MS IA : IA710
Classification tree, bagging, and random forest

Pavlo Mozharovskyi

LTCI, Télécom Paris, Institut Polytechnique de Paris

October 2019

Today and on Monday

Framework and problematic
- The task of classification
- Simplest examples and first problems

Classification tree
- Algorithm
- Tuning

Bagging
- Motivation
- Algorithm
- An example

Random forest
- Algorithm
- Interpretation
- Consistency results

Literature

Learning materials include but are not limited to:

- Hastie, T., Tibshirani, R., and Friedman, J. (2009).
  The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition).
  Springer.
  - Section 8.7.
  - Section 9.2.
  - Chapter 15.

- Slides of the lecture.

- Biau, Devroye, Lugosi (2008).
  Consistency of random forests and other averaging classifiers.
  Journal of Machine Learning Research, 9, 2015–2033.

Contents

Binary supervised classification

Notation:

- **Given:** for the random pair (X, Y) in $\mathbb{R}^d \times \{0, 1\}$ consisting of a random observation X and its random binary label Y (class), a sample of n i.i.d.: $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$.
- **Goal:** predict the label of the new (unseen before) observation $\boldsymbol{x}$.
- **Method:** construct a classification rule:

$$g : \mathbb{R}^d \to \{0, 1\}, \boldsymbol{x} \mapsto g(\boldsymbol{x}),$$

  so $g(\boldsymbol{x})$ is the prediction of the label for observation $\boldsymbol{x}$.
- **Criterion:** of the performance of g is the **error probability**:

$$\mathbb{P}\big(g(X) \neq Y\big) = \int_X \mathbb{1}\big(g(\boldsymbol{x}) \neq Y\big) \mu_X(d\boldsymbol{x}).$$

  where μ_X is the probability measure of X.
- **The best solution:** is to know the distribution of (X,Y):

$$g(\boldsymbol{x}) = \mathbb{E}[Y|X = \boldsymbol{x}].$$

Bayes classification rule

Bayes formula for the probability of event A conditioned on event B:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}\,.$$

In the context of binary supervised classification:

$$P(Y = 0|X = \boldsymbol{x}) = \frac{P(X = \boldsymbol{x}|Y = 0)\,P(Y = 0)}{P(X = \boldsymbol{x})}$$

and

$$P(Y = 1|X = \boldsymbol{x}) = \frac{P(X = \boldsymbol{x}|Y = 1)\,P(Y = 1)}{P(X = \boldsymbol{x})}\,.$$

When deciding which class to assign $\boldsymbol{x}$ we choose "1" if

$$P(Y = 1|X = \boldsymbol{x}) > P(Y = 0|X = \boldsymbol{x}) \quad \text{or} \quad \frac{P(Y = 1|X = \boldsymbol{x})}{P(Y = 0|X = \boldsymbol{x})} > 1\,.$$

So choose "1" if $\dfrac{P(X = \boldsymbol{x}|Y = 1)\,P(Y = 1)}{P(X = \boldsymbol{x}|Y = 0)\,P(Y = 0)} = \dfrac{f_1(\boldsymbol{x})\pi_1}{f_0(\boldsymbol{x})\pi_0} > 1$ and "0" if not.

Iris data

Fisher's iris data: is this the same flower?

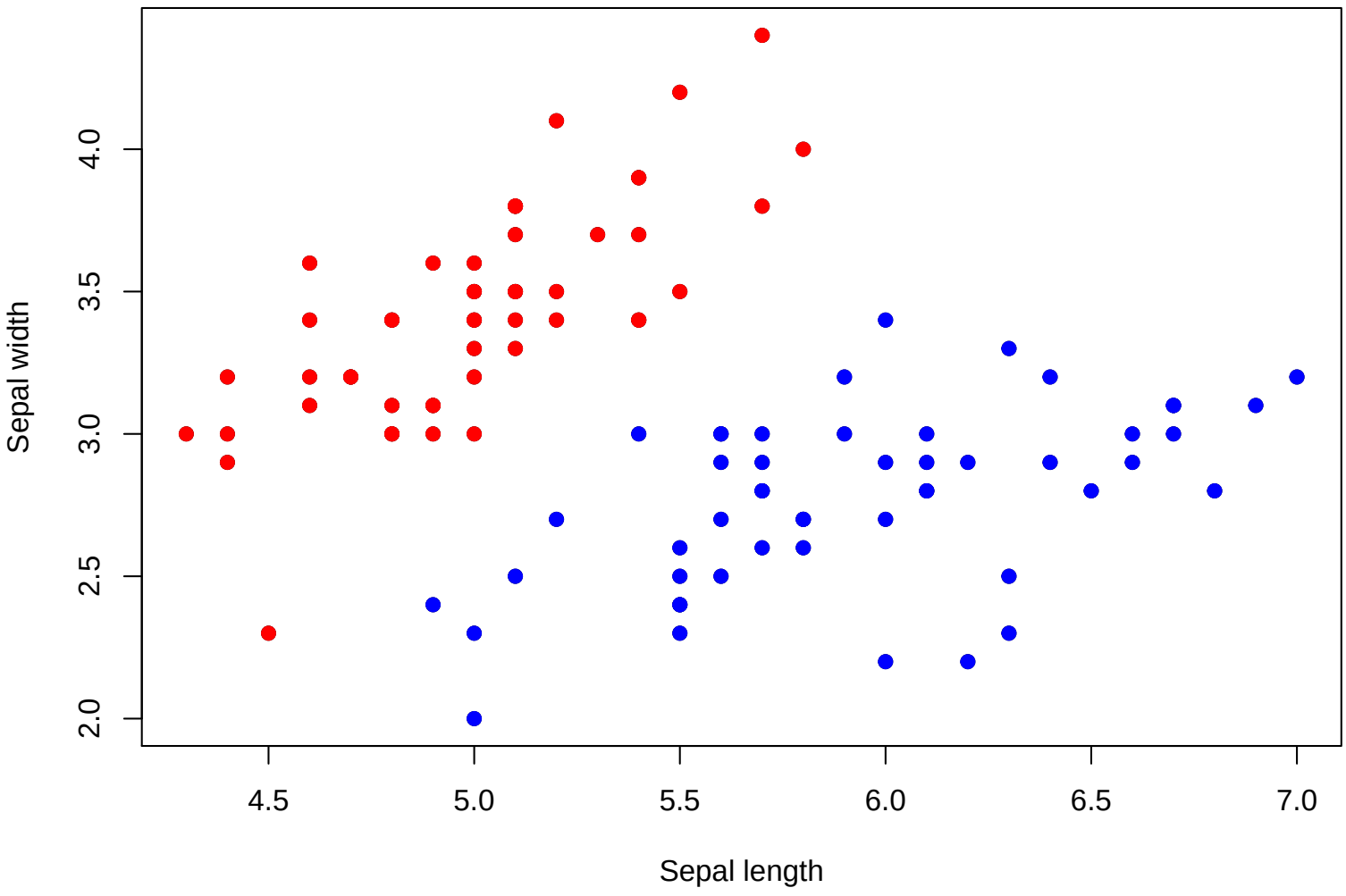
Iris **setosa**

Iris **versicolor**

Iris data

Iris **setosa**		Iris **versicolor**	
Sepal length (cm)	Sepal width (cm)	Sepal length (cm)	Sepal width (cm)
5.1	3.5	7	3.2
4.9	3	6.4	3.2
4.7	3.2	6.9	3.1
4.6	3.1	5.5	2.3
5	3.6	6.5	2.8
5.4	3.9	5.7	2.8
4.6	3.4	6.3	3.3
5	3.4	4.9	2.4
4.4	2.9	6.6	2.9
...	...	...	...
...	...	...	...
...	...	...	...
4.6	3.2	6.2	2.9
5.3	3.7	5.1	2.5
5	3.3	5.7	2.8

Iris data



Linear discriminant analysis

- **Assumptions:** Both classes are normally distributed with the same covariance matrix, *i.e.* $X|Y = j \sim N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\,, j = 0, 1$ or

$$f_j(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_j)}}\, e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)}\,, \quad \text{for } j = 0, 1$$

$$\text{and} \quad \boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}\,.$$

- **Plug-in into Bayes:**

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \frac{P(Y=1|X=\boldsymbol{x})}{P(Y=0|X=\boldsymbol{x})} > 1\,, \\ 0 & \text{else}\,; \end{cases}$$

$$\text{or} \quad g(\boldsymbol{x}) = \mathbb{1}\left(\log \frac{\pi_1 f_1(\boldsymbol{x})}{\pi_0 f_0(\boldsymbol{x})} > 0\right).$$

Linear discriminant analysis

$$
\begin{aligned}
\log \frac{\pi_1 f_1(\boldsymbol{x})}{\pi_0 f_0(\boldsymbol{x})} &= \log \frac{\pi_1}{\pi_0} + \log \frac{\frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_1)}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}}{\frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_0)}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)}} \\
&= \log \frac{\pi_1}{\pi_0} + \log \frac{\sqrt{\det(\boldsymbol{\Sigma}_0)}}{\sqrt{\det(\boldsymbol{\Sigma}_1)}} \\
&+ \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) \\
&= \log \frac{\pi_1}{\pi_0} + \log \frac{\sqrt{\det(\boldsymbol{\Sigma}_0)}}{\sqrt{\det(\boldsymbol{\Sigma}_1)}} \\
&+ \frac{1}{2}\left(\boldsymbol{x}^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 - \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{x} + \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right) \\
&- \frac{1}{2}\left(\boldsymbol{x}^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1\right) \\
&= \ldots
\end{aligned}
$$

Exploit $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$ to simplify.

Linear discriminant analysis

$$
\begin{aligned}
\log \frac{\pi_1 f_1(\boldsymbol{x})}{\pi_0 f_0(\boldsymbol{x})} &= \log \frac{\pi_1}{\pi_0} + \log \frac{\sqrt{\det(\boldsymbol{\Sigma})}}{\sqrt{\det(\boldsymbol{\Sigma})}} \\
&+ \frac{1}{2}\left(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\mu_0 - \mu_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \mu_0^T\boldsymbol{\Sigma}^{-1}\mu_0\right) \\
&- \frac{1}{2}\left(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\mu_1 - \mu_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \mu_1^T\boldsymbol{\Sigma}^{-1}\mu_1\right) \\
&= \log \frac{\pi_1}{\pi_0} + \frac{1}{2}\mu_0^T\boldsymbol{\Sigma}^{-1}\mu_0 - \frac{1}{2}\mu_1^T\boldsymbol{\Sigma}^{-1}\mu_1 \\
&+ \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{1-}(\mu_1 - \mu_0) + \frac{1}{2}(\mu_1 - \mu_0)^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} \\
&= \log \frac{\pi_1}{\pi_0} - \frac{1}{2}(\mu_1 - \mu_0)^T\boldsymbol{\Sigma}^{-1}(\mu_1 - \mu_0) \\
&+ \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\mu_1 - \mu_0)\,.
\end{aligned}
$$

Linear discriminant analysis (algorithm)

- **Learning:**

  Let
  - $I_0 = \{i \,:\, y_i = 0\,,\ i = 1, ..., n\}$ $(n_0 = \#I_0)$;
  - $I_1 = \{i \,:\, y_i = 1\,,\ i = 1, ..., n\}$ $(n_1 = \#I_1)$.

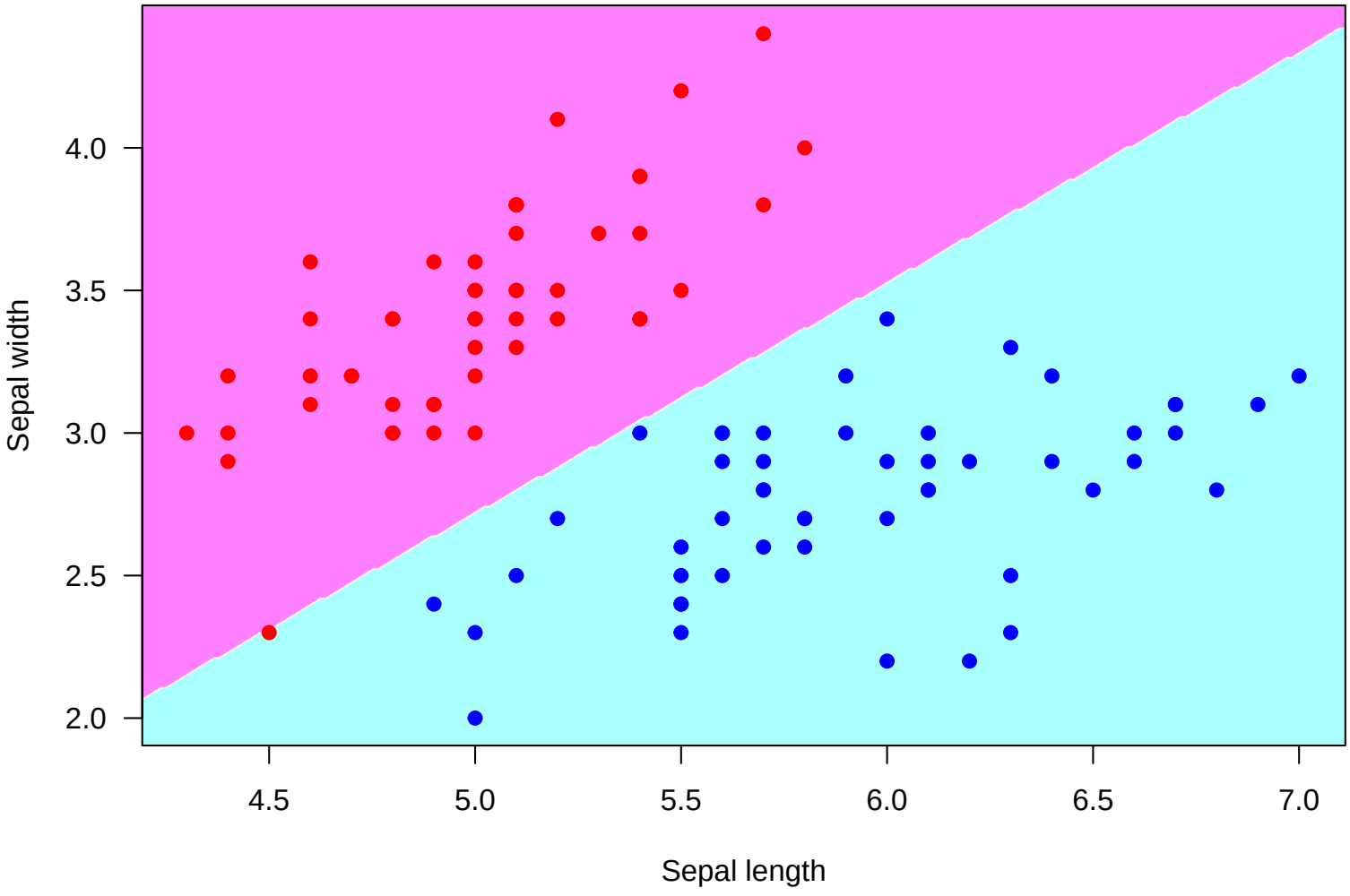
  Estimate
  - Priors: $p_0 = \frac{n_0}{n}$, $p_1 = \frac{n_1}{n}$;
  - Means: $\bar{\boldsymbol{x}}_0 = \frac{1}{n_0}\sum_{i\in I_0} \boldsymbol{x}_i$, $\bar{\boldsymbol{x}}_1 = \frac{1}{n_1}\sum_{i\in I_1} \boldsymbol{x}_i$, $(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_0)$;
  - Common covariance matrix:

    $$\boldsymbol{S} = \frac{1}{n-2}\left(\sum_{i\in I_0}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_0)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_0)^T + \sum_{i\in I_1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_1)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_1)^T\right) .$$

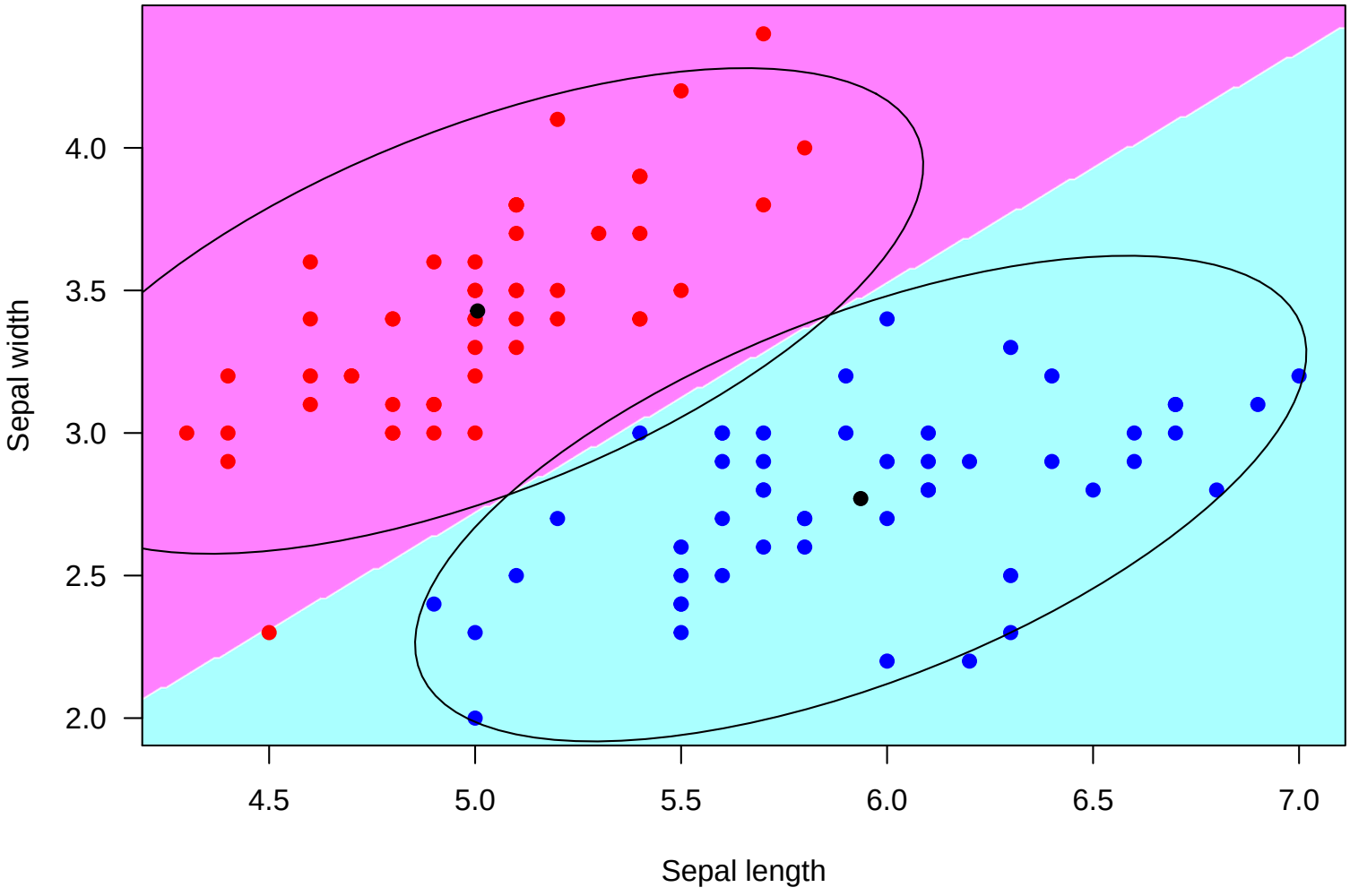
- **Classification:** For a new observation $\boldsymbol{x}$

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \log\frac{p_1}{p_0} - \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_0)^T\boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_0) \\ & \quad + \boldsymbol{x}^T\boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_0) > 0\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Linear discriminant analysis (iris data)



Linear discriminant analysis (iris data)



k-nearest neighbors (algorithm)

For $\boldsymbol{x} \in \mathbb{R}^d$ and some integer $0 < k < n$, let a set $I_k(\boldsymbol{x})$ index the k-nearest neighbors of the point $\boldsymbol{x}$:

$$I_k(\boldsymbol{x}) = \{i(1), ..., i(k)\},$$

where $\|\boldsymbol{x} - \boldsymbol{x}_{i(1)}\| \leq \|\boldsymbol{x} - \boldsymbol{x}_{i(2)}\| \leq ... \leq \|\boldsymbol{x} - \boldsymbol{x}_{i(n)}\|$ is an ascending order. k is to be set, *e.g.* chosen by the means of cross-validation.

