Cellular Tree Classifiers

Gérard Biau & Luc Devroye

Paris, December 2013
Outline

1. Context
2. Cellular tree classifiers
3. A mathematical model
4. Are there consistent cellular tree classifiers?
5. A non-randomized solution
6. Random forests
Big Data is a collection of data sets so large and complex that it becomes impossible to process using classical tools.
Big Data is a collection of data sets so large and complex that it becomes impossible to process using classical tools.

It includes data sets with sizes beyond the ability of commonly-used software tools to process the data within a tolerable elapsed time.
Big Data is a collection of data sets so large and complex that it becomes impossible to process using classical tools.

It includes data sets with sizes beyond the ability of commonly-used software tools to process the data within a tolerable elapsed time.

As of 2012, every day 2.5 quintillion \((2.5 \times 10^{18})\) bytes of data were created.
The Big Data era

- Big Data is a collection of data sets so large and complex that it becomes impossible to process using classical tools.

- It includes data sets with sizes beyond the ability of commonly-used software tools to process the data within a tolerable elapsed time.

- As of 2012, every day 2.5 quintillion \((2.5 \times 10^{18})\) bytes of data were created.

- Megabytes and gigabytes are old-fashioned.
The challenges involved in Big Data problems are interdisciplinary.
The Big Data era

- The challenges involved in Big Data problems are **interdisciplinary**.

- **Data analysis** in the Big Data regime requires consideration of:
The Big Data era

- The challenges involved in Big Data problems are **interdisciplinary**.

- **Data analysis** in the Big Data regime requires consideration of:
  
  - **Systems issues**: How to store, index and transport data at massive scales?
The Big Data era

- The challenges involved in Big Data problems are interdisciplinary.

- Data analysis in the Big Data regime requires consideration of:

  - **Systems issues**: How to store, index and transport data at massive scales?

  - **Statistical issues**: How to cope with errors and biases of all kinds? How to develop models and procedures that work when both $n$ and $p$ are astronomical?
The Big Data era

The challenges involved in Big Data problems are interdisciplinary.

Data analysis in the Big Data regime requires consideration of:

▷ Systems issues: How to store, index and transport data at massive scales?

▷ Statistical issues: How to cope with errors and biases of all kinds? How to develop models and procedures that work when both $n$ and $p$ are astronomical?

▷ Algorithmic issues: How to perform computations?
The challenges involved in Big Data problems are interdisciplinary.

Data analysis in the Big Data regime requires consideration of:

- **Systems issues**: How to store, index and transport data at massive scales?

- **Statistical issues**: How to cope with errors and biases of all kinds? How to develop models and procedures that work when both $n$ and $p$ are astronomical?

- **Algorithmic issues**: How to perform computations?

Big Data requires massively parallel softwares running on tens, hundreds, or even thousands of servers.
Greedy algorithms build solutions incrementally, usually with little effort.
Greedy algorithms build solutions incrementally, usually with little effort.

Such procedures form a result piece by piece, always choosing the next item that offers the most obvious and immediate benefit.
Greedy algorithms

- **Greedy algorithms** build solutions incrementally, usually with little effort.

- Such procedures form a result *piece by piece*, always choosing the next item that offers the most obvious and immediate benefit.

- Greedy methods have an *autonomy* that makes them ideally suited for *distributive or parallel computation*.
Greedy algorithms

- Our goal is to formalize the setting and to provide a foundational discussion of various properties of tree classifiers that are designed following these principles.
Greedy algorithms

- Our goal is to formalize the setting and to provide a foundational discussion of various properties of tree classifiers that are designed following these principles.

- They may find use in a world with new computational models in which parallel or distributed computation is feasible and even the norm.
Outline

1 Context

2 Cellular tree classifiers

3 A mathematical model

4 Are there consistent cellular tree classifiers?

5 A non-randomized solution

6 Random forests
Classification
Basics of classification

- We have a random pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\) with an unknown distribution.
Basics of classification

- We have a random pair $(X, Y) \in \mathbb{R}^d \times \{0, 1\}$ with an unknown distribution.

