of n observations described by p features $X_1, X_2, ..., X_p$ and one **quantitative** response variable Y.

Building a regression tree

Suppose we have a set of n observations described by p features $X_1, X_2, ..., X_p$ and one **quantitative** response variable Y.

There are two steps for building a tree:

Building a regression tree

Suppose we have a set of n observations described by p features $X_1, X_2, ..., X_p$ and one **quantitative** response variable Y.

There are two steps for building a tree:

- Divide the feature space in J **non-overlapping regions** $R_1, R_2, ..., R_J$.

Building a regression tree

Suppose we have a set of n observations described by p features $X_1, X_2, ..., X_p$ and one **quantitative** response variable Y.

There are two steps for building a tree:

- Divide the feature space in J **non-overlapping regions** $R_1, R_2, ..., R_J$.
- For each observation in region R_j, compute the mean of the response values:

$$\hat{y}_{R_j} = \frac{1}{\#R_j} \sum_{i \in R_j} y_i$$

Building a regression tree

Q: How the regions are constructed ?

Building a regression tree

Q: How the regions are constructed ?

A: Using a top-down, greedy approach: **recursive binary splitting**

Building a regression tree

Q: How the regions are constructed ?

A: Using a top-down, greedy approach: **recursive binary splitting**

Goal: find regions $R_1, ..., R_J$ that **minimize the residual sum of squares**, defined by:

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

Building a regression tree

Q: How the regions are constructed ?

A: Using a top-down, greedy approach: **recursive binary splitting**

Goal: find regions $R_1, ..., R_J$ that **minimize the residual sum of squares**, defined by:

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

Note: RSS is also called the **training error**

Building a regression tree

Recursive binary splitting:

Building a regression tree

Recursive binary splitting:

- Consider the **feature** X_j

Building a regression tree

Recursive binary splitting:

- Consider the **feature** X_j
- Select a **cutpoint** $s \in \mathcal{S}(X_j)$

Building a regression tree

Recursive binary splitting:

- Consider the **feature** X_j
- Select a **cutpoint** $s \in \mathcal{S}(X_j)$
- **Split** the feature space in two regions R_l and R_r such that:

$$R_l(j,s) = \{X | X_j < s\} \text{ and } R_r(j,s) = \{X | X_j \geq s\}$$

Building a regression tree

Recursive binary splitting:

- Consider the **feature** X_j
- Select a **cutpoint** $s \in \mathcal{S}(X_j)$
- **Split** the feature space in two regions R_l and R_r such that:

$$R_l(j,s) = \{X | X_j < s\} \text{ and } R_r(j,s) = \{X | X_j \geq s\}$$

Finally we keep the pair (j,s) that **minimize**

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

Building a regression tree

This process is repeated for the new regions obtained

Building a regression tree

This process is repeated for the new regions obtained

Continues until a **stopping criterion** (e.g minimum size of a region) is verified

Building a regression tree

This process is repeated for the new regions obtained

Continues until a **stopping criterion** (e.g minimum size of a region) is verified

The resulting tree is call the **maximal tree**

Pruning the maximal tree

The maximal tree might be too complex (in terms of depth)

Pruning the maximal tree

The maximal tree might be too complex (in terms of depth)

A smaller tree with less splits (regions): better interpretation, lower variance

Pruning the maximal tree

The maximal tree might be too complex (in terms of depth)

A smaller tree with less splits (regions): better interpretation, lower variance

Pruning consists in growing a very large tree T_0 and reduce it back to obtain a **subtree**

Pruning the maximal tree

Cost complexity pruning or **Weakest link pruning**

Pruning the maximal tree

Cost complexity pruning or **Weakest link pruning**

We define a measure that take into account both training error and tree complexity

Pruning the maximal tree

Cost complexity pruning or **Weakest link pruning**

We define a measure that take into account both training error and tree complexity

Goal: For a given value α (**complexity parameter**), find a subtree $T \subset T_0$ that minimizes

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

Example: Predicting Baseball Players' Salaries

Data: **Hitters** dataset (Major League Baseball Data from the 1986 and 1987 seasons) containing 322 observations of 20 features.

Example: Predicting Baseball Players' Salaries

Data: **Hitters** dataset (Major League Baseball Data from the 1986 and 1987 seasons) containing 322 observations of 20 features.

Goal: Predict a baseball player's `Salary` based on `Years` (number of years played in major leagues) and `Hits` (number of hits made in the previous season).

Name	Years	Hits	Salary
Alan Ashby	14	81	475.0
Alvin Davis	3	130	480.0
Andre Dawson	11	141	500.0
Andres Galarraga	2	87	91.5
Alfredo Griffin	11	169	750.0

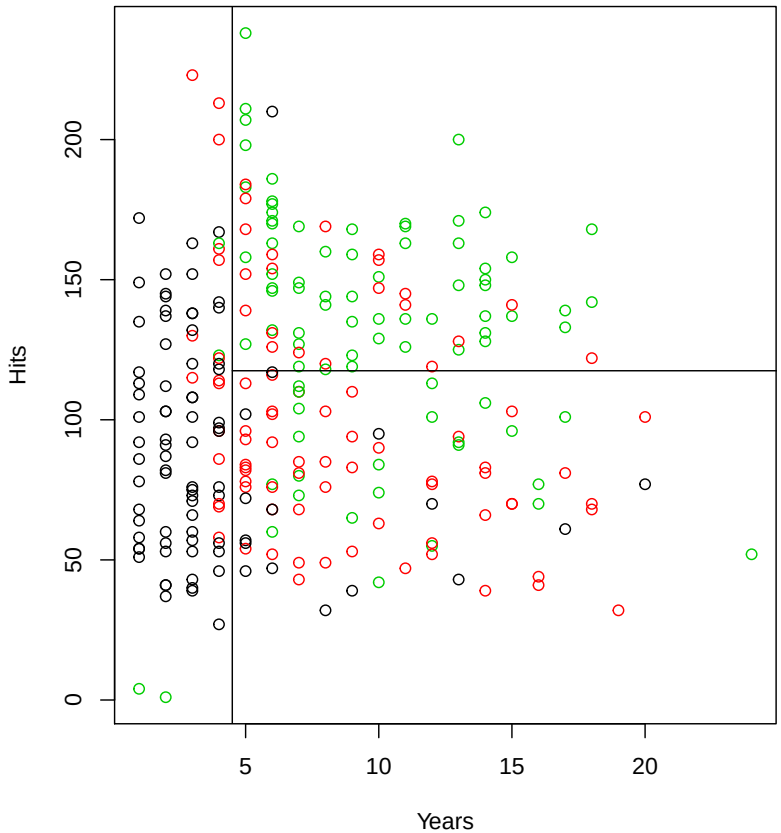
Figure: A sample of baseball players

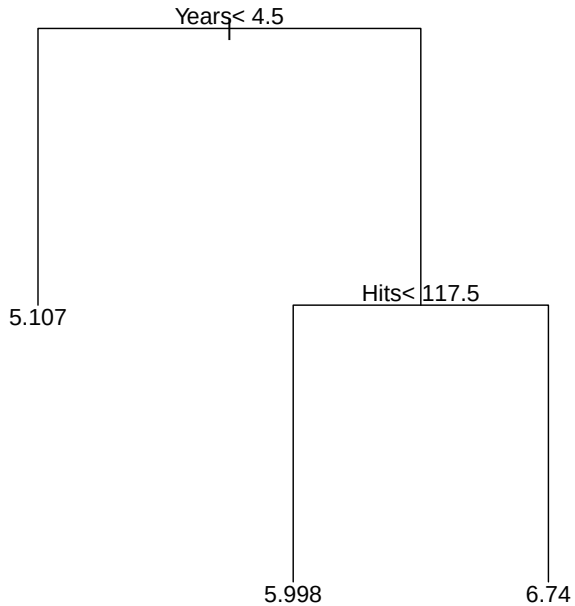
Name	Years	Hits	Salary
Alan Ashby	14	81	475.0
Alvin Davis	3	130	480.0
Andre Dawson	11	141	500.0
Andres Galarraga	2	87	91.5
Alfredo Griffin	11	169	750.0

Figure: A sample of baseball players

We denote:

- X_1: `Years` (exogeneous)
- X_2: `Hits` (exogeneous)
- Y: log(`Salary`) (endogeneous)


Hits
0
50
100
150
200
5
10
15
20
Years


Years< 4.5
5.107
Hits< 117.5
5.998
6.74

Classification tree

A classification tree is similar to a regression tree

Classification tree

A classification tree is similar to a regression tree

Used to predict a **qualitative** response variable:
$Y \in \{0, ..., K\}, K \in \mathbb{N}^*$

Classification tree

A classification tree is similar to a regression tree

Used to predict a **qualitative** response variable:
$Y \in \{0, ..., K\}, K \in \mathbb{N}^*$

Assign to each region R_j the **most commonly occuring class** of observations in R_j

Classification tree

A classification tree is similar to a regression tree

Used to predict a **qualitative** response variable:
$Y \in \{0, ..., K\}, K \in \mathbb{N}^*$

Assign to each region R_j the **most commonly occuring class** of observations in R_j

Problem: RSS cannot be used in classification: an alternative to RSS is the **classification error rate**

Classification error rate

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

Classification error rate

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

$$E = 1 - \max_k(\hat{p}_{mk})$$

Classification error rate

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

$$E = 1 - \max_k(\hat{p}_{mk})$$

Here, $\hat{p}_{mk}$ is the proportion of training observations in the m^{th} region that are from the k^{th} class

Classification error rate

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

$$E = 1 - \max_k(\hat{p}_{mk})$$

Here, $\hat{p}_{mk}$ is the proportion of training observations in the m^{th} region that are from the k^{th} class

Note: in practice, two other measures are used

Gini index

The Gini index is defined as follows:

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

Gini index

The Gini index is defined as follows:

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

This is a measure of total variance across the K classes

Gini index

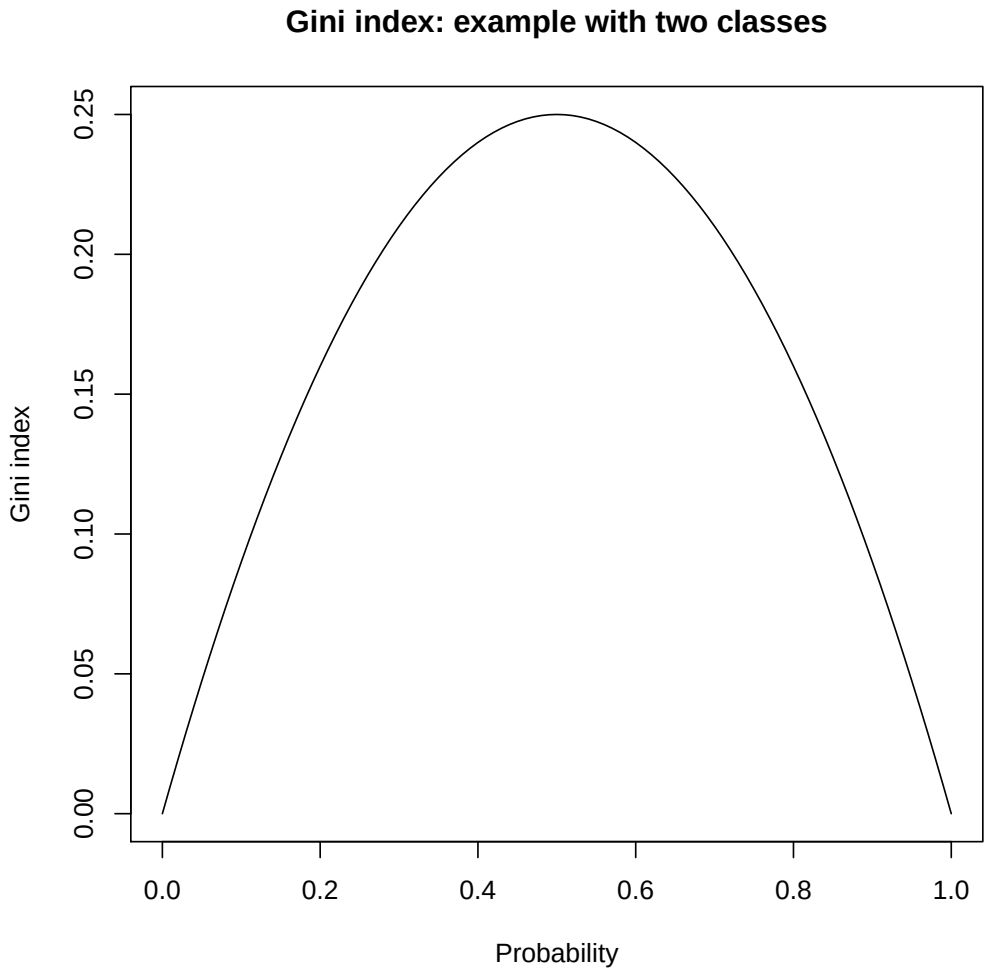
The Gini index is defined as follows:

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

This is a measure of total variance across the K classes

The Gini index is a measure of **purity**: small values indicate that a node contains a predominant class

Gini index


Gini index: example with two classes
Gini index
0.00
0.05
0.10
0.15
0.20
0.25
0.0
0.2
0.4
0.6
0.8
1.0
Probability

Cross-entropy

An alternative to the Gini index

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

Cross-entropy

An alternative to the Gini index

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

This is also a measure of purity: small values indicate that a node is **pure**

Cross-entropy

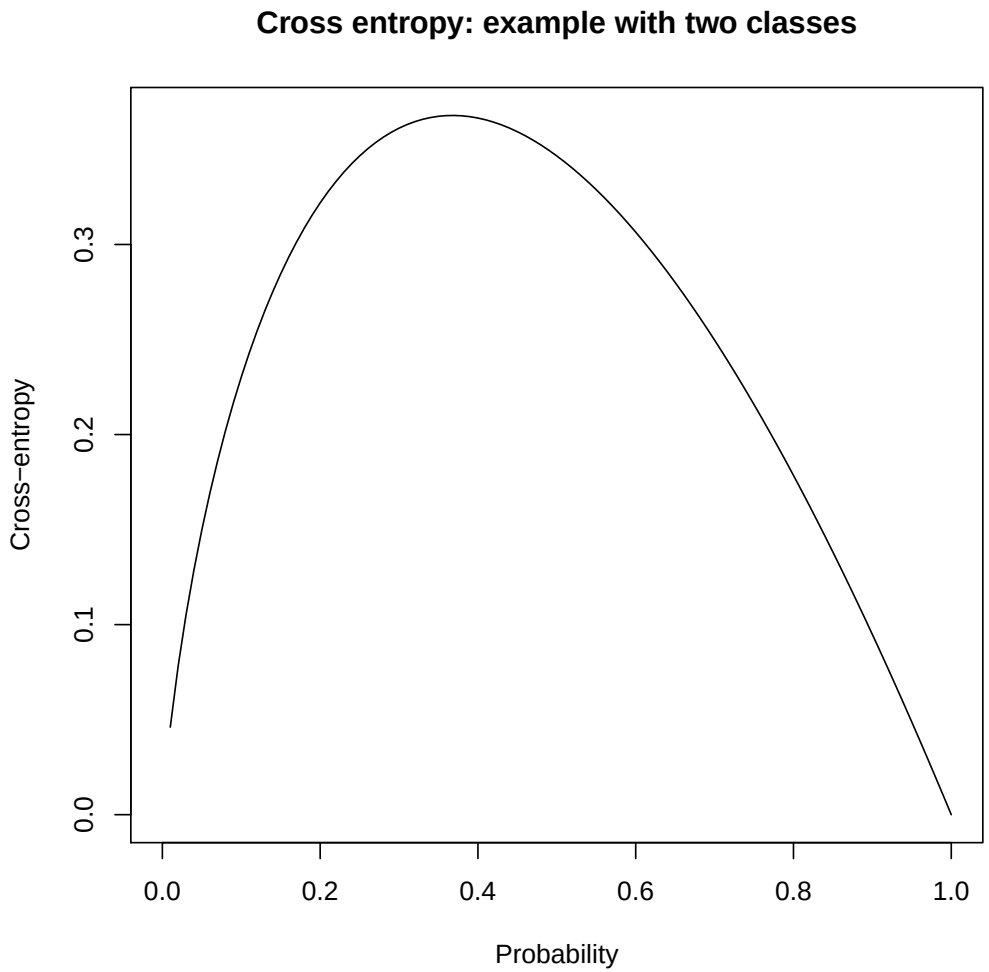
An alternative to the Gini index

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

This is also a measure of purity: small values indicate that a node is **pure**

Note: Gini index and cross-entropy are similar numerically

Cross-entropy



Example: predicting heart disease

Data: **Heart** dataset containing 303 observations of 14 features.

Example: predicting heart disease

Data: **Heart** dataset containing 303 observations of 14 features.

Goal: Predict the presence of heart disease `HD` based on different predictors such as `Age`, `Sex`, `Chol` (a cholesterol measurement)...

Age	Sex	Chestpain	...	HD
63	1	typical	...	No
67	1	asymptomatic	...	Yes
67	1	asymptomatic	...	Yes
37	1	nonanginal	...	No
41	0	nontypical	...	No