Then the k-**nearest neighbors** (kNN) algorithm **classifies** a new observation as follows:

- Calculate classes' proportion in the k-neighborhood:

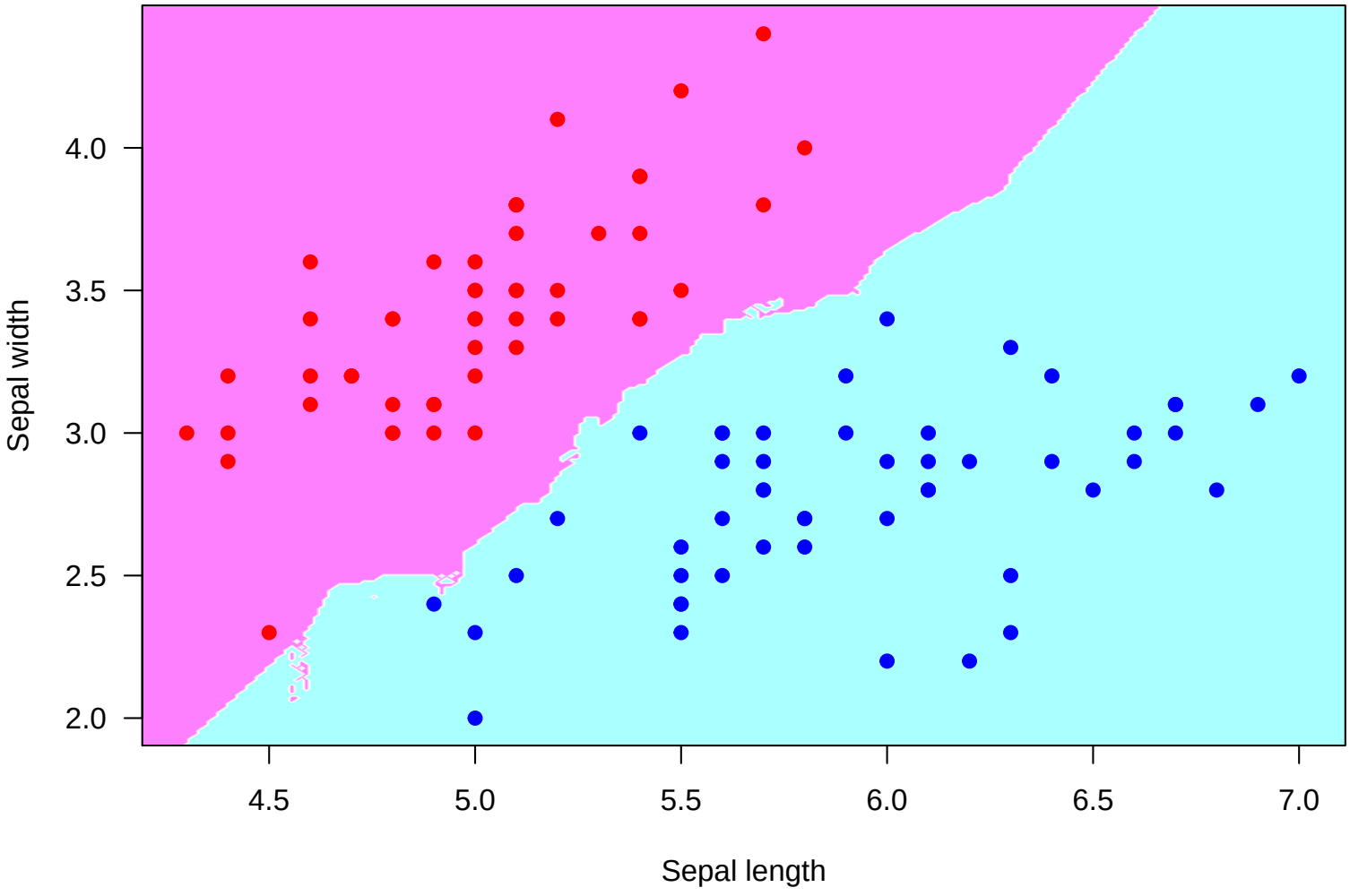
$$p_k(\boldsymbol{x}) = \frac{\sum_{i \in I_k(\boldsymbol{x})} \mathbb{1}(y_i = 1)}{\sum_{i \in I_k(\boldsymbol{x})} \mathbb{1}(y_i = 0)}.$$

- Assign the class based on majority:

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p_k(\boldsymbol{x}) > 1, \\ 0 & \text{otherwise}. \end{cases}$$

- Deal with ties, *e.g.* decide randomly, or choose odd ks only.

k-nearest neighbors (iris data, k=9)



k-nearest neighbors classifier (universal consistency)

Under certain assumptions, kNN is universally consistent, *i.e.* approaches the classification error of the Bayes classifier with increasing length of the training sample n.

Theorem (Stone, 1977)

If $k \to \infty$ and $\frac{k}{n} \to 0$ then the kNN in $\mathbb{R}^d$ with Euclidean distance is universally consistent, i.e.

$$\lim_{n\to\infty} \mathbb{E}\Big[\int_X \big(g_n(\boldsymbol{x}) - \mathbb{E}[Y|X=\boldsymbol{x}]\big)^2 \mu_X(d\boldsymbol{x})\Big] = 0\,,$$

for any probability measure of (X, Y).

In general for kernel-based methods with h being the bandwidth:

Theorem (Devroye-Krzyżak, 1989)

If $h \to 0$ and $nh^d \to +\infty$ then the kernel-based classifier is universally consistent.

Rate of convergence

Nonparametric methods suffer from the **curse of dimensionality**: if the number of exploratory variables is large, the spherical neighborhood is filled poorly, which reduces the rate of convergence.

Consider the kNN regression estimate:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{k} \sum_{i \in I_k(\boldsymbol{x})} y_i \,.$$

Theorem (Györfi, Kohler, Krzyżak, Walk, 2002)

If the regression function is Lipschitz continuous then for the kNN estimator it holds

$$\mathbb{E}\Big[\int_X \big(\hat{f}_n(\boldsymbol{x}) - \mathbb{E}[Y|X=\boldsymbol{x}]\big)^2 \mu_X(d\boldsymbol{x})\Big] = O(n^{-\frac{2}{d+2}}) \,.$$

In practice non-parametric estimators possess poor performance in high-dimensional spaces.

Possible solution: aggregation methods

Aggregation methods allow, to a certain extent, deal with

1. **curse of dimensionality**;
2. **sensibility** of the method w.r.t. the choice of parameters;
3. **preserve previous properties** while being computationally tractable.

These proposed approaches are based on the **aggregation**, *i.e.*:

1. construct an ensemble of $g_1, ..., g_B$ of **weak learning algorithms**;
2. aggregate them into the **final classifier**

$$g(\boldsymbol{x}) = \frac{1}{B} \sum_{k=1}^{B} g_k(\boldsymbol{x}) \, .$$

The key concepts:

- **bagging and random forests**;
- **boosting**.

Contents

Classification tree (algorithm)

Growing a tree (training)

Input:

- Training sample $\big((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Measure of impurity $Q^{(m)}(T)$ for node m of tree T.
- Stopping criteria $S^{(m)}(T)$ for node m of tree T.

1. Define the root node by the region $R^{(0)}$ containing the entire sample, set $m = 0$.
2. If $S^{(m)}(T)$ is fulfilled then stop for this node.
3. Find a split (one-variable threshold) diving node region $R^{(m)}$ into two nodes with subregions $R^{(m_L)}$ and $R^{(m_R)}$ to minimize $Q^{(m)}(T)$.
4. Repeat steps 2–3 for all leaves till global stopping.

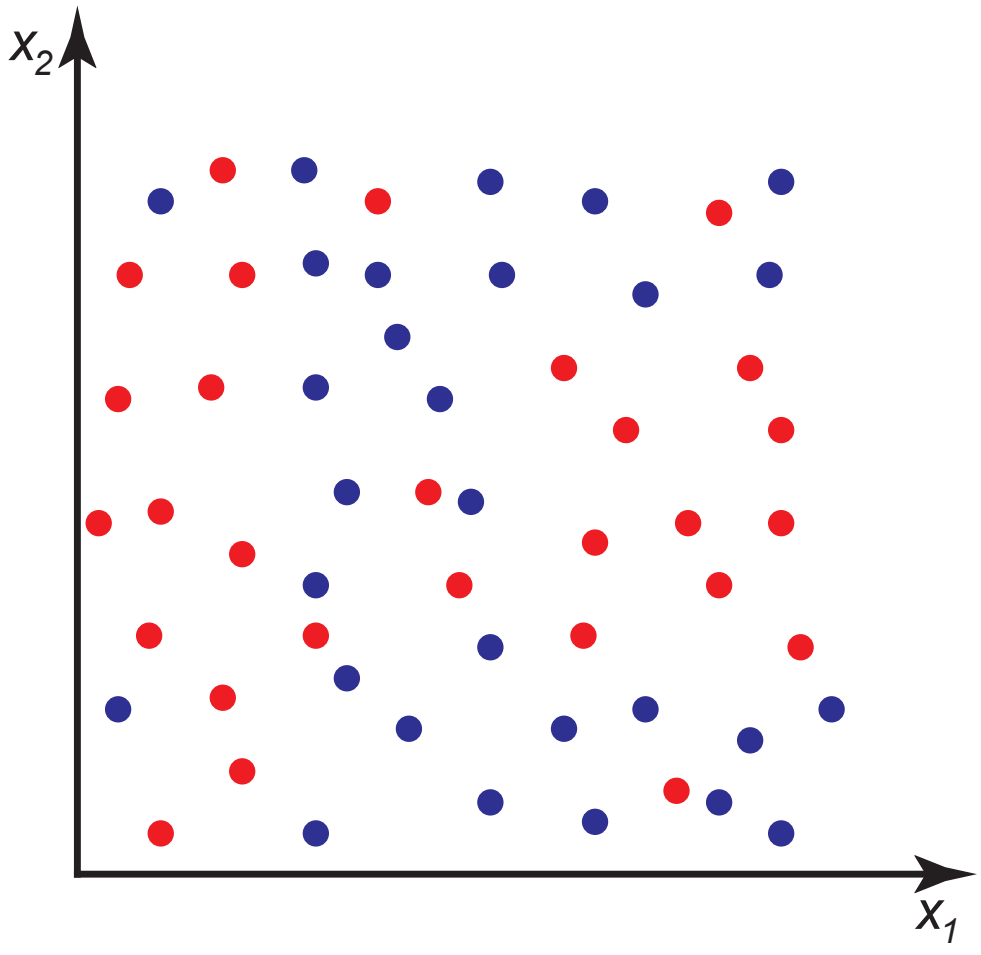
Output: The tree T.

Descending the tree (classification)

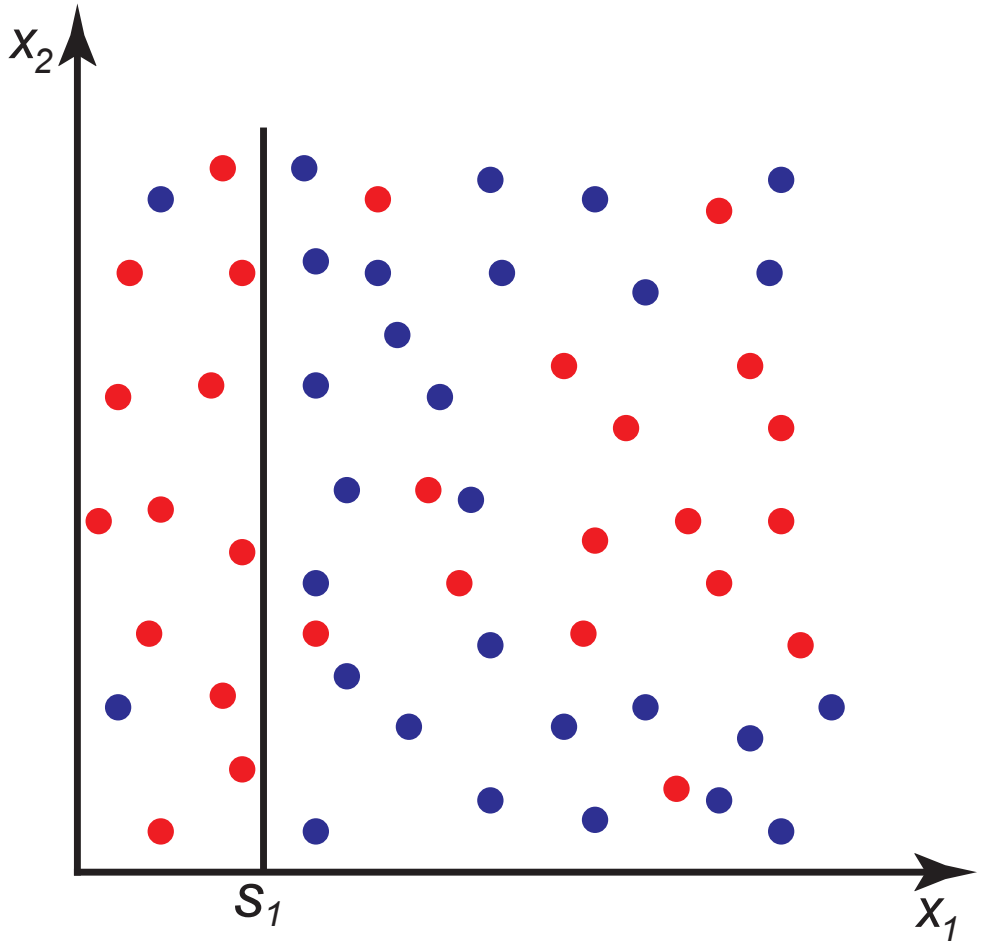
- Descend the tree till a terminal node, in each node m classify the new observation $\boldsymbol{x} \in \mathbb{R}^d$ to class $k(m)$

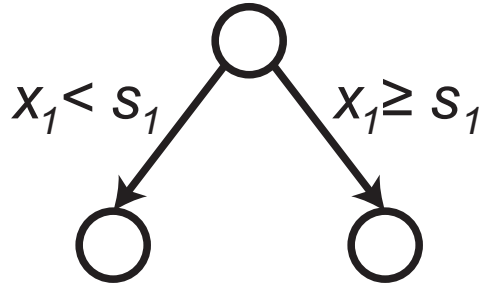
$$k(m) = \underset{j \in \{0,1\}}{\arg\max} \sum_{i \in R^{(m)}} I(y_i = j)\,.$$

Classification tree (illustration)

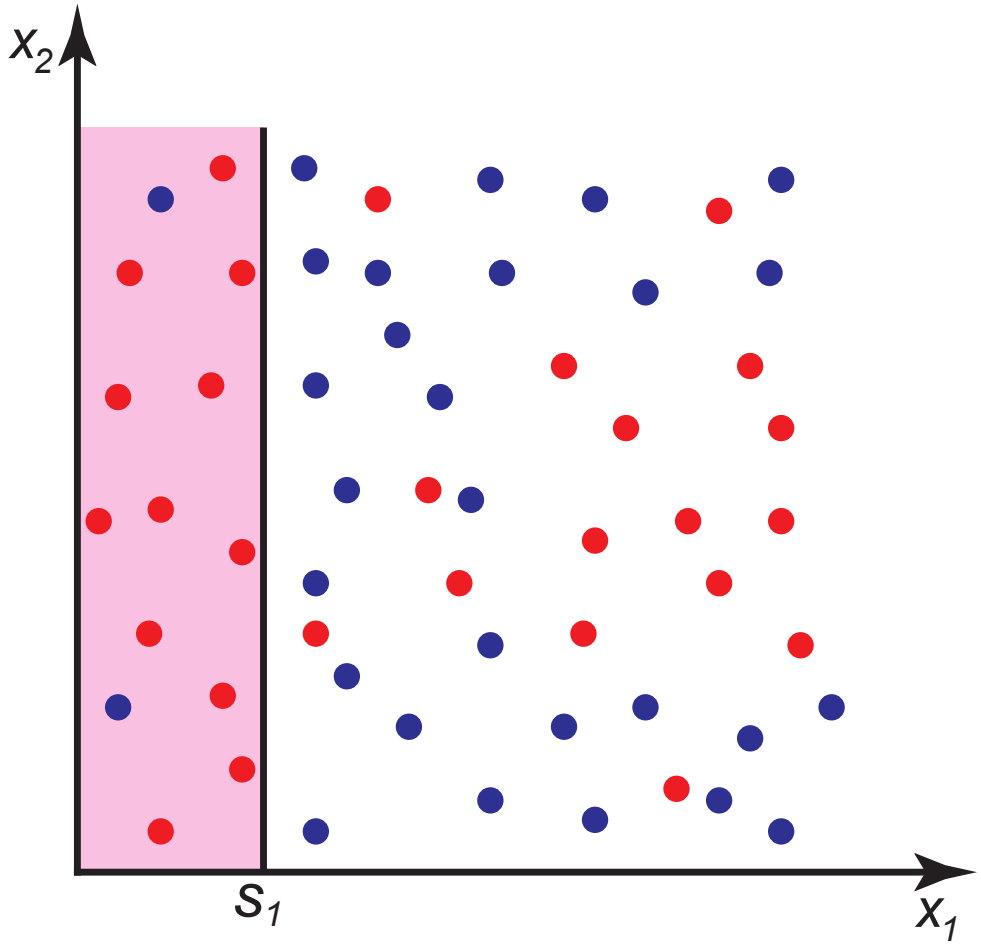


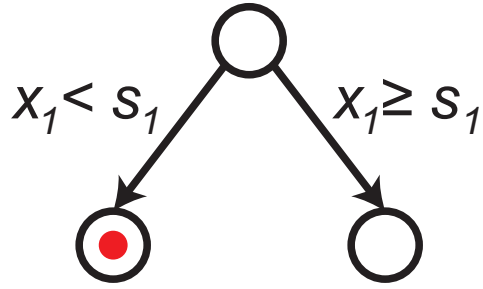
Classification tree (illustration)



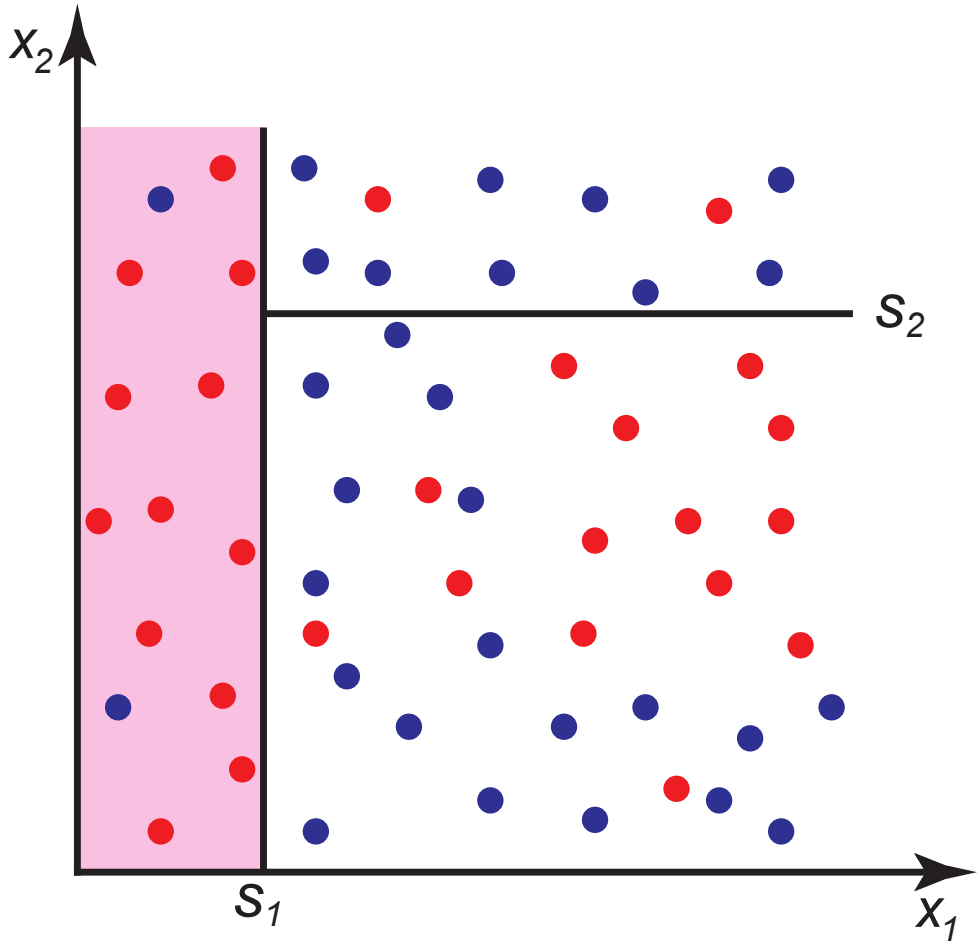


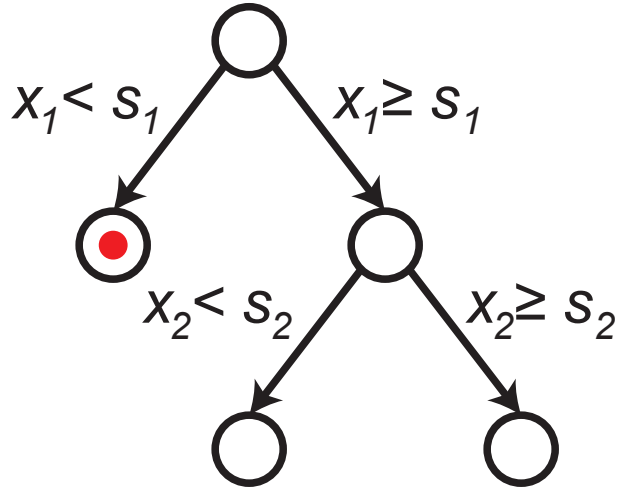
Classification tree (illustration)