- **Goal:** Design a classifier $g : \mathbb{R}^d \rightarrow \{0, 1\}$. 
Basics of classification

- We have a random pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\) with an unknown distribution.

- **Goal**: Design a classifier \(g : \mathbb{R}^d \rightarrow \{0, 1\}\).

- The probability of error is \(L(g) = \mathbb{P}\{g(X) \neq Y\}\).
Basics of classification

- We have a random pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\) with an unknown distribution.

- **Goal:** Design a classifier \(g : \mathbb{R}^d \to \{0, 1\}\).

- The probability of error is \(L(g) = \mathbb{P}\{g(X) \neq Y\}\).

- The Bayes classifier

  \[
g^*(x) = \begin{cases} 
  1 & \text{if } \mathbb{P}\{Y = 1|X = x\} > 1/2 \\
  0 & \text{otherwise}
\end{cases}
\]

has the smallest probability of error, that is

\[
L^* = L(g^*) = \inf_{g: \mathbb{R}^d \to \{0, 1\}} \mathbb{P}\{g(X) \neq Y\}.
\]
Basics of classification

- The data: \( D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \), i.i.d. copies of \((X, Y)\).
Basics of classification

- The data: $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, i.i.d. copies of $(X, Y)$.

- A classifier $g_n(x)$ is a function of $x$ and $\mathcal{D}_n$. 

Basics of classification

- **The data**: \( \mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \), i.i.d. copies of \((X, Y)\).

- A classifier \( g_n(x) \) is a function of \( x \) and \( \mathcal{D}_n \).

- The **probability of error** is

  \[
  L(g_n) = \mathbb{P}\{g_n(X) \neq Y | \mathcal{D}_n\}.
  \]
Basics of classification

- The data: $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, i.i.d. copies of $(X, Y)$.

- A classifier $g_n(x)$ is a function of $x$ and $\mathcal{D}_n$.

- The probability of error is
  \[
  L(g_n) = \mathbb{P}\{g_n(X) \neq Y | \mathcal{D}_n\}.
  \]

- It is consistent if
  \[
  \lim_{n \to \infty} \mathbb{E}L(g_n) = L^*.
  \]
The data: \( D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \), i.i.d. copies of \((X, Y)\).

A classifier \( g_n(x) \) is a function of \( x \) and \( D_n \).

The probability of error is
\[
L(g_n) = \mathbb{P}\{g_n(X) \neq Y | D_n\}.
\]

It is consistent if
\[
\lim_{n \to \infty} \mathbb{E}L(g_n) = L^*.
\]

It is universally consistent if it is consistent for all possible distributions of \((X, Y)\).
Tree classifiers

- Many popular classifiers are universally consistent.
Tree classifiers

- Many popular classifiers are universally consistent.

- These include several brands of histogram rules, $k$-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.
Many popular classifiers are universally consistent.

These include several brands of histogram rules, $k$-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.

Tree methods loom large for several reasons:
Many popular classifiers are universally consistent.

These include several brands of histogram rules, \( k \)-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.

Tree methods loom large for several reasons:

▷ All procedures that partition space can be viewed as special cases of partitions generated by trees.
Many popular classifiers are universally consistent.

These include several brands of histogram rules, \( k \)-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.

Tree methods **loom large** for several reasons:

- All procedures that **partition space** can be viewed as special cases of partitions generated by trees.
- Tree classifiers are **conceptually simple**, and explain the data very well.
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
The tree structure is usually data dependent, and it is in the construction itself that trees differ.
The tree structure is usually data dependent, and it is in the construction itself that trees differ.

Thus, there are virtually infinitely many possible strategies to build classification trees.
The tree structure is usually data dependent, and it is in the construction itself that trees differ.

Thus, there are virtually infinitely many possible strategies to build classification trees.