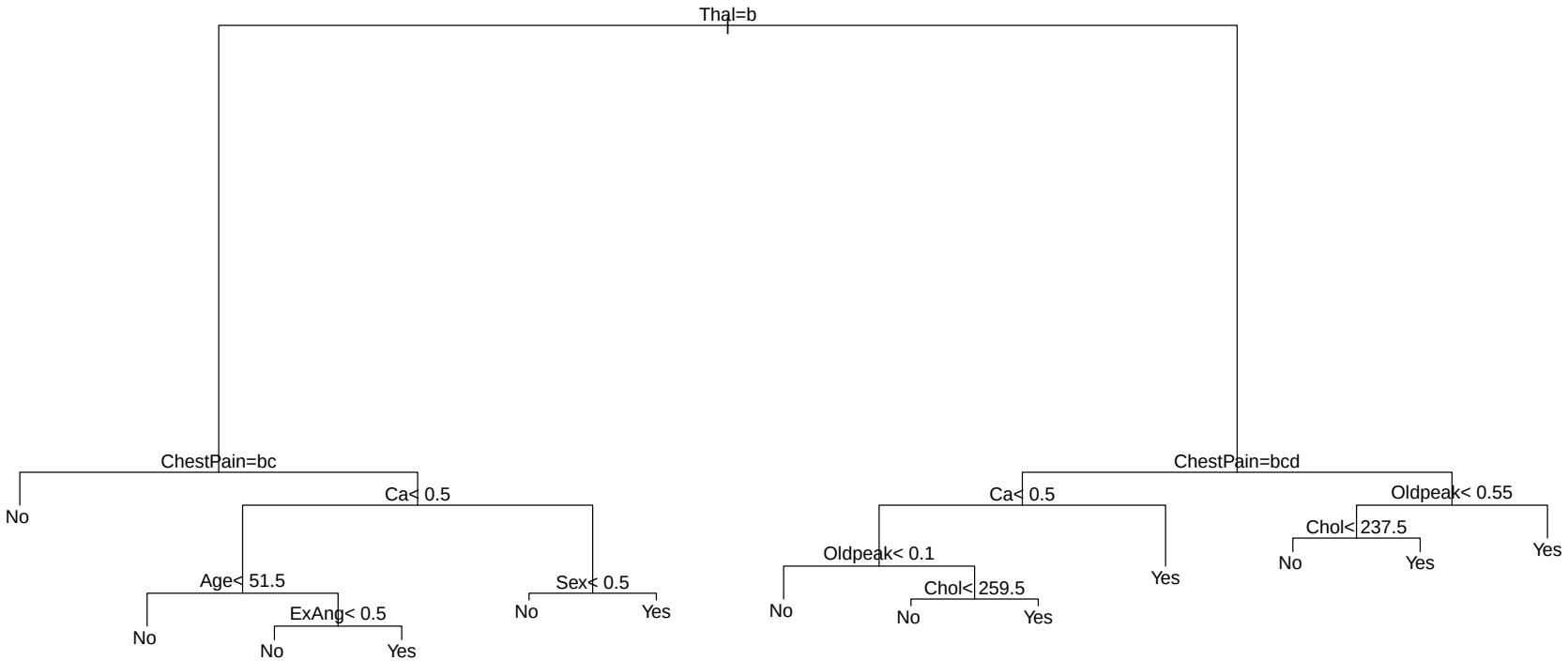
Figure: A sample of patients

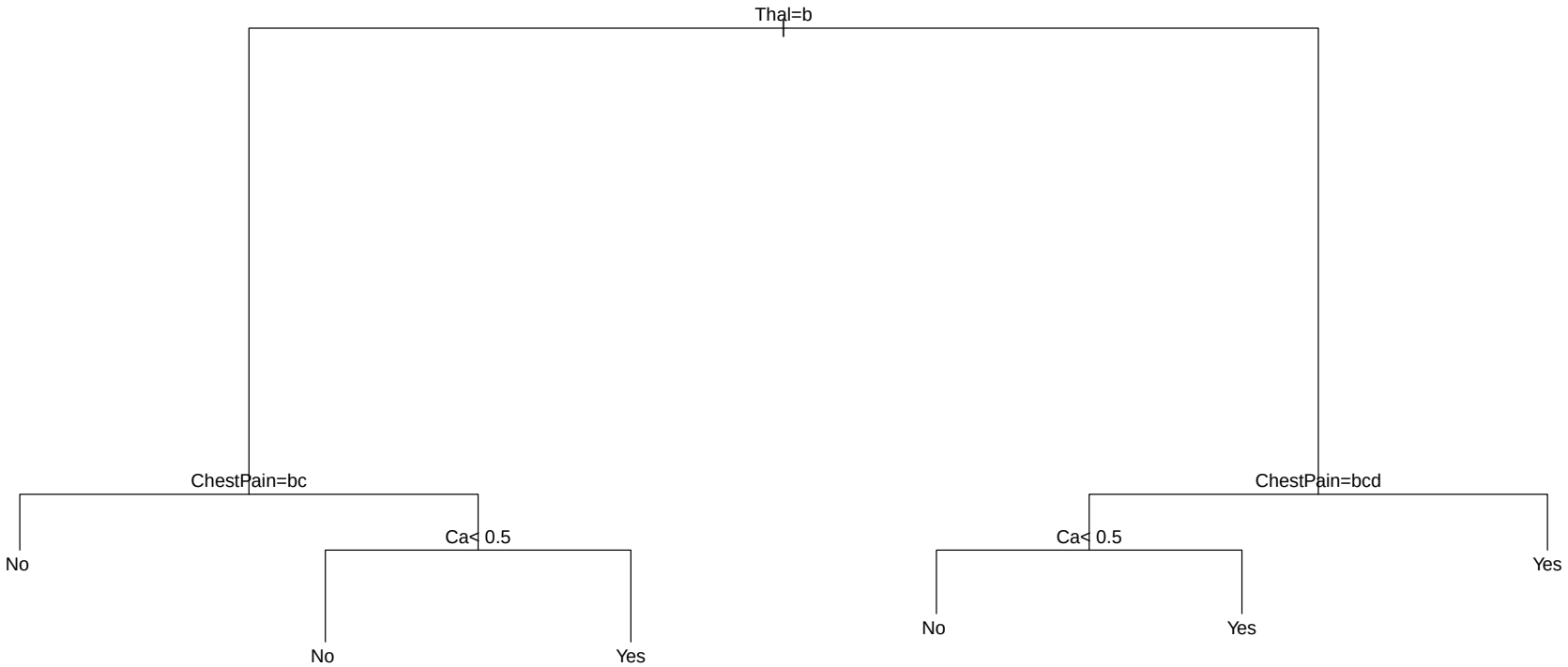
Age	Sex	Chestpain	...	HD
63	1	typical	...	No
67	1	asymptomatic	...	Yes
67	1	asymptomatic	...	Yes
37	1	nonanginal	...	No
41	0	nontypical	...	No

Figure: A sample of patients

We denote:

- X_1: `Age` (exogeneous)
- X_2: `Sex` (exogeneous)
- ...
- Y: `HD` (endogeneous)


Thal=b
ChestPain=bc
Ca< 0.5
No
Age< 51.5
Sex< 0.5
No
ExAng< 0.5
No
No
Yes
Yes
ChestPain=bcd
Ca< 0.5
Oldpeak< 0.55
Oldpeak< 0.1
Yes
Chol< 237.5
No
Chol< 259.5
Yes
No
No
Yes
Yes


Thal=b
ChestPain=bc
ChestPain=bcd
Ca< 0.5
Ca< 0.5
No
No
Yes
No
Yes
Yes

Trees: pros and cons

Pros:

- Easy to use
- Help decision making
- Graphical, interpretable
- Deal with both quantitative and qualitative features

Cons:

- Not the best method in terms of prediction accuracy
- Trees are unstable

3 Advanced tree-based methods
- Bagging and Random Forest
- Clustering using binary trees
- Variable importance

Bagging

Bootstrap aggregation (or **bagging**) is a procedure for reducing the variance of a statistial learning method

Bagging

Bootstrap aggregation (or **bagging**) is a procedure for reducing the variance of a statistial learning method

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

Bagging

Bootstrap aggregation (or **bagging**) is a procedure for reducing the variance of a statistial learning method

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

Averaging a set of obervations reduces variance

Bagging

Idea:

- Calculate B predictions denoted $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ using B separate training sets

Bagging

Idea:

- Calculate B predictions denoted $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ using B separate training sets
- Average the B predictions to obtain a single low-variance statistical learning model:

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

Bagging

Idea:

- Calculate B predictions denoted $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ using B separate training sets
- Average the B predictions to obtain a single low-variance statistical learning model:

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

- Problem: In practice, we do not have access to multiple training sets

Bagging

Solution:

- Generate B bootstrapped training datasets

Bagging

Solution:

- Generate B bootstrapped training datasets
- Train a model (a tree) on the b^{th} bootstrapped training dataset in ordre to get $\hat{f}*b(x)$

Bagging

Solution:

- Generate B bootstrapped training datasets
- Train a model (a tree) on the b^{th} bootstrapped training dataset in ordre to get $\hat{f}*b(x)$
- Average all the predictions:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}*b(x)$$

Random forest

- RF is an improvement over bagged trees, using a "trick" that **decorrelated** the bootstrapped trees

Random forest

- RF is an improvement over bagged trees, using a "trick" that **decorrelated** the bootstrapped trees

- A random sample of m predictors is chosen among the full set of p predictors at each split of the tree

Random forest

- RF is an improvement over bagged trees, using a "trick" that **decorrelated** the bootstrapped trees
- A random sample of m predictors is chosen among the full set of p predictors at each split of the tree
- Typically, we choose $m = \sqrt{p}$, useful when we have a high number of correlated predictors

Unsupervised learning

- Two issues in unsupervised learning: **Clustering** and **Density estimation**

Unsupervised learning

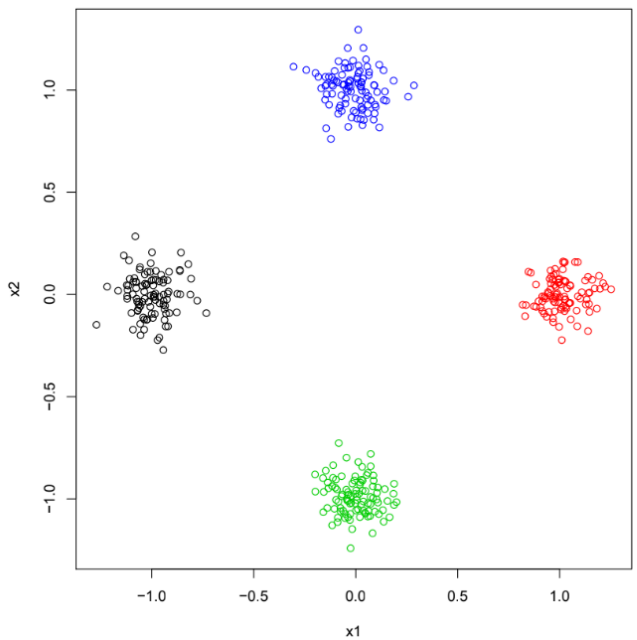
- Two issues in unsupervised learning: **Clustering** and **Density estimation**
- We have no endogeneous feature Y

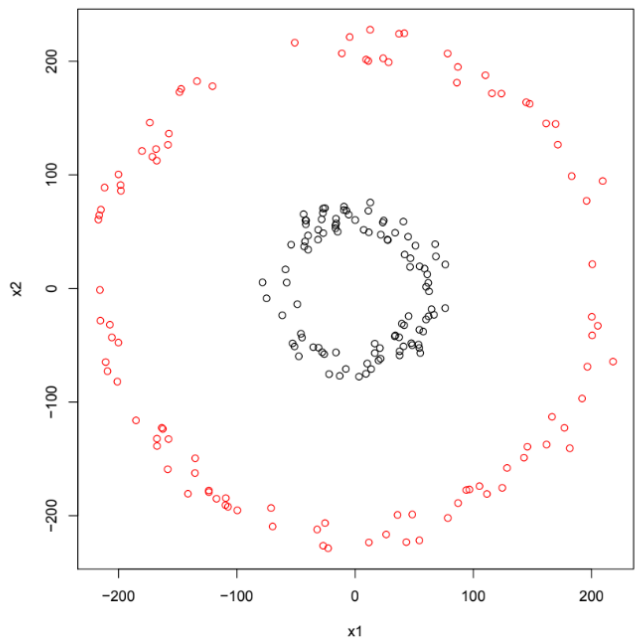
Unsupervised learning

- Two issues in unsupervised learning: **Clustering** and **Density estimation**
- We have no endogeneous feature Y
- We focus on **clustering**

Unsupervised learning

- Two issues in unsupervised learning: **Clustering** and **Density estimation**
- We have no endogeneous feature Y
- We focus on **clustering**





Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

- hierarchical methods (e.g HCA)

Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

- hierarchical methods (e.g HCA)
- partitional methods (e.g K-means)

Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

- hierarchical methods (e.g HCA)
- partitional methods (e.g K-means)
- density-based methods

Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

- hierarchical methods (e.g HCA)
- partitional methods (e.g K-means)
- density-based methods

Classical problems:

Clustering

Suppose we have a set of training observations $\{X_i\}, i = 1, ..., n$, we try to find a **partition** of this set in K clusters, minimizing **within-cluster heterogeneity** and maximizing **between-cluster heterogeneity**. There are three main approaches:

- hierarchical methods (e.g HCA)
- partitional methods (e.g K-means)
- density-based methods

Classical problems:

- Measure the **performance** of a partition
- Choose the optimal value of K, the **number of clusters**

Clustering using unsupervised binary trees

CUBT is a top-down hierarchical clustering methods that works in **3 steps**:

Clustering using unsupervised binary trees

CUBT is a top-down hierarchical clustering methods that works in **3 steps**:

- **growing** the maximal tree: recursive binary partitioning

Clustering using unsupervised binary trees

CUBT is a top-down hierarchical clustering methods that works in **3 steps**:

- **growing** the maximal tree: recursive binary partitioning
- **pruning** the tree (dissimilarity-based pruning)

Clustering using unsupervised binary trees

CUBT is a top-down hierarchical clustering methods that works in **3 steps**:

- **growing** the maximal tree: recursive binary partitioning
- **pruning** the tree (dissimilarity-based pruning)
- **joining** the leaves of the tree (alternative pruning)

Similarities with CART

CUBT has many similarities with CART:

Similarities with CART

CUBT has many similarities with CART:

- Efficiency

Similarities with CART

CUBT has many similarities with CART:

- Efficiency
- Flexibility

Similarities with CART

CUBT has many similarities with CART:

- Efficiency
- Flexibility
- Interpretability

Similarities with CART

CUBT has many similarities with CART:

- Efficiency
- Flexibility
- Interpretability
- Good convergence properties

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$.

