Big Data & Greedy Trees

Gérard Biau & Luc Devroye

Lyon, December 2012

Outline

Outline

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.

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.

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 (2.5×10^{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 (2.5×10^{18}) bytes of data were created.
- Megabytes and gigabytes are old-fashioned.

The Big Data era

- 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 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?
- Big Data requires massively parallel softwares running on tens, hundreds, or even thousands of servers.

Greedy algorithms

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

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 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

- 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.
- In the short term, parallelism will take hold in massive datasets and complex systems.

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.

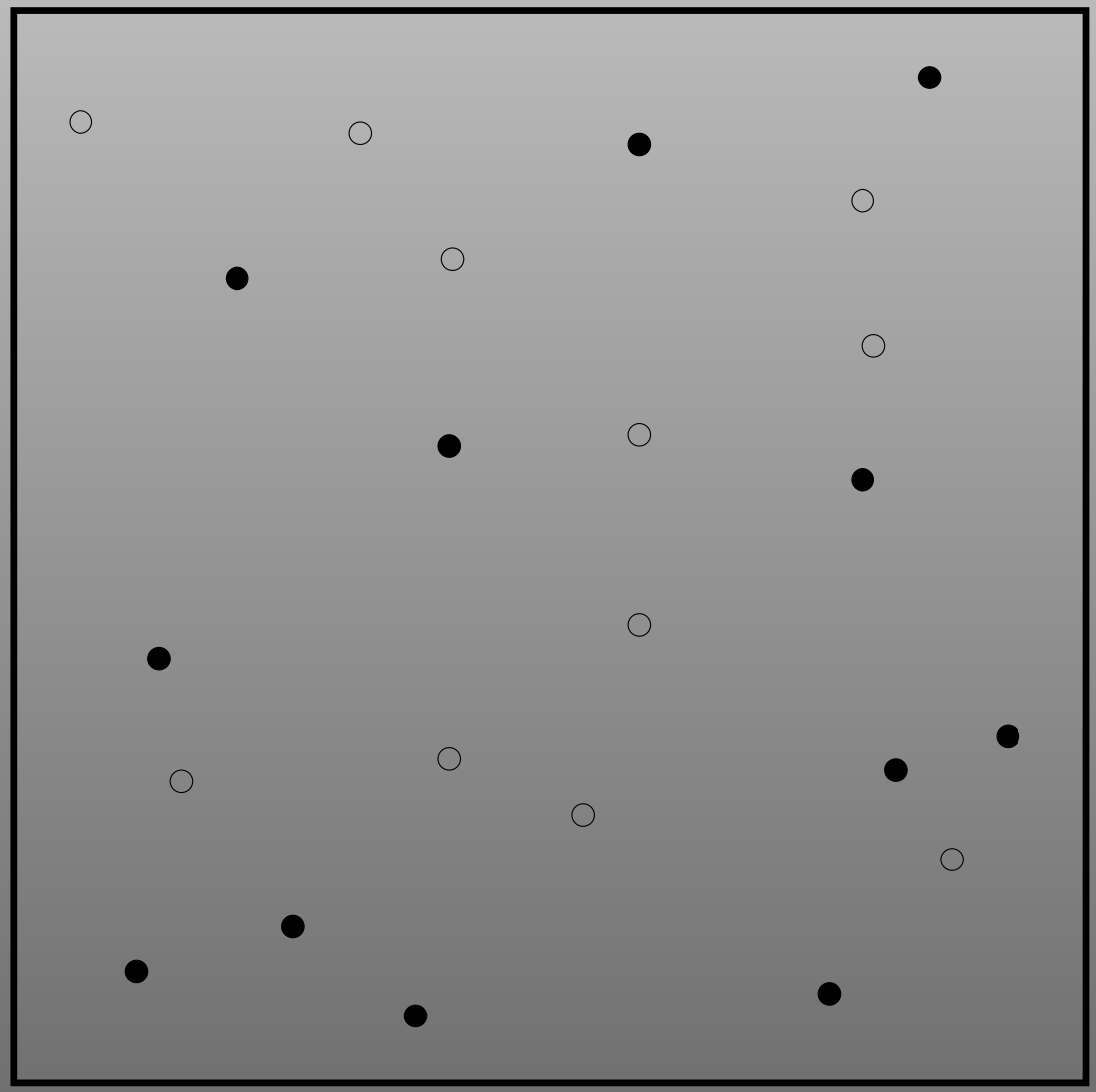
Greedy algorithms

- 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.

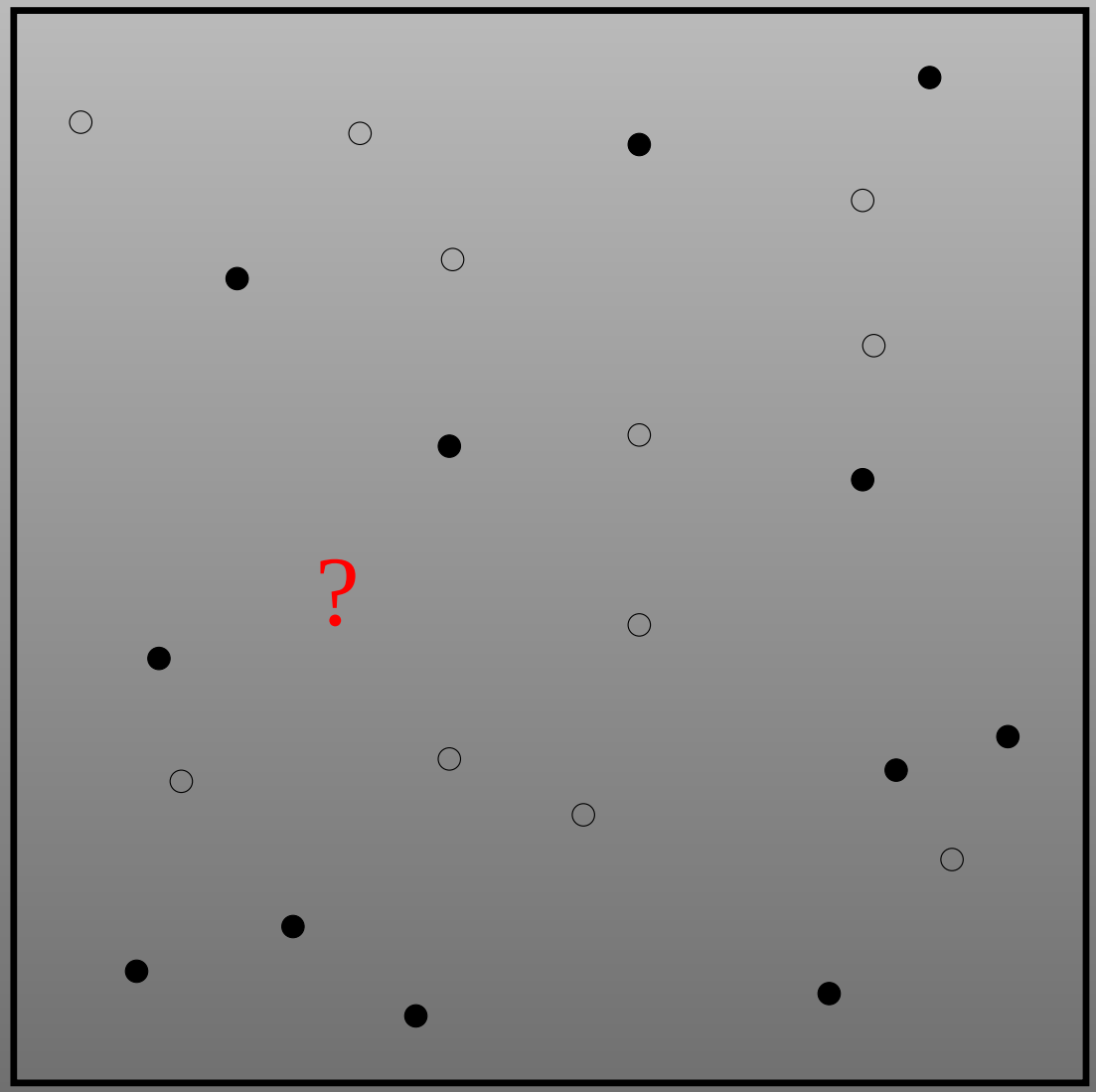
- They may find use in a world with new computational models in which parallel or distributed computation is feasible and even the norm.

Outline

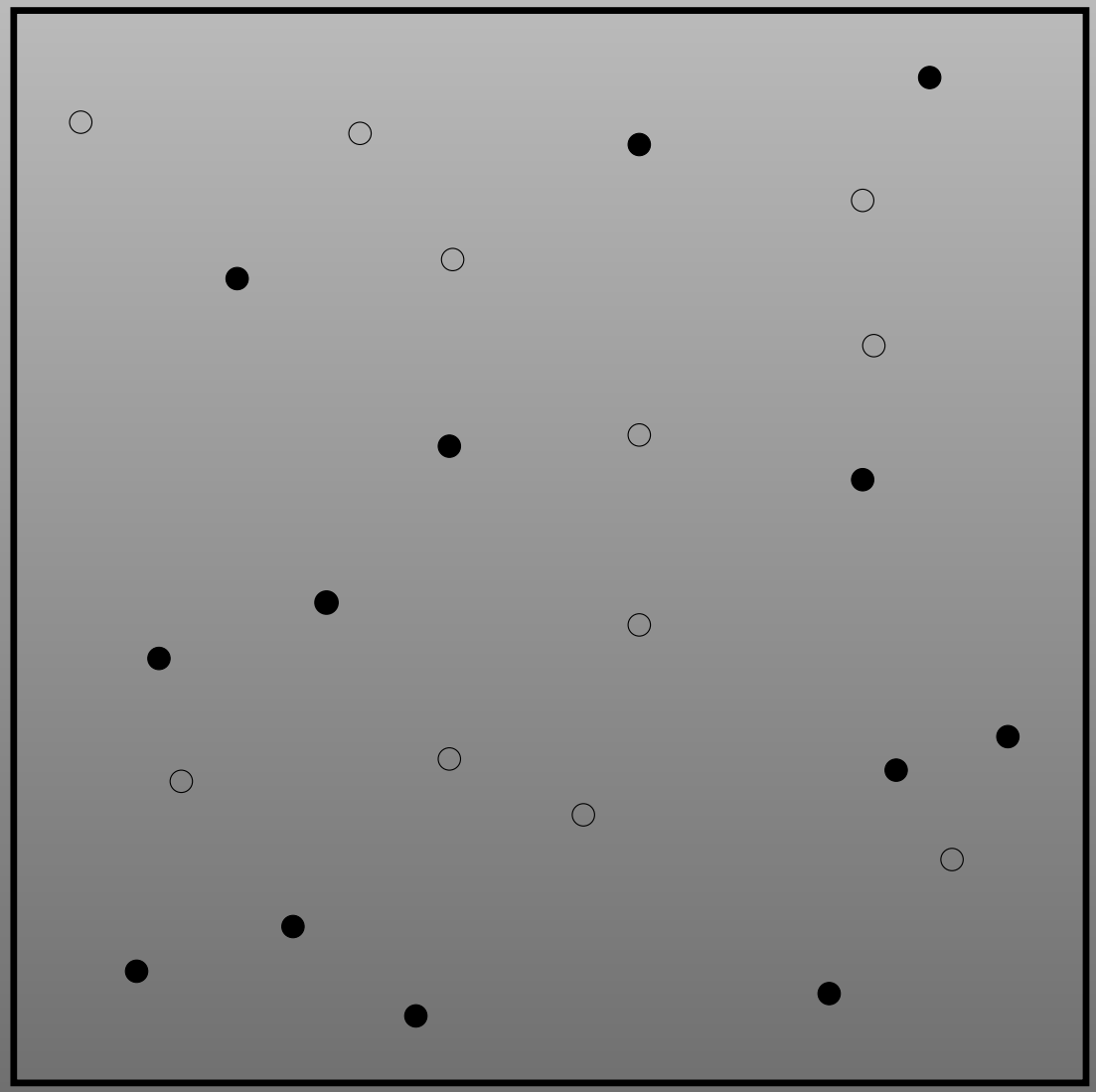
Classification



Classification



Classification



Basics of classification

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

Basics of classification

- We have a random pair $(\mathbf{X}, Y) \in \mathbb{R}^d \times \{0, 1\}$ with an unknown distribution.
- Goal: Design a measurable classifier $g : \mathbb{R}^d \to \{0, 1\}$.

Basics of classification

- We have a random pair $(\mathbf{X}, Y) \in \mathbb{R}^d \times \{0, 1\}$ with an unknown distribution.
- Goal: Design a measurable classifier $g : \mathbb{R}^d \to \{0, 1\}$.
- The probability of error is $L(g) = \mathbb{P}\{g(\mathbf{X}) \neq Y\}$.

Basics of classification

- We have a random pair $(\mathbf{X}, Y) \in \mathbb{R}^d \times \{0, 1\}$ with an unknown distribution.
- Goal: Design a measurable classifier $g : \mathbb{R}^d \to \{0, 1\}$.
- The probability of error is $L(g) = \mathbb{P}\{g(\mathbf{X}) \neq Y\}$.
- The Bayes classifier

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

has the smallest probability of error, that is

$$L^\star = L(g^\star) = \inf_{g:\mathbb{R}^d \to \{0,1\}} \mathbb{P}\{g(\mathbf{X}) \neq Y\}.$$

Basics of classification

- The data: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, i.i.d. copies of $(\mathbf{X}, Y)$.

Basics of classification

- The data: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, i.i.d. copies of $(\mathbf{X}, Y)$.
- A classifier $g_n(\mathbf{x})$ is a measurable function of $\mathbf{x}$ and $\mathcal{D}_n$.

Basics of classification

- The data: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, i.i.d. copies of $(\mathbf{X}, Y)$.
- A classifier $g_n(\mathbf{x})$ is a measurable function of $\mathbf{x}$ and $\mathcal{D}_n$.
- The probability of error is

$$L(g_n) = \mathbb{P}\{g_n(\mathbf{X}) \neq Y | \mathcal{D}_n\}.$$

Basics of classification

- The data: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, i.i.d. copies of $(\mathbf{X}, Y)$.
- A classifier $g_n(\mathbf{x})$ is a measurable function of $\mathbf{x}$ and $\mathcal{D}_n$.
- The probability of error is

$$L(g_n) = \mathbb{P}\{g_n(\mathbf{X}) \neq Y | \mathcal{D}_n\}.$$

- It is consistent if

$$\lim_{n \to \infty} \mathbb{E}L(g_n) = L^\star.$$

Basics of classification

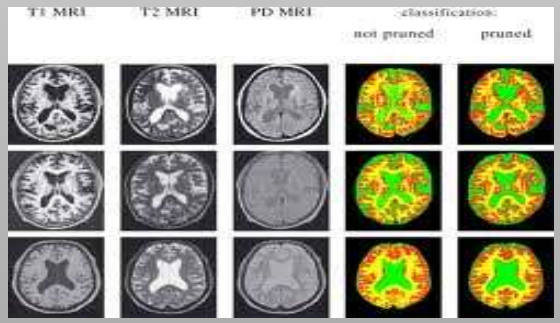
- The data: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, i.i.d. copies of $(\mathbf{X}, Y)$.
- A classifier $g_n(\mathbf{x})$ is a measurable function of $\mathbf{x}$ and $\mathcal{D}_n$.
- The probability of error is

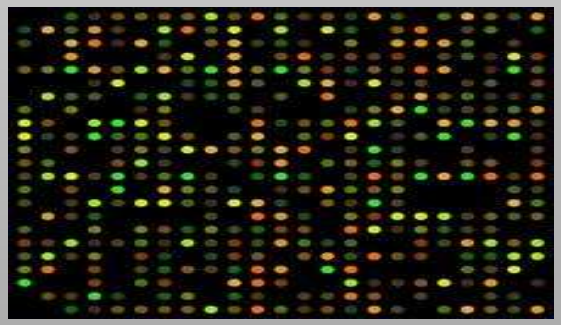
$$L(g_n) = \mathbb{P}\{g_n(\mathbf{X}) \neq Y | \mathcal{D}_n\}.$$

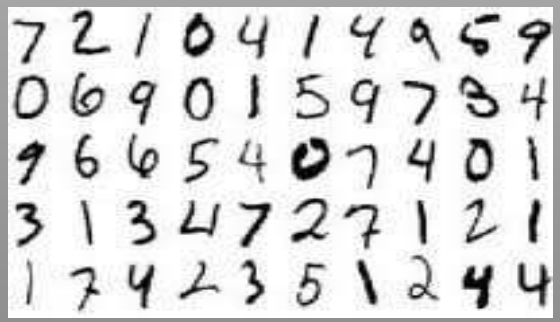
- It is consistent if

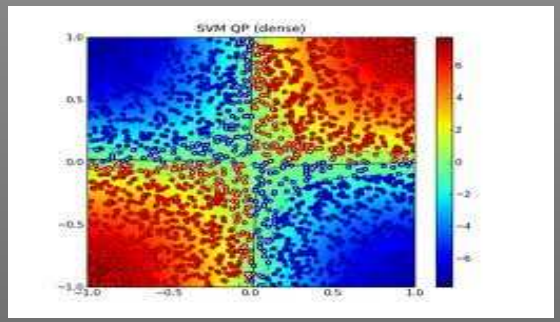
$$\lim_{n \to \infty} \mathbb{E}L(g_n) = L^\star.$$

- It is universally consistent if it is consistent for all possible distributions of $(\mathbf{X}, Y)$.