Despite this great diversity, all tree species end up with two fundamental questions at each node:

1. Should the node be split?
2. In the affirmative, what are its children?
Cellular trees proceed from a different philosophy.
The cellular spirit

- Cellular trees proceed from a different philosophy.
- A cellular tree should be able to answer questions ① and ② using local information only.
1. Context
2. Cellular tree classifiers
3. **A mathematical model**
4. Are there consistent cellular tree classifiers?
5. **A non-randomized solution**
6. Random forests
Let $C$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.
Let $C$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.

Example: $C = \{\text{hyperplanes of } \mathbb{R}^d\}$. 
A model

- Let $C$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.

- Example: $C = \{ \text{hyperplanes of } \mathbb{R}^d \}$.

- The class is parametrized by a vector $\sigma \in \mathbb{R}^p$. 
Let $\mathcal{C}$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.

Example: $\mathcal{C} = \{ \text{hyperplanes of } \mathbb{R}^d \}$.

The class is parametrized by a vector $\sigma \in \mathbb{R}^p$.

There is a splitting function $f(x, \sigma)$ such that $\mathbb{R}^d$ is partitioned into

$$A = \{ x \in \mathbb{R}^d : f(x, \sigma) \geq 0 \} \quad \text{and} \quad B = \{ x \in \mathbb{R}^d : f(x, \sigma) < 0 \}.$$
A cellular split may be seen as a family of mappings

\[ \sigma : (\mathbb{R}^d \times \{0, 1\})^n \to \mathbb{R}^p. \]
A model

A cellular split may be seen as a family of mappings

\[ \sigma : (\mathbb{R}^d \times \{0, 1\})^n \rightarrow \mathbb{R}^p. \]

We see that \( \sigma \) is a model for the cellular decision ②.
A model

- A cellular split may be seen as a family of mappings

\[ \sigma : (\mathbb{R}^d \times \{0, 1\})^n \rightarrow \mathbb{R}^p. \]

- We see that \( \sigma \) is a model for the cellular decision ②.

- In addition, there is a second family of mappings \( \theta \), but this time with a boolean output.
A cellular split may be seen as a family of mappings

$$\sigma : (\mathbb{R}^d \times \{0, 1\})^n \rightarrow \mathbb{R}^p.$$  

We see that $\sigma$ is a model for the cellular decision ②.

In addition, there is a second family of mappings $\theta$, but this time with a boolean output.

It is a stopping rule and models the cellular decision ①.
A cellular machine partitions the data recursively.
A cellular machine partitions the data recursively.

- If $\theta(D_n) = 0$, the root cell is final, and the space is not split.
A cellular machine partitions the data \textit{recursively}.

- If \( \theta(D_n) = 0 \), the root cell is \textit{final}, and the space is \textit{not split}.
- Otherwise, \( \mathbb{R}^d \) is \textit{split} into

\[
A = \{ x : f(x, \sigma(D_n)) \geq 0 \} \quad \text{and} \quad B = \{ x : f(x, \sigma(D_n)) < 0 \}.
\]
A cellular machine partitions the data recursively.

- If $\theta(D_n) = 0$, the root cell is final, and the space is not split.
- Otherwise, $\mathbb{R}^d$ is split into

$$A = \{x : f(x, \sigma(D_n)) \geq 0\} \quad \text{and} \quad B = \{x : f(x, \sigma(D_n)) < 0\}.$$  
- The data $D_n$ are partitioned into two groups.
A cellular machine partitions the data recursively.

- If $\theta(D_n) = 0$, the root cell is final, and the space is not split.
- Otherwise, $\mathbb{R}^d$ is split into

$$A = \{ x : f(x, \sigma(D_n)) \geq 0 \} \quad \text{and} \quad B = \{ x : f(x, \sigma(D_n)) < 0 \} .$$

- The data $D_n$ are partitioned into two groups.
- The groups are sent to child cells, and the process is repeated.
Cellular procedure

- A cellular machine partitions the data **recursively**.

  ▶ If $\theta(D_n) = 0$, the root cell is **final**, and the space is **not split**.