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$. The **child nodes** of t are denoted t_L and t_R, defined as follows:

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$. The **child nodes** of t are denoted t_L and t_R, defined as follows:

$$t_L = \{x \in \mathbb{R}^p | x_j \leq a\} \text{ and } t_R = \{x \in \mathbb{R}^p | x_j > a\}$$

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$. The **child nodes** of t are denoted t_L and t_R, defined as follows:

$$t_L = \{x \in \mathbb{R}^p | x_j \leq a\} \text{ and } t_R = \{x \in \mathbb{R}^p | x_j > a\}$$

Let $X_t = \{X | X \in t\}$, $\alpha_t = P(X \in t)$ and $R(t)$ a heterogeneity measure (**deviance**) of t, defined as:

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$. The **child nodes** of t are denoted t_L and t_R, defined as follows:

$$t_L = \{x \in \mathbb{R}^p | x_j \leq a\} \text{ and } t_R = \{x \in \mathbb{R}^p | x_j > a\}$$

Let $X_t = \{X | X \in t\}$, $\alpha_t = P(X \in t)$ and $R(t)$ a heterogeneity measure (**deviance**) of t, defined as:

$$R(t) = \alpha_t trace(cov(X_t))$$

Step 1: Growing the maximal tree

Let t be a tree node containing a set of observations in $\mathbb{R}^p$. The **child nodes** of t are denoted t_L and t_R, defined as follows:

$$t_L = \{x \in \mathbb{R}^p | x_j \leq a\} \text{ and } t_R = \{x \in \mathbb{R}^p | x_j > a\}$$

Let $X_t = \{X | X \in t\}$, $\alpha_t = P(X \in t)$ and $R(t)$ a heterogeneity measure (**deviance**) of t, defined as:

$$R(t) = \alpha_t trace(cov(X_t))$$

The **best split** of t is defined by the pair $(j, a) \in \{1, ...p\} \times \mathbb{R}$ maximizing

$$\Delta(t, j, a) = R(t) - R(t_L) - R(t_R)$$

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

- All observations in t are the same

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

- All observations in t are the same
- There are less than *minsize* observations in t

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

- All observations in t are the same
- There are less than *minsize* observations in t
- $\Delta(t, j, a) < \textit{mindev} \times R(S)$

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

- All observations in t are the same
- There are less than *minsize* observations in t
- $\Delta(t, j, a) < \textit{mindev} \times R(S)$

The clustering **tree** represents the **partition**.

Step 1: Growing the maximal tree

We denote S the initial training dataset, each node t is split recursively until the following **stopping criteria** are verified:

- All observations in t are the same
- There are less than *minsize* observations in t
- $\Delta(t, j, a) < mindev \times R(S)$

The clustering **tree** represents the **partition**. Each **leaf** represents a **cluster**.

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Pruning criterion

If $d^{\delta}(L, R) \leq \textit{mindist}$ then t_L and t_R are aggregated

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Pruning criterion

If $d^{\delta}(L, R) \leq \textit{mindist}$ then t_L and t_R are aggregated

$d^{\delta}(L, R)$ is an empirical dissimilarity measure between t_L and t_R:

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Pruning criterion

If $d^\delta(L,R) \leq \textit{mindist}$ then t_L and t_R are aggregated

$d^\delta(L,R)$ is an empirical dissimilarity measure between t_L and t_R:

$$d^\delta(L,R) = \max(\bar{d}_L^\delta, \bar{d}_R^\delta)$$

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Pruning criterion

If $d^\delta(L,R) \leq \textit{mindist}$ then t_L and t_R are aggregated

$d^\delta(L,R)$ is an empirical dissimilarity measure between t_L and t_R:

$$d^\delta(L,R) = \max(\bar{d}_L^\delta, \bar{d}_R^\delta)$$

where $\forall \delta \in [0,1]$

Step 2: Pruning the tree

We denote t_L and t_R the leaves obtained by splitting t.

Pruning criterion

If $d^\delta(L,R) \leq mindist$ then t_L and t_R are aggregated

$d^\delta(L,R)$ is an empirical dissimilarity measure between t_L and t_R:

$$d^\delta(L,R) = \max(\bar{d}_L^\delta, \bar{d}_R^\delta)$$

where $\forall \delta \in [0,1]$

$$\bar{d}_L^\delta = \frac{1}{\delta n_L} \sum_{i=1}^{\delta n_L} d_i \text{ and } \bar{d}_R^\delta = \frac{1}{\delta n_R} \sum_{j=1}^{\delta n_R} d_j$$

Step 2: Dissimilarity measure

$\forall X_i \in t_R$ and $X_j \in t_L$ we have:

Step 2: Dissimilarity measure

$\forall X_i \in t_R$ and $X_j \in t_L$ we have:

$$\tilde{d}_i = \min_{x \int_R} d(X_i, x) \text{ and } \tilde{d}_j = \min_{x \int_L} d(x, X_j)$$

Step 2: Dissimilarity measure

$\forall X_i \in t_R$ and $X_j \in t_L$ we have:

$$\tilde{d}_i = \min_{x \int_R} d(X_i, x) \text{ and } \tilde{d}_j = \min_{x \int_L} d(x, X_j)$$

and their ordered versions $\{d_i\}$ and $\{d_j\}$,

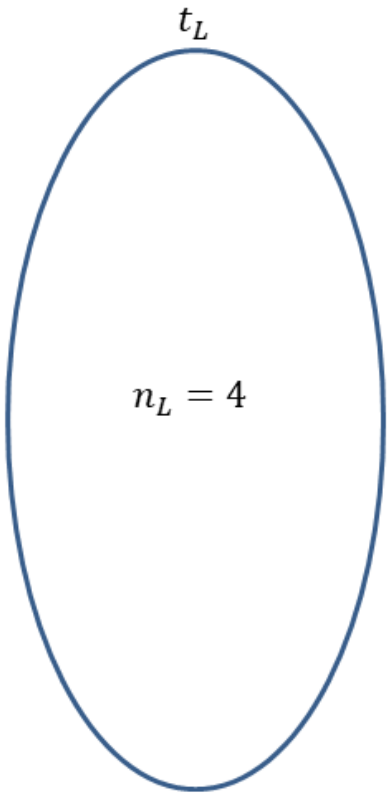
Step 2: Dissimilarity measure

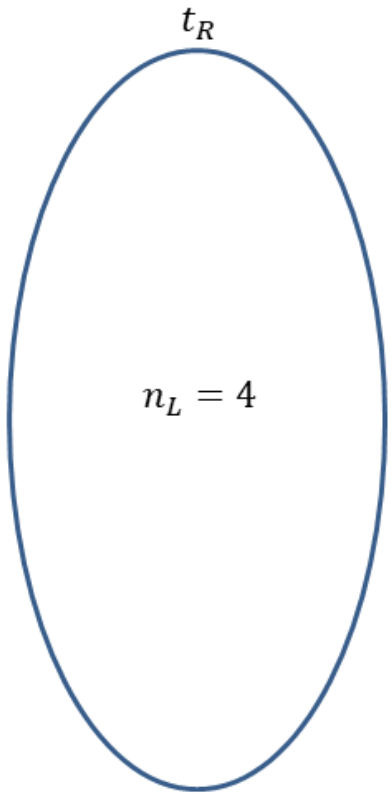
$\forall X_i \in t_R$ and $X_j \in t_L$ we have:

$$\tilde{d}_i = \min_{x \int_R} d(X_i, x) \text{ and } \tilde{d}_j = \min_{x \int_L} d(x, X_j)$$

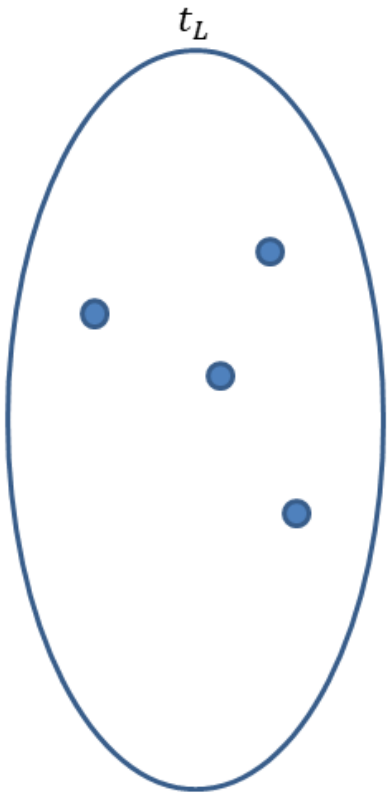
and their ordered versions $\{d_i\}$ and $\{d_j\}$, where d(X,y) is the Euclidean distance