T1 MRI
T2 MRI
PD MRI
classification:
not pruned
pruned






SVM QP (dense)

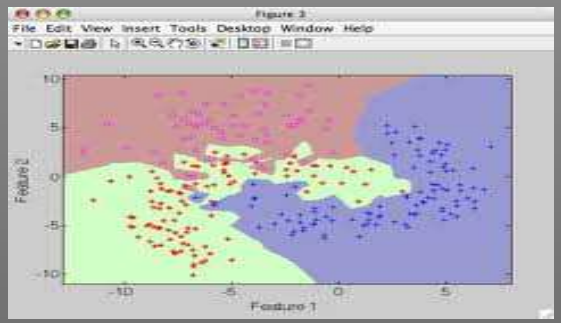


Figure 3
File Edit View Insert Tools Desktop Window Help
Feature 2
Feature 1

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.

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:

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:
  - All procedures that partition space can be viewed as special cases of partitions generated by trees.

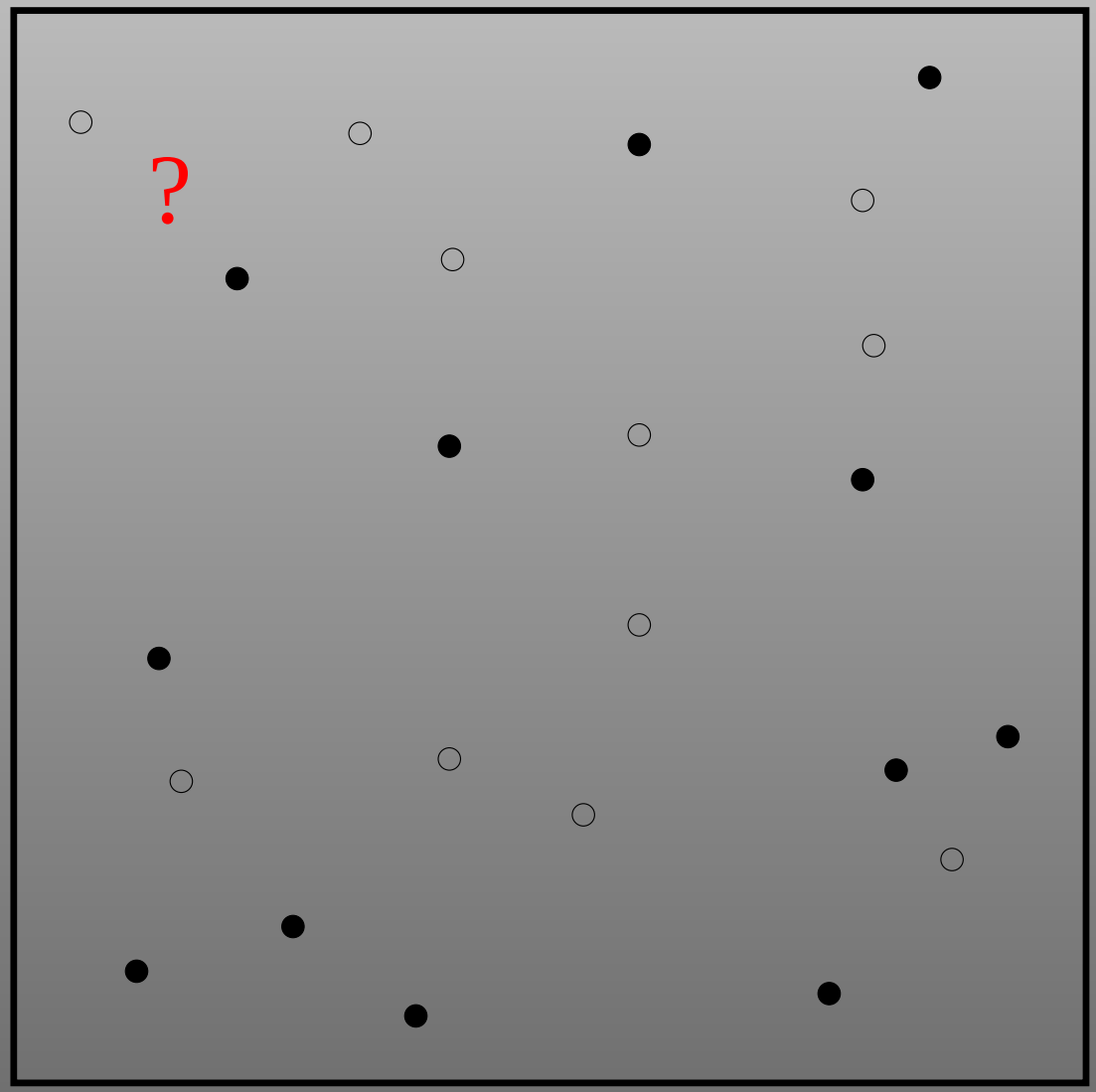
Tree classifiers

- 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:
  - All procedures that partition space can be viewed as special cases of partitions generated by trees.
  - Simple neural networks that use voting methods can also be regarded as trees.

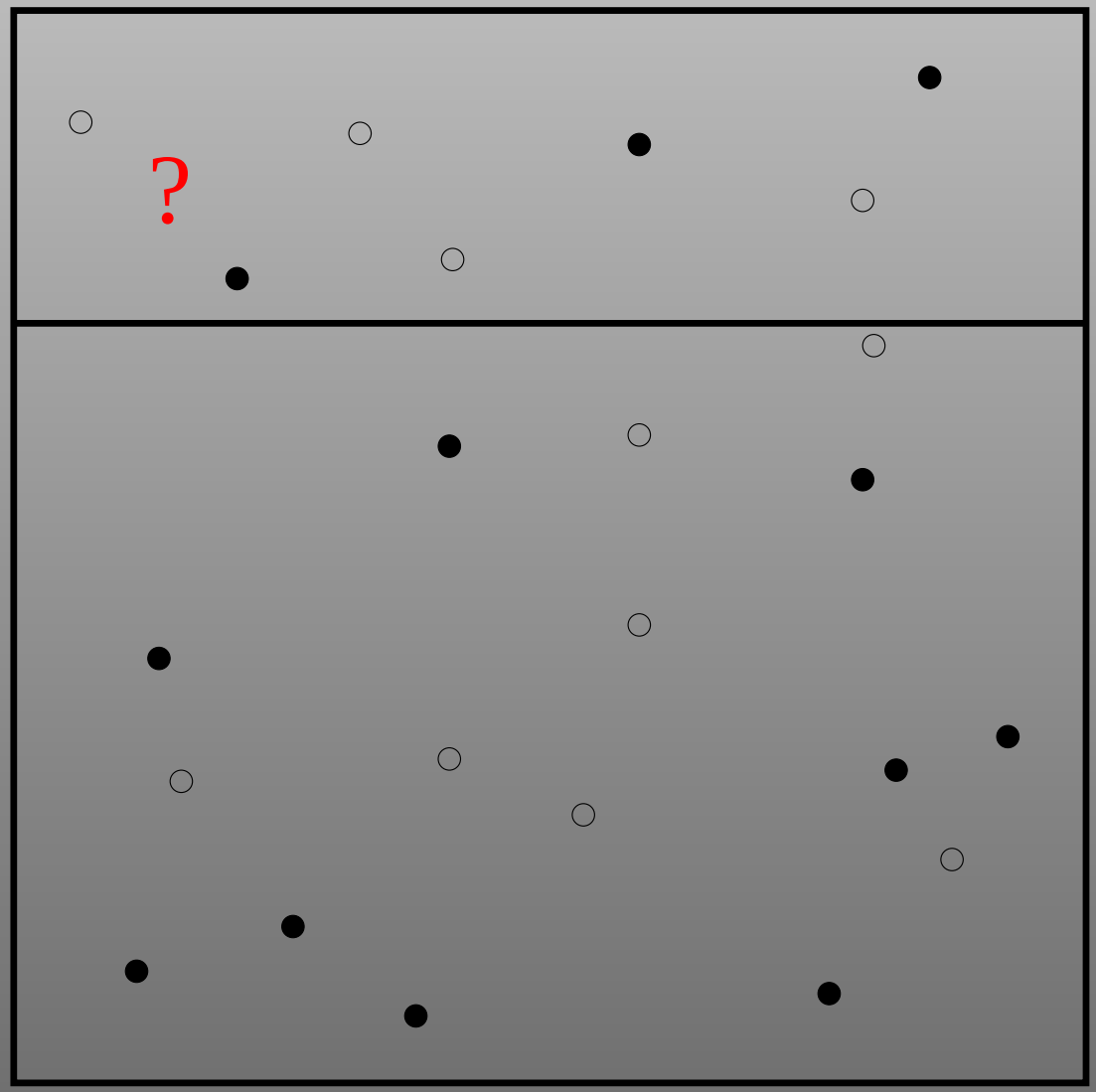
Tree classifiers

- 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:
  - All procedures that partition space can be viewed as special cases of partitions generated by trees.
  - Simple neural networks that use voting methods can also be regarded as trees.
  - Tree classifiers are conceptually simple, and explain the data very well.