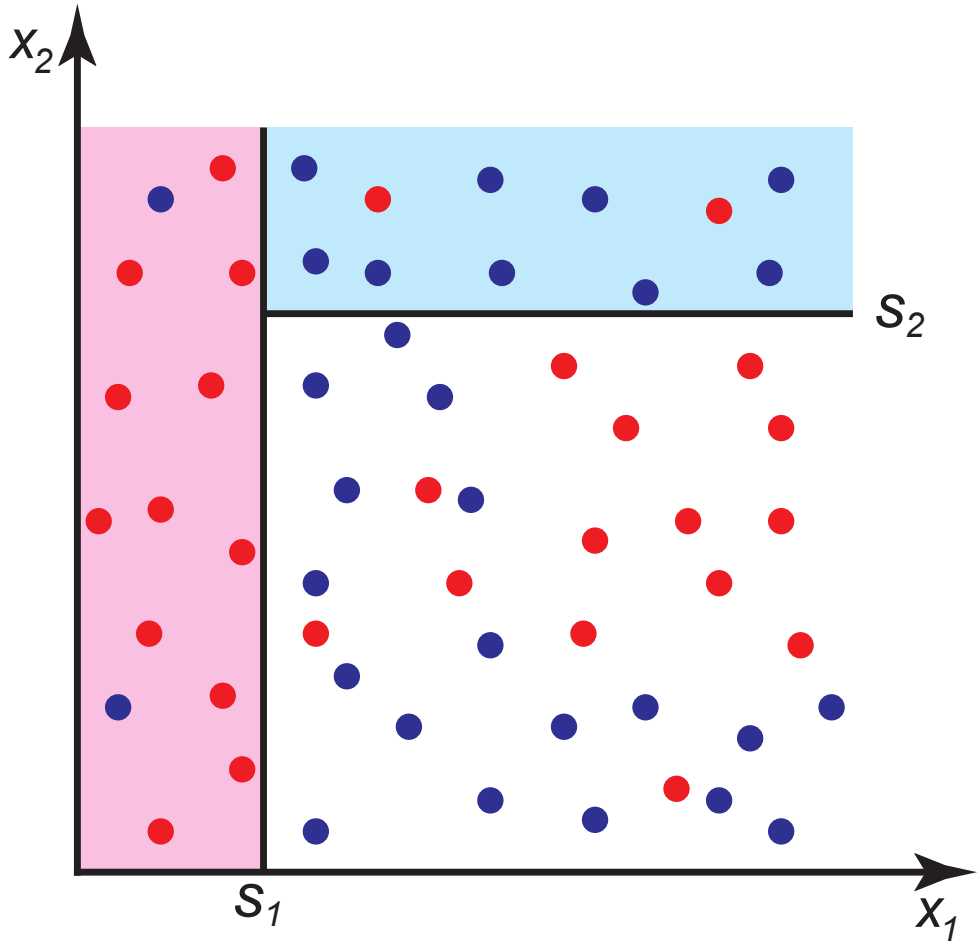


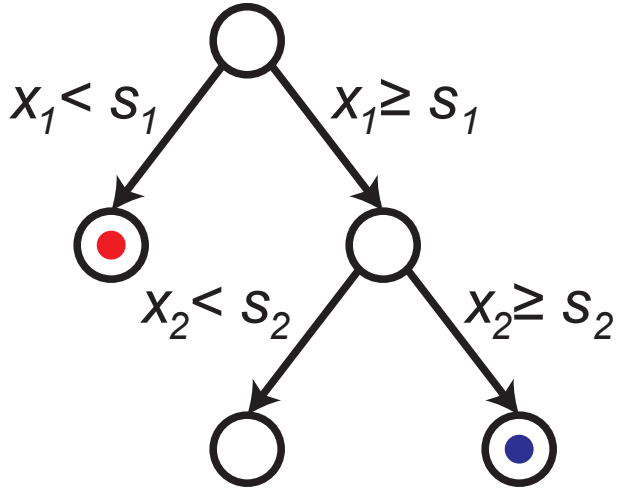
Classification tree (illustration)



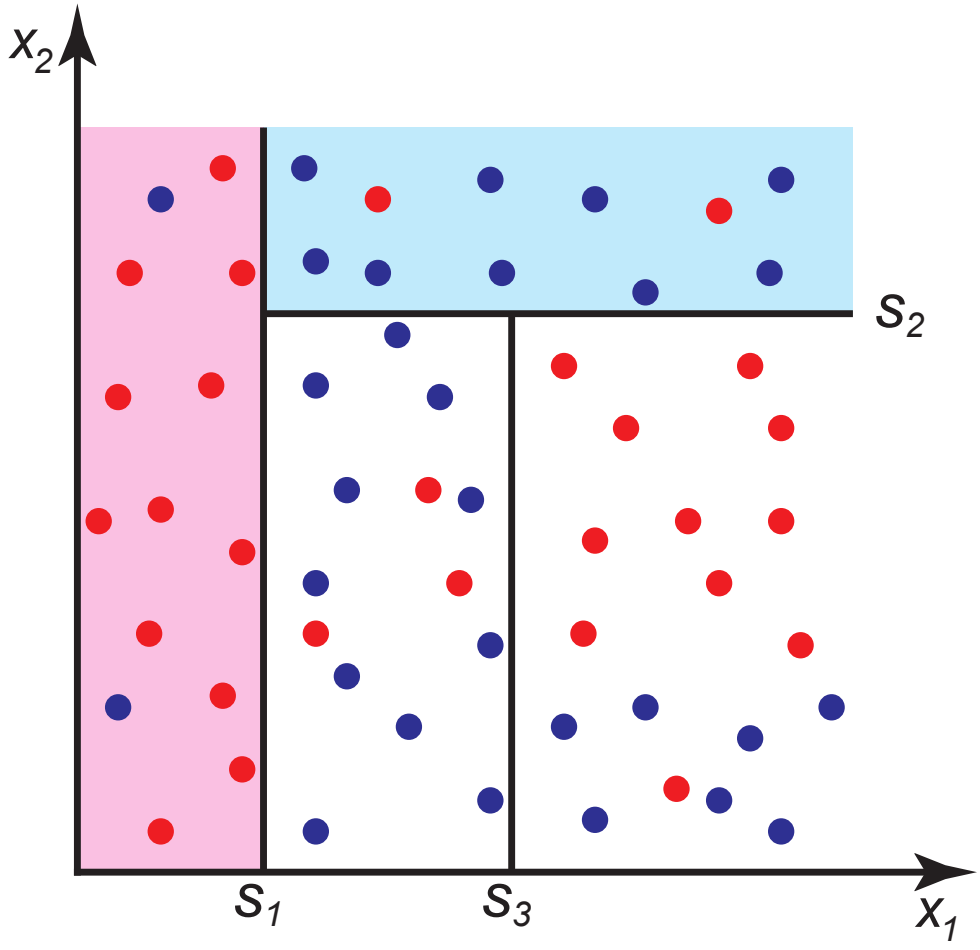


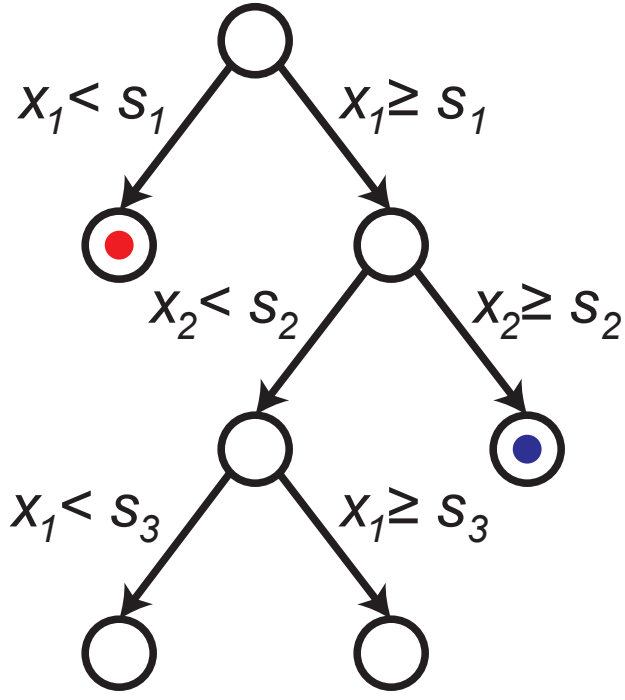
Classification tree (illustration)



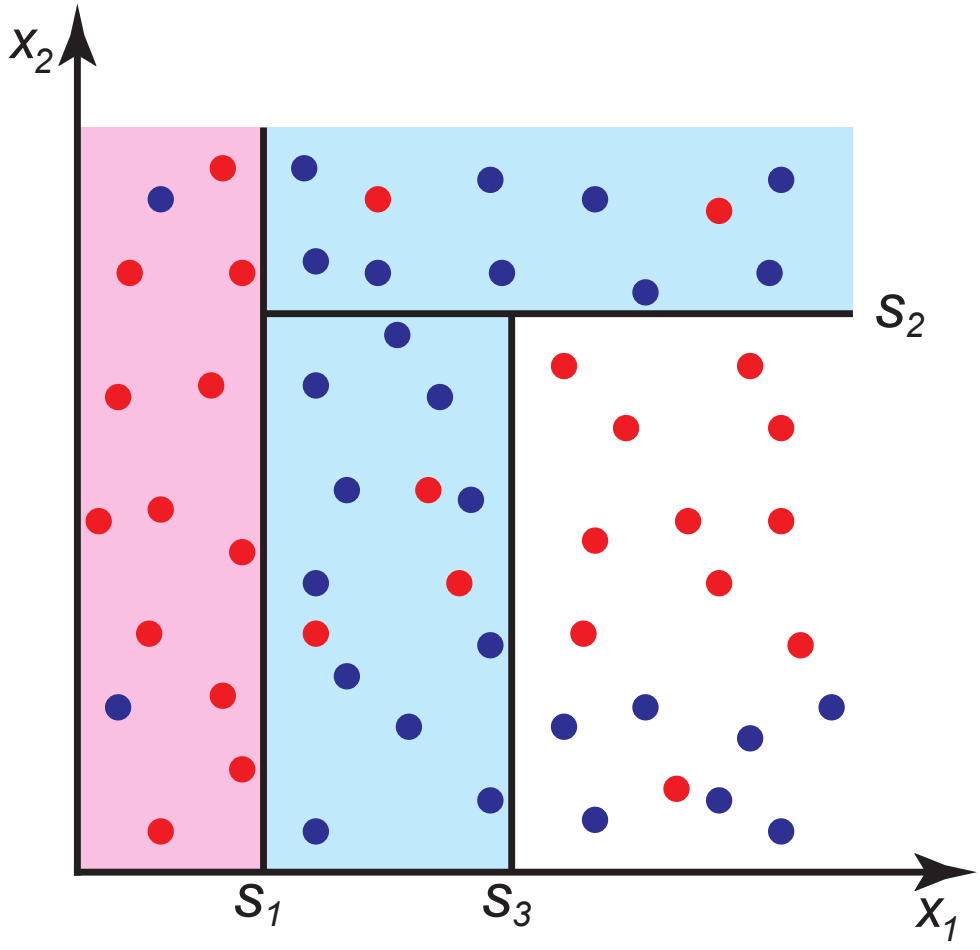


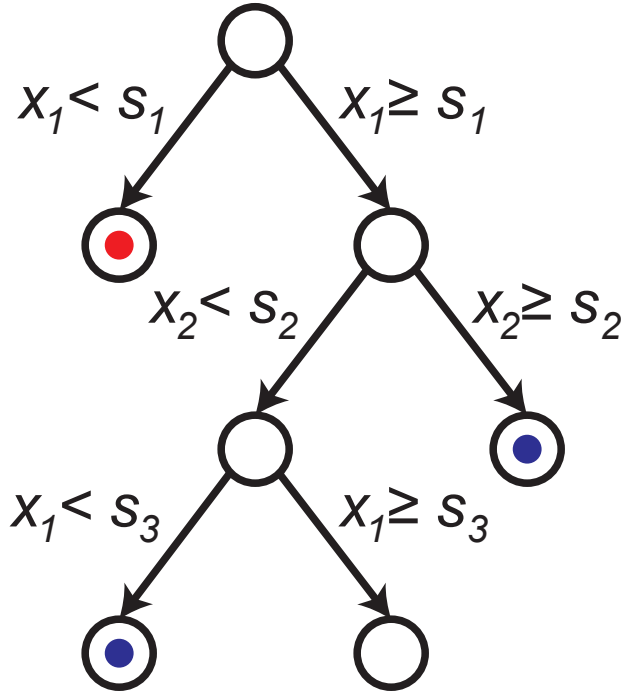
Classification tree (illustration)



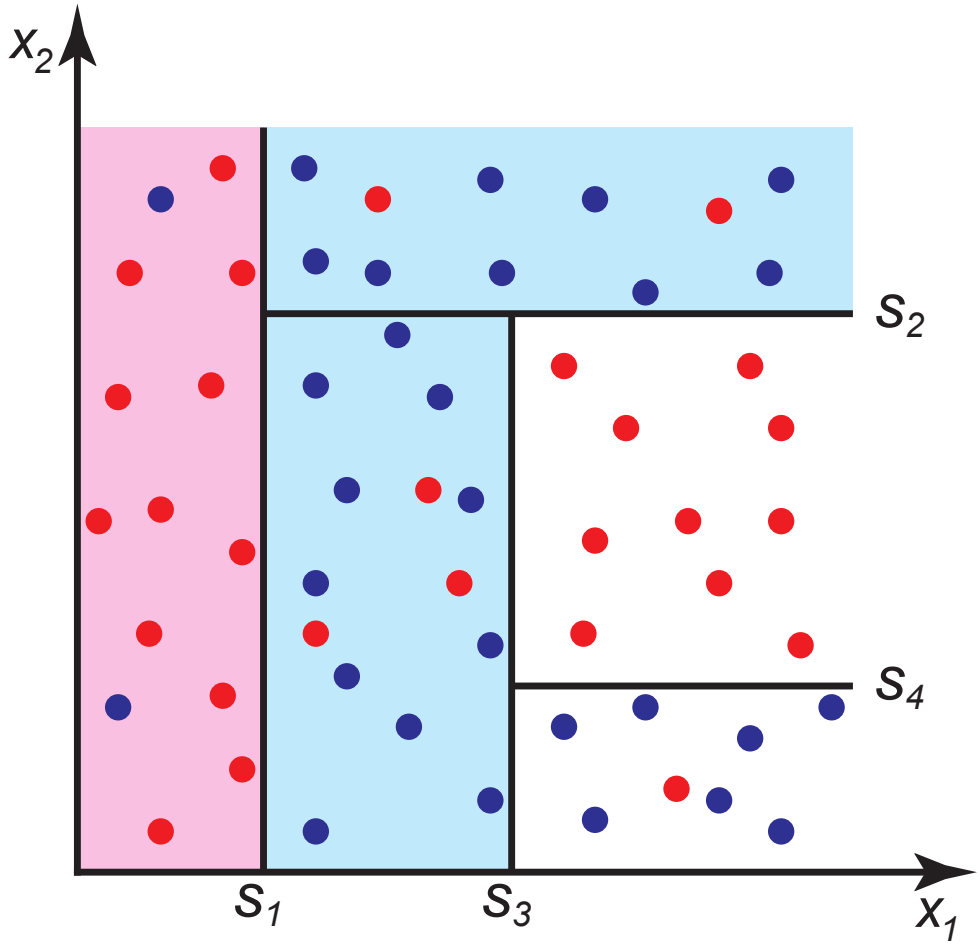


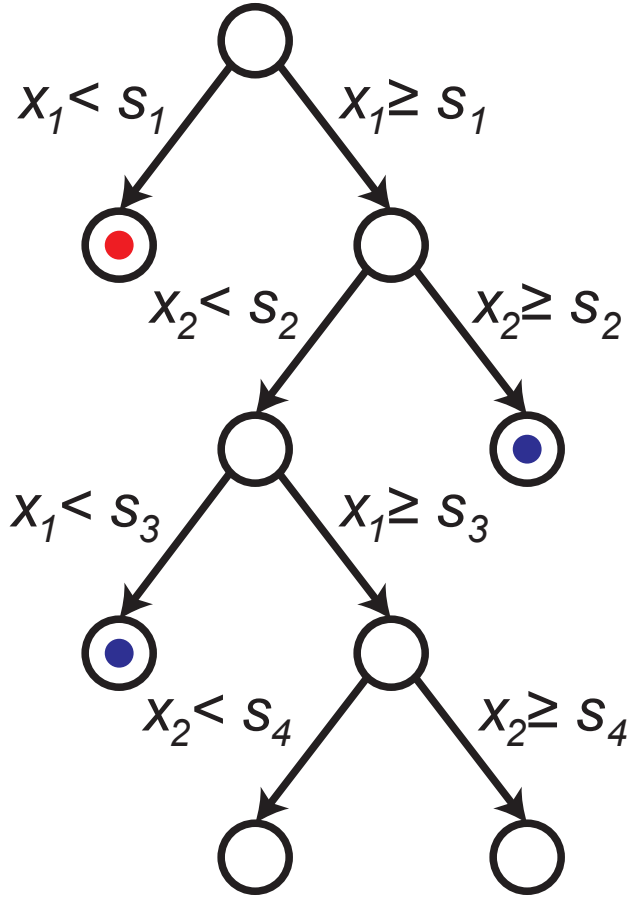
Classification tree (illustration)



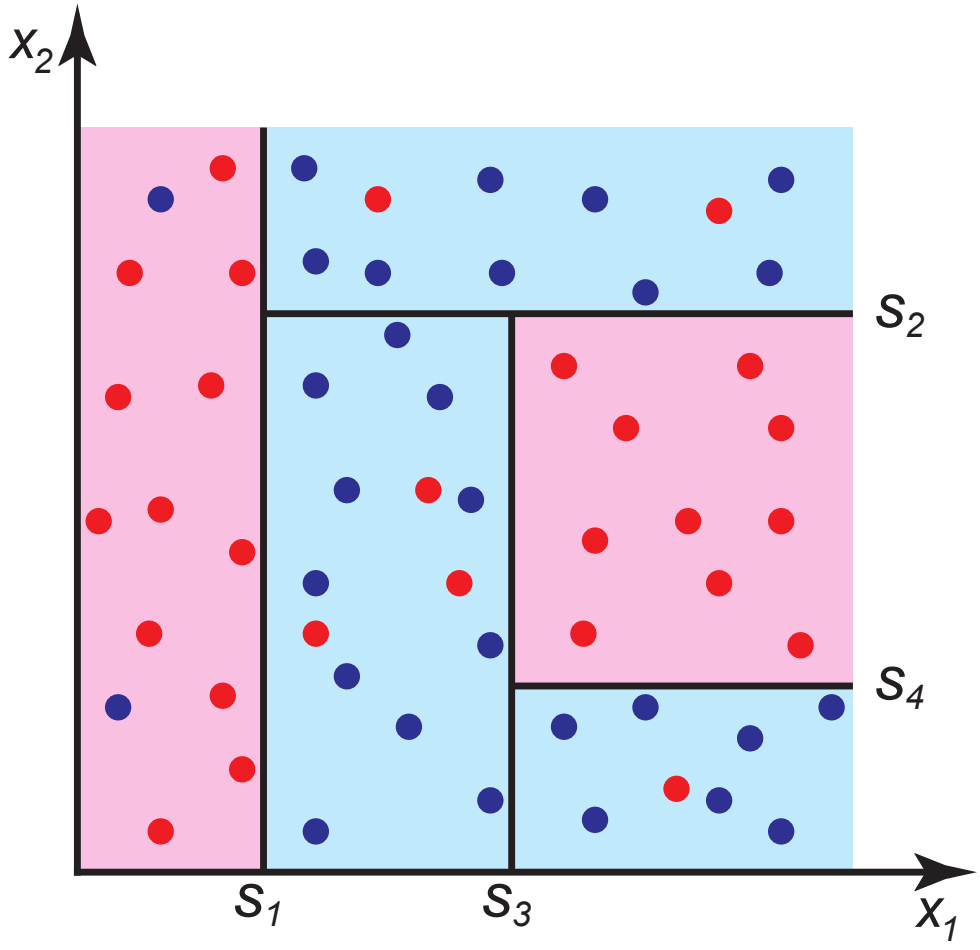


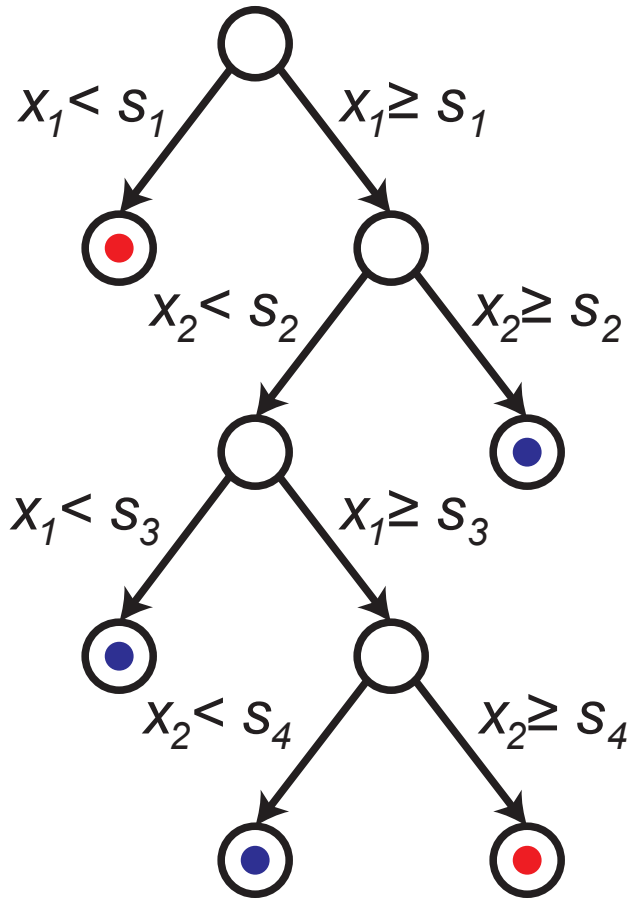
Classification tree (illustration)