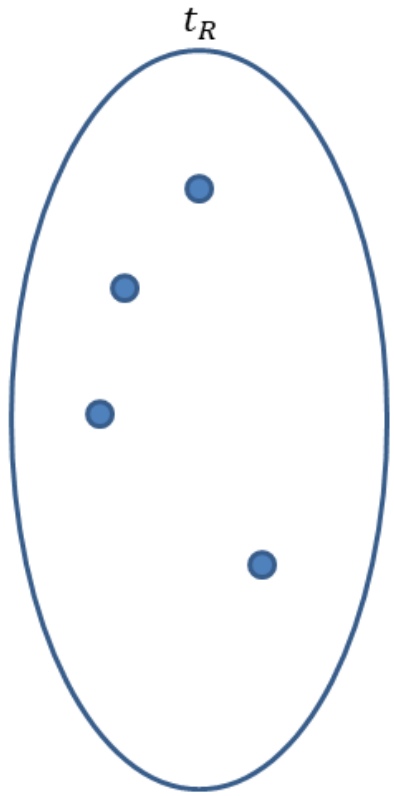
Dissimilarity measure: illustration



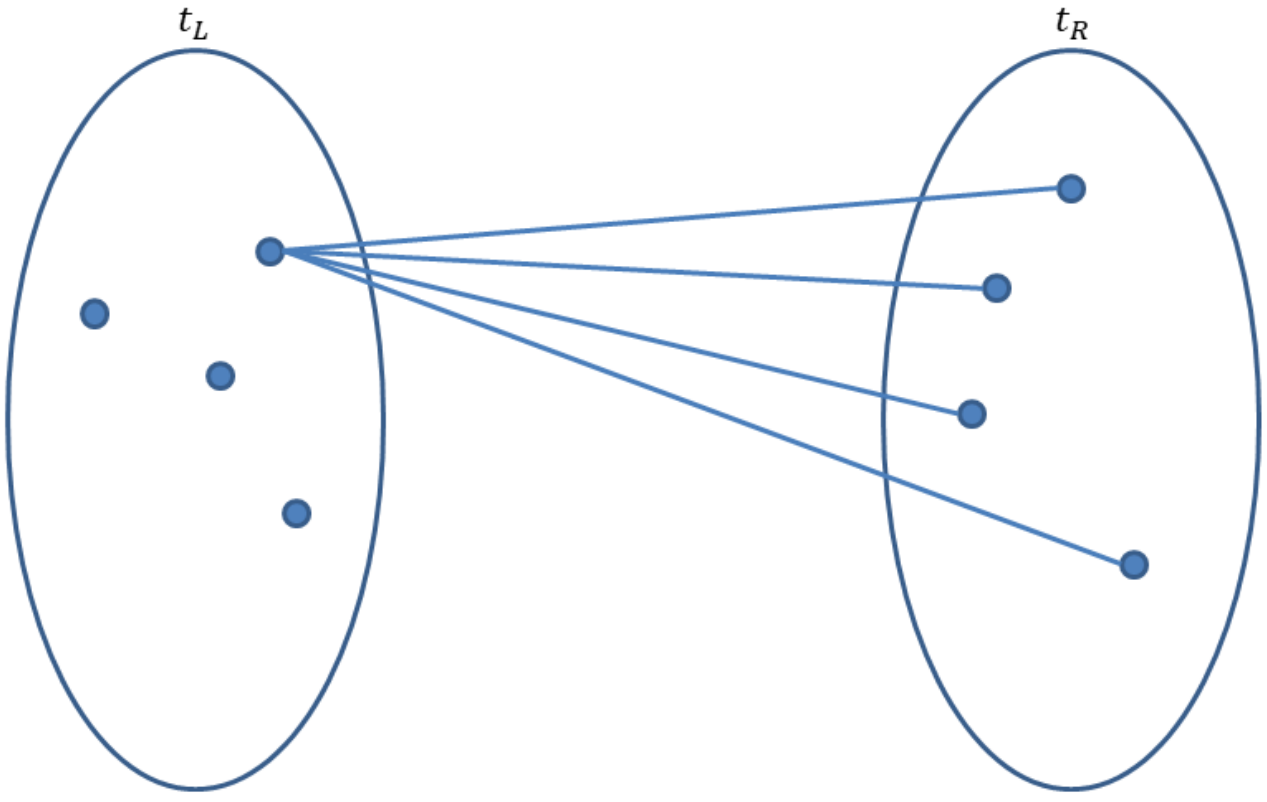


Dissimilarity measure: illustration

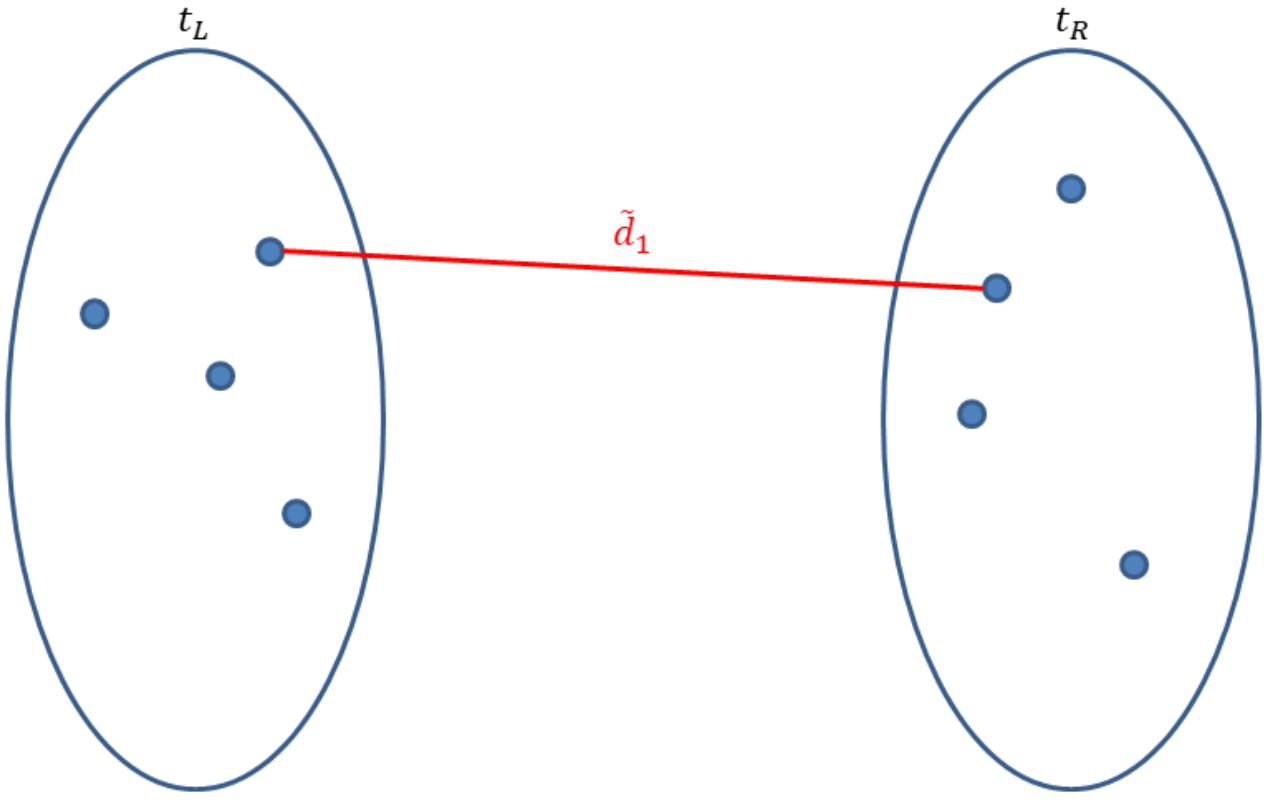




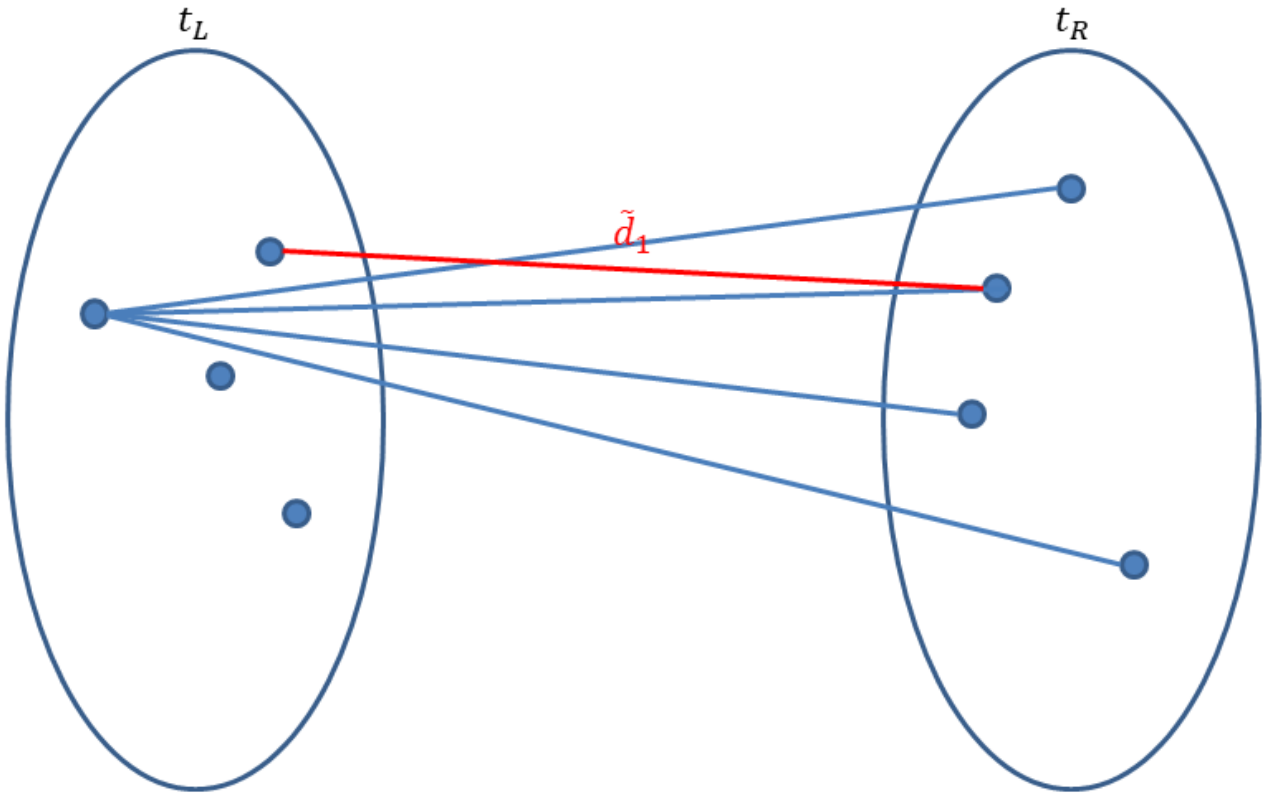
Dissimilarity measure: illustration



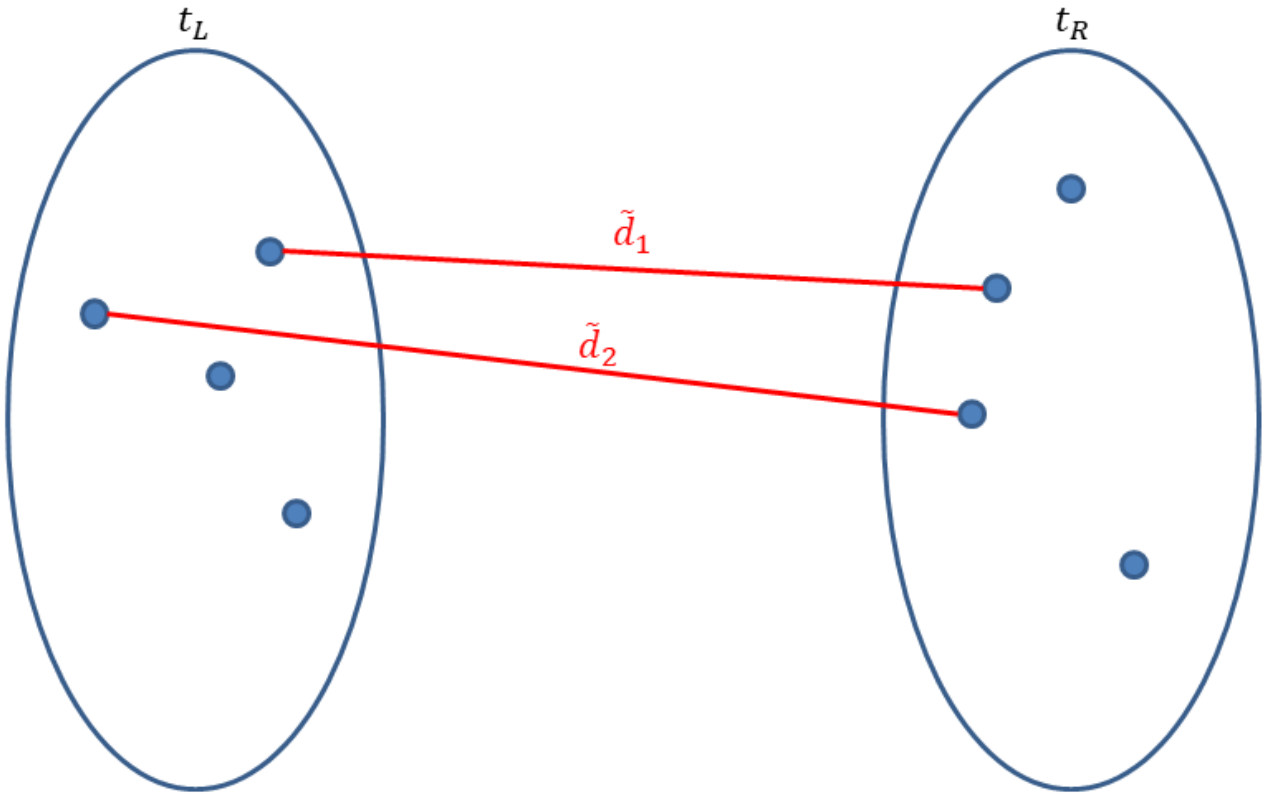
Dissimilarity measure: illustration



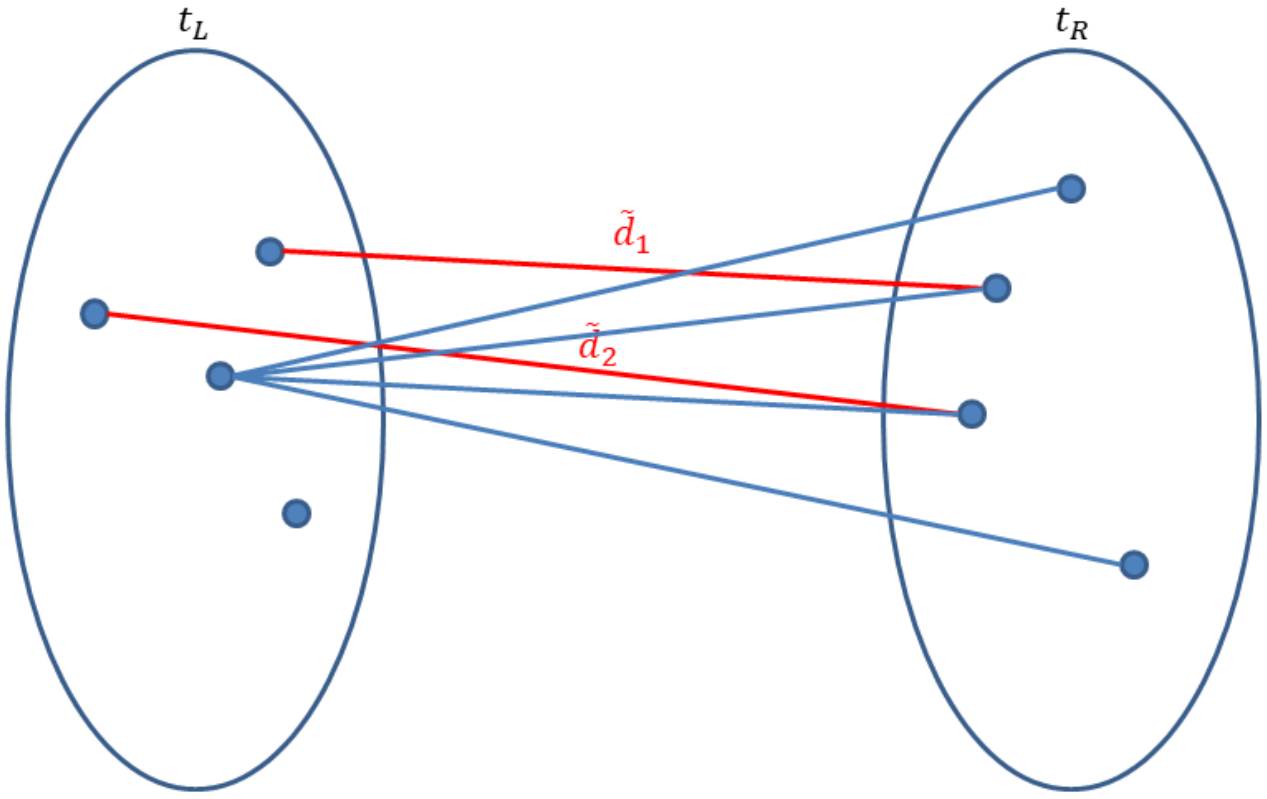
Dissimilarity measure: illustration



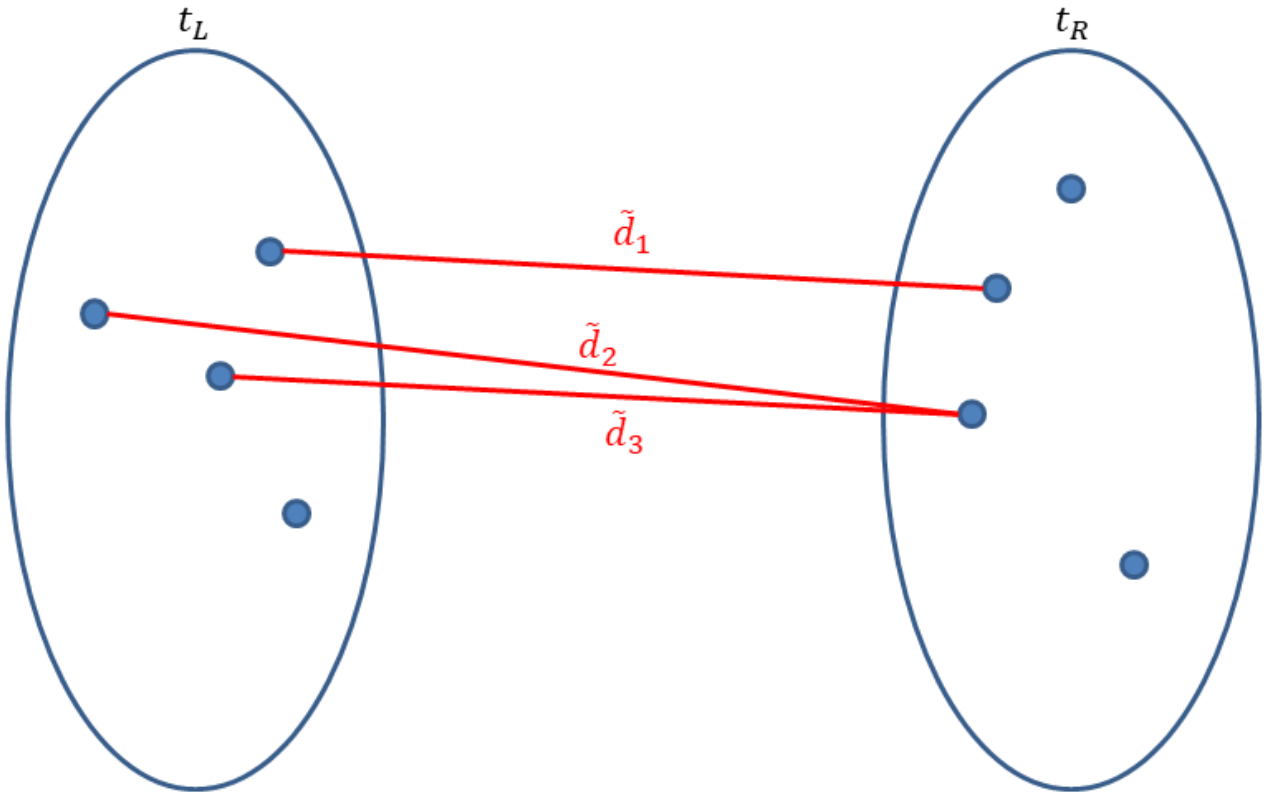
Dissimilarity measure: illustration



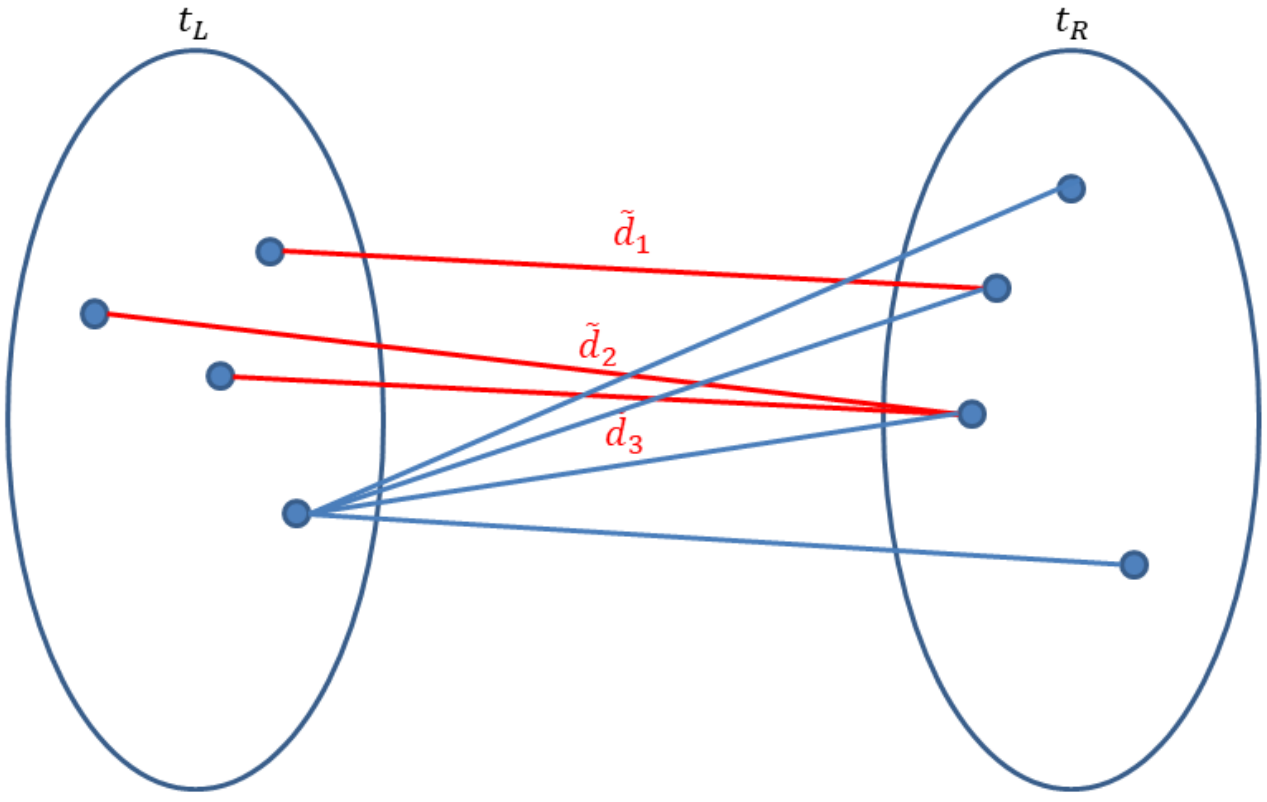
Dissimilarity measure: illustration



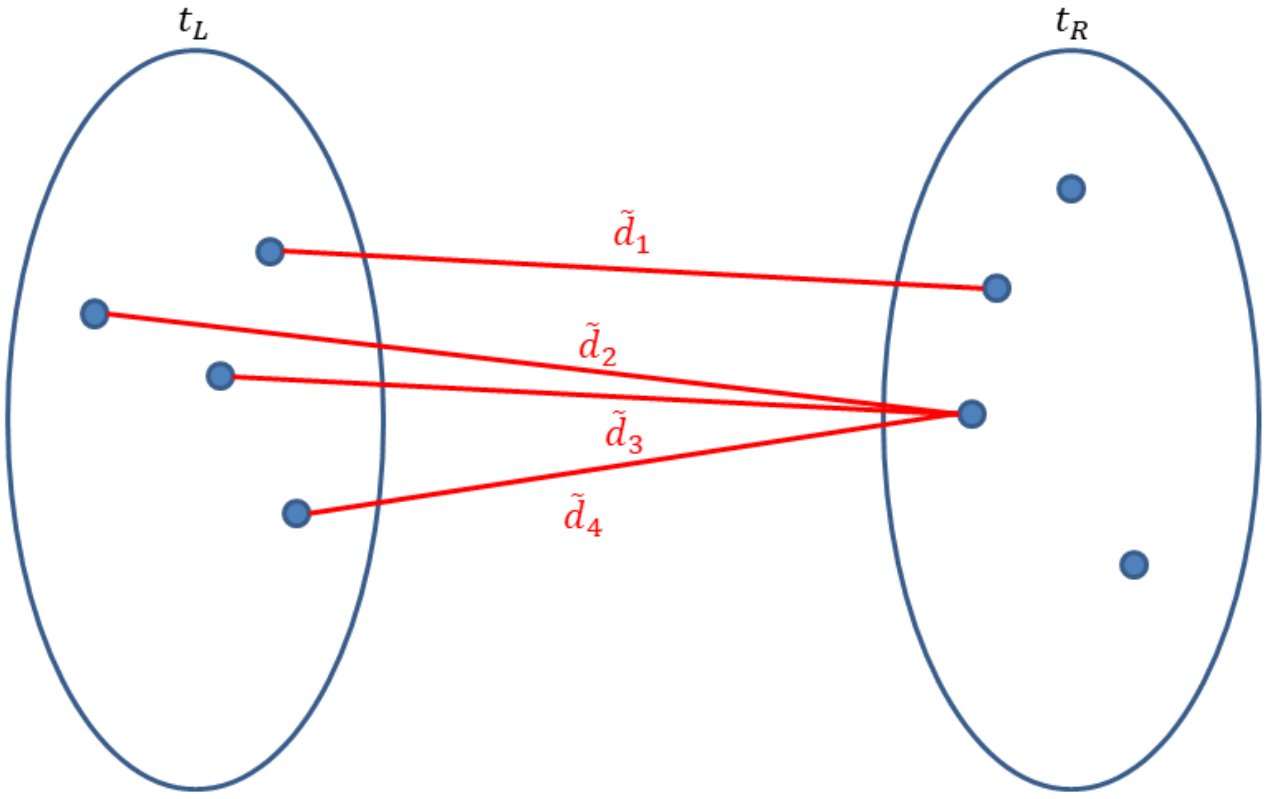
Dissimilarity measure: illustration



Dissimilarity measure: illustration



Dissimilarity measure: illustration



Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

1. $\Delta(t_L, t_R) = R(t_L \cup t_R) - R(t_L) - R(t_R)$

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

1. $\Delta(t_L, t_R) = R(t_L \cup t_R) - R(t_L) - R(t_R)$
2. $\Delta(t_L, t_R) = d^{\delta}(L, R)$

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

1. $\Delta(t_L, t_R) = R(t_L \cup t_R) - R(t_L) - R(t_R)$
2. $\Delta(t_L, t_R) = d^{\delta}(L, R)$