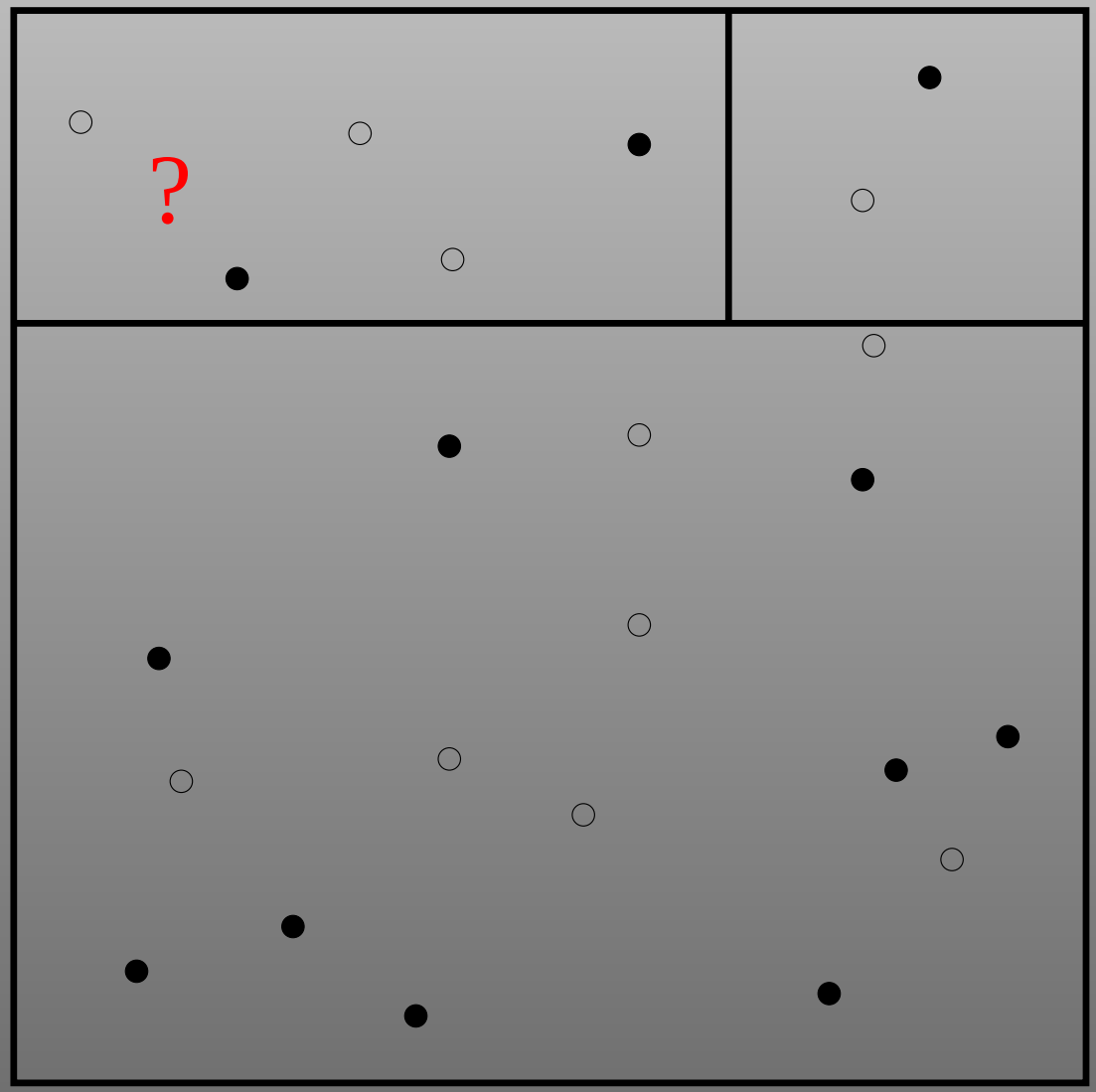
Trees



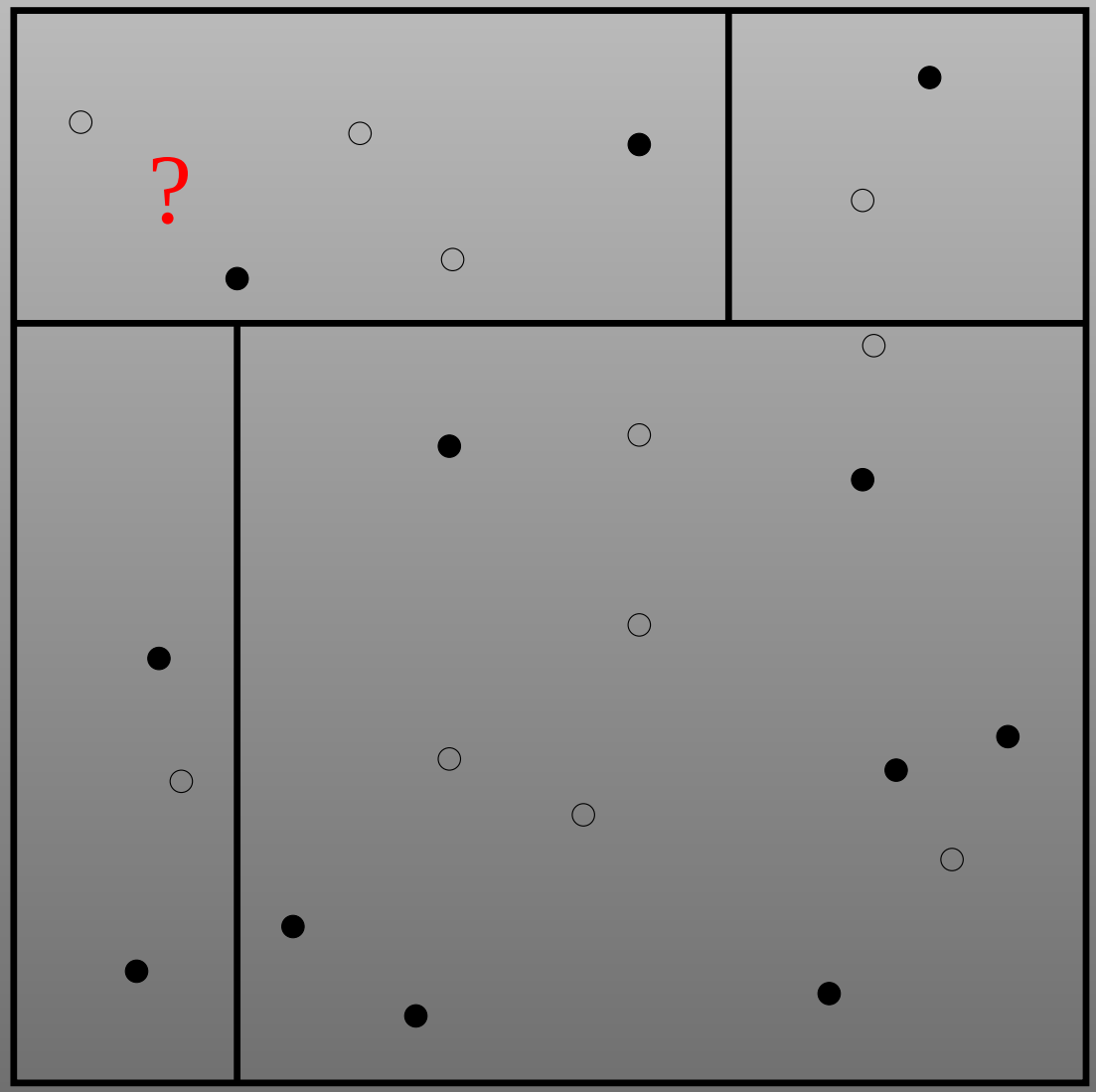
Trees



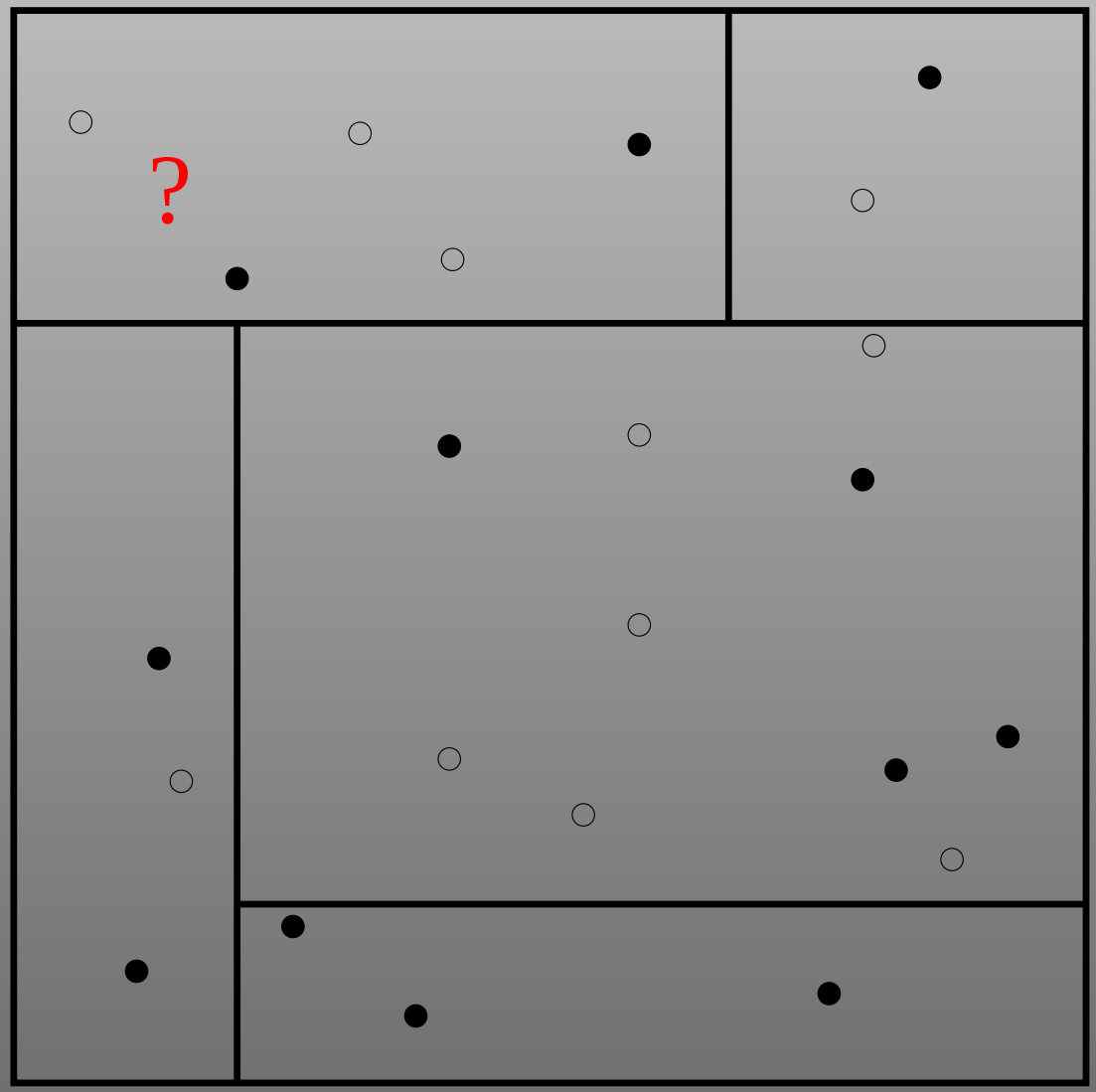
Trees



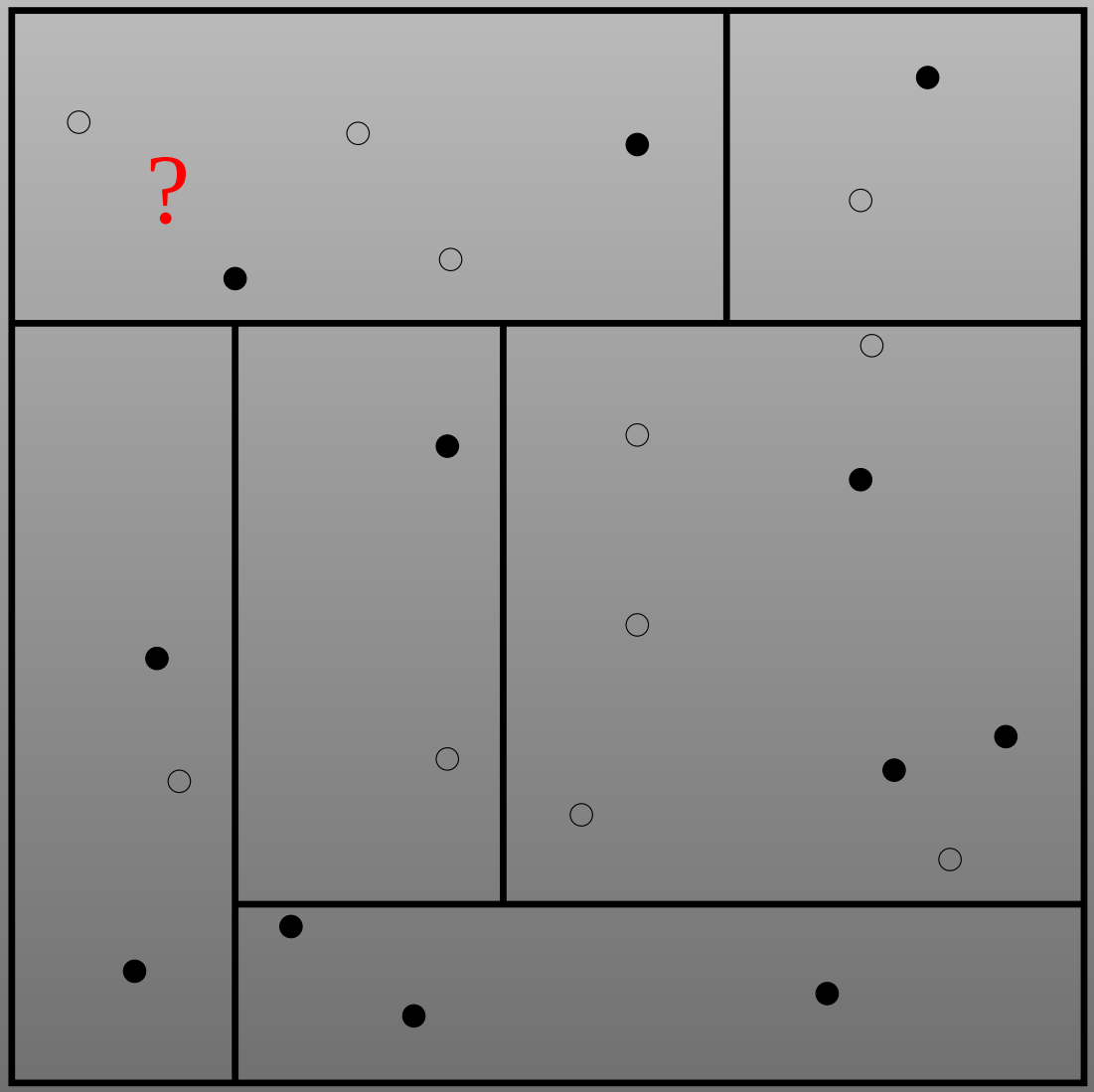
Trees



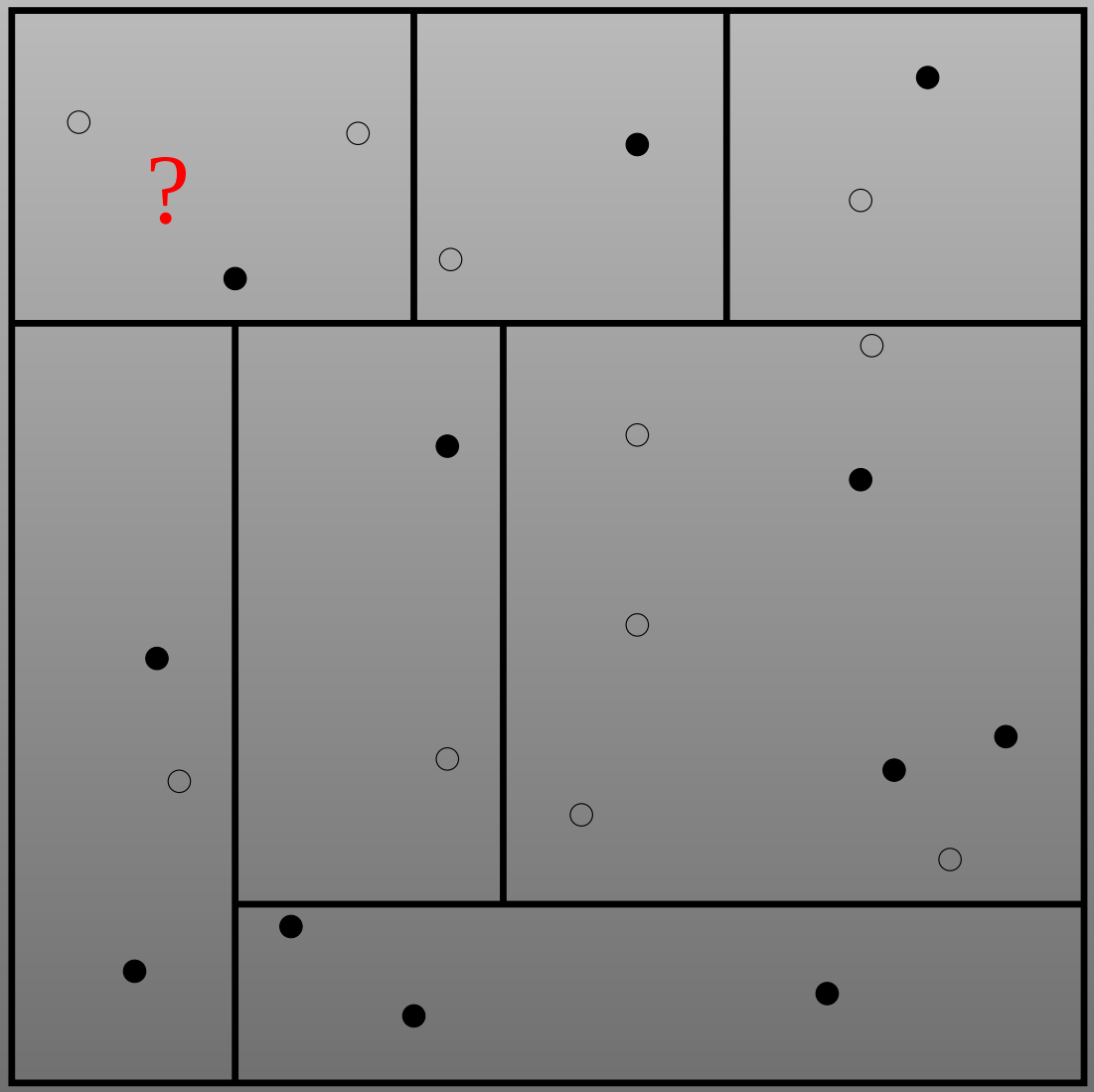
Trees



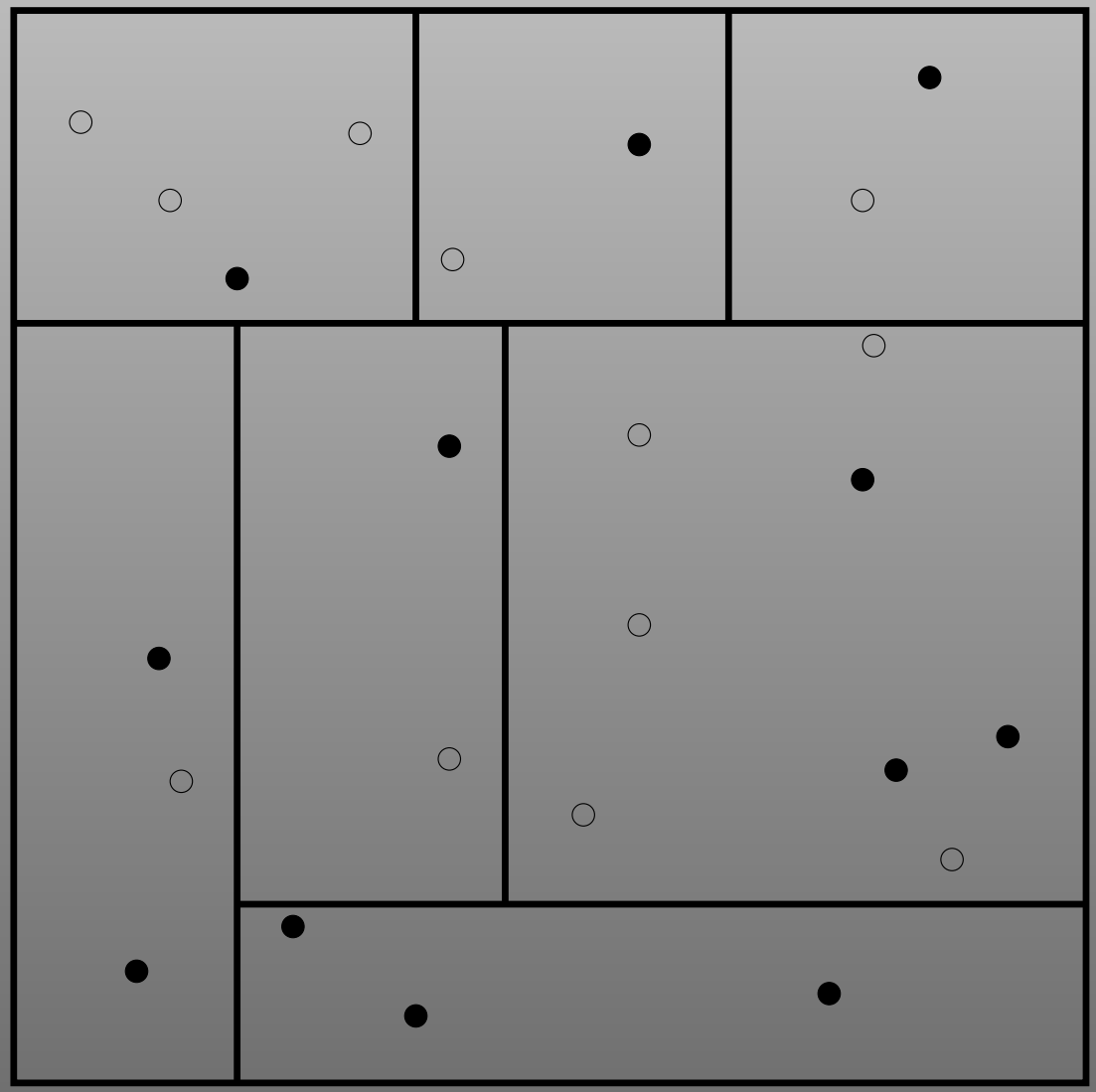
Trees



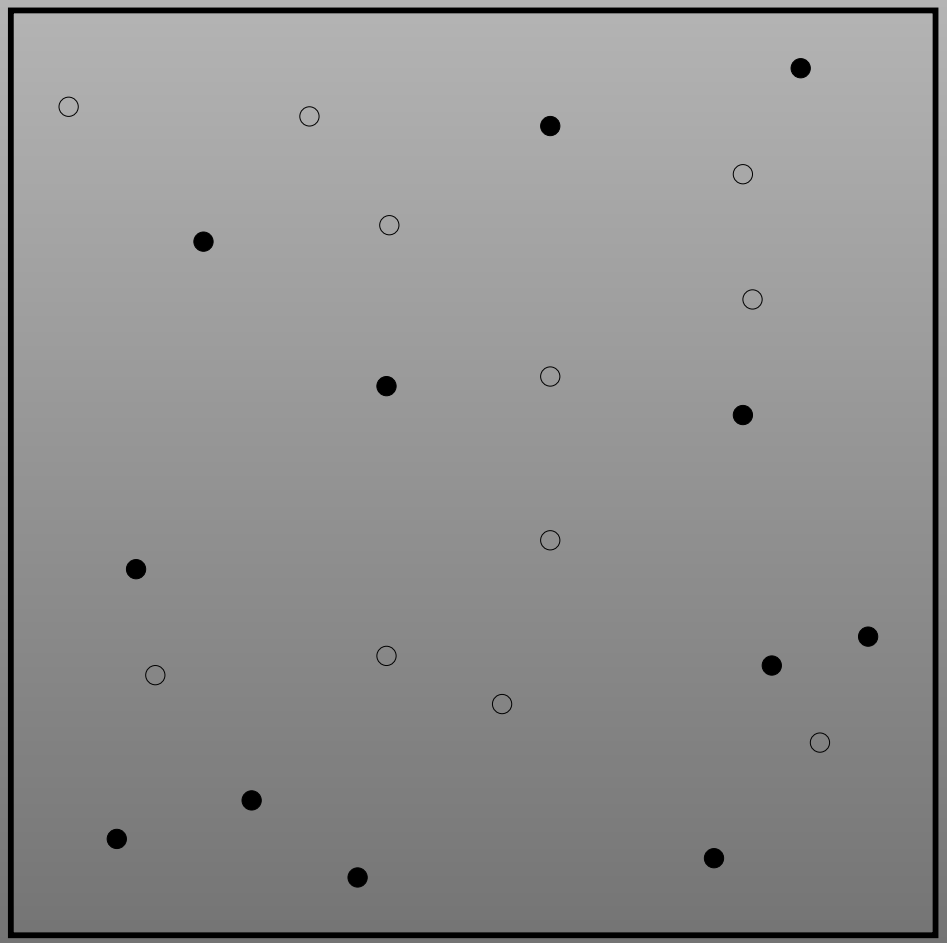
Trees



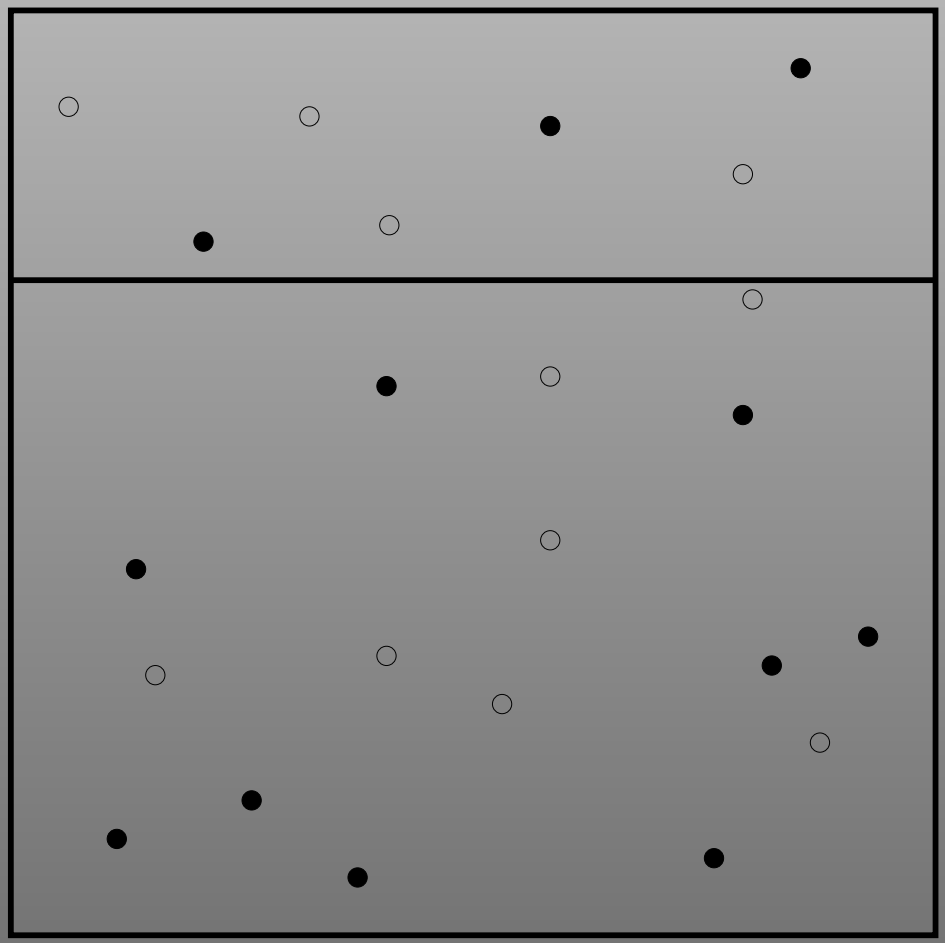
Trees

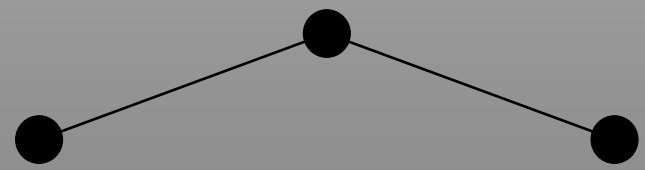


Trees

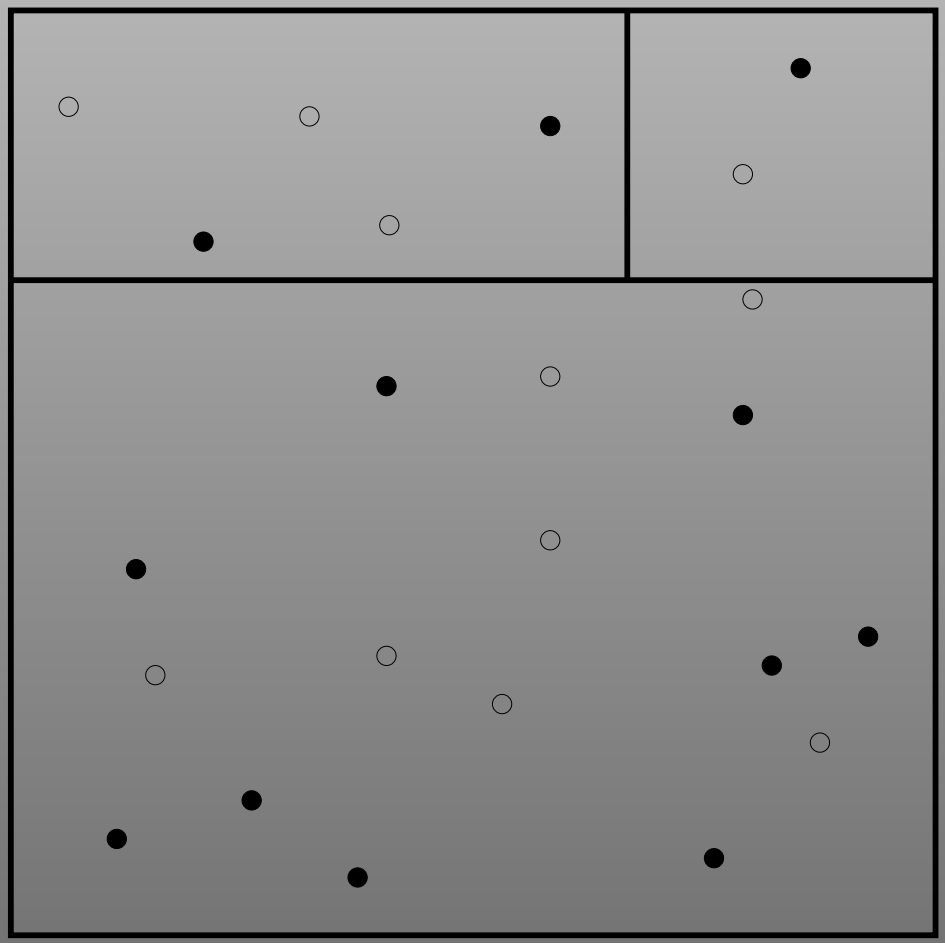


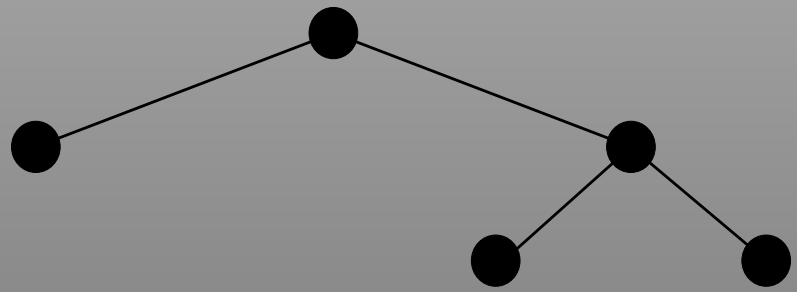
Trees



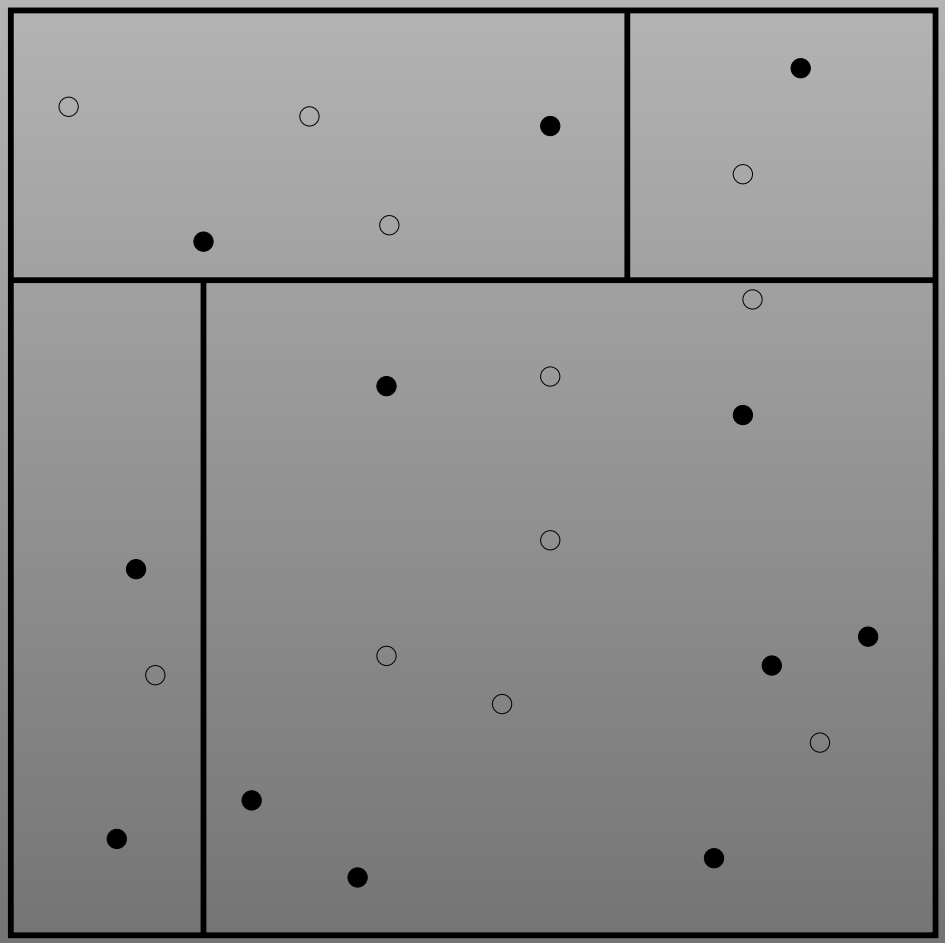


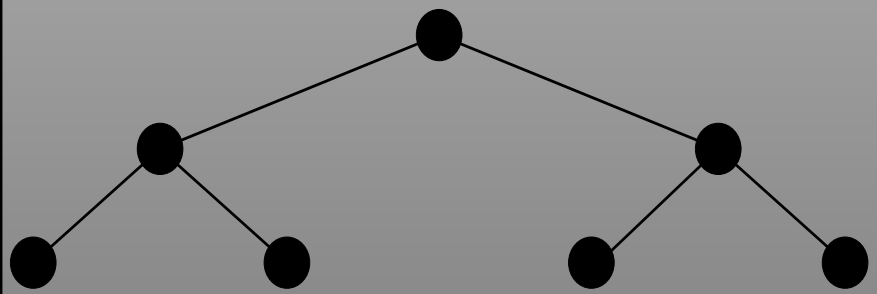
Trees



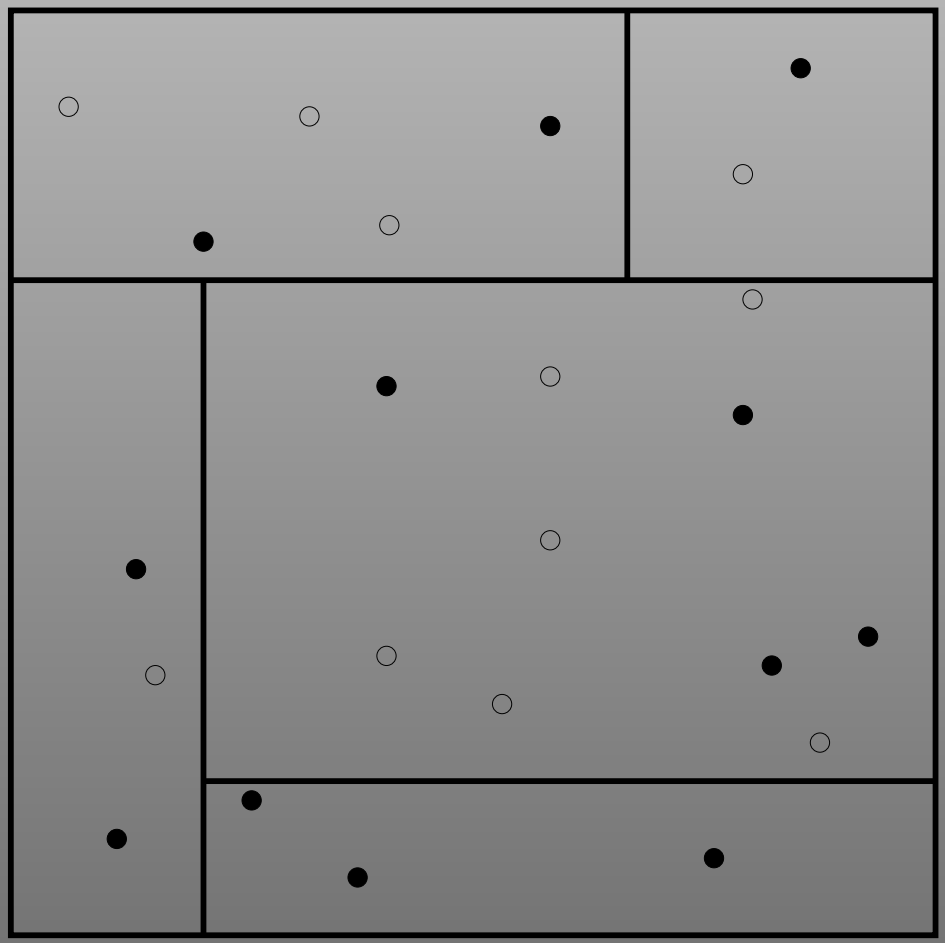


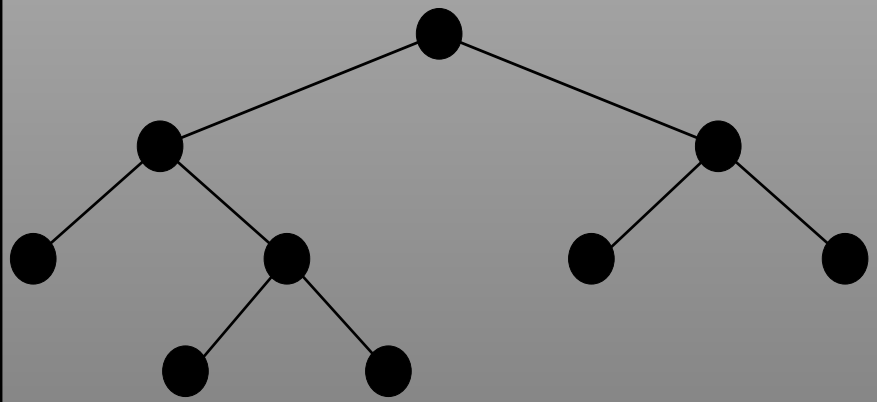
Trees



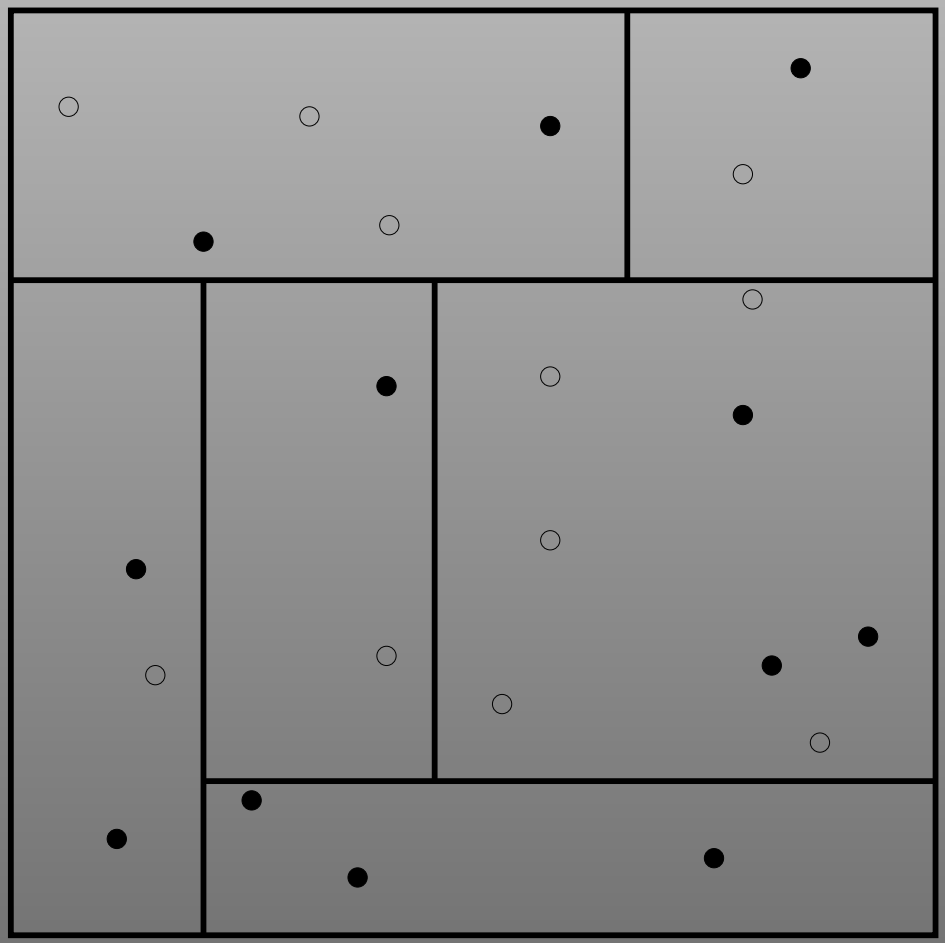


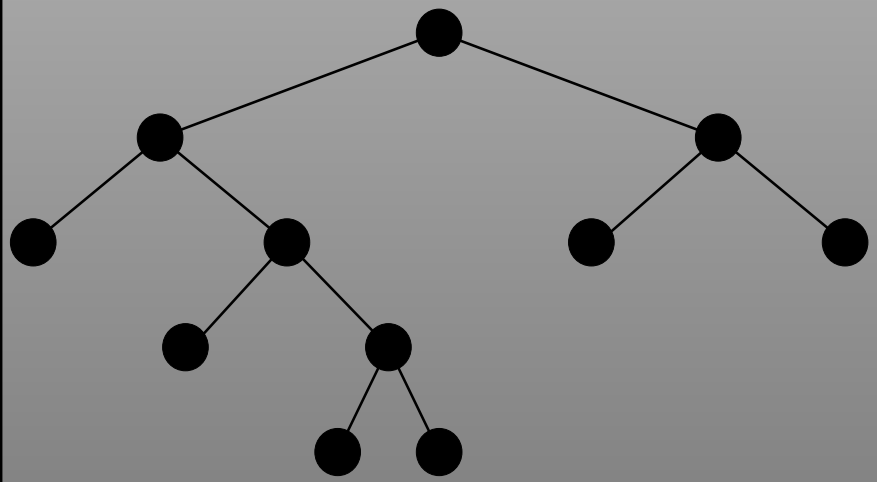
Trees



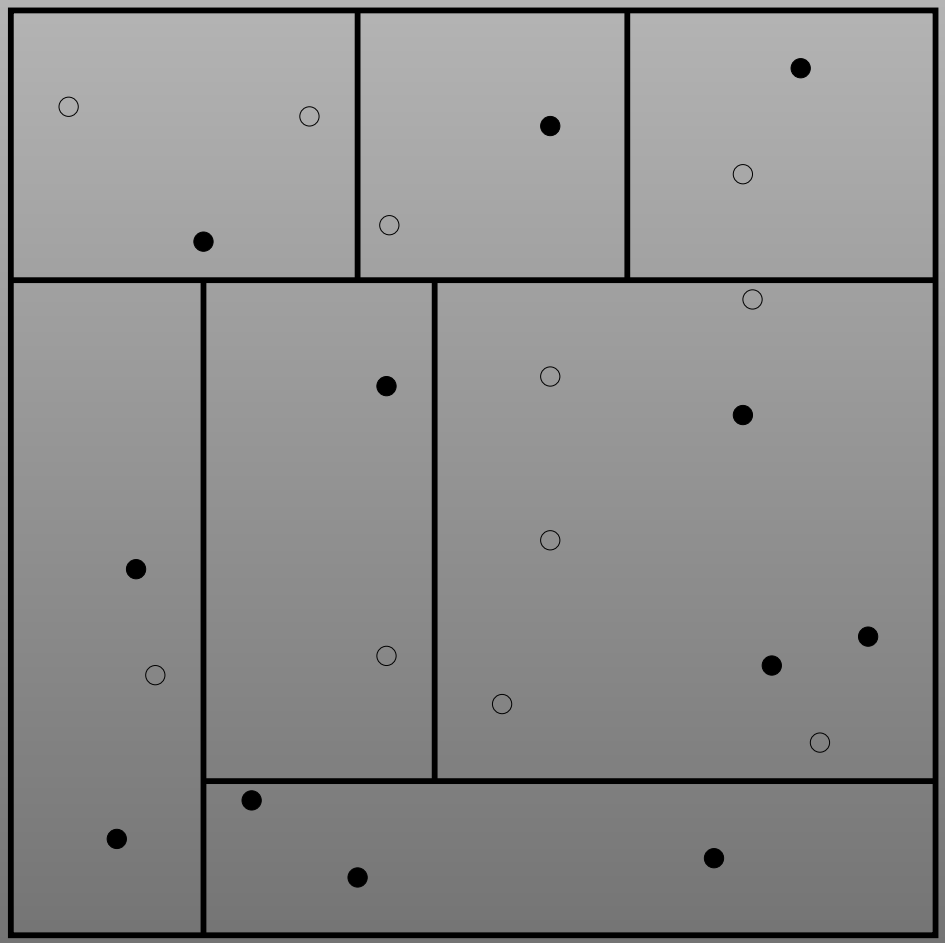


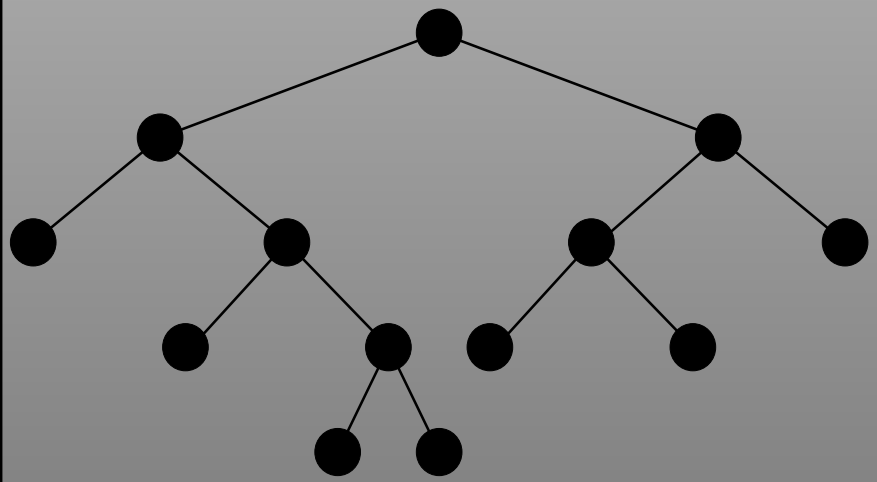
Trees



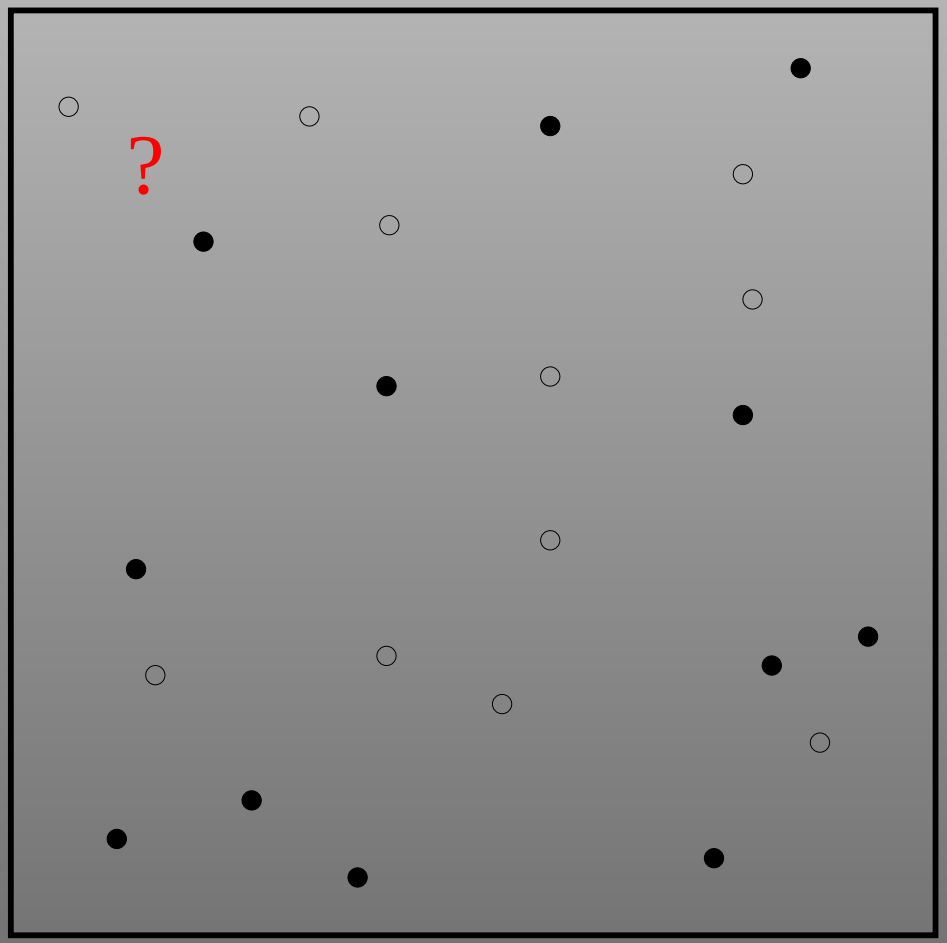


Trees

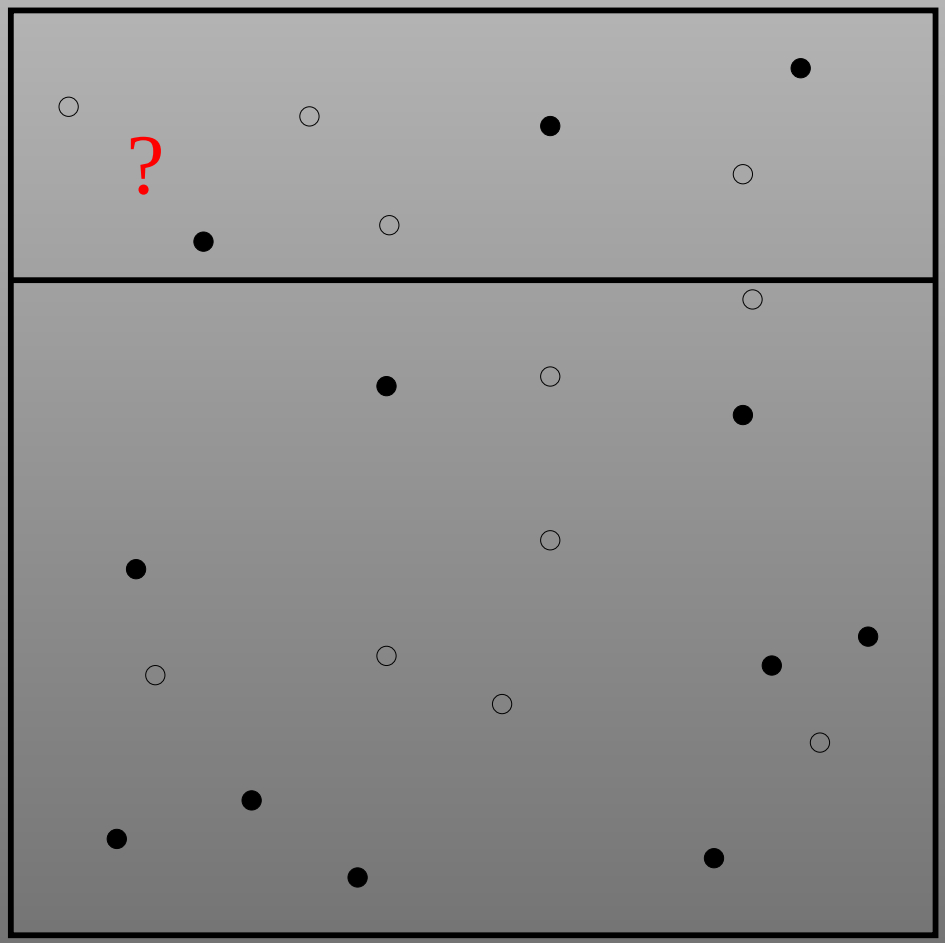


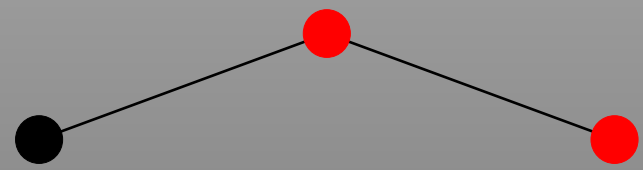


Trees

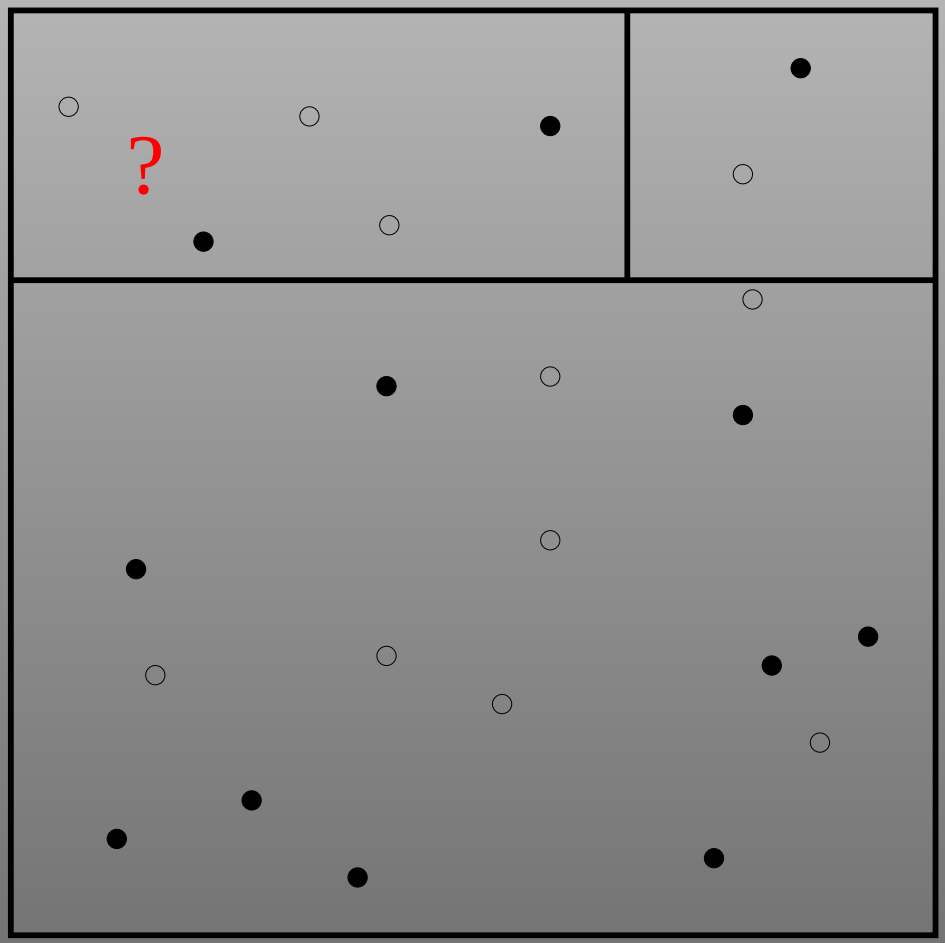