Classification tree (illustration)





Classification tree: choice of impurity measure

Let $n^{(m)} = \#\{\boldsymbol{x} \mid \boldsymbol{x} \in R^{(m)}\}$ be the number of observations in region $R^{(m)}$. Then the classification accuracy of node m classifying to class k is

$$\hat{p}_k^{(m)} = \frac{1}{n^{(m)}} \sum_{\boldsymbol{x}_i \in R^{(m)}} I(y_i = k)\,.$$

Possible choices for Q:

- Misclassification error:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}}(1 - \hat{p}_{k(m_L)}^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}}(1 - \hat{p}_{k(m_R)}^{(m_R)})\,,$$

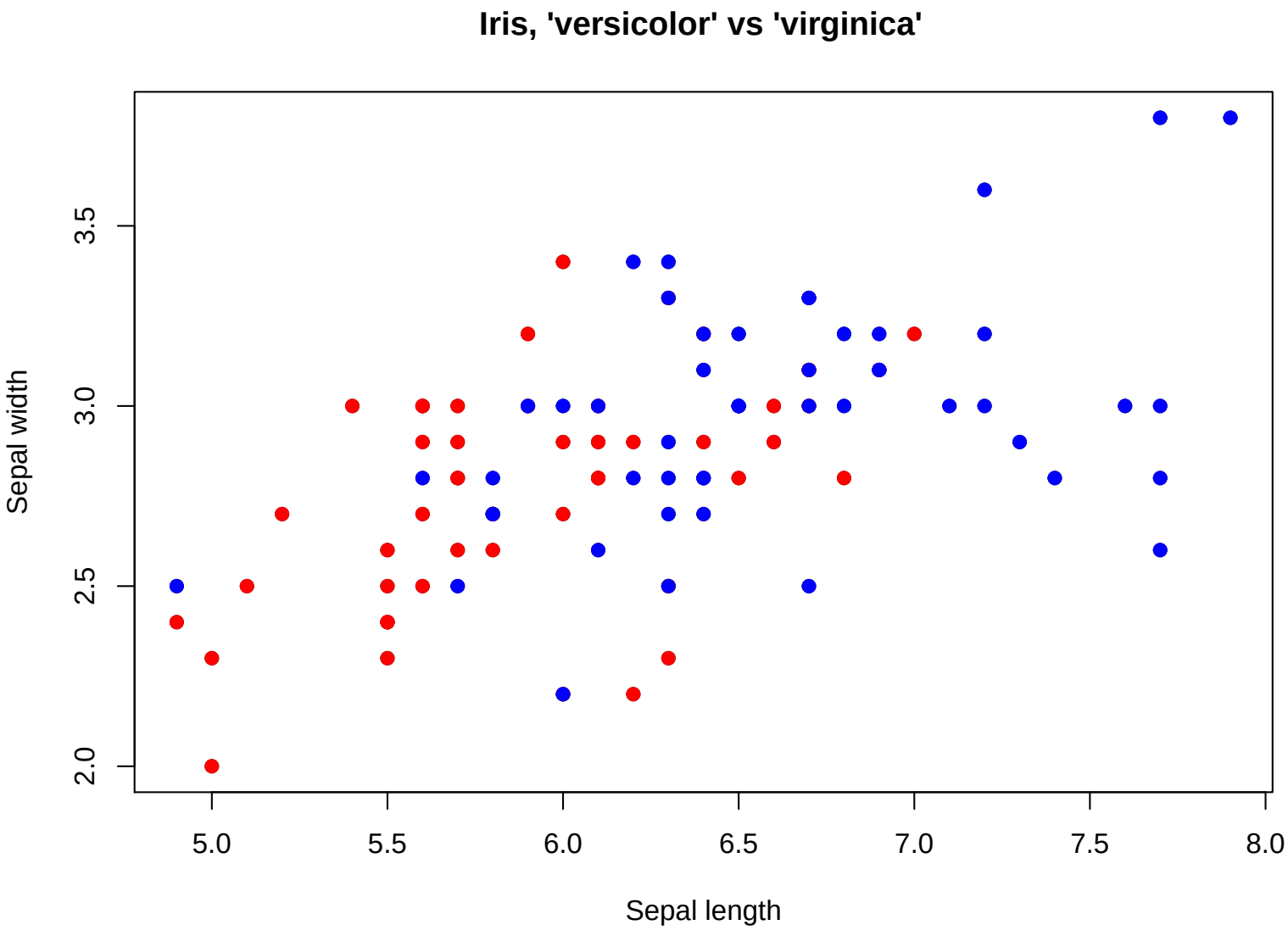
- Gini index:

$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}} 2\hat{p}_k^{(m_L)}(1 - \hat{p}_k^{(m_L)}) + \frac{n^{(m_R)}}{n^{(m)}} 2\hat{p}_k^{(m_R)}(1 - \hat{p}_k^{(m_R)})\,,$$

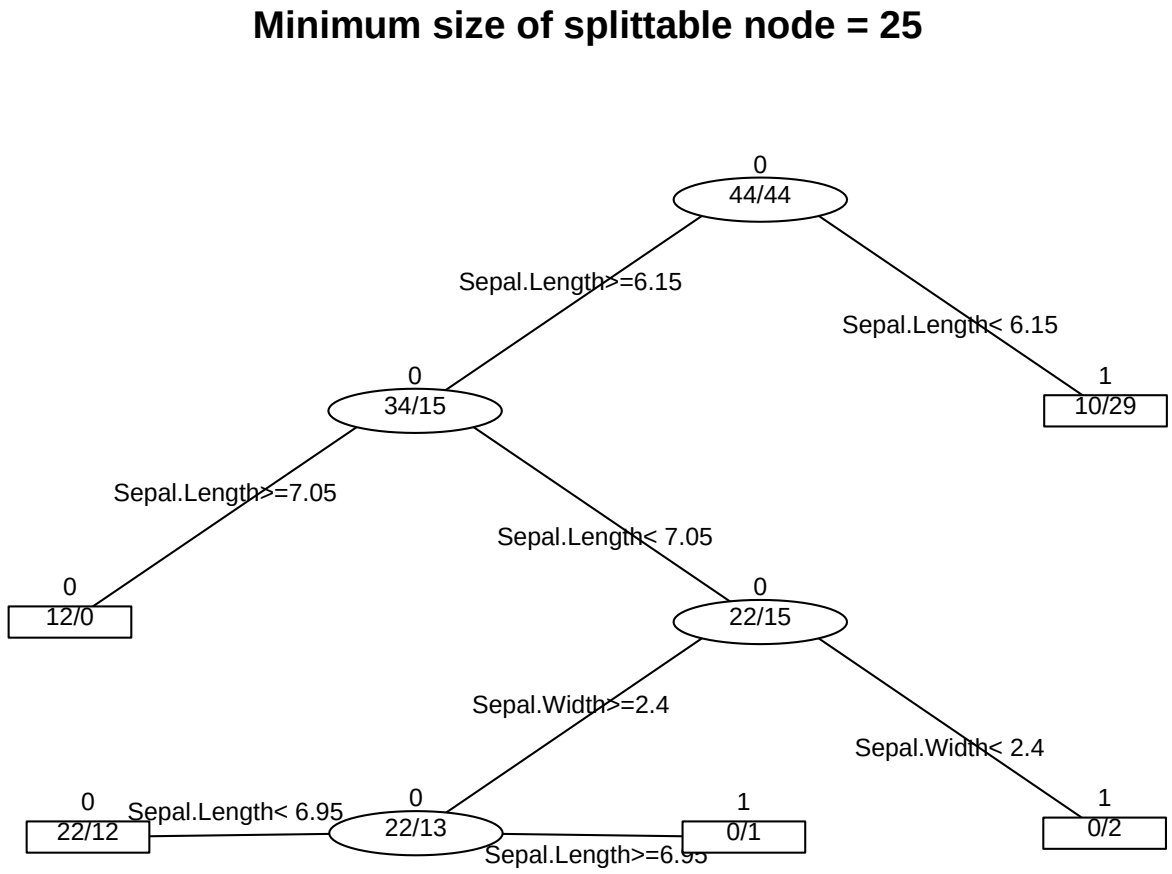
- Cross-entropy (deviance):

$$\begin{aligned} Q^{(m)}(T) = \; - \; & \Big(\frac{n^{(m_L)}}{n^{(m)}} \big(\hat{p}_k^{(m_L)} \log \hat{p}_k^{(m_L)} + (1 - \hat{p}_k^{(m_L)}) \log(1 - \hat{p}_k^{(m_L)}) \big) \\ & + \frac{n^{(m_R)}}{n^{(m)}} \big(\hat{p}_k^{(m_R)} \log \hat{p}_k^{(m_R)} + (1 - \hat{p}_k^{(m_R)}) \log(1 - \hat{p}_k^{(m_R)}) \big) \Big)\,. \end{aligned}$$