Let N_L be the total number of leaves and K the expected number of classes.

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

1. $\Delta(t_L, t_R) = R(t_L \cup t_R) - R(t_L) - R(t_R)$
2. $\Delta(t_L, t_R) = d^{\delta}(L, R)$

Let N_L be the total number of leaves and K the expected number of classes. $\forall (L, R) \in \{1, ..., N_L\}$ and $L \neq R$ we have $(\tilde{L}, \tilde{R}) = argmin_{L,R} \Delta(t_L, t_R)$

Step 3: Joining the leaves

We aggregate leaves that are not issued from a same parent

Two joining criteria

Leaves are compared using:

1. $\Delta(t_L, t_R) = R(t_L \cup t_R) - R(t_L) - R(t_R)$
2. $\Delta(t_L, t_R) = d^{\delta}(L, R)$

Let N_L be the total number of leaves and K the expected number of classes. $\forall (L, R) \in \{1, ..., N_L\}$ and $L \neq R$ we have $(\tilde{L}, \tilde{R}) = argmin_{L,R} \Delta(t_L, t_R)$

Joining the leaves

$t_{\tilde{L}}$ and $t_{\tilde{R}}$ are replaced by their union $t_{\tilde{L}} \cup t_{\tilde{R}}$ and $N_L = N_L - 1$.
Stop when $N_L = K$.

CUBT: pros and cons

Pros:

- Decisional method
- Interpretable
- Extensions to other types of data
- Adapted to parallel computing
- Partition of the feature space, not only the training dataset