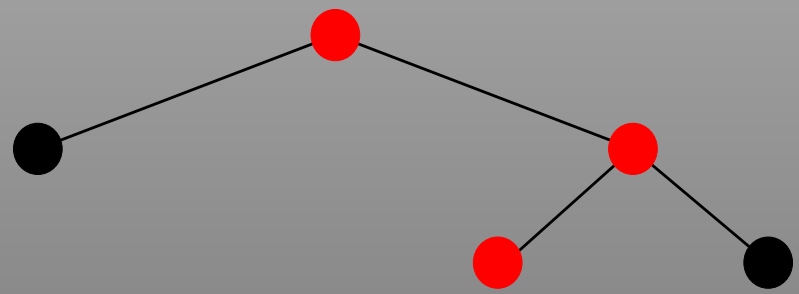
Trees



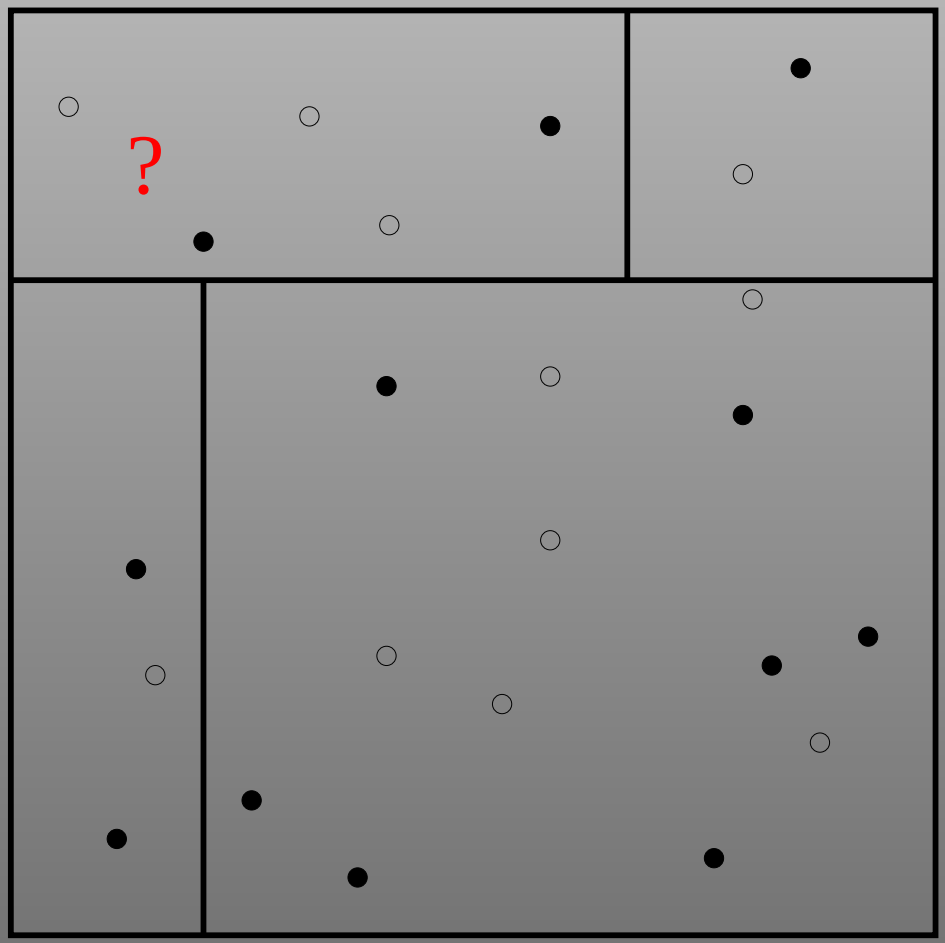


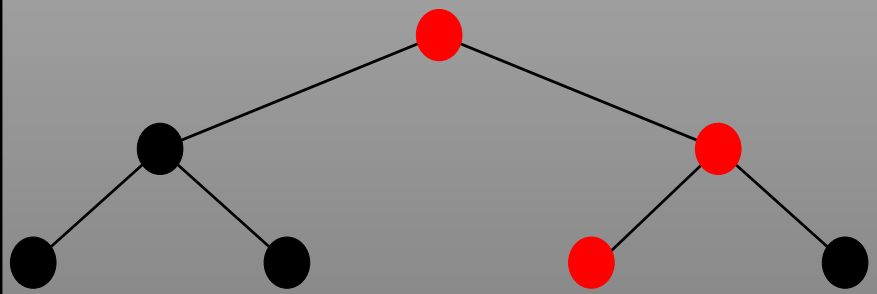
Trees



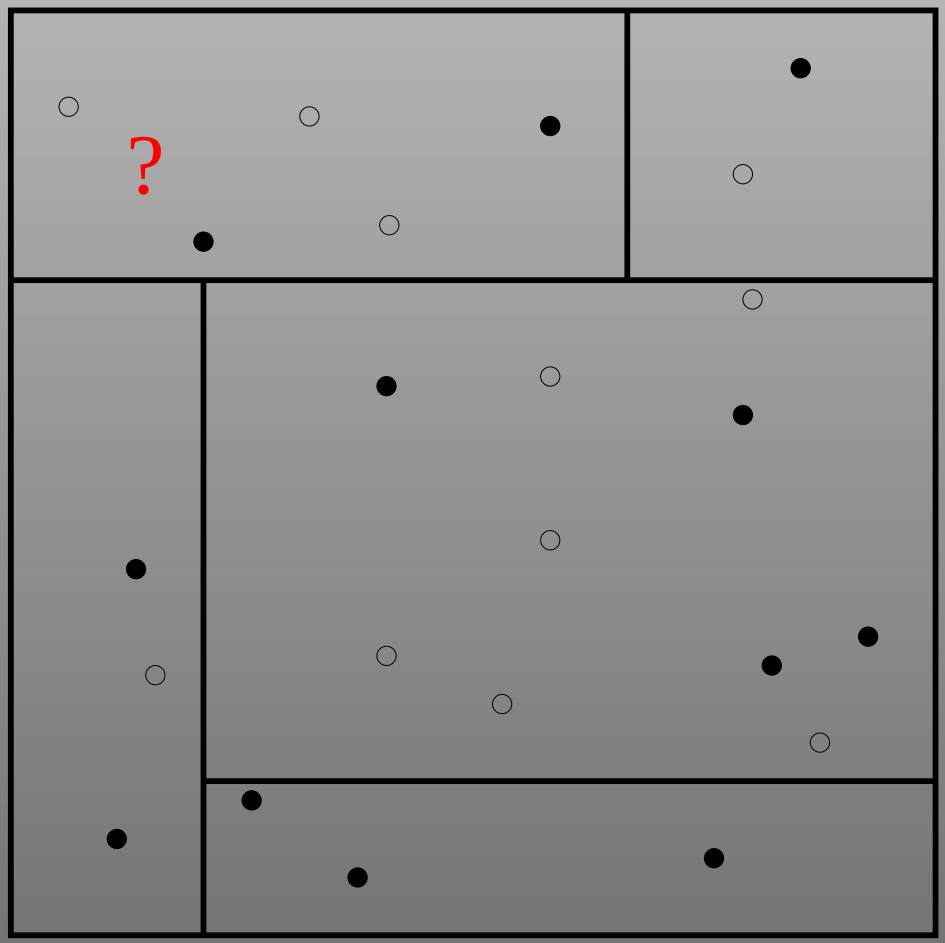


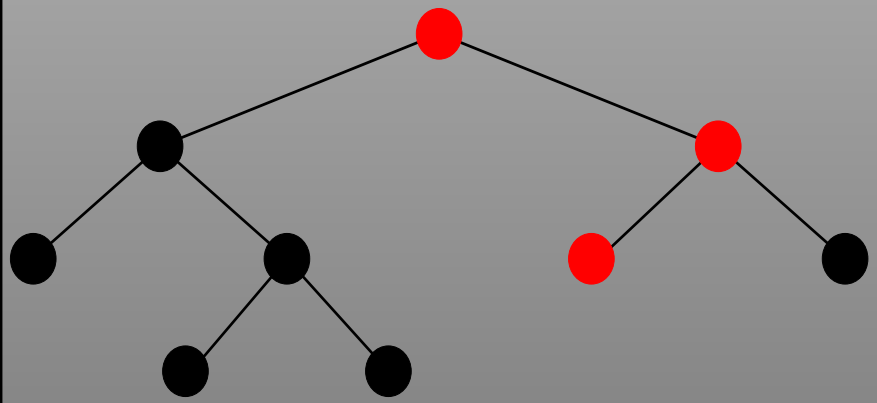
Trees



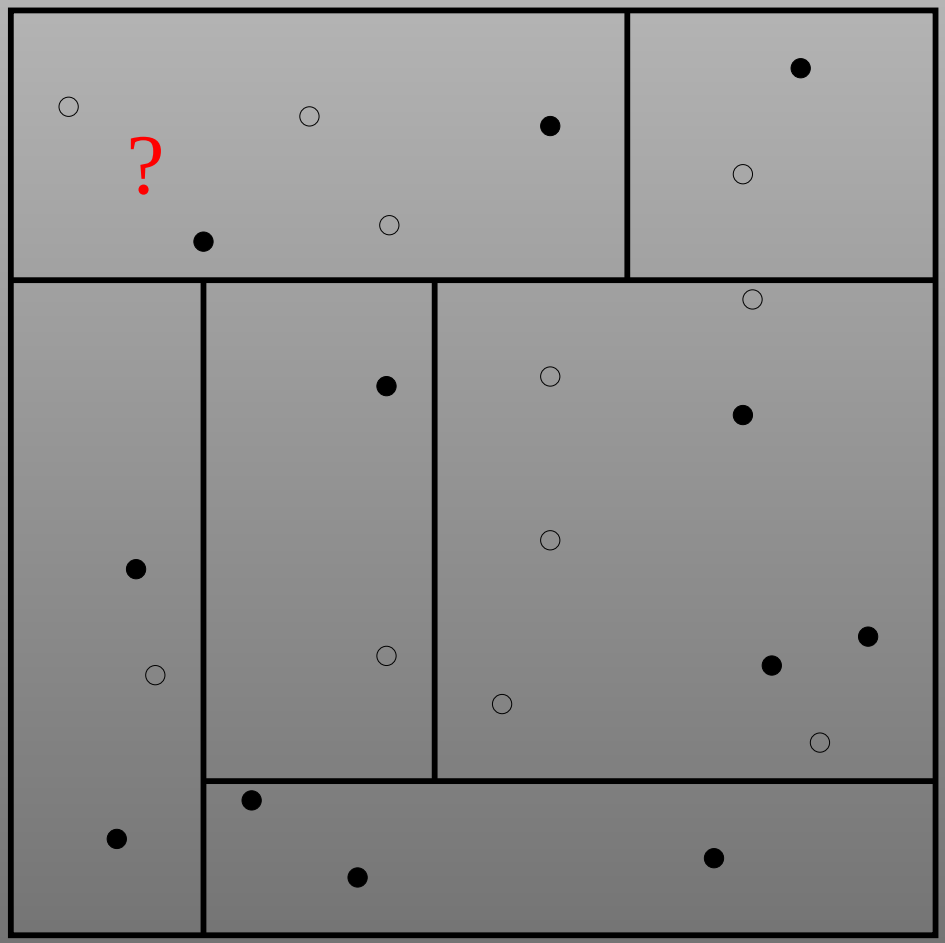


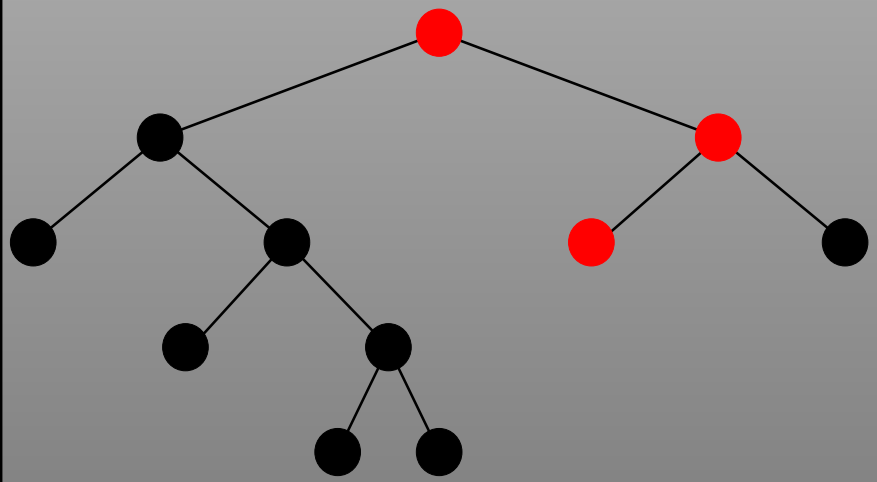
Trees



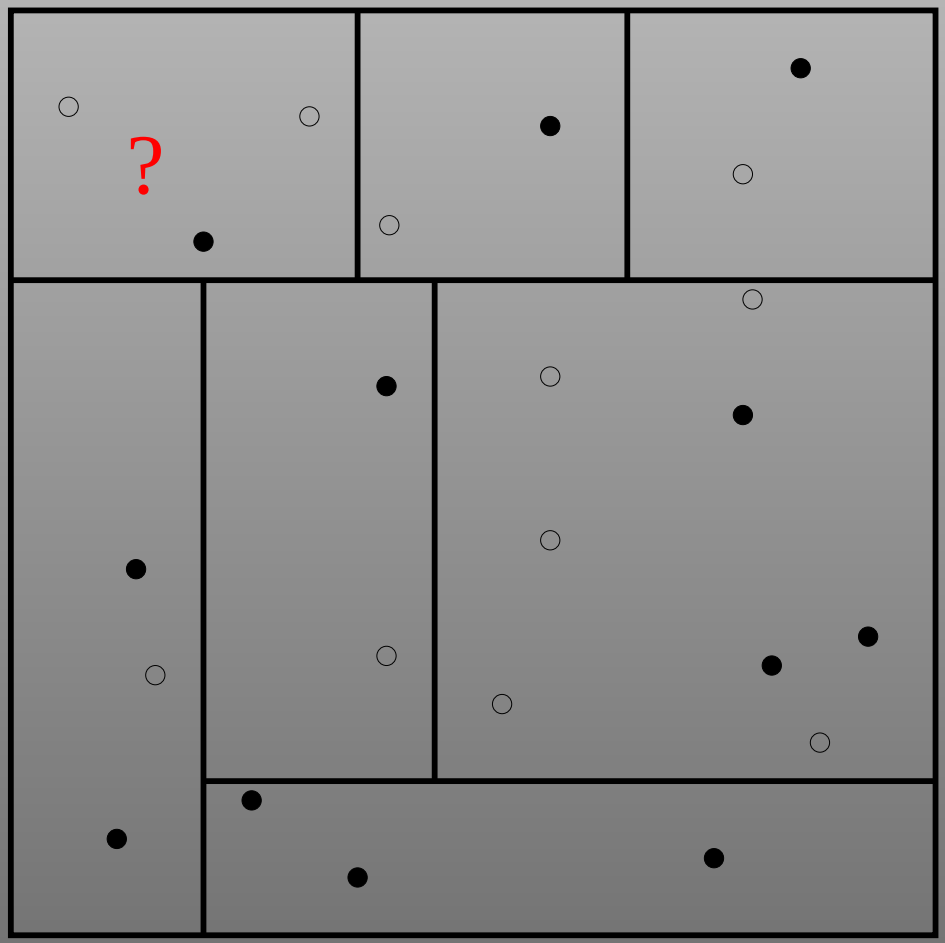


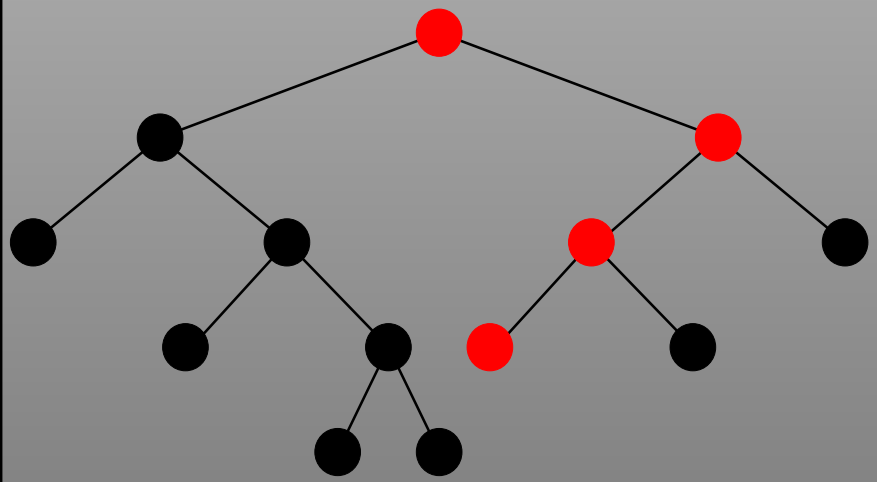
Trees





Trees





Tree classifiers

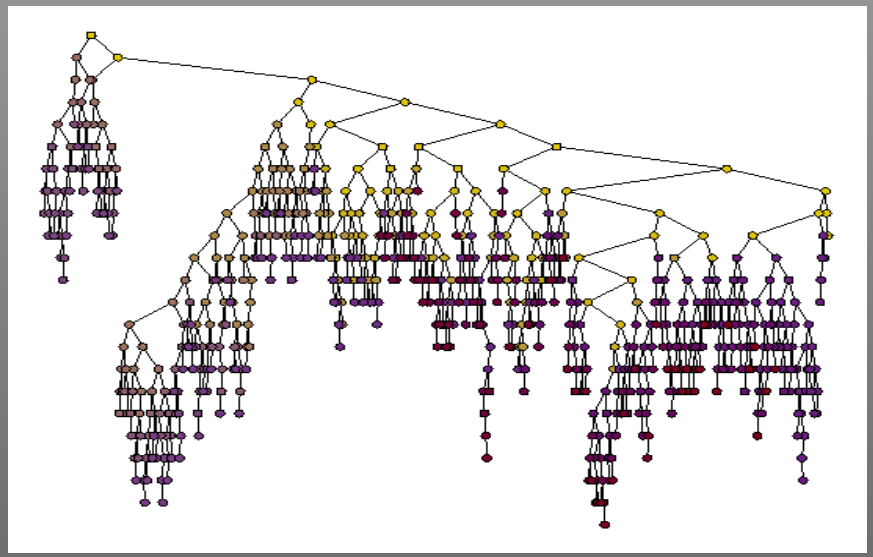
- The design of trees can be cumbersome.

Tree classifiers

- The design of trees can be cumbersome.
- Optimizations could face a huge combinatorial and computational hurdle.

Tree classifiers

- The design of trees can be cumbersome.
- Optimizations could face a huge combinatorial and computational hurdle.
- The greedy paradigm addresses these concerns.

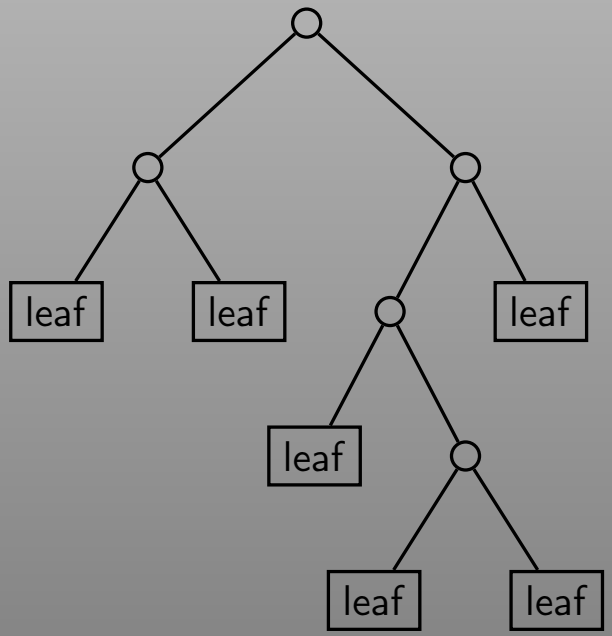


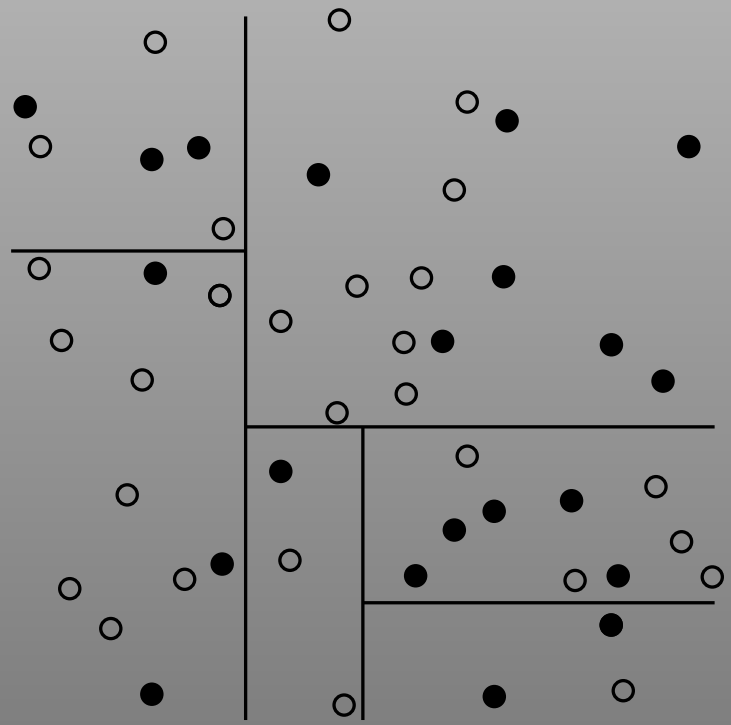
Trees

- Classification trees partition $\mathbb{R}^d$ into regions, often hyperrectangles parallel to the axes.

Trees