  ▶ Otherwise, $\mathbb{R}^d$ is split into

    $$ A = \{x : f(x, \sigma(D_n)) \geq 0\} \quad \text{and} \quad B = \{x : f(x, \sigma(D_n)) < 0\} \, . $$

  ▶ The data $D_n$ are partitioned into **two groups**.

  ▶ The groups are sent to child cells, and the process is **repeated**.

- Final classification proceeds by a **majority vote**.
An important remark

Is any classifier a cellular tree?
An important remark

Is any classifier a cellular tree?

Set $\theta = 1$ if we care at the root, and $\theta = 0$ elsewhere.

The root node is split by the classifier into a set

$$A = \{x \in \mathbb{R}^d : g_n(x) = 1\}$$

and its complement, and both child nodes are leaves.
An important remark

Is any classifier a cellular tree?

1. Set $\theta = 1$ if we care at the root, and $\theta = 0$ elsewhere.
2. The root node is split by the classifier into a set

$$A = \{x \in \mathbb{R}^d : g_n(x) = 1\}$$

and its complement, and both child nodes are leaves.

This is not allowed.
Outline

1. Context
2. Cellular tree classifiers
3. A mathematical model
4. Are there consistent cellular tree classifiers?
5. A non-randomized solution
6. Random forests
An example

- At first sight, there are *no* consistent cellular tree classifiers.
An example

- At first sight, there are no consistent cellular tree classifiers.

- **Example:** The $k$-median tree.
At first sight, there are no consistent cellular tree classifiers.

Example: The $k$-median tree.

When $d = 1$, split by finding the median element among the $X_i$'s.
At first sight, there are **no** consistent cellular tree classifiers.

**Example:** The $k$-median tree.

- When $d = 1$, split by finding the **median** element among the $X_i$'s.
- Keep doing this for $k$ rounds.
An example

- At first sight, there are no consistent cellular tree classifiers.

- **Example:** The $k$-median tree.
  - When $d = 1$, split by finding the median element among the $X_i$’s.
  - Keep doing this for $k$ rounds.
  - In $d$ dimensions, rotate through the coordinates.
An example

- At first sight, there are no consistent cellular tree classifiers.

- Example: The $k$-median tree.
  - When $d = 1$, split by finding the median element among the $X_i$’s.
  - Keep doing this for $k$ rounds.
  - In $d$ dimensions, rotate through the coordinates.

- This rule is consistent, provided $k \to \infty$ and $k2^k/n \to 0$. 
An example

- At first sight, there are no consistent cellular tree classifiers.

- Example: The $k$-median tree.
  - When $d = 1$, split by finding the median element among the $X_i$’s.
  - Keep doing this for $k$ rounds.
  - In $d$ dimensions, rotate through the coordinates.

- This rule is consistent, provided $k \to \infty$ and $k 2^k/n \to 0$.

This is not cellular.
The cellular splitting method $\sigma$ mimics the median tree classifier.
A randomized solution

- The cellular splitting method \( \sigma \) mimics the median tree classifier.

- The dimension to cut is chosen \textit{uniformly at random}. 
A randomized solution

- The cellular splitting method $\sigma$ mimics the median tree classifier.

- The dimension to cut is chosen uniformly at random.

- The selected dimension is then split at the median.
A randomized solution

- The cellular splitting method $\sigma$ mimics the median tree classifier.
- The dimension to cut is chosen uniformly at random.
- The selected dimension is then split at the median.
- The novelty is in the choice of the decision function $\theta$. 
A randomized solution

- The cellular splitting method $\sigma$ mimics the median tree classifier.

- The dimension to cut is chosen uniformly at random.

- The selected dimension is then split at the median.

- The novelty is in the choice of the decision function $\theta$.

- This function ignores the data altogether and uses a randomized decision that is based on the size of the input.
Consider a nonincreasing function $\varphi : \mathbb{N} \to (0, 1]$. 
Consider a nonincreasing function $\varphi : \mathbb{N} \rightarrow (0, 1]$.