Cons:

- Same as CART
- Trees are unstable

Motivation and objectives

Motivation

Motivation and objectives

Motivation

- Feature selection

Motivation and objectives

Motivation

- Feature selection
- Dimension reduction

Motivation and objectives

Motivation

- Feature selection
- Dimension reduction
- Missing data

Motivation and objectives

Motivation

- Feature selection
- Dimension reduction
- Missing data

Objectives

Motivation and objectives

Motivation

- Feature selection
- Dimension reduction
- Missing data

Objectives

- Define variable importance in CUBT

Motivation and objectives

Motivation

- Feature selection
- Dimension reduction
- Missing data

Objectives

- Define variable importance in CUBT
- Analyze its stability

Motivation and objectives

Motivation

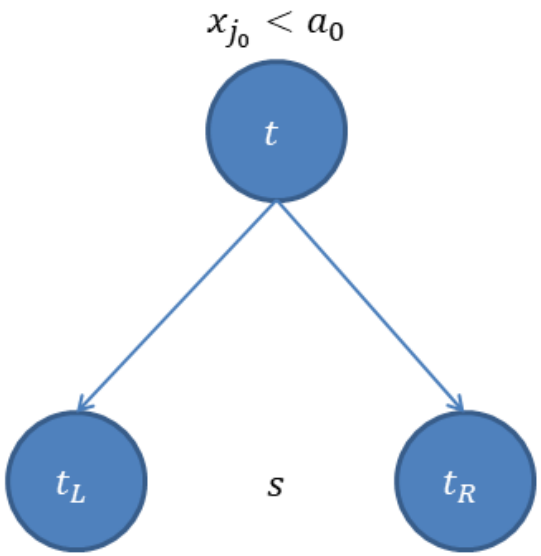
- Feature selection
- Dimension reduction
- Missing data

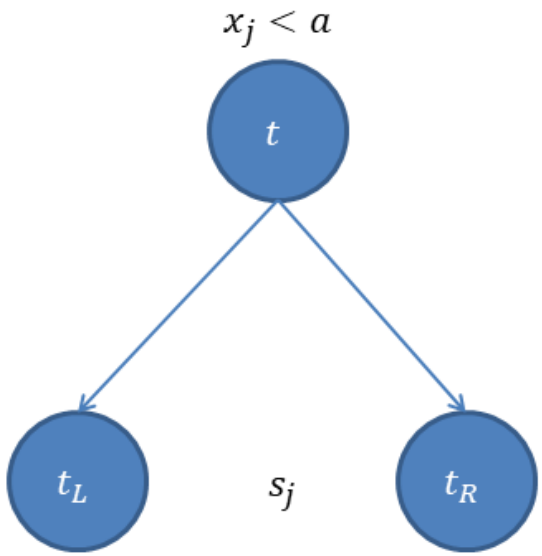
Objectives

- Define variable importance in CUBT
- Analyze its stability
- Compare to other methods

Competitive splits

To compute the importance of a feature j, we define the **competitive split** of a feature j_0 in a node t.