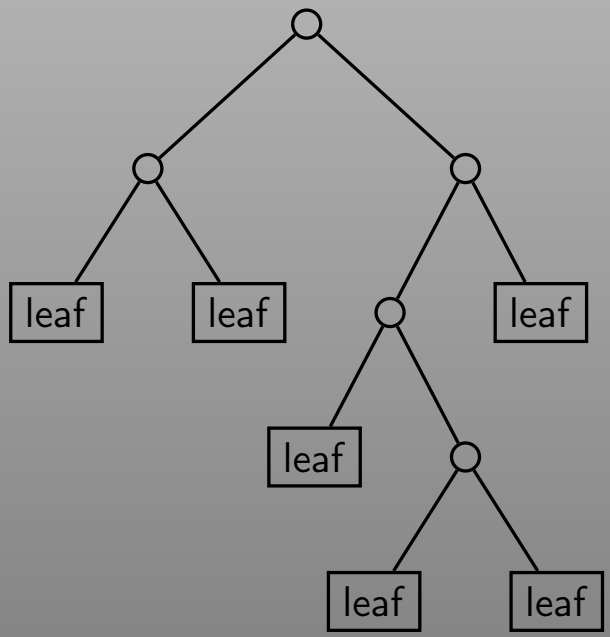
- Classification trees partition $\mathbb{R}^d$ into regions, often hyperrectangles parallel to the axes.

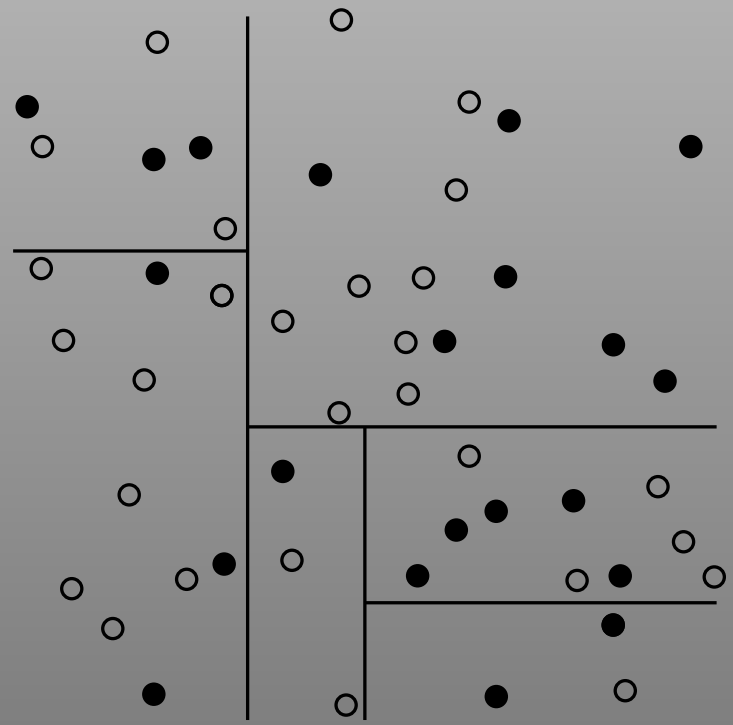




Trees

- Classification trees partition $\mathbb{R}^d$ into regions, often hyperrectangles parallel to the axes.





- The tree classifier takes the simple form

$$g_n(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^n \mathbf{1}_{[\mathbf{X}_i \in A, Y_i = 1]} > \sum_{i=1}^n \mathbf{1}_{[\mathbf{X}_i \in A, Y_i = 0]}, \quad \mathbf{x} \in A \\ 0 & \text{otherwise.} \end{cases}$$

Trees

- The tree structure is usually data dependent, and it is in the construction itself that trees differ.

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.

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?

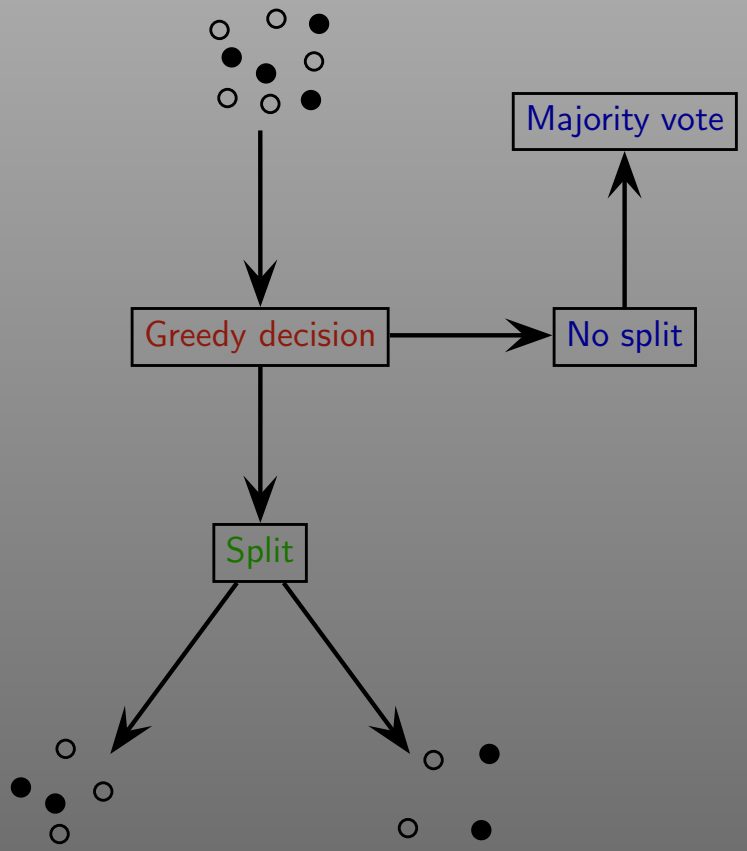
② In the affirmative, what are its children?

The greedy spirit

- Greedy trees proceed from a different philosophy.

The greedy spirit

- Greedy trees proceed from a different philosophy.
- A greedy tree should be able to answer questions ① and ② using local information only.