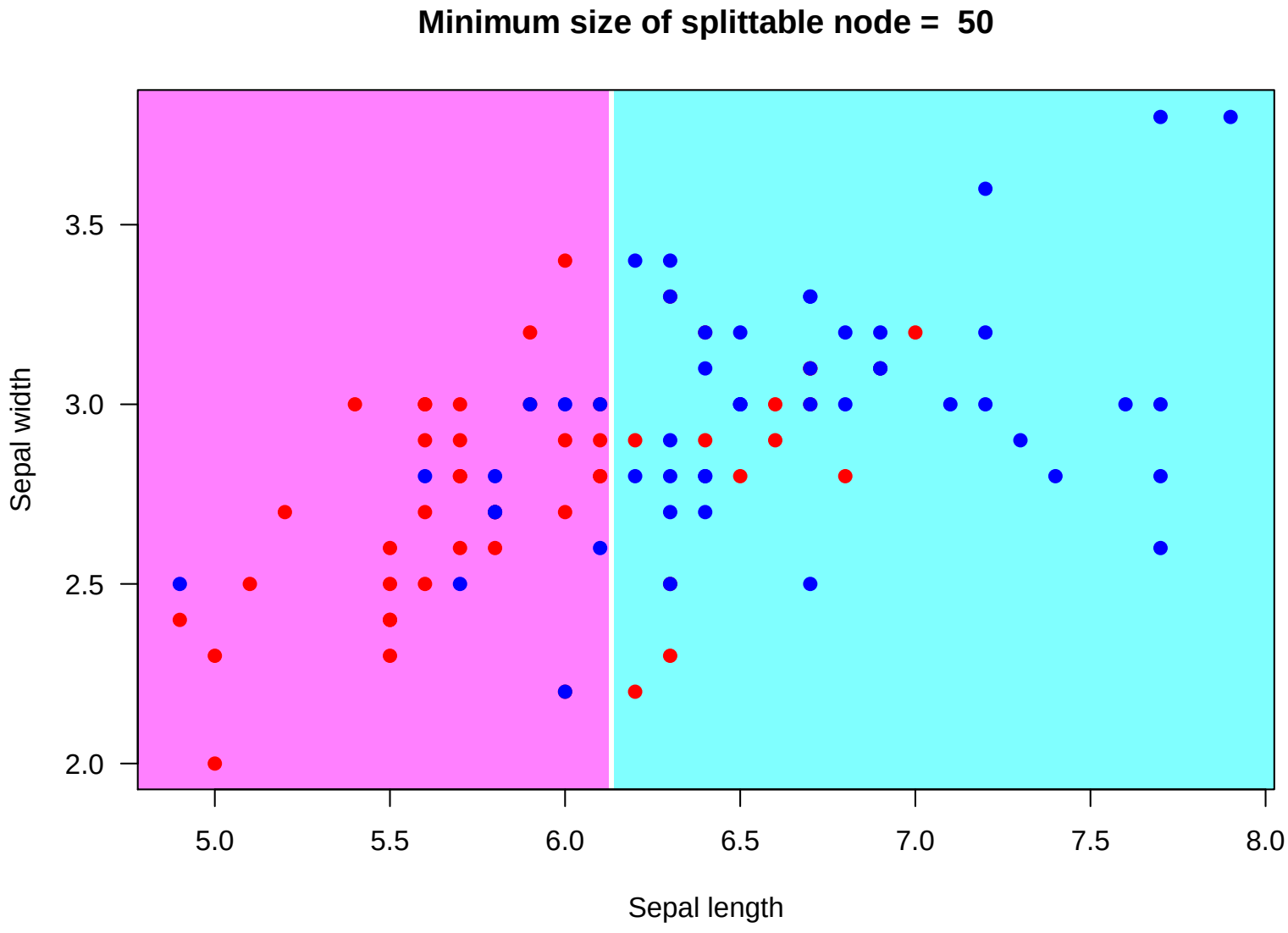
Classification tree: iris data



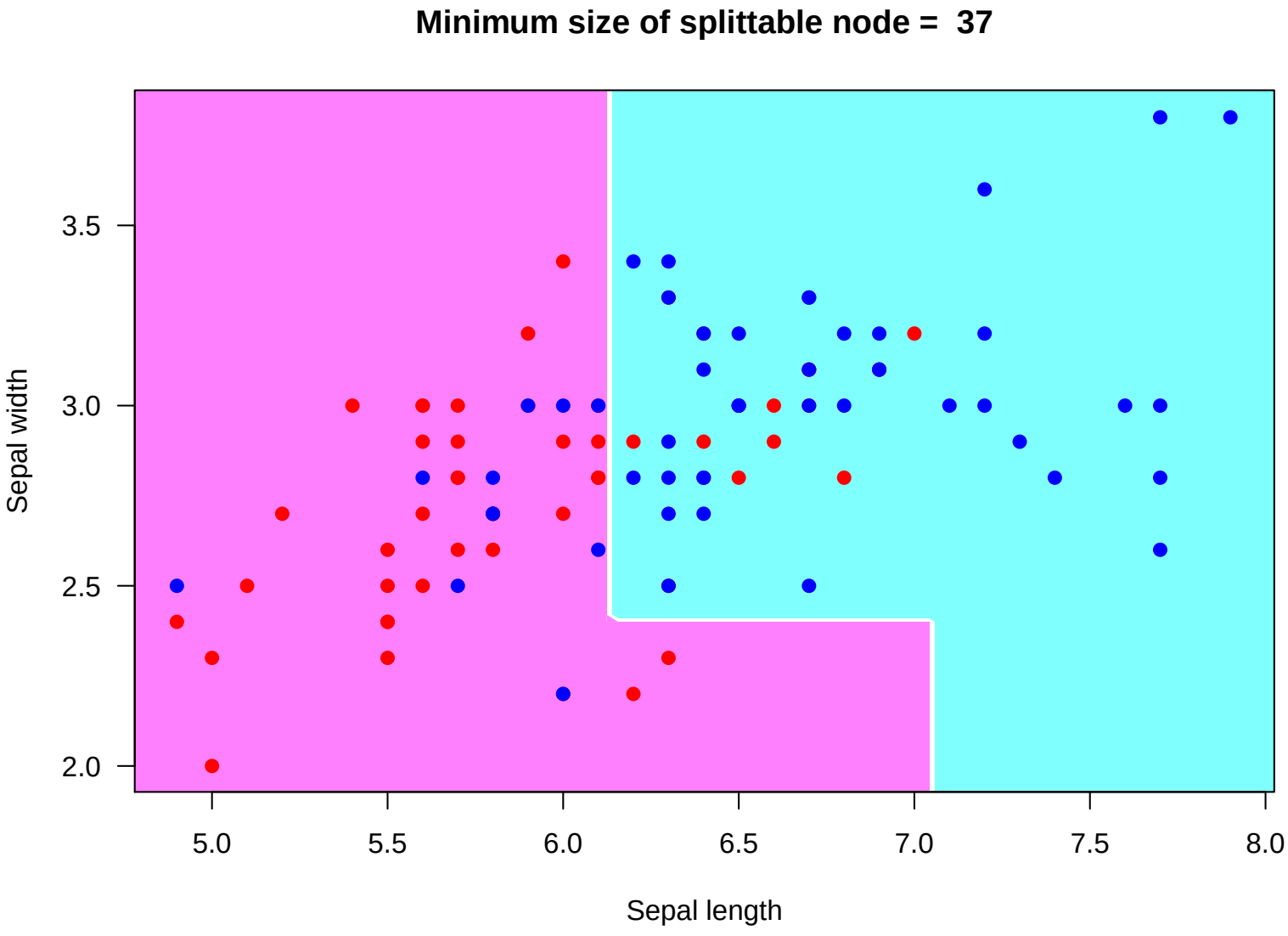
Classification tree: iris data



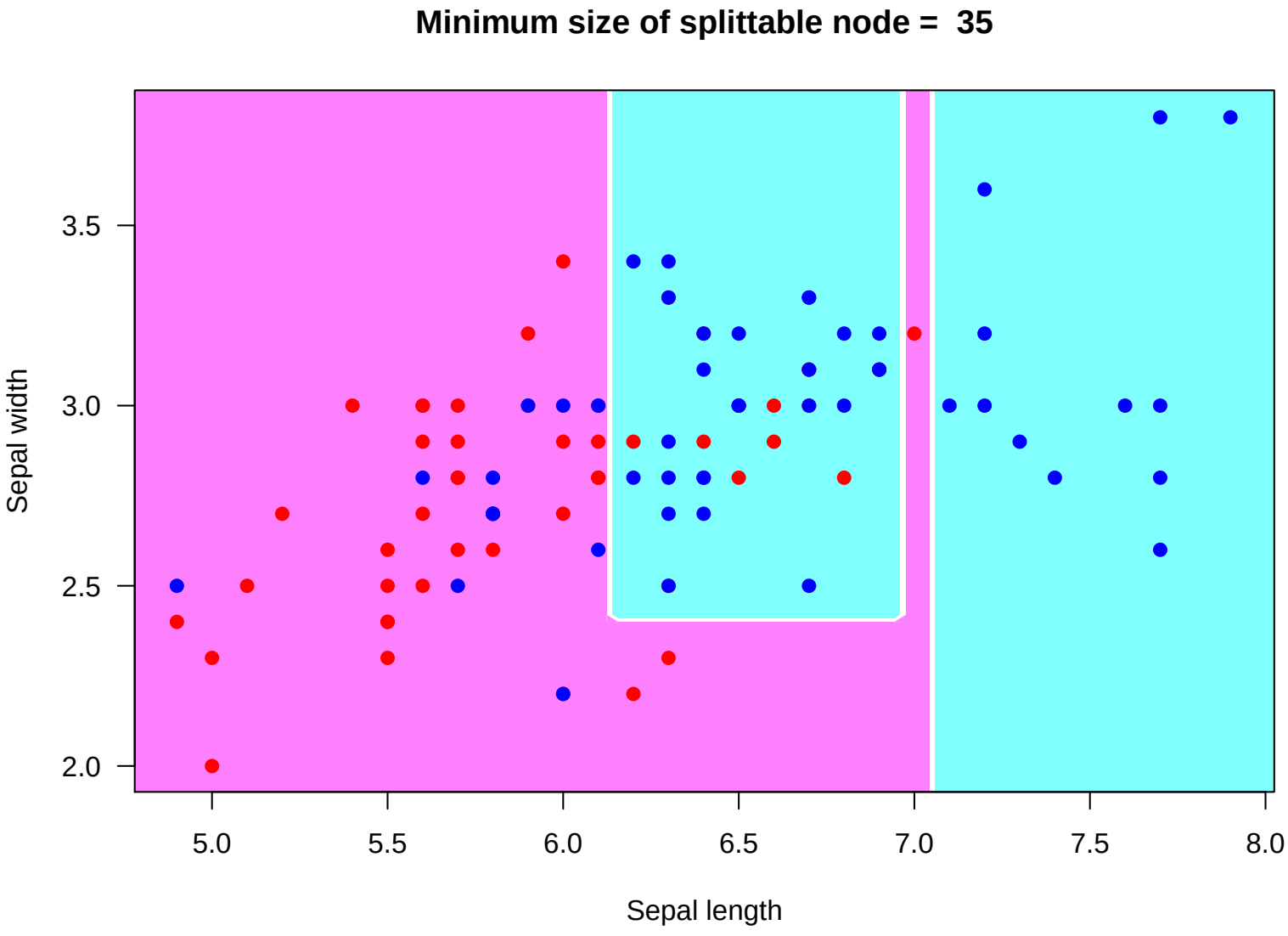
Classification tree: iris data



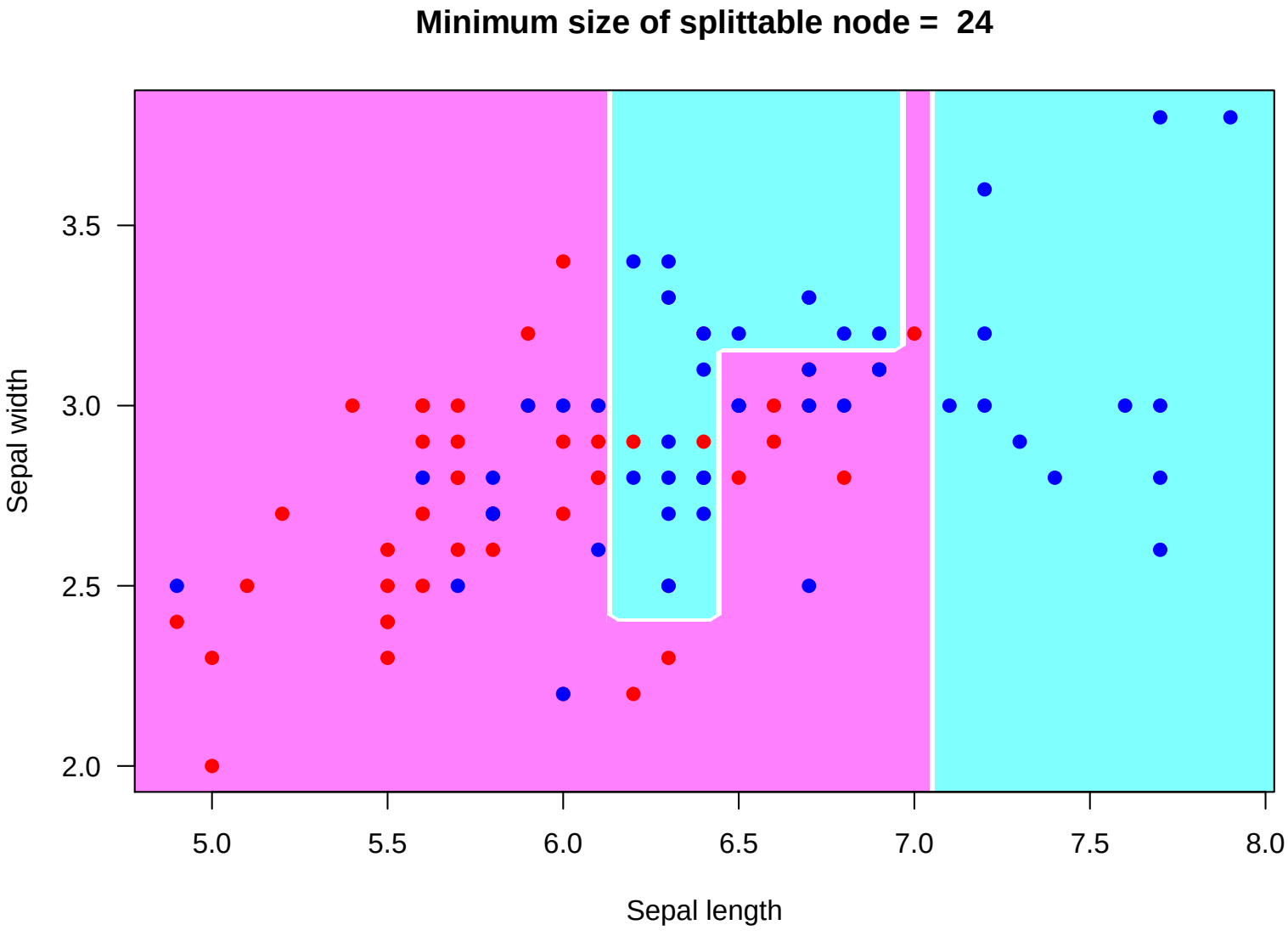
Classification tree: iris data



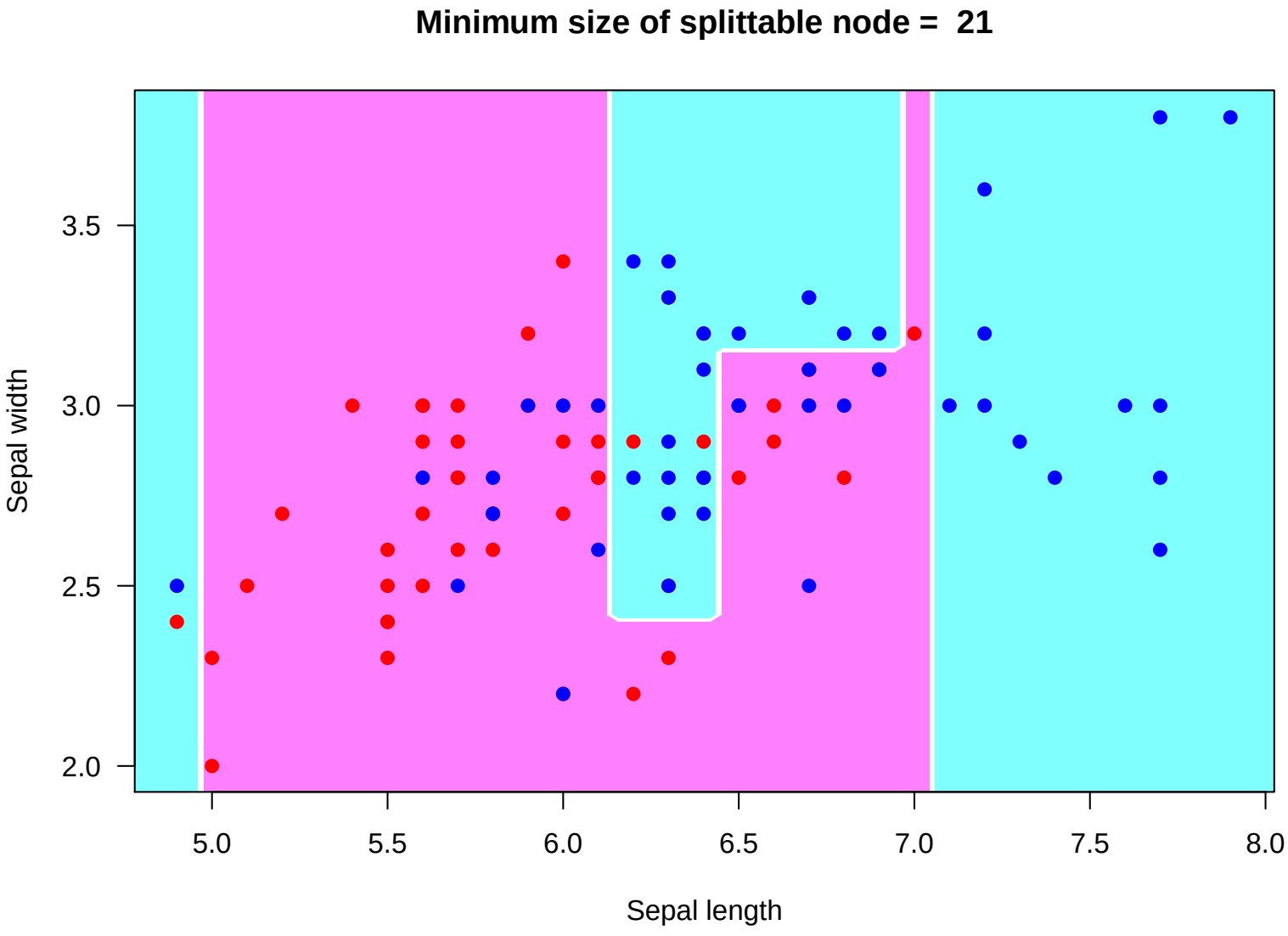
Classification tree: iris data



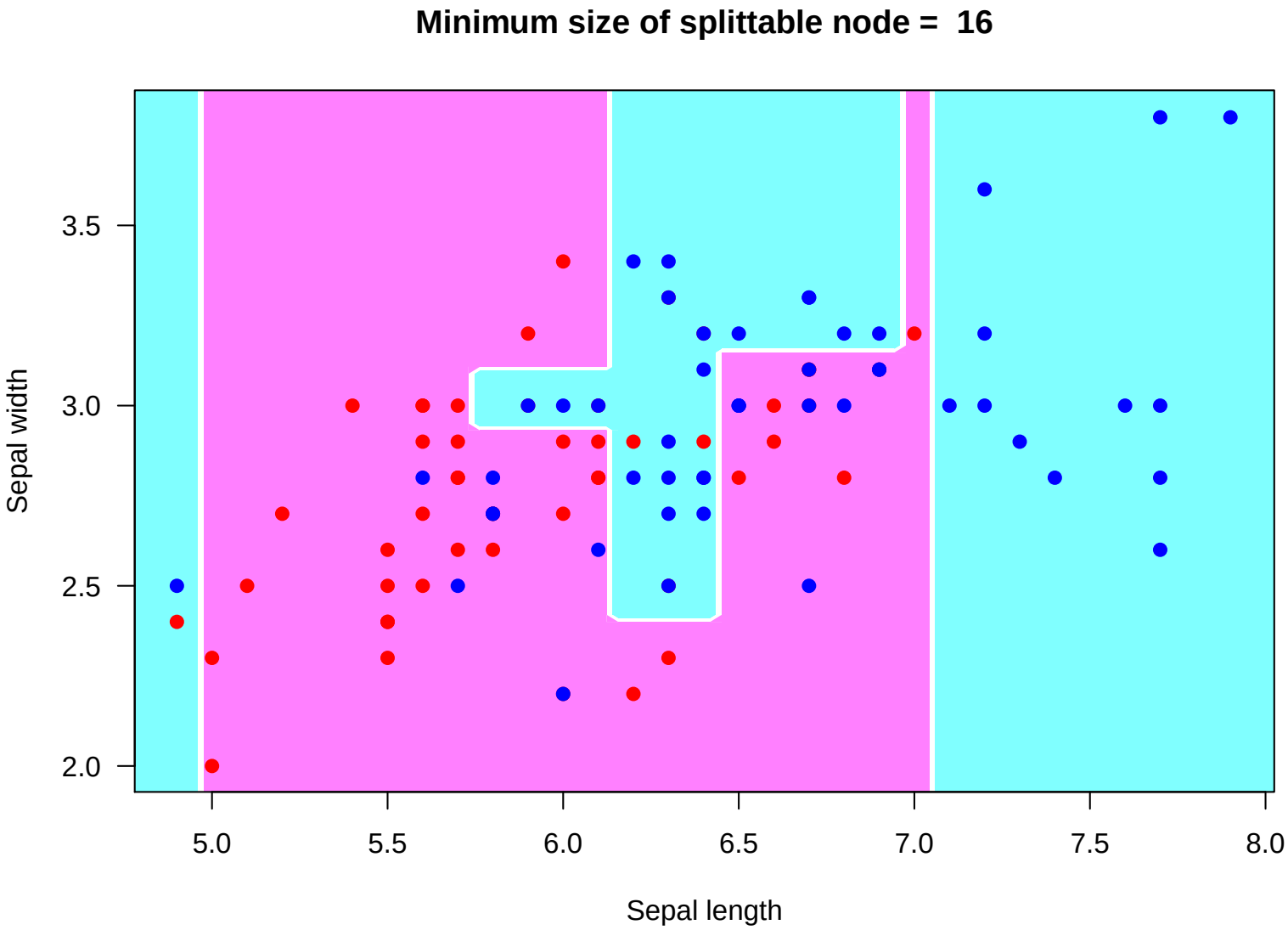
Classification tree: iris data



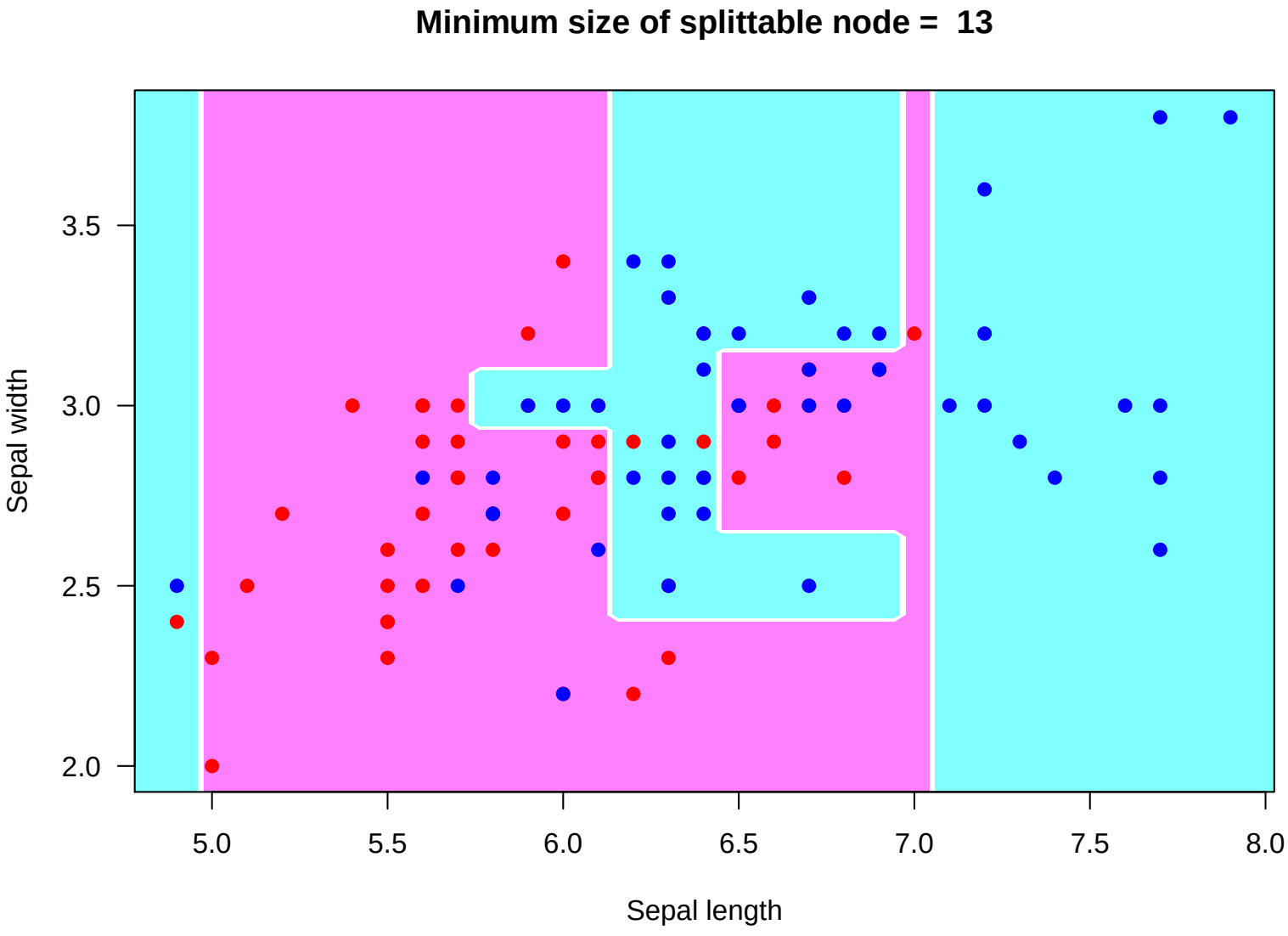
Classification tree: iris data



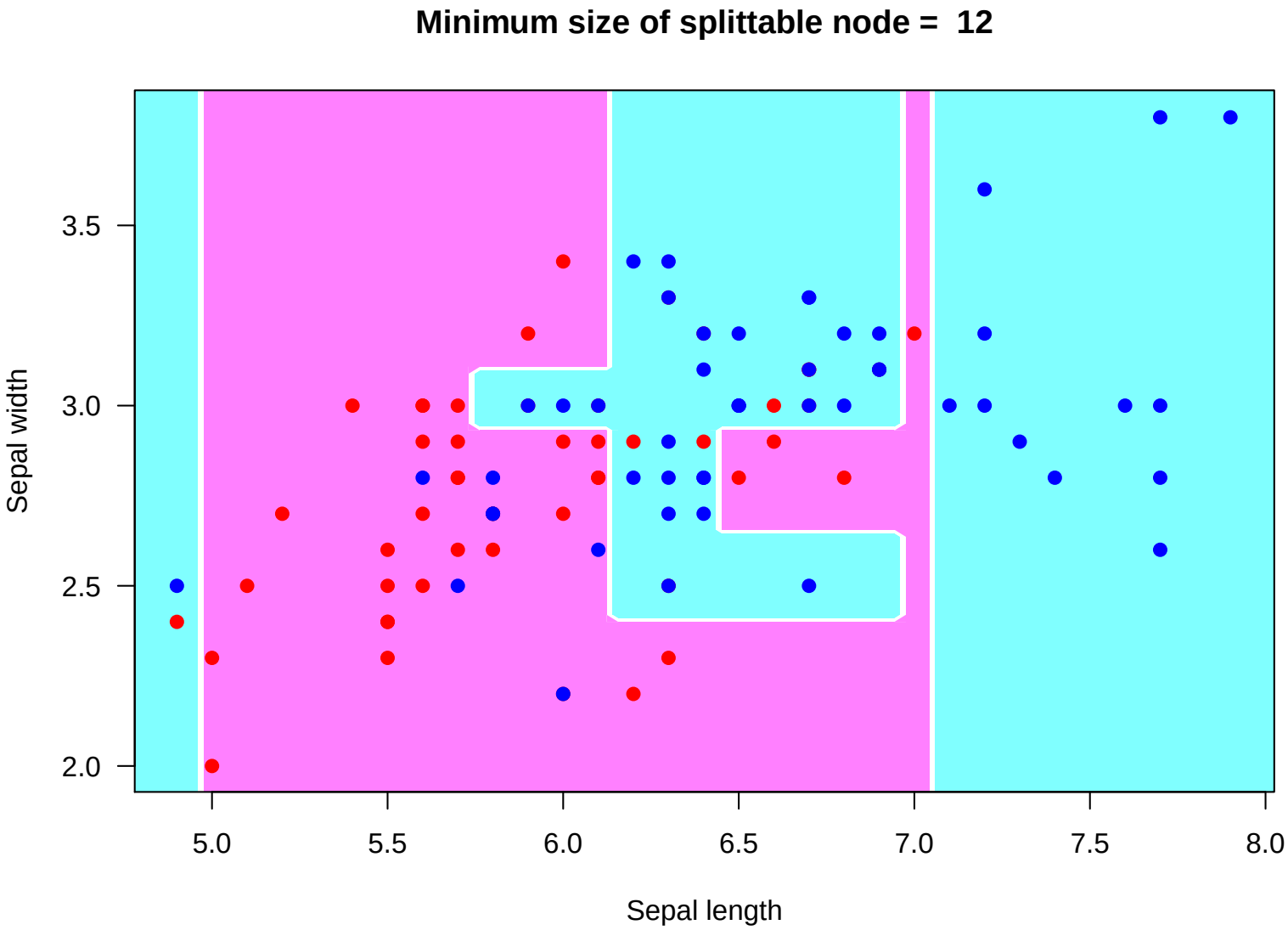
Classification tree: iris data



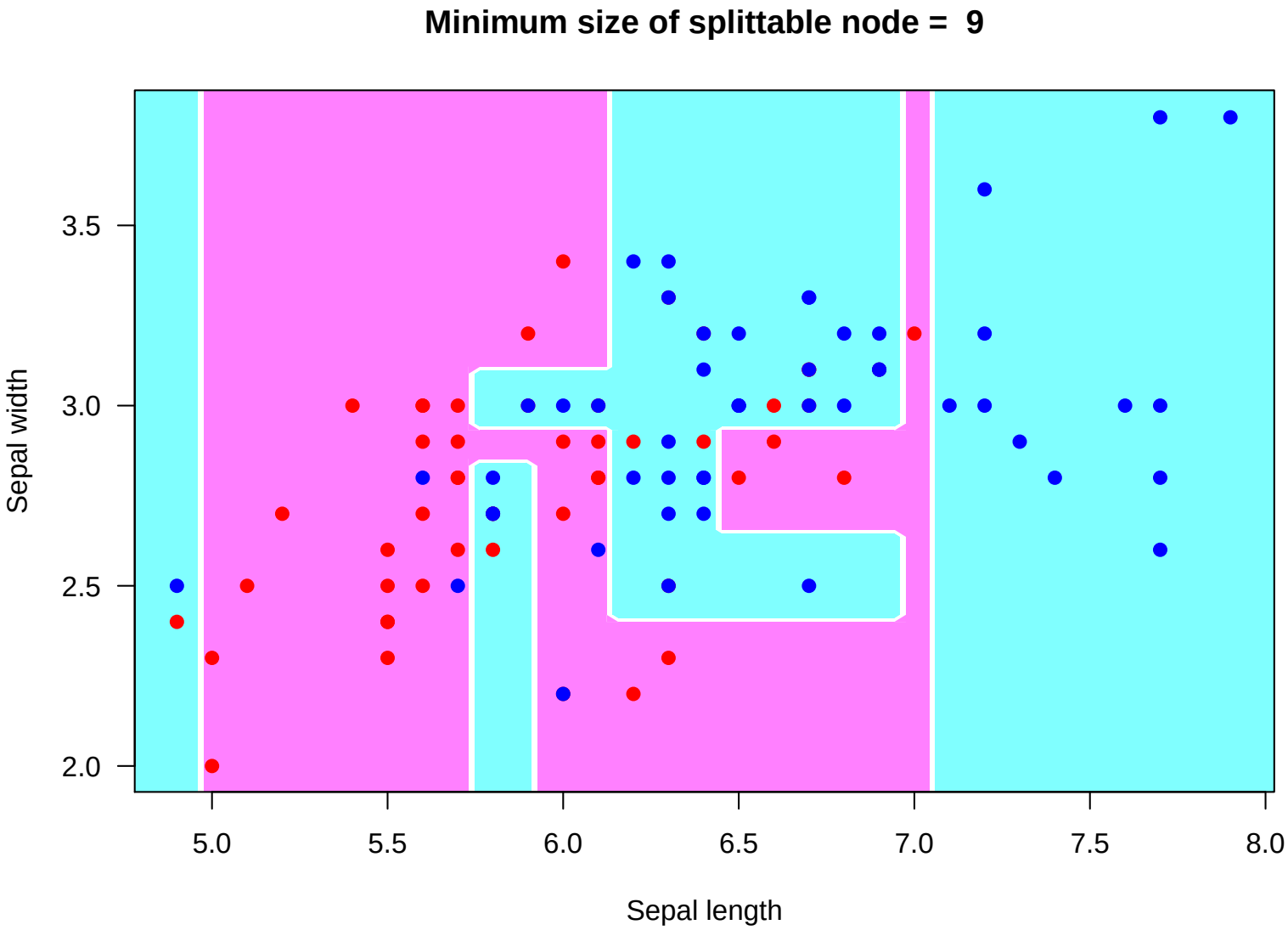
Classification tree: iris data



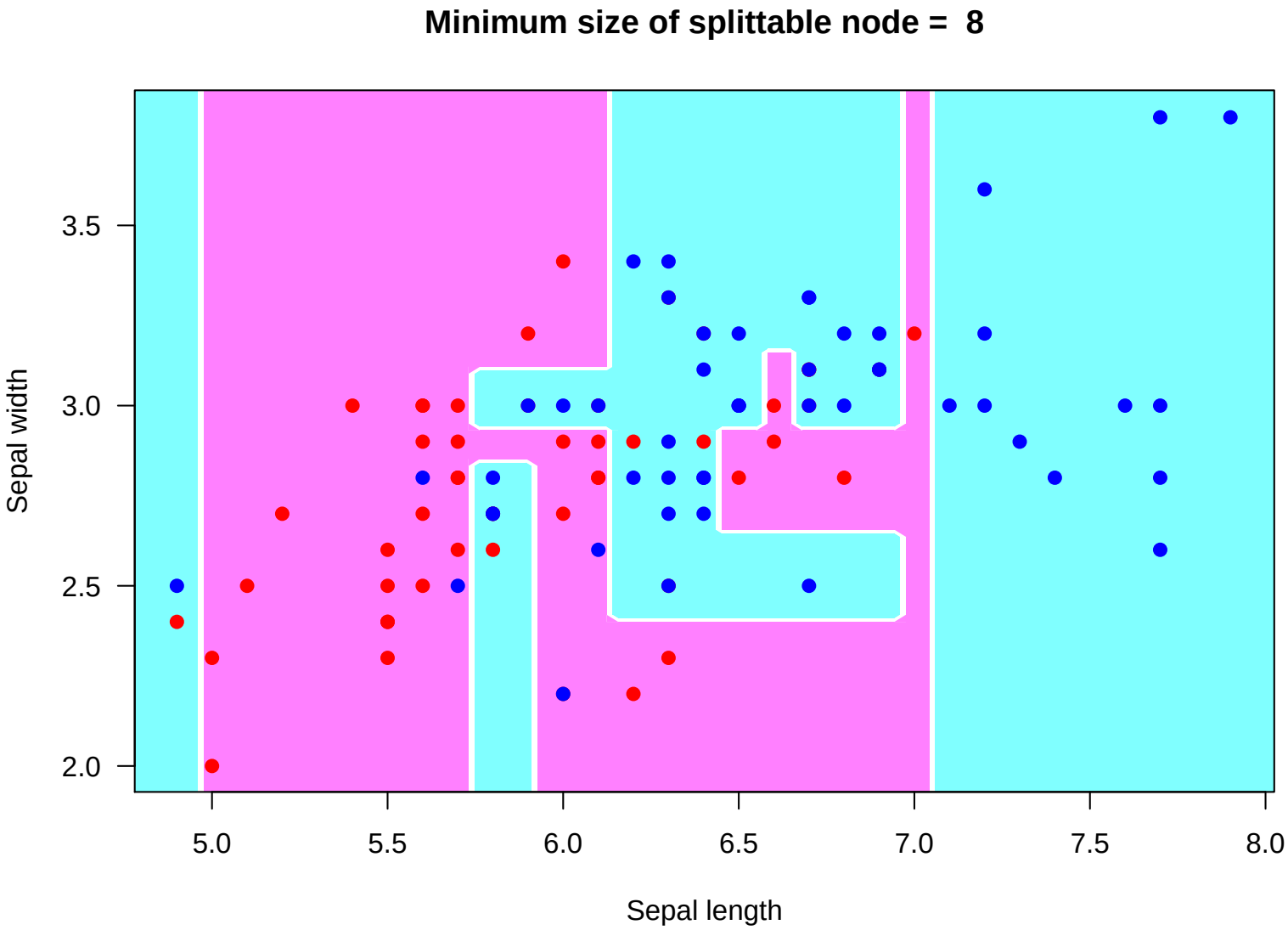
Classification tree: iris data



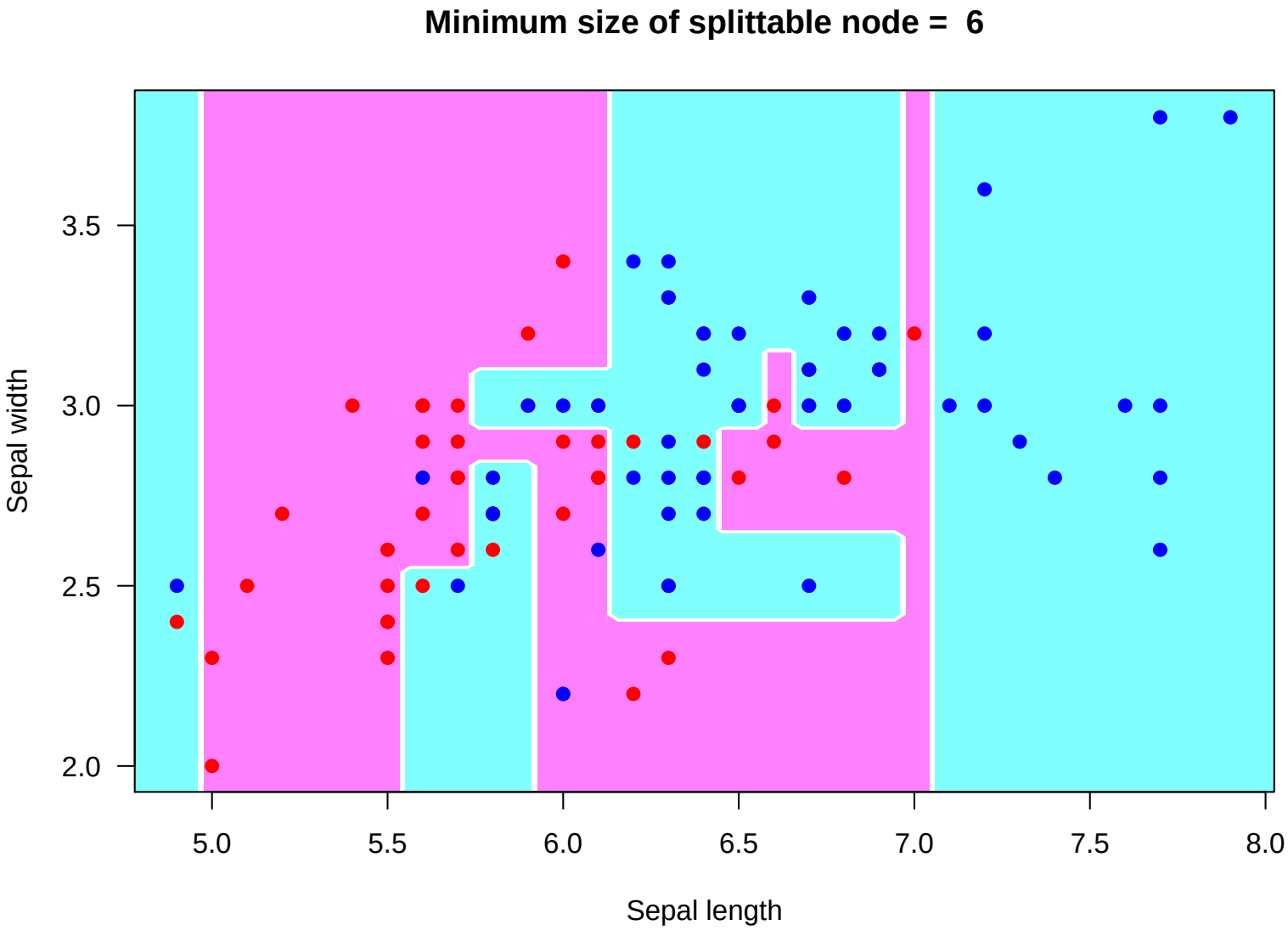