Competitive splits

The probability that an observation is sent to the **left node** for both splits is

Competitive splits

The probability that an observation is sent to the **left node** for both splits is

$$p(t_L \cap \ t'_L) = \frac{\#\{t_L \cap t'_L\}}{n_t}$$

Competitive splits

The probability that an observation is sent to the **left node** for both splits is

$$p(t_L \cap \ t'_L) = \frac{\#\{t_L \cap t'_L\}}{n_t}$$

Given a observation is in t, the probability that both splits sent it to the left is

Competitive splits

The probability that an observation is sent to the **left node** for both splits is

$$p(t_L \cap \ t'_L) = \frac{\#\{t_L \cap t'_L\}}{n_t}$$

Given a observation is in t, the probability that both splits sent it to the left is

$$p_{LL}(s, s_j) = \frac{p(t_L \cap \ t'_L)}{p(t)}$$

Competitive splits

The probability that an observation is sent to the **left node** for both splits is

$$p(t_L \cap \ t'_L) = \frac{\#\{t_L \cap t'_L\}}{n_t}$$

Given a observation is in t, the probability that both splits sent it to the left is

$$p_{LL}(s, s_j) = \frac{p(t_L \cap \ t'_L)}{p(t)}$$

p_{RR} can be defined equivalently

Surrogate splits and variable importance

We define an association measure between s_j and s

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

$\tilde{s}_j$ is a **surrogate split** of s if

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

$\tilde{s}_j$ is a **surrogate split** of s if $p(s, \tilde{s}_j) = max_{s_j} p(s, s_j)$

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

$\tilde{s}_j$ is a **surrogate split** of s if $p(s, \tilde{s}_j) = max_{s_j} p(s, s_j)$ The importance of the variable j is given by

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

$\tilde{s}_j$ is a **surrogate split** of s if $p(s, \tilde{s}_j) = max_{s_j} p(s, s_j)$ The importance of the variable j is given by

$$Imp(X_j) = \sum_t \Delta(R(\tilde{s}_j, t))$$

Surrogate splits and variable importance

We define an association measure between s_j and s

$$p(s, s_j) = p_{LL} + p_{RR}$$

$\tilde{s}_j$ is a **surrogate split** of s if $p(s, \tilde{s}_j) = max_{s_j} p(s, s_j)$ The importance of the variable j is given by

$$Imp(X_j) = \sum_t \Delta(R(\tilde{s}_j, t))$$

which is the **loss of deviance** induced if each node is replaced by the surrogate split defined on X_j

Conclusion

- CUBT is an interpretable clustering method

Conclusion

- CUBT is an interpretable clustering method
- Measure of variable importance in CUBT

Conclusion

- CUBT is an interpretable clustering method
- Measure of variable importance in CUBT
- Heuristics have been proposed for tuning the method

Conclusion

- CUBT is an interpretable clustering method
- Measure of variable importance in CUBT
- Heuristics have been proposed for tuning the method
- Stability of variable importance

4 Applications to Economics: Causal Trees

Causal trees

Recursive partitioning for heterogeneous causal effect . Susan Athey and Guido Imbens (2016). In PNAS.

Causal trees

Recursive partitioning for heterogeneous causal effect . Susan Athey and Guido Imbens (2016). In PNAS.

- Estimating **heterogeneity in causal effects**

Causal trees

Recursive partitioning for heterogeneous causal effect . Susan Athey and Guido Imbens (2016). In PNAS.

- Estimating **heterogeneity in causal effects**
- Evaluating the **differences in treatment effects** across subsets of population

Causal trees

Recursive partitioning for heterogeneous causal effect . Susan Athey and Guido Imbens (2016). In PNAS.

- Estimating **heterogeneity in causal effects**
- Evaluating the **differences in treatment effects** across subsets of population
- Using **recursive binary partitioning**

Causal trees

Recursive partitioning for heterogeneous causal effect . Susan Athey and Guido Imbens (2016). In PNAS.

- Estimating **heterogeneity in causal effects**
- Evaluating the **differences in treatment effects** across subsets of population
- Using **recursive binary partitioning**

This approach is based on regression trees, modified to take into account the treatment effects in causal models

The problem

- We have N **units**, indexed by $i = 1, ..., N$

The problem

- We have N **units**, indexed by $i = 1, ..., N$
- We suppose the existence of **potential outcomes** for each unit, $(Y_i(0), Y_i(1))$ (Rubin Causal Model)

The problem

- We have N **units**, indexed by $i = 1, ..., N$
- We suppose the existence of **potential outcomes** for each unit, $(Y_i(0), Y_i(1))$ (Rubin Causal Model)
- **Difference in potential outcomes**, $\tau_i = Y_i(1) - Y_i(0)$

The problem

- We have N **units**, indexed by $i = 1, ..., N$
- We suppose the existence of **potential outcomes** for each unit, $(Y_i(0), Y_i(1))$ (Rubin Causal Model)
- **Difference in potential outcomes**, $\tau_i = Y_i(1) - Y_i(0)$
- $W_i \in \{0, 1\}$, binary indicator of the treatment (0 for control, 1 for treatment)

The problem

- We have N **units**, indexed by $i = 1, ..., N$
- We suppose the existence of **potential outcomes** for each unit, $(Y_i(0), Y_i(1))$ (Rubin Causal Model)
- **Difference in potential outcomes**, $\tau_i = Y_i(1) - Y_i(0)$
- $W_i \in \{0, 1\}$, binary indicator of the treatment (0 for control, 1 for treatment)
- The outcome is

$$Y_i^{obs} = Y_i(W_i) = \begin{cases} Y_i(0) & \text{if } W_i = 0 \\ Y_i(1) & \text{if } W_i = 1 \end{cases}$$

The problem

Let X_i be a **vector of features** (covariates, variables), not affected by the treatment

The problem

Let X_i be a **vector of features** (covariates, variables), not affected by the treatment

We assume **unconfoundedness**, $W_i(Y_i(0), Y_i(1))|X_i$

The problem

Let X_i be a **vector of features** (covariates, variables), not affected by the treatment

We assume **unconfoundedness**, $W_i(Y_i(0), Y_i(1))|X_i$

We try to estimate the **conditional average treatment effect** (CATE)

$$\tau(x) = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = x]$$

Honest estimation

Causal trees are based on CART, the difference is in the estimation:

Honest estimation

Causal trees are based on CART, the difference is in the estimation:

- Estimation CATE rather than predicting outcomes

Honest estimation

Causal trees are based on CART, the difference is in the estimation:

- Estimation CATE rather than predicting outcomes
- "Honest" estimation: use separate datasets for constructing the partition (the regions) and estimating the effects within the leaves

Honest estimation

Causal trees are based on CART, the difference is in the estimation:

- Estimation CATE rather than predicting outcomes
- "Honest" estimation: use separate datasets for constructing the partition (the regions) and estimating the effects within the leaves
- In CART, the estimation is "adaptive" (same dataset)

Honest estimation

Suppose the feature space is the union of two subsamples L and R

Honest estimation

Suppose the feature space is the union of two subsamples L and R

Given a sample S, the average outcomes in the two subsamples are $\bar{Y}_L$ and $\bar{Y}_R$

Honest estimation

Suppose the feature space is the union of two subsamples L and R

Given a sample S, the average outcomes in the two subsamples are $\bar{Y}_L$ and $\bar{Y}_R$

Algorithm: Split the feature space if the difference in average outcomes exceeds a threshold c:

Honest estimation

Suppose the feature space is the union of two subsamples L and R

Given a sample S, the average outcomes in the two subsamples are $\bar{Y}_L$ and $\bar{Y}_R$

Algorithm: Split the feature space if the difference in average outcomes exceeds a threshold c:

$$\pi(S) = \begin{cases} \{\{L, R\}\} & \text{if } \bar{Y}_L - \bar{Y}_R \leq c \\ \{\{L\}, \{R\}\} & \text{if } \bar{Y}_L - \bar{Y}_R > c \end{cases}$$

Honest estimation

- Let Π a **tree** defining a partition on the feature space.

Honest estimation

- Let Π a **tree** defining a partition on the feature space.
- Let $l(x, \Pi)$ the leaf $l \in \Pi$ such that $x \in l$

Honest estimation

- Let Π a **tree** defining a partition on the feature space.
- Let $I(x, \Pi)$ the leaf $I \in \Pi$ such that $x \in I$
- The estimated outcomes is

Honest estimation

- Let Π a **tree** defining a partition on the feature space.
- Let $l(x, \Pi)$ the leaf $l \in \Pi$ such that $x \in l$
- The estimated outcomes is

$$\hat{\mu}(x, S, \Pi) = \frac{1}{\#(i \in S : X_i \in l(x, \Pi)} \sum_{i \in S : X_i \in l(x, \Pi)} Y_i$$

Honest target

The goal is to find a partition Π, that **minimizes** the **mean square error** (MSE) computed using a test sample S^{test} and an estimation sample S^{est}, as follows:

Honest target

The goal is to find a partition Π, that **minimizes** the **mean square error** (MSE) computed using a test sample S^{test} and an estimation sample S^{est}, as follows:

$$MSE(S^{test}, S^{est}, \Pi) = \frac{1}{\#(S^{test})} \sum_{i \in S^{test}} ((Y_i - \hat{\mu}(x, S^{est}, \Pi)) - Y_i^2)$$

Honest target

The goal is to find a partition Π, that **minimizes** the **mean square error** (MSE) computed using a test sample S^{test} and an estimation sample S^{est}, as follows:

$$MSE(S^{test}, S^{est}, \Pi) = \frac{1}{\#(S^{test})} \sum_{i \in S^{test}} ((Y_i - \hat{\mu}(x, S^{est}, \Pi)) - Y_i^2)$$

In the same way, the authors also define "honest" splitting and cross-valdation.

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation
- New method for constructing trees for causal effects

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation
- New method for constructing trees for causal effects
- Obtain a set of treatment effects and confidence intervals in each region

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation
- New method for constructing trees for causal effects
- Obtain a set of treatment effects and confidence intervals in each region
- Can be used to explore **randomized controlled trials** (medical studies, field experiments)

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation
- New method for constructing trees for causal effects
- Obtain a set of treatment effects and confidence intervals in each region
- Can be used to explore **randomized controlled trials** (medical studies, field experiments)
- Identify subsets of population with lower-than-average or higher-than-average treatment effects

Conclusion and further reading...

- This approach was compared to 3 other approaches (transformed outcome trees, fit-based trees, square T-statistic trees)
- Honest versus adaptative estimation
- New method for constructing trees for causal effects
- Obtain a set of treatment effects and confidence intervals in each region
- Can be used to explore **randomized controlled trials** (medical studies, field experiments)
- Identify subsets of population with lower-than-average or higher-than-average treatment effects
- From trees to forests: *Generalized Random Forests.* Susan Athey, Julie Tibshirani and Stefan Wager(2018). In arxiv.

End of lecture.