Outline

A model

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

A model

- Let $\mathcal{C}$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.
- Example: $\mathcal{C} = \left\{ C = \prod_{j=1}^{d} (a_j, b_j) : -\infty \leq a_j < b_j \leq +\infty \right\}$.

A model

- Let $\mathcal{C}$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.
- Example: $\mathcal{C} = \left\{ C = \prod_{j=1}^{d} (a_j, b_j) : -\infty \leq a_j < b_j \leq +\infty \right\}$.
- Let $\{(\mathbf{X}_i, Y_i) : i \in I\}$ be the subset of observations falling in a given node.

A model

- Let $\mathcal{C}$ be a class of possible subsets of $\mathbb{R}^d$ that can be used for splits.
- Example: $\mathcal{C} = \left\{ C = \prod_{j=1}^{d} (a_j, b_j) : -\infty \leq a_j < b_j \leq +\infty \right\}$.
- Let $\{(\mathbf{X}_i, Y_i) : i \in I\}$ be the subset of observations falling in a given node.
- A split that uses a set $C \in \mathcal{C}$ results in two sets of indices:

$$I' = \{i \in I : \mathbf{X}_i \in C\} \quad \text{and} \quad I'' = \{i \in I : \mathbf{X}_i \notin C\}.$$

A model

- A greedy split may be seen as a mapping

$$\sigma : (\mathbb{R}^d \times \{0, 1\})^{|I|} \to \mathbb{R}^p.$$

A model

- A greedy split may be seen as a mapping

$$\sigma : (\mathbb{R}^d \times \{0, 1\})^{|I|} \to \mathbb{R}^p.$$

- We see that σ is a model for the greedy decision ②.

A model

- A greedy split may be seen as a mapping

$$\sigma : (\mathbb{R}^d \times \{0, 1\})^{|I|} \to \mathbb{R}^p.$$

- We see that σ is a model for the greedy decision ②.

- In addition, there is a second mapping θ, but this time with a boolean output.

A model

- A greedy split may be seen as a mapping

$$\sigma : (\mathbb{R}^d \times \{0,1\})^{|I|} \to \mathbb{R}^p.$$

- We see that σ is a model for the greedy decision ②.
- In addition, there is a second mapping θ, but this time with a boolean output.
- It is a stopping rule and models the greedy decision ①.

Greedy procedure

- A greedy machine partitions the data recursively.

Greedy procedure

- A greedy machine partitions the data recursively.
  - A node = a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node.