Then, if $U$ is the uniform $[0, 1]$ random variable associated with node $A$,

$$\theta = 1_{[U > \varphi(N(A))]}.$$
Consistency

- Consider a nonincreasing function \( \varphi : \mathbb{N} \to (0, 1] \).

- Then, if \( U \) is the uniform \([0, 1]\) random variable associated with node \( A \),

\[
\theta = 1_{\{U > \varphi(N(A))\}}.
\]

**Theorem**

Let \( \beta \) be a real number in \((0, 1)\). Define

\[
\varphi(n) = \begin{cases} 
1 & \text{if } n < 3 \\
1/\log^\beta n & \text{if } n \geq 3
\end{cases}
\]

Then

\[
\lim_{n \to \infty} \mathbb{E} L(g_n) = L^* \quad \text{as } n \to \infty.
\]
Outline

1. Context
2. Cellular tree classifiers
3. A mathematical model
4. Are there consistent cellular tree classifiers?
5. A non-randomized solution
6. Random forests
A full $2^d$-ary tree

- At the root, we find the median in direction 1.
A full $2^d$-ary tree

- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.
A full $2^d$-ary tree

- At the root, we find the median in direction 1.

- Then on each of the two subsets, we find the median in direction 2.

- Then on each of the four subsets, we find the median in direction 3, and so forth.
A full $2^d$-ary tree

- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.
- Then on each of the four subsets, we find the median in direction 3, and so forth.
- Repeating this for $k$ levels of nodes leads to $2^{dk}$ leaf regions.
The stopping rule $\theta$

The quality of the classifier at node $A$ is assessed by

$$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[X_i \in A, Y_i = 1], \sum_{i=1}^{n} 1[X_i \in A, Y_i = 0] \right).$$
The stopping rule $\theta$

- The **quality** of the classifier at node $A$ is assessed by

$$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1_{X_i \in A, Y_i = 1}, \sum_{i=1}^{n} 1_{X_i \in A, Y_i = 0} \right).$$

- Define the **nonnegative integer** $k^+$ by

$$k^+ = \left\lfloor \alpha \log_2 (N(A) + 1) \right\rfloor.$$
The stopping rule $\theta$

- The **quality** of the classifier at node $A$ is assessed by

$$
\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, Y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, Y_i = 0] \right).
$$

- Define the **nonnegative integer** $k^+$ by

$$
k^+ = \lfloor \alpha \log_2 (N(A) + 1) \rfloor.
$$

- Set

$$
\hat{L}_n(A, k^+) = \sum_{A_j \in \mathcal{P}_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.
$$
The stopping rule $\theta$

- The quality of the classifier at node $A$ is assessed by

$$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, y_i = 0] \right).$$

- Define the nonnegative integer $k^+$ by

$$k^+ = \lfloor \alpha \log_2 (N(A) + 1) \rfloor.$$

- Set

$$\hat{L}_n(A, k^+) = \sum_{A_j \in \mathcal{P}_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.$$

- Both $\hat{L}_n(A)$ and $\hat{L}_n(A, k^+)$ may be evaluated on the basis of the data points falling in $A$ only.
The stopping rule $\theta$

- The **quality** of the classifier at node $A$ is assessed by
  
  $$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, y_i = 0] \right).$$

- Define the **nonnegative integer** $k^+$ by
  
  $$k^+ = \lfloor \alpha \log_2(N(A) + 1) \rfloor.$$

- Set
  
  $$\hat{L}_n(A, k^+) = \sum_{A_j \in \mathcal{P}_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.$$

- Both $\hat{L}_n(A)$ and $\hat{L}_n(A, k^+)$ may be evaluated on the basis of the data points falling in $A$ only.