Classification tree: iris data



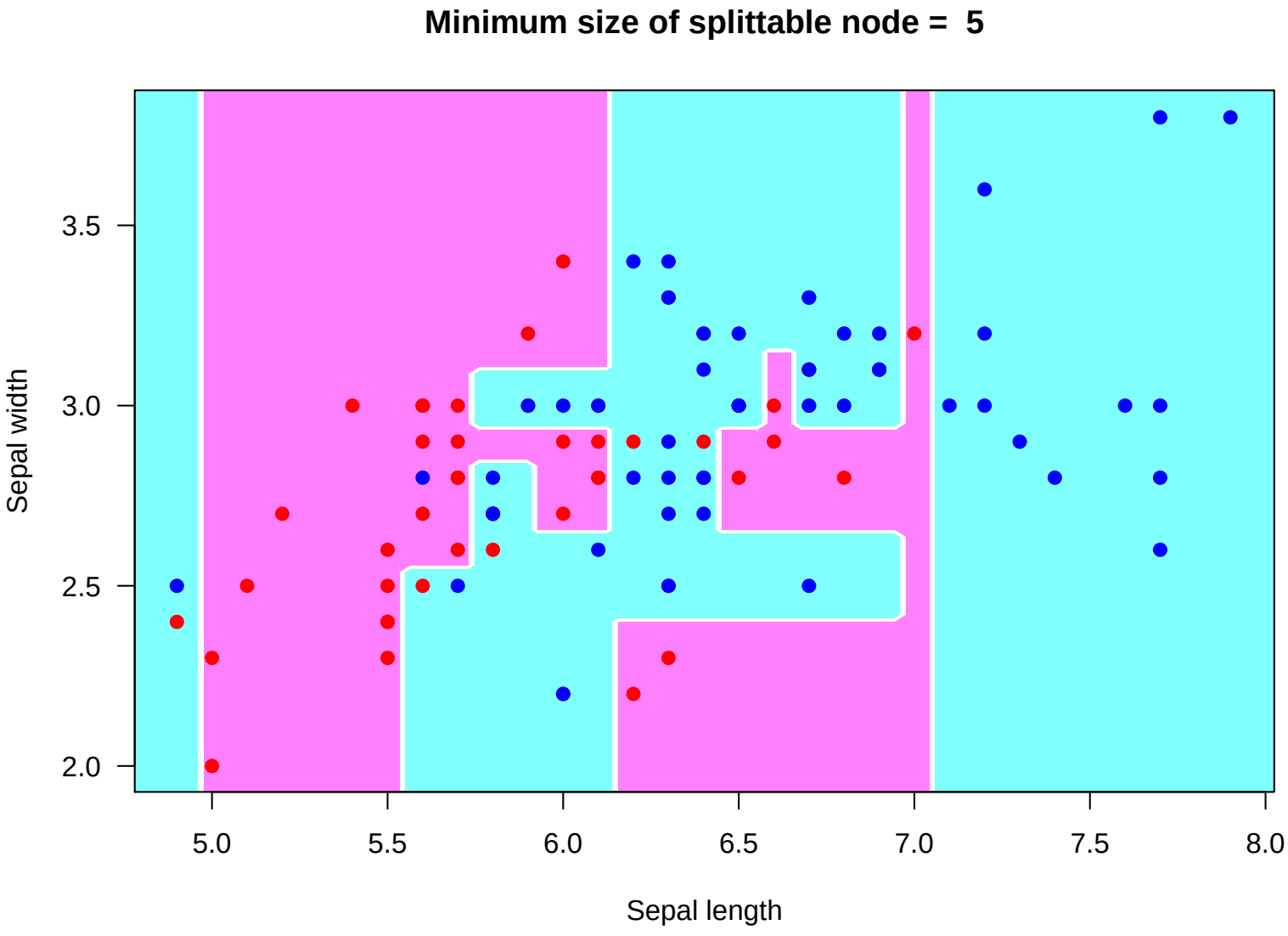
Classification tree: iris data



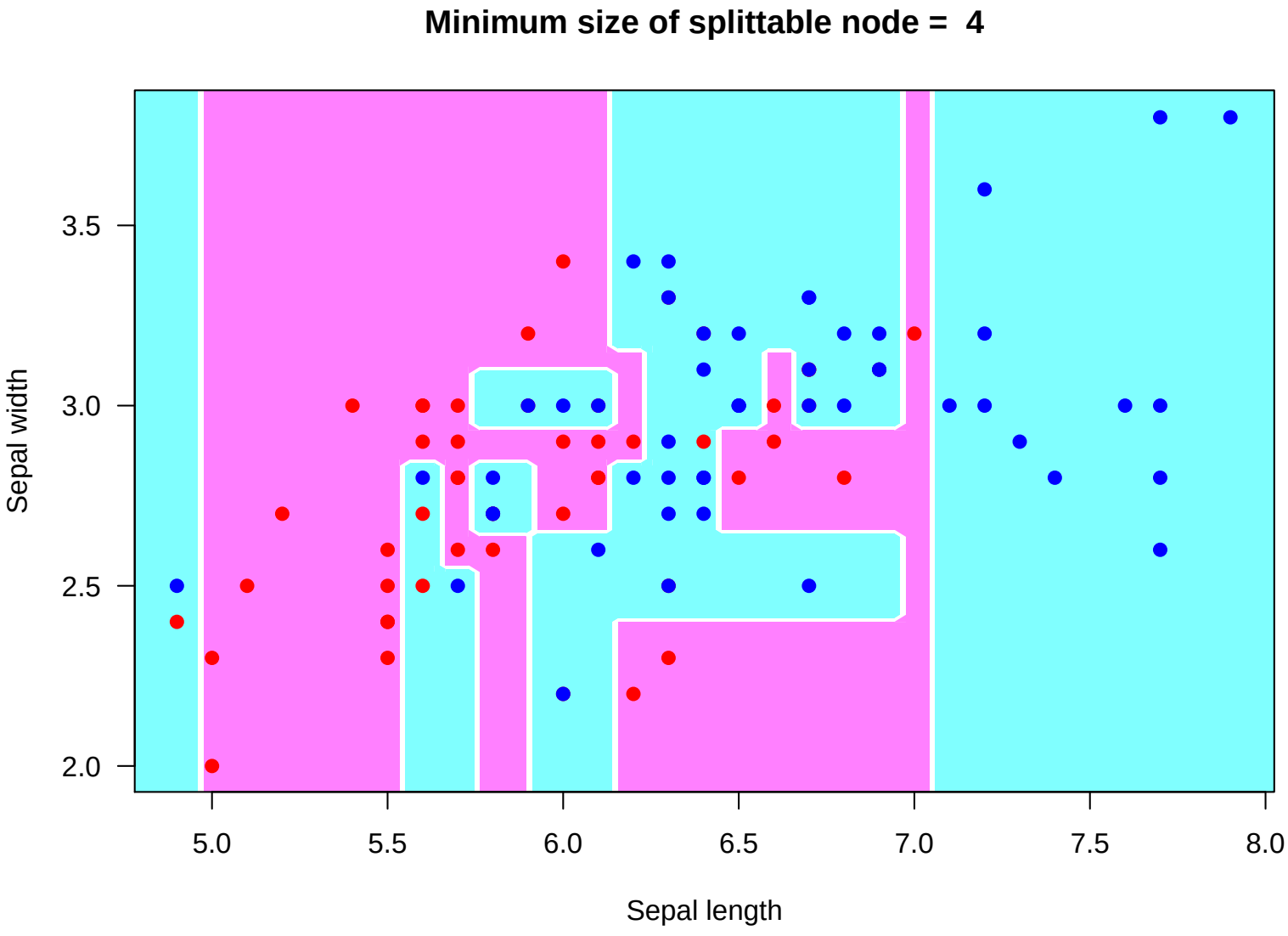
Classification tree: iris data



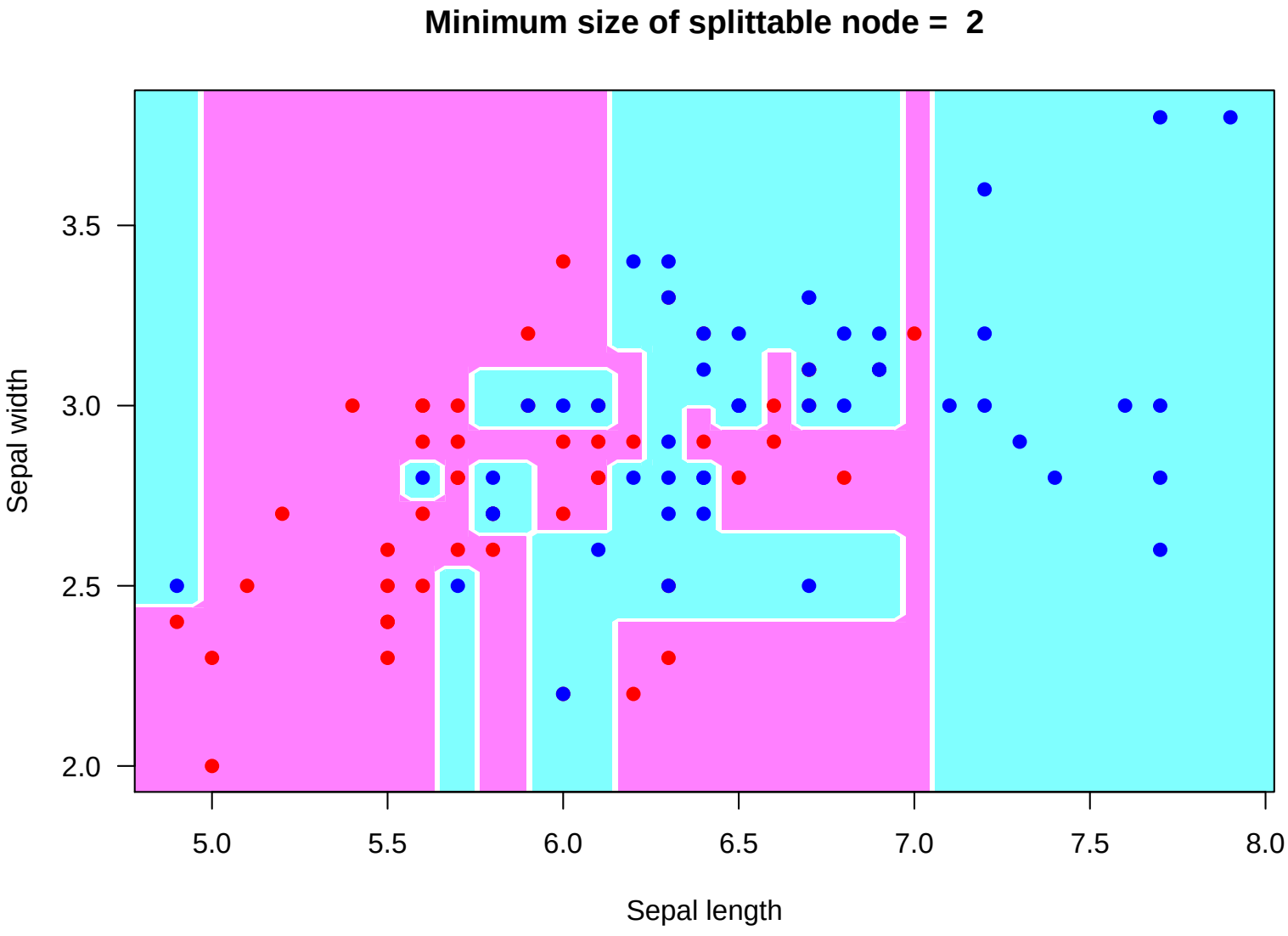
Classification tree: iris data



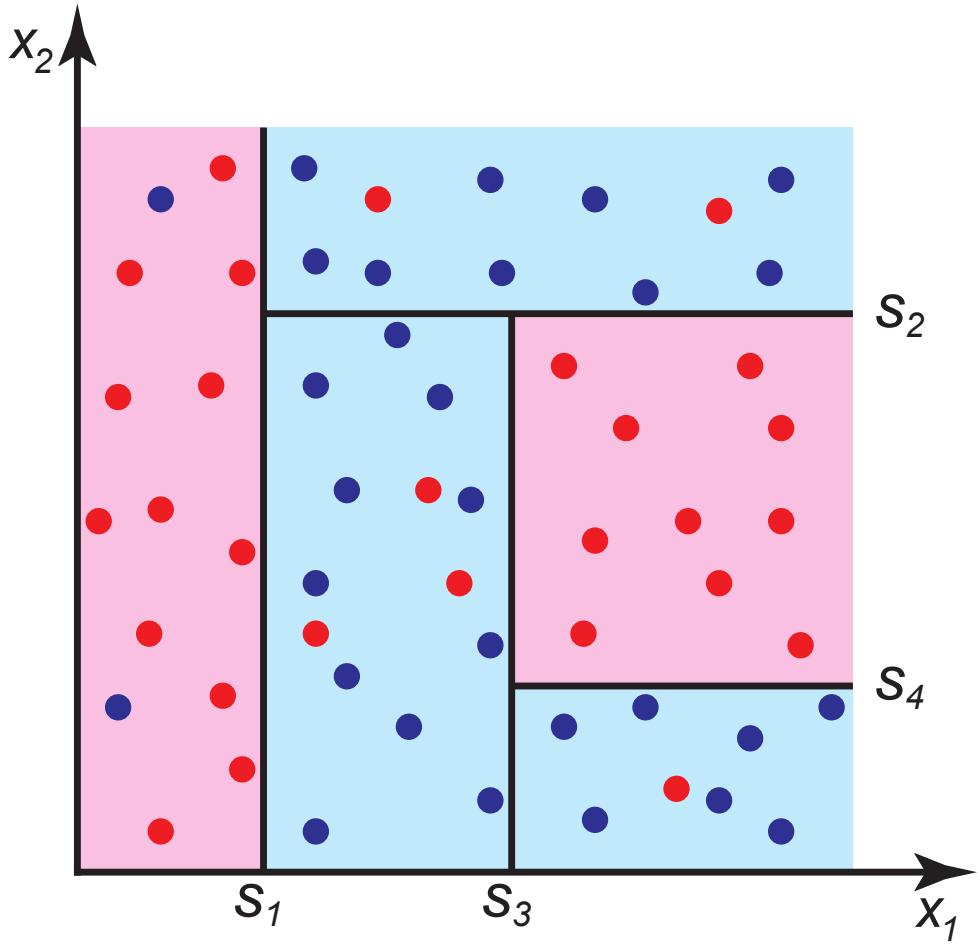
Classification tree: iris data

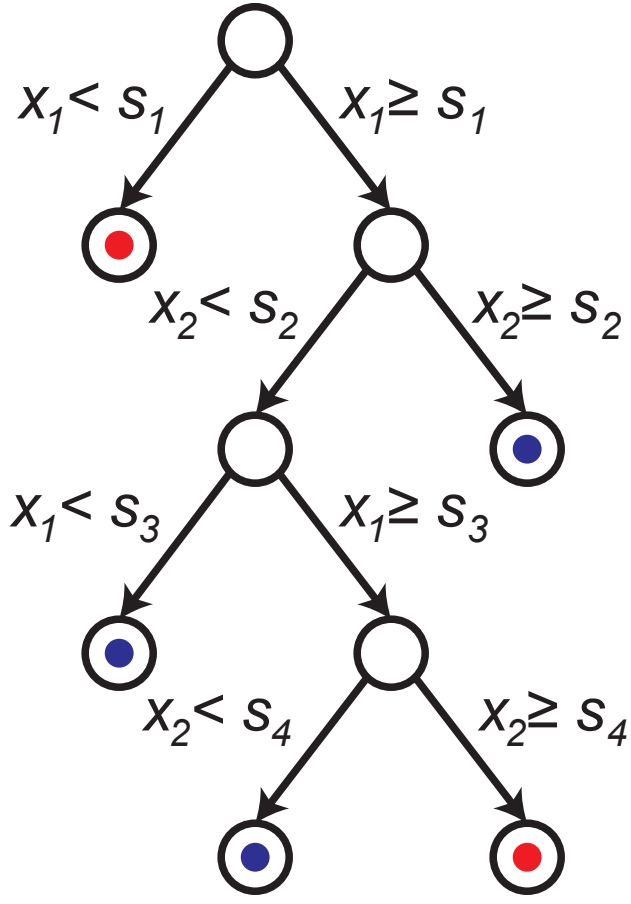


Classification tree: iris data



Reminder: classification tree





Regression tree

- Regression tree is grown in a **(very) similar to the classification tree manner** with slight changes:
- Suppose that **a partition** into M regions $R_1, R_2, ..., R_M$ **is given**.
- The **response is** then modeled as **a constant** c_m **in each region**:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\boldsymbol{x} \in R_m) \,.$$