Greedy procedure

- A greedy machine partitions the data recursively.
  - A node = a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node.
  - For a node A,
$$I = \{i = 1, \ldots, n : \mathbf{X}_i \in A\}.$$

Greedy procedure

- A greedy machine partitions the data recursively.
  - A node = a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node.
  - For a node A,
    $$I = \{i = 1, \ldots, n : \mathbf{X}_i \in A\}.$$
  - Compute the decision
    $$\theta\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \{0, 1\}$$
    and the parameter of the split
    $$\sigma\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \mathcal{C}.$$

Greedy procedure

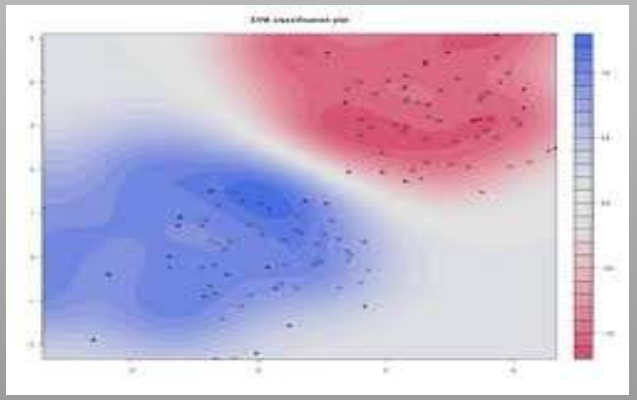
- A greedy machine partitions the data recursively.
  - A node = a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node.
  - For a node A,
    $$I = \{i = 1, \ldots, n : \mathbf{X}_i \in A\}.$$
  - Compute the decision
    $$\theta\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \{0, 1\}$$
    and the parameter of the split
    $$\sigma\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \mathcal{C}.$$
  - If $\theta = 0$: STOP. Otherwise, the node spawns two children.

Greedy procedure

- A greedy machine partitions the data recursively.
  - A node = a subset of $\mathbb{R}^d$, starting with $\mathbb{R}^d$ for the root node.
  - For a node A,
    $$I = \{i = 1, \ldots, n : \mathbf{X}_i \in A\}.$$
  - Compute the decision
    $$\theta\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \{0, 1\}$$
    and the parameter of the split
    $$\sigma\left((\mathbf{X}_i, Y_i) : i \in I\right) \in \mathcal{C}.$$
  - If $\theta = 0$: STOP. Otherwise, the node spawns two children.
- Final classification proceeds by a majority vote.

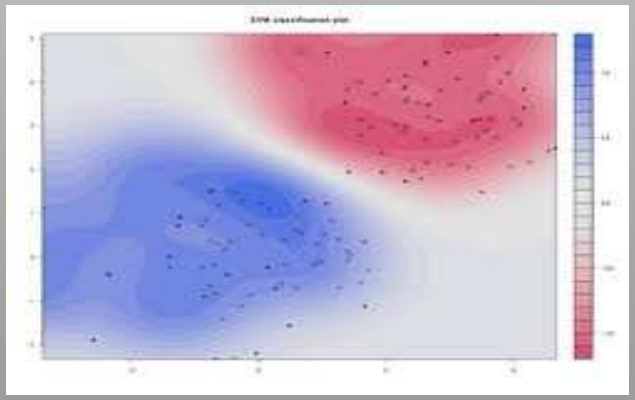
An important remark

Is any classifier a greedy tree?



An important remark

Is any classifier a greedy tree?



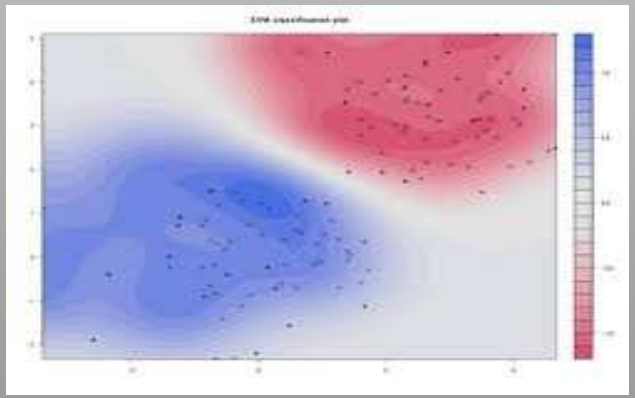
1. Set $\theta = 1$ if we care at the root, and $\theta = 0$ elsewhere.
2. The root node is split by the classifier into a set

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

and its complement, and both child nodes are leaves.

An important remark

Is any classifier a greedy 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

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

and its complement, and both child nodes are leaves.

This is not allowed.

Outline

An example

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

An example

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

An example

- At first sight, there are no consistent greedy tree classifiers.
- Example: The k-median tree.
  - When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'s.

An example

- At first sight, there are no consistent greedy tree classifiers.
- Example: The k-median tree.
  - When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'s.
  - Keep doing this for k rounds.

An example

- At first sight, there are no consistent greedy tree classifiers.
- Example: The k-median tree.
  - When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'s.
  - Keep doing this for k rounds.
  - In d dimensions, rotate through the coordinates.

An example

- At first sight, there are no consistent greedy tree classifiers.
- Example: The k-median tree.
  - When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'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 greedy tree classifiers.
- Example: The k-median tree.
  - When $d = 1$, split by finding the median element among the $\mathbf{X}_i$'s.
  - Keep doing this for k rounds.
  - In d dimensions, rotate through the coordinates.
- This rule is consistent, provided $k \to \infty$ and $k2^k/n \to 0$.

This is not greedy.

A randomized solution

- The greedy splitting method σ mimics the median tree classifier.

A randomized solution

- The greedy splitting method σ mimics the median tree classifier.
- The dimension to cut is chosen uniformly at random.

A randomized solution

- The greedy splitting method σ 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 greedy splitting method σ 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 θ.

A randomized solution

- The greedy splitting method σ 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 θ.
- This function ignores the data altogether and uses a randomized decision that is based on the size of the input.

Consistency

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

Consistency

- Consider a nonincreasing function $p : \mathbb{N} \to (0, 1)$.
- Then, if U is the uniform $[0, 1]$ random variable associated with node A,

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

Consistency

- Consider a nonincreasing function $p : \mathbb{N} \to (0, 1)$.
- Then, if U is the uniform $[0, 1]$ random variable associated with node A,

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

Theorem

Let β be a real number in $(0, 1)$. Define

$$p(n) = \begin{cases} 1 & \text{if } n < 3 \\ 1/\log^{\beta} n & \text{if } n \geq 3. \end{cases}$$

Then

$$\lim_{n \to \infty} \mathbb{E}L(g_n) = L^{\star} \quad \text{as } n \to \infty.$$

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.
- The median classifier does not use the ordering in the data.

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.
- The median classifier does not use the ordering in the data.
- One can use the randomness present in the permutation of the data.

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.
- The median classifier does not use the ordering in the data.
- One can use the randomness present in the permutation of the data.
- This corresponds to $\approx n \log_2 n$ independent fair coin flips.

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.
- The median classifier does not use the ordering in the data.
- One can use the randomness present in the permutation of the data.
- This corresponds to $\approx n \log_2 n$ independent fair coin flips.
- The total number of bits required to carry out all computations is

$$\mathcal{O}\left((3 + \log_2 d) 2^{\log^{\beta+\varepsilon} n}\right).$$

Can one do without randomization?

- The solution is in the hypothesis that the data elements are i.i.d.
- The median classifier does not use the ordering in the data.
- One can use the randomness present in the permutation of the data.
- This corresponds to $\approx n \log_2 n$ independent fair coin flips.
- The total number of bits required to carry out all computations is

$$\mathcal{O}\left((3 + \log_2 d) 2^{\log^{\beta+\varepsilon} n}\right).$$

There is sufficient randomness at hand to do the job.

Outline

A full 2^d-ary tree

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

A full 2^d-ary tree

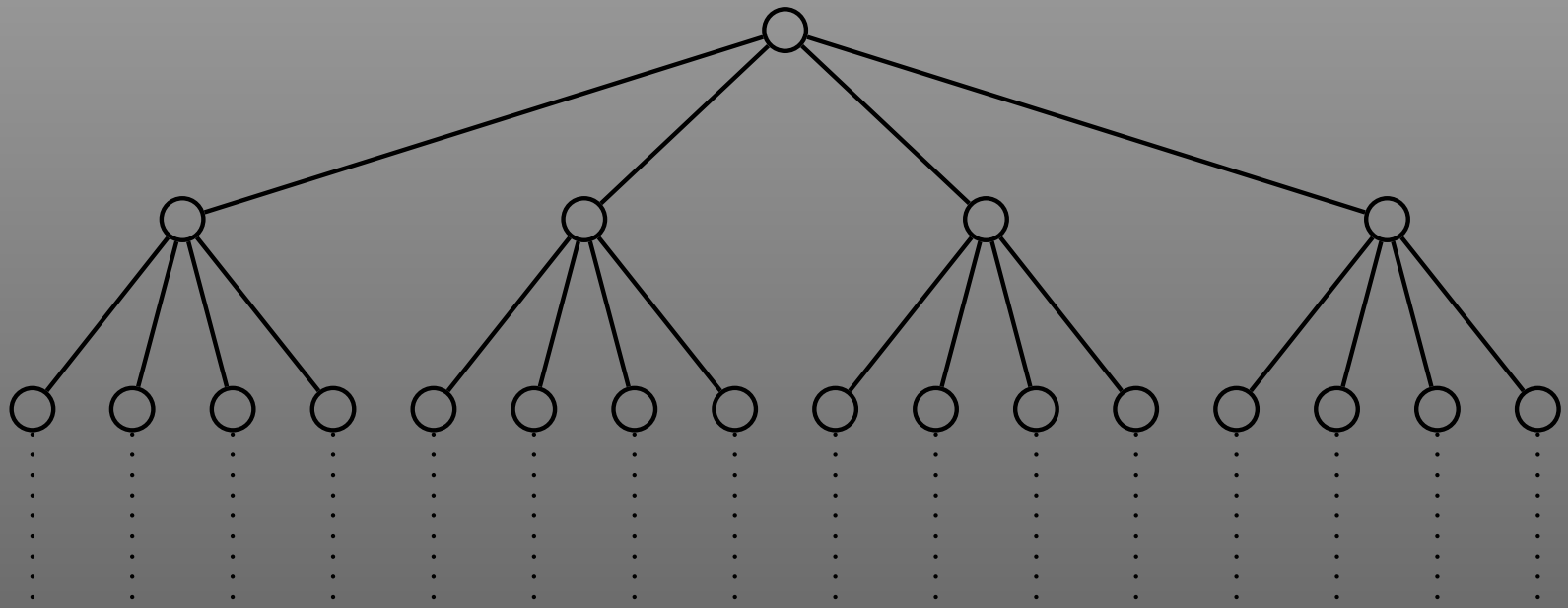
- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.

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 θ

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

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

The stopping rule θ

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

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

- Define the nonnegative integer k^+ by

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

The stopping rule θ

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

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

- 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 θ

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

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

- 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 θ

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

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

- 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 greedy.

The stopping rule θ

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 θ

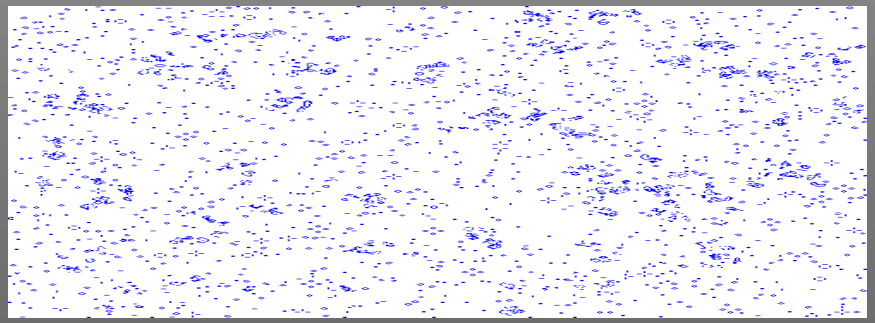
Put $\theta = 0$ if

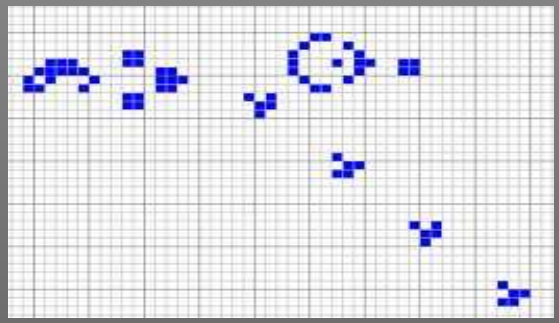
$$\left|\hat{L}_n(A) - \hat{L}_n(A, k^+)\right| \leq \left(\frac{1}{N(A)+1}\right)^{\beta}.$$

Theorem

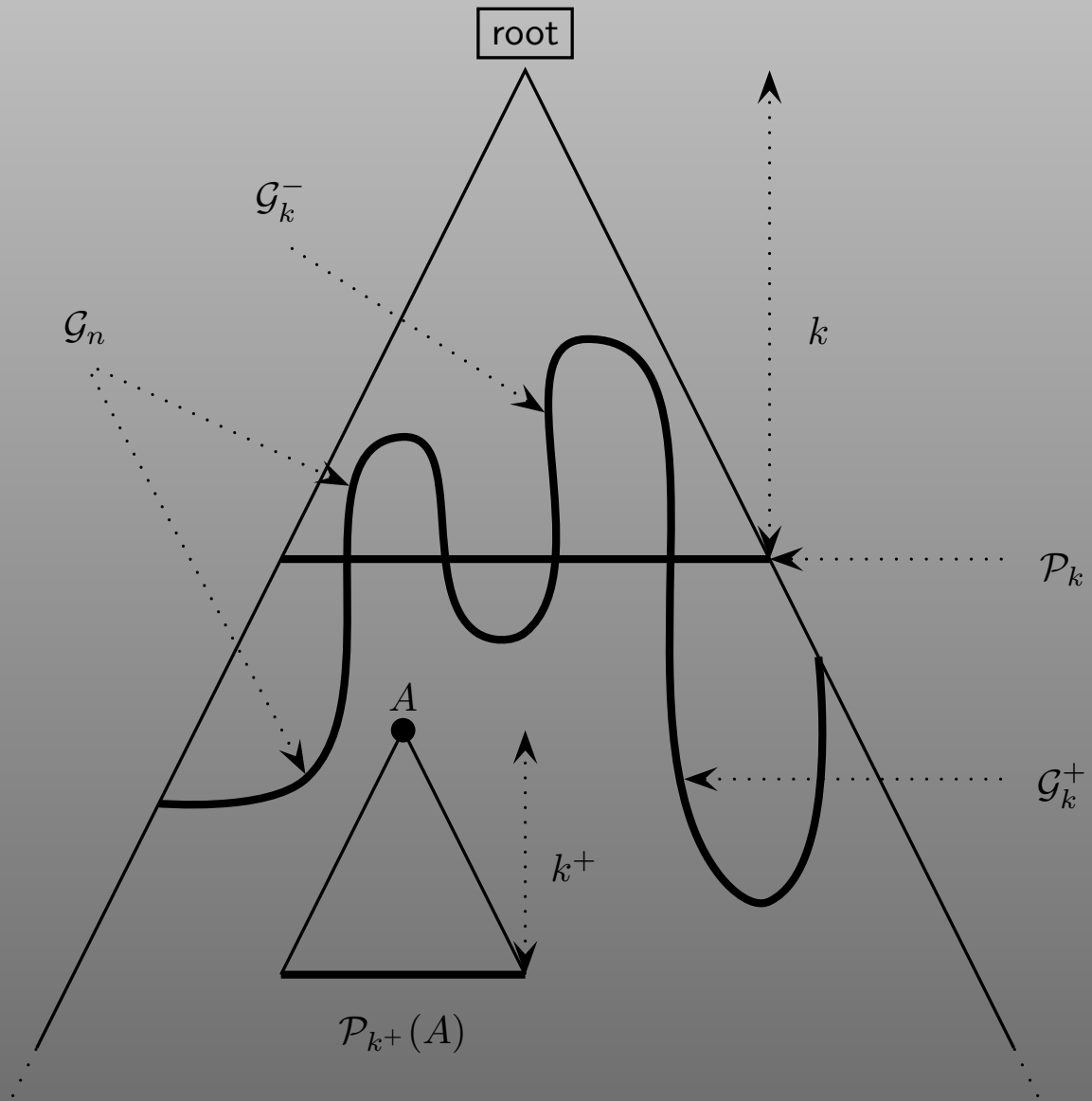
Take $1 - d\alpha - 2\beta > 0$. Then

$$\lim_{n\to\infty} \mathbb{E}L(g_n) = L^\star \quad \text{as } n \to \infty.$$





Proof


root
$\mathcal{G}_k^-$
$\mathcal{G}_n$
k
$\mathcal{P}_k$
A
$\mathcal{G}_k^+$
k^+
$\mathcal{P}_{k^+}(A)$

Proof

Let $\psi(n,k) = L_k^\star - L^\star$. Set

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

Then

$$\frac{2^{dk_n^\star}}{n} \to 0 \quad \text{as } n \to \infty.$$