This is **cellular**.
The stopping rule $\theta$

Put $\theta = 0$ if

$$\left| \hat{L}_n(A) - \hat{L}_n(A, k^+) \right| \leq \left( \frac{1}{N(A) + 1} \right)^\beta.$$
The stopping rule $\theta$

Put $\theta = 0$ if

$$\left| \hat{L}_n(A) - \hat{L}_n(A, k^+) \right| \leq \left( \frac{1}{N(A) + 1} \right)^{\beta}.$$

Theorem

Take $1 - d\alpha - 2\beta > 0$. Then

$$\lim_{n \to \infty} \mathbb{E}L(g_n) = L^* \quad \text{as} \ n \to \infty.$$
1. Context
2. Cellular tree classifiers
3. A mathematical model
4. Are there consistent cellular tree classifiers?
5. A non-randomized solution
6. Random forests
Leo Breiman (1928-2005)
Leo Breiman (1928-2005)
From trees to forests

- Leo Breiman promoted random forests.
Leo Breiman promoted random forests.

Idea: Using tree averaging as a means of obtaining good rules.
Leo Breiman promoted random forests.

Idea: Using tree averaging as a means of obtaining good rules.

The base trees are simple and randomized.
Leo Breiman promoted random forests.

Idea: Using tree averaging as a means of obtaining good rules.

The base trees are simple and randomized.

Breiman’s ideas were decisively influenced by


Ho (1998, random subspace method).

Leo Breiman promoted random forests.

Idea: Using tree averaging as a means of obtaining good rules.

The base trees are simple and randomized.

Breiman’s ideas were decisively influenced by


stat.berkeley.edu/users/breiman/RandomForests
Random forests

- They are serious competitors to state-of-the-art methods.
Random forests

- They are serious competitors to state-of-the-art methods.
- They are fast and easy to implement.
Random forests

- They are serious competitors to state-of-the-art methods.
- They are fast and easy to implement.
- They can handle a very large number of input variables.
Random forests

- They are serious competitors to state-of-the-art methods.
- They are fast and easy to implement.
- They can handle a very large number of input variables.

- The algorithm is difficult to analyze and its mathematical properties remain to date largely unknown.
Random forests

- They are serious competitors to state-of-the-art methods.
- They are fast and easy to implement.
- They can handle a very large number of input variables.

- The algorithm is difficult to analyze and its mathematical properties remain to date largely unknown.
- Most theoretical studies have concentrated on isolated parts or stylized versions of the procedure.
Three basic ingredients

1. Randomization and no-pruning
   - For each tree, select at random, at each node, a small group of input coordinates to split.
Three basic ingredients

1. Randomization and no-pruning

- For each tree, select at random, at each node, a small group of input coordinates to split.

- Calculate the best split based on these features and cut.
Three basic ingredients

1. Randomization and no-pruning
   - For each tree, select at random, at each node, a small group of input coordinates to split.
   - Calculate the best split based on these features and cut.
   - The tree is grown to maximum size.
Three basic ingredients

2. Aggregation

- Final predictions are obtained by aggregating over the ensemble.
Three basic ingredients

2. Aggregation

- Final predictions are obtained by aggregating over the ensemble.
- It is fast and easily parallelizable.
Three basic ingredients

3. Bagging

- The subspace randomization scheme is blended with bagging.
Three basic ingredients

3. Bagging

- The subspace randomization scheme is blended with bagging.
- Bühlmann and Yu (2002).
- Biau, Cérou and Guyader (2010).
A training sample: \( \mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \).
A training sample: $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$.

A generic pair: $(X, Y)$ satisfying $\mathbb{E}Y^2 < \infty$. 
A training sample: $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$.

A generic pair: $(X, Y)$ satisfying $\mathbb{E}Y^2 < \infty$.

Our goal: Estimate the regression function $r(x) = \mathbb{E}[Y|X = x]$. 
A training sample: \( \mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \).

A generic pair: \((X, Y)\) satisfying \( \mathbb{E}Y^2 < \infty \).

Our goal: Estimate the regression function \( r(x) = \mathbb{E}[Y|X = x] \).

Quality criterion: \( \mathbb{E}[r_n(X) - r(X)]^2 \).
A random forest is a collection of randomized base regression trees

\[ \{r_n(x, \Theta_m, D_n), m \geq 1\} \]
The model

- A random forest is a collection of randomized base regression trees

\[ \{ r_n(\mathbf{x}, \Theta_m, \mathcal{D}_n), m \geq 1 \} \].