- Adopt **sum of squares as impurity measure**:

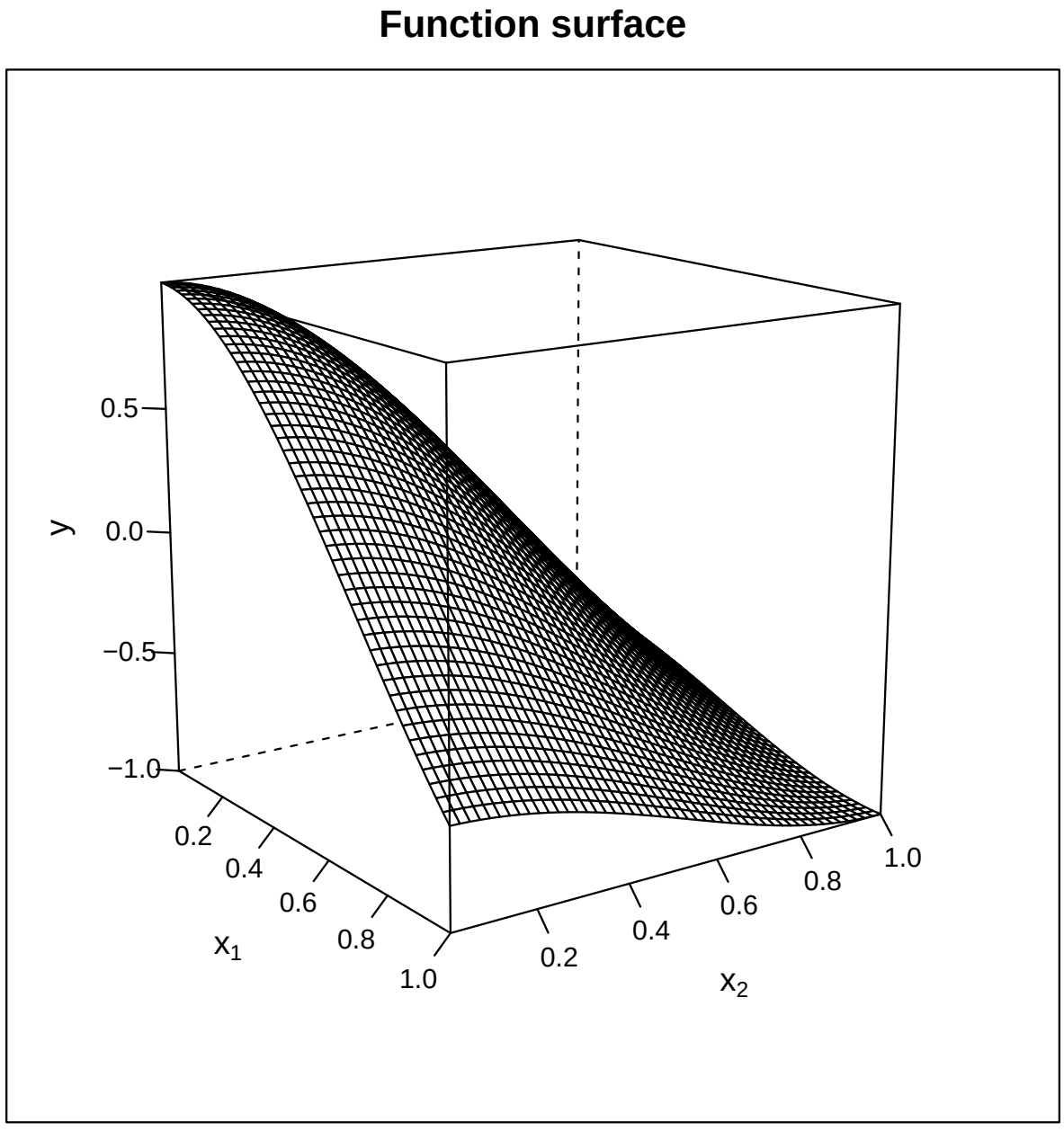
$$Q^{(m)}(T) = \frac{n^{(m_L)}}{n^{(m)}} \sum_{\boldsymbol{x}_i \in R_{m_L}} (y_i - \hat{c}_{m_L})^2 + \frac{n^{(m_R)}}{n^{(m)}} \sum_{\boldsymbol{x}_i \in R_{m_R}} (y_i - \hat{c}_{m_R})^2 \,.$$

- One can check that the **optimal choice of** $\hat{c}_m$ for region R_m **is the average over** R_m:

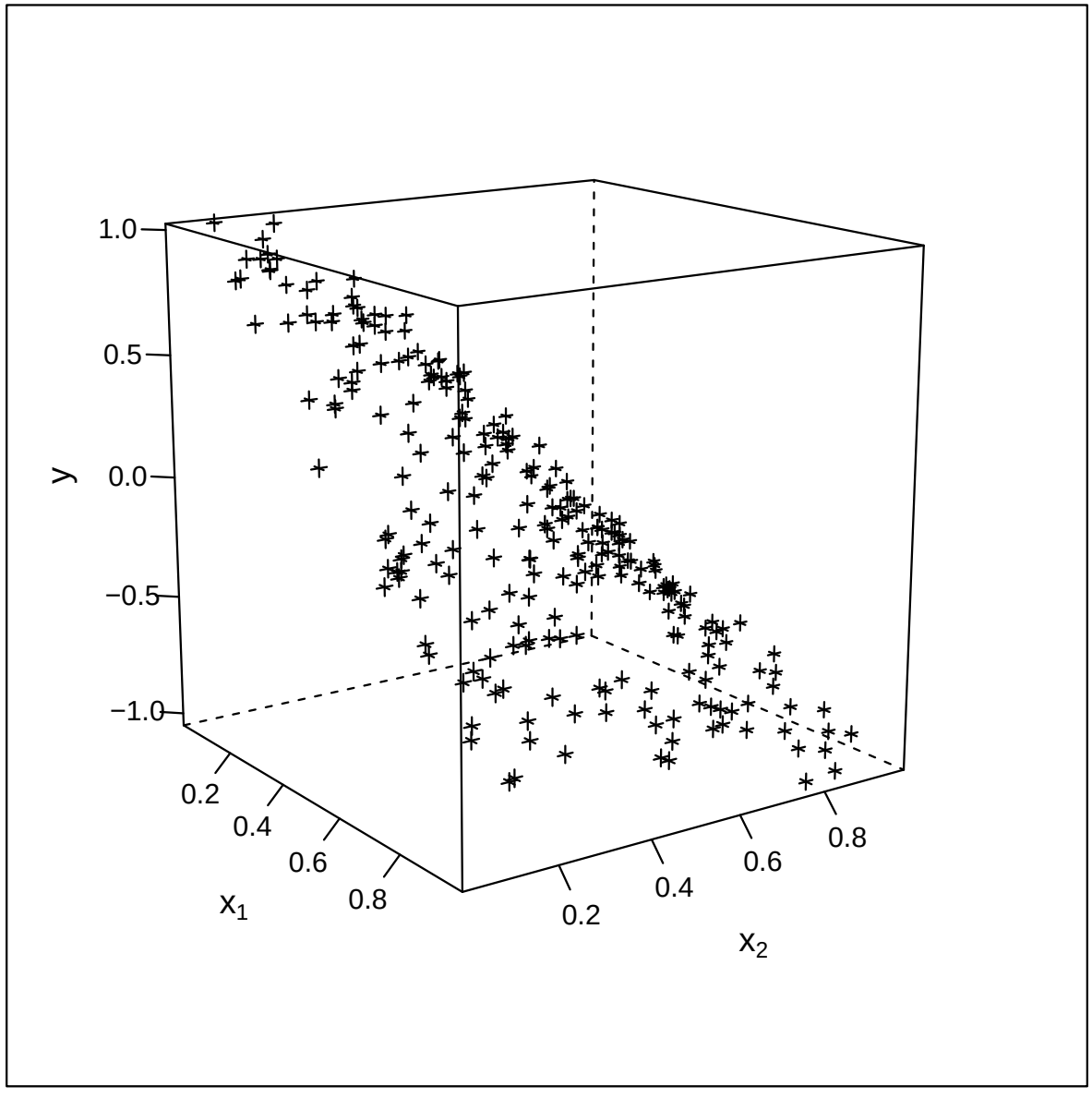
$$\hat{c}_m = \frac{1}{n^{(m)}} \sum_{\boldsymbol{x}_i \in R_m} y_i \,,$$

with $n^{(m)}$ being the number of observations $\boldsymbol{x}_i \in \mathcal{D}_n$ in region R_m.

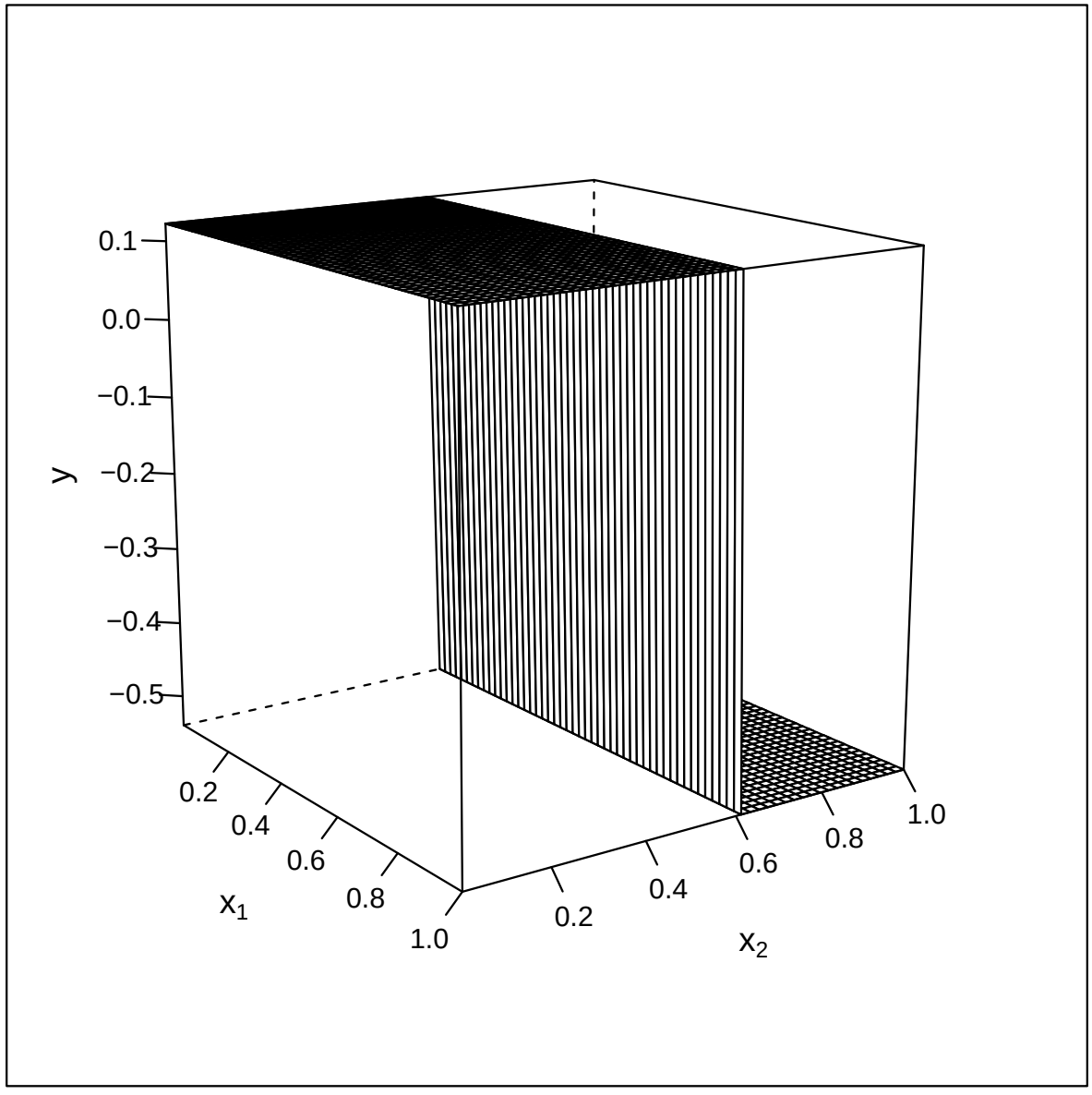
Regression tree (illustration)



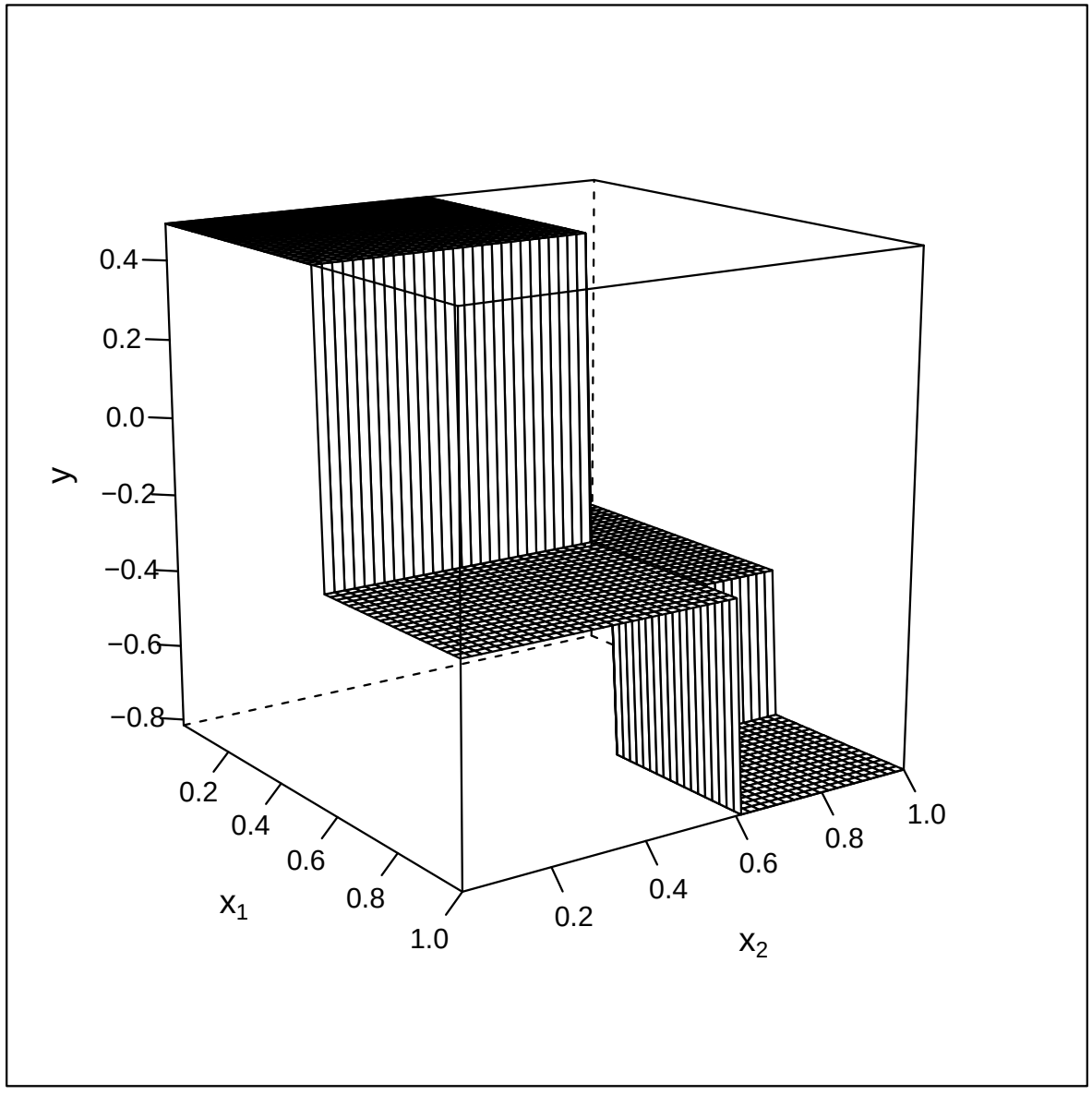
Regression tree (illustration)