- These random trees are combined to form the aggregate

\[ \bar{r}_n(\mathbf{X}, \mathcal{D}_n) = \mathbb{E}_\Theta [ r_n(\mathbf{X}, \Theta, \mathcal{D}_n) ] . \]
The model

- A random forest is a collection of randomized base regression trees
  \[ \{r_n(x, \Theta_m, D_n), m \geq 1\} \].

- These random trees are combined to form the aggregate
  \[ \bar{r}_n(X, D_n) = \mathbb{E}_{\Theta} [r_n(X, \Theta, D_n)] \].

- \( \Theta \) is independent of \( X \) and the training sample \( D_n \).
The procedure
The procedure
The procedure
The procedure
The procedure
The forest

- A local averaging estimate:

\[
\bar{r}_n(x) = \frac{\sum_{i=1}^{n} K(x, X_i)Y_i}{\sum_{j=1}^{n} K(x, X_j)}.
\]
A local averaging estimate:

\[
\bar{r}_n(x) = \frac{\sum_{i=1}^{n} K(x, X_i)Y_i}{\sum_{j=1}^{n} K(x, X_j)}.
\]

Centered cuts:

\[
K(x, z) = \sum_{k_{n_1}, \ldots, k_{n_d}} \frac{k_n!}{k_{n_1}! \cdots k_{n_d}!} \prod_{j=1}^{d} p_{n_j}^{k_{n_j}} 1[k_{n_j} < \alpha_j].
\]
The forest

- A local averaging estimate:

\[ \tilde{r}_n(x) = \frac{\sum_{i=1}^{n} K(x, X_i)Y_i}{\sum_{j=1}^{n} K(x, X_j)}. \]

- Centered cuts:

\[ K(x, z) = \sum_{k_{n1}, \ldots, k_{nd} \atop \sum_{j=1}^{d} k_{nj} = k_n} \frac{k_n!}{k_{n1}! \ldots k_{nd}!} \prod_{j=1}^{d} p_{nj}^{k_{nj}} 1[k_{nj} < \alpha_j]. \]

- Uniform cuts:

\[ K(x, z) = \sum_{k_{n1}, \ldots, k_{nd} \atop \sum_{j=1}^{d} k_{nj} = k_n} \frac{k_n!}{k_{n1}! \ldots k_{nd}!} \prod_{m=1}^{d} p_{nm}^{k_{nm}} \frac{1}{k_{nm}!} \int_{0}^{\infty} -\log |x_m - z_m| t^{k_{nm}} e^{-t} dt. \]
Theorem

The random forests estimate \( \bar{r}_n \) is consistent whenever \( p_{nj} \log k_n \to \infty \) for all \( j = 1, \ldots, d \) and \( k_n/n \to 0 \) as \( n \to \infty \).
Consistency

The random forests estimate $\bar{r}_n$ is consistent whenever $p_{nj} \log k_n \to \infty$ for all $j = 1, \ldots, d$ and $k_n/n \to 0$ as $n \to \infty$.

- In the purely random model, $p_{nj} = 1/d$, independently of $n$ and $j$. 
Consistency

**Theorem**

The random forests estimate \( \bar{r}_n \) is **consistent** whenever \( p_{nj} \log k_n \to \infty \) for all \( j = 1, \ldots, d \) and \( k_n/n \to 0 \) as \( n \to \infty \).

- In the **purely random** model, \( p_{nj} = 1/d \), independently of \( n \) and \( j \).

- A more **in-depth** analysis is needed.
There is empirical evidence that many signals in high-dimensional spaces admit a sparse representation.
There is empirical evidence that many signals in high-dimensional spaces admit a sparse representation.

- Images wavelet coefficients.
There is empirical evidence that many signals in high-dimensional spaces admit a \textit{sparse representation}.

- Images wavelet coefficients.
- High-throughput technologies.
There is empirical evidence that many signals in high-dimensional spaces admit a **sparse representation**.

- Images wavelet coefficients.
- High-throughput technologies.

Sparse estimation is playing an **increasingly important role** in the statistics and machine learning communities.
There is empirical evidence that many signals in high-dimensional spaces admit a \textit{sparse representation}.

- Images wavelet coefficients.
- High-throughput technologies.

Sparse estimation is playing an \textit{increasingly important role} in the statistics and machine learning communities.

Several methods have recently been developed in both fields, which rely upon the notion of \textit{sparsity}.
The regression function $r(X) = \mathbb{E}[Y|X]$ depends in fact only on a nonempty subset $S$ of the $d$ features.
The regression function $r(X) = \mathbb{E}[Y|X]$ depends in fact only on a nonempty subset $S$ of the $d$ features.

In other words:

$$r(X) = \mathbb{E}[Y|X_S].$$
The regression function \( r(X) = \mathbb{E}[Y|X] \) depends in fact only on a nonempty subset \( S \) of the \( d \) features.

In other words:

\[
r(X) = \mathbb{E}[Y|X_S].
\]

In this dimension reduction scenario, the ambient dimension \( d \) can be very large, much larger than \( n \).
The regression function \( r(\mathbf{X}) = \mathbb{E}[Y|\mathbf{X}] \) depends in fact only on a nonempty subset \( S \) of the \( d \) features.

In other words:

\[
r(\mathbf{X}) = \mathbb{E}[Y|\mathbf{X}_S].
\]

In this dimension reduction scenario, the ambient dimension \( d \) can be very large, much larger than \( n \).

As such, the value \( S \) characterizes the sparsity: The smaller \( S \), the sparser \( r \).
Ideally, $p_{nj} = 1/S$ for $j \in S$. 
Ideally, $p_{nj} = \frac{1}{S}$ for $j \in S$.

To stick to reality, we will rather require

$$p_{nj} = \frac{1}{S} \left( 1 + \xi_{nj} \right).$$
Ideally, $p_{nj} = 1/S$ for $j \in S$.

To stick to reality, we will rather require

$$p_{nj} = \frac{1}{S} (1 + \xi_{nj}).$$

Today, an assumption. Tomorrow, a lemma.
Main result

**Theorem**

If $p_{nj} = (1/S)(1 + \xi_{nj})$ for $j \in S$, with $\xi_{nj} \log n \to 0$ as $n \to \infty$, then for the choice

$$k_n \propto n^{1/(1 + \frac{0.75}{S \log 2})},$$

we have

$$\limsup_{n \to \infty} \sup_{(X, Y) \in \mathcal{F}_S} \frac{\mathbb{E} \left[ r_n(X) - r(X) \right]^2}{n^{-0.75} \frac{1}{S \log 2 + 0.75}} \leq \Lambda.$$
Main result

Theorem

If \( p_{nj} = \frac{1}{S} (1 + \xi_{nj}) \) for \( j \in S \), with \( \xi_{nj} \log n \to 0 \) as \( n \to \infty \), then for the choice

\[
k_n \propto n^{1/(1 + \frac{0.75}{S \log 2})},
\]

we have

\[
\limsup_{n \to \infty} \sup_{(X,Y) \in F_S} \frac{\mathbb{E} \left[ \bar{r}_n(X) - r(X) \right]^2}{n^{-0.75} S \log 2 + 0.75} \leq \Lambda.
\]

Take-home message

The rate \( n^{-0.75} S \log 2 + 0.75 \) is strictly faster than the usual minimax rate \( n^{-2/(d+2)} \) as soon as \( S \leq \lceil 0.54d \rceil \).
Dimension reduction
Proof
Proof

Let $\psi(n, k) = L^*_k - L^*$. Set

$$k^*_n = \min \left\{ \ell \geq 0 : \psi(n, \ell) < \sqrt{\left( \frac{2d\ell}{n} \right)^{1-d\alpha}} \right\}.$$ 

Then

$$\frac{2d k^*_n}{n} \rightarrow 0 \quad \text{as} \quad n \rightarrow \infty.$$