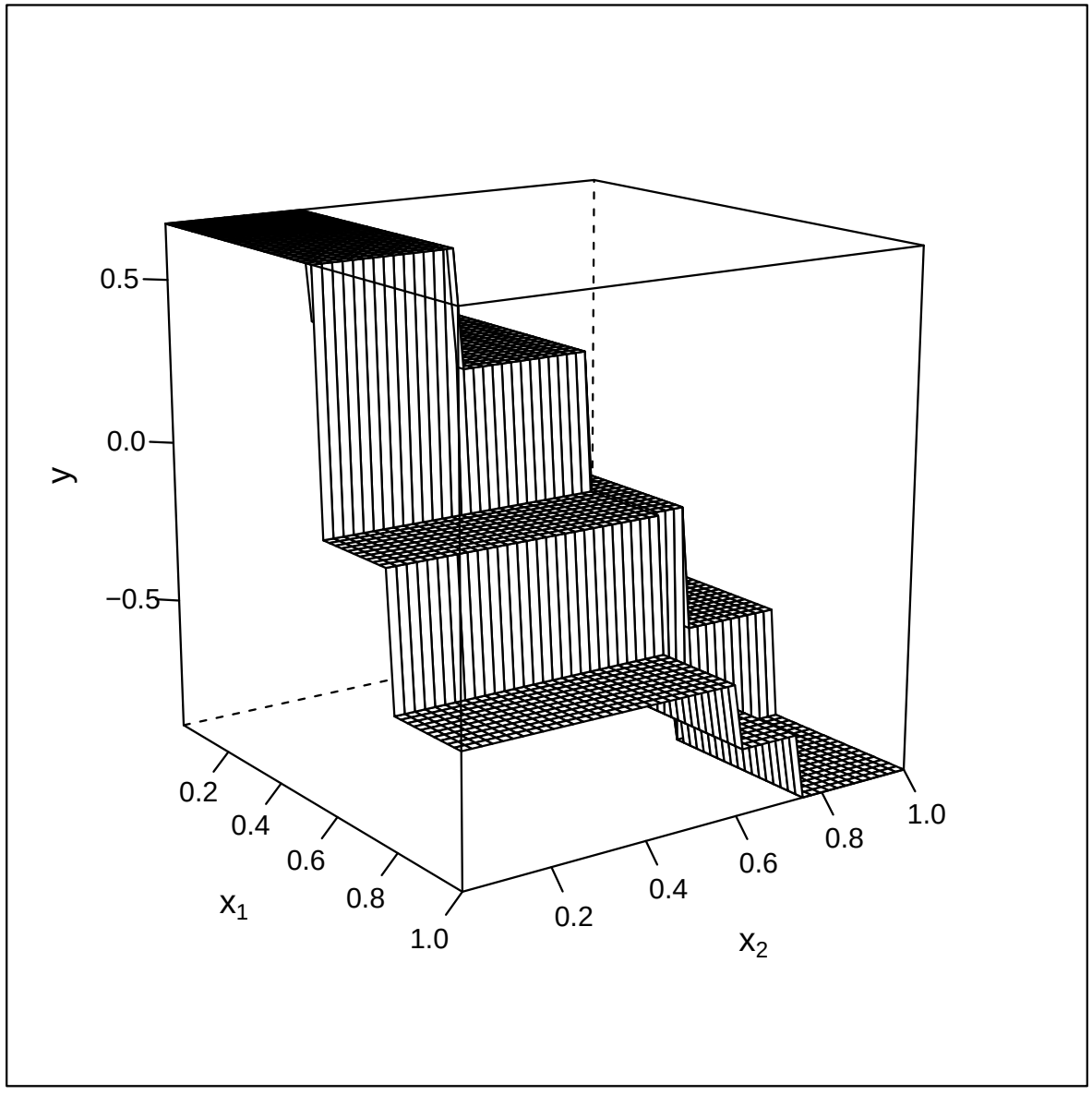
Regression tree (illustration)



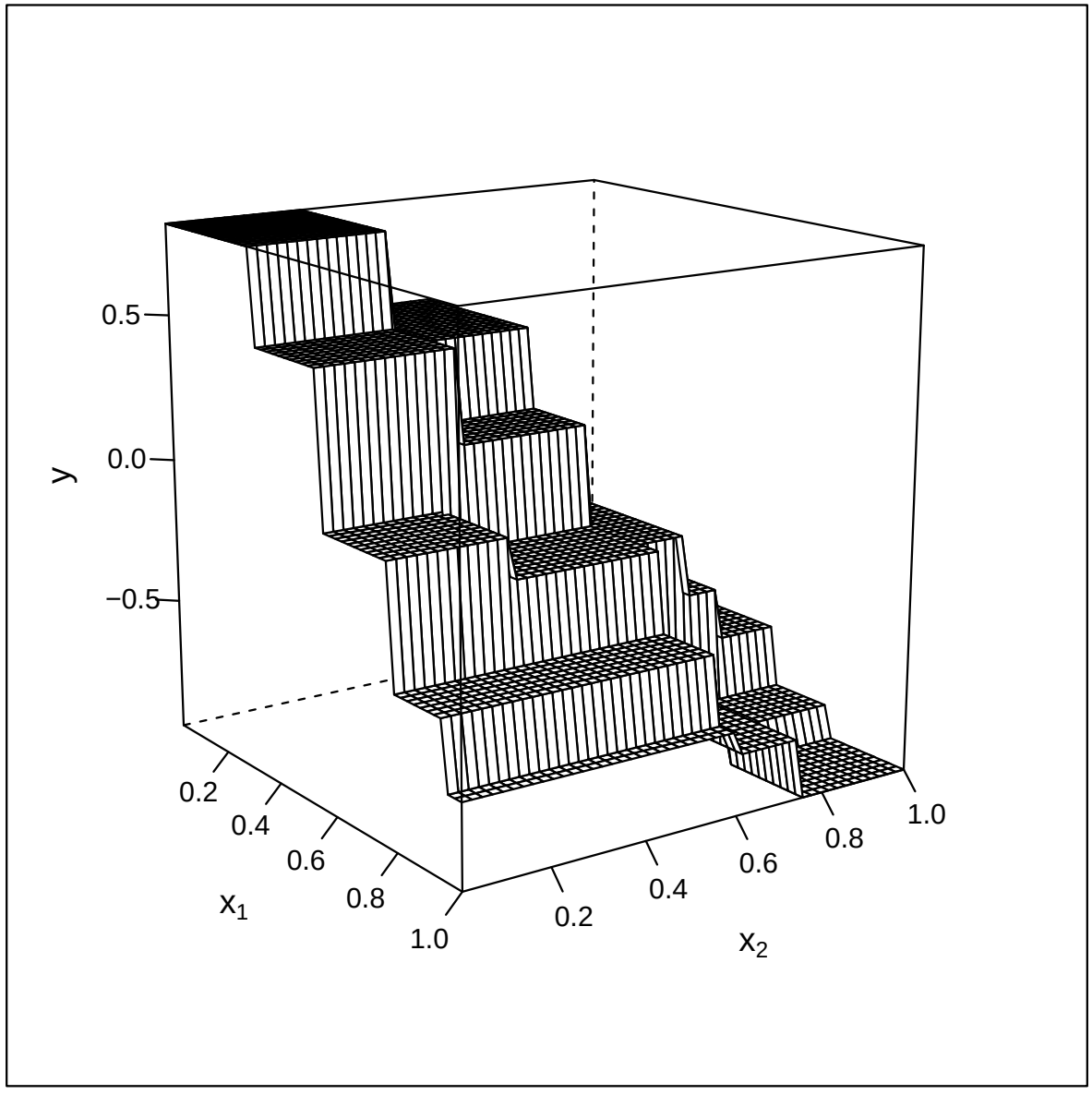
Regression tree (illustration)



Regression tree (illustration)



Regression tree (illustration)



Classification tree: tuning and properties

- A possible (and wide-spread) choice for the stopping criteria S is to restrict **the number of points in region** $R^{(m)}$ to be split to some constant n_{min}:

$$S^{(m)}(T) = I(n^{(m)} < n_{min}) .$$

- Often the classification tree is constructed in two stages:
  1. n_{min} **is set very small** and the tree is grown.
  2. **Pruning** of tree is conducted based on some parameter α, which consists in choosing a **subtree** that **minimizes** a **cost-complexity** criterion, *e.g.*

$$Q^{(m)}(T) = \sum_{m=1}^{\#T} n^{(m)} Q^{(m)}(T) + \alpha \#T ,$$

  where $\#T$ stands for the number of nodes in the tree.
  Pruning parameter α is chosen by the means of **cross-validation**.

- The key **advantage** of the classification tree is its **interpretability**, as the feature space partition is fully described by a single tree.

Contents

The key idea

- The "Wisdom of Crowds" (Surowiecki, 2004): The **collective knowledge** of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting.

- **Bagging** implements this way of thinking standing for a range of methods following the general idea introduced by Léo Breiman (1996).

- **Bagging** is a shortcut for **B**ootstrap **Agg**regat**ing**.

- The main idea is to construct a single estimator that consists of **a number of basic classifiers** (weak learners) (taught on a bootstrapped samples) aggregated by averaging (voting).

Motivation (regression)

- Consider the standard regression setting

$$Y = g(X) + \epsilon.$$

- The single bagged estimator

$$\hat{g}_B(\boldsymbol{x}) = \frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x})$$

  is the estimator of g obtained by aggregating estimators $g_1, ..., g_B$.

- $g_k(\boldsymbol{x}) = g_k(\boldsymbol{x}; (X_1, Y_1), ..., (X_n, Y_n))$ as well as $\hat{g}_B(\boldsymbol{x}) = \hat{g}_B(\boldsymbol{x}; (X_1, Y_1), ..., (X_n, Y_n))$ are random variables.

- One can measure the improvement of aggregating by comparing performance of $\hat{g}_B(\boldsymbol{x})$ and those of $g_k(x)$, $k = 1, ..., B$ in terms of **bias** and **variance**.

Bias and variance (regression)

- **Assumption:** Random variables $g_1, ..., g_B$ are i.i.d.
- **Bias:**

$$\mathbb{E}[\hat{g}_B(\boldsymbol{x})] = \mathbb{E}[g_k(\boldsymbol{x})]\,.$$

Conclusion

Aggregation does not modify the bias.

- **Variance:**

$$\mathbb{V}[\hat{g}_B(\boldsymbol{x})] = \frac{1}{B}\mathbb{V}[g_k(\boldsymbol{x})]\,.$$

Conclusion

Aggregation reduces the variance.

Motivation (classification)

- Let $g_1, ..., g_B$ be an ensemble of **basic classifiers**.
- **Assumption:** Each basic classifier has an independent error $\epsilon < 0.5$ for some observation $\boldsymbol{x}$ (let $y = 1$ being the correct decision):

$$\mathbb{P}\big(g_k(\boldsymbol{x}) \neq 1\big) = \epsilon < 0.5 \quad \text{for} \quad k = 1, ..., B\,,$$
$$g_1(\boldsymbol{x}), ..., g_B(\boldsymbol{x}) \quad \text{are } \textbf{i.i.d}\,.$$

- Further, let the aggregated classifier be

$$g^{agg}(\boldsymbol{x}) = \mathbb{1}\Big(\frac{1}{B}\sum_k g_k(\boldsymbol{x}) > 0.5\Big)\,.$$

- Then $\sum_k g_k(\boldsymbol{x})$ will have binomial distribution

$$\sum_k g_k(\boldsymbol{x}) \sim Bin(B, 1-\epsilon)$$

and classification error of $\boldsymbol{x}$ will decrease with increasing B:

$$\mathbb{P}\big(g^{agg}(\boldsymbol{x}) \neq 1\big) = \sum_{k=1}^{B/2} \binom{B}{k} (1-\epsilon)^k \epsilon^{B-k} \xrightarrow[B\to\infty]{} 0\,.$$

Motivation (classification)

Theorem (Chernoff-Hoeffding, Bernoulli scheme)

If $X_1, ..., X_n$ are i.i.d. random variables taking values in $\{0, 1\}$, then for any $\eta > 0$ it holds

$$\mathbb{P}\Big(\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i > \eta\Big) < \exp(-2\eta^2 n)\,.$$

One can express classification error as

$$\mathbb{P}\big(g^{agg}(\boldsymbol{x}) \neq 1\big) = \mathbb{P}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x}) < 0.5\Big)\,.$$

By a sequence of simple transformations we obtain

$$\underbrace{(1-\epsilon)}_{\mathbb{E}[g_1(\boldsymbol{x})]} - \frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x}) > 0.5 - \epsilon\,.$$

Applying the Chernoff-Hoeffding inequality gives

$$\mathbb{P}\big(g^{agg}(\boldsymbol{x}) \neq 1\big) < \exp\Big(-\frac{1}{2}B(1-2\epsilon)^2\Big)\,.$$

Motivation

- The derivations from above exploit the **assumption** that random variables $g_1(\boldsymbol{x}), ..., g_B(\boldsymbol{x})$ are **independent and identically distributed**.
- As the classifiers $g_1, ..., g_B$ are constructed using the same training sample $\mathcal{D}_n$, the assumption of **independence** is not really credible.
- The idea is thus to **introduce a source of randomness** into the sample used to train each single classifier g_k , $k = 1, ..., B$.
- Resort to the idea of the **bootstrap**.

Bagging (algorithm)

Training

Input:

- Training sample $\big((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Basic classifier $g(\cdot)$.
- Number of estimators to aggregate B.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}_n^k$ from $\mathcal{D}_n$ using bootstrap.
2. Learn g_k on $\mathcal{D}_n^k$.

Output: The aggregated classifier $g^{agg}(\cdot) = \mathbb{1}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\cdot) > 0.5\Big)$.

Classification

- Classify the new observation $\boldsymbol{x}$ as

$$g^{agg}(\boldsymbol{x}) = \mathbb{1}\left(\frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x}) > 0.5\right).$$

Drawing bootstrap samples

- **Bootstrap** is a technique based on random sampling, which allows for estimating the sampling distribution of almost any statistics.
- Bootstrap drawings are represented by B random variables θ_k, $k = 1, ..., B$.
- Bootstrap drawings are usually conducted with the **same distribution** independently, *i.e.* $\theta_1, ..., \theta_B$ are **i.i.d.** according to the same distribution θ.
- In general for the sample consisting of n observations, two techniques are used to draw bootstrap samples:
  - draw (choose randomly) n **observations with replacements**,
  - draw (choose randomly) $l < n$ **observations without replacement**.
- Thus aggregated classifiers contain two sources of randomness:
  - due to the $\mathcal{D}_n$ being a random draw from distribution of (X, Y),
  - due to the bootstrap drawing.

Choice of the parameters

- There are two choices to be done:
  - **base classifier** $g(\cdot)$,
  - **number of** bootstrap **iterations** B.

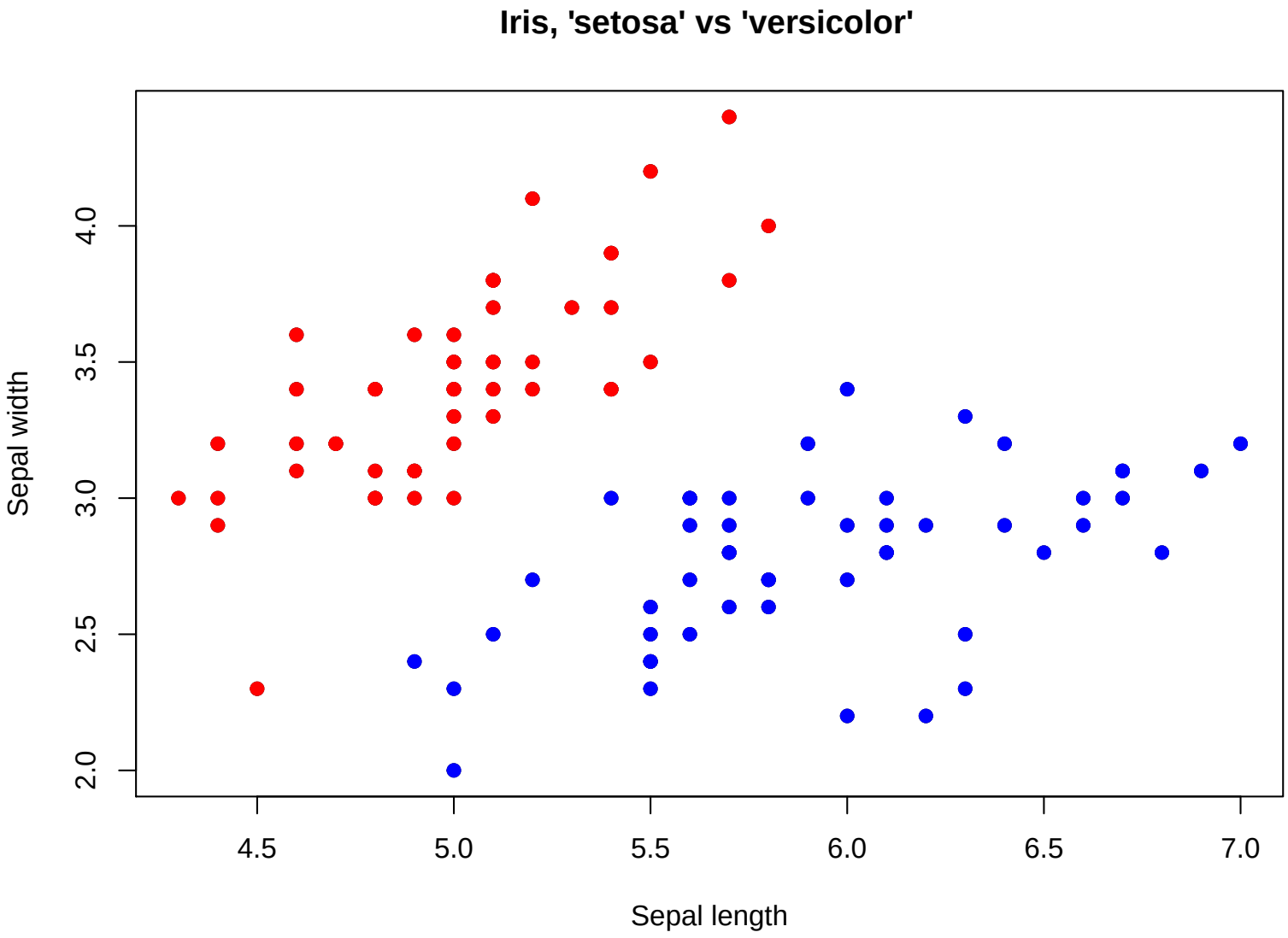
- W.r.t. the law of large numbers we have

$$\begin{aligned}
\lim_{B\to\infty} g^{agg}(\boldsymbol{x}) &= \lim_{B\to\infty} \mathbb{1}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x}) > 0.5\Big) \\
&= \mathbb{1}\Big(\lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} g(\boldsymbol{x}, \theta_k, \mathcal{D}_n)\Big) \\
&= \mathbb{1}\big(\mathbb{E}_\theta[g(\boldsymbol{x}, \theta, \mathcal{D}_n)]\big)\,.
\end{aligned}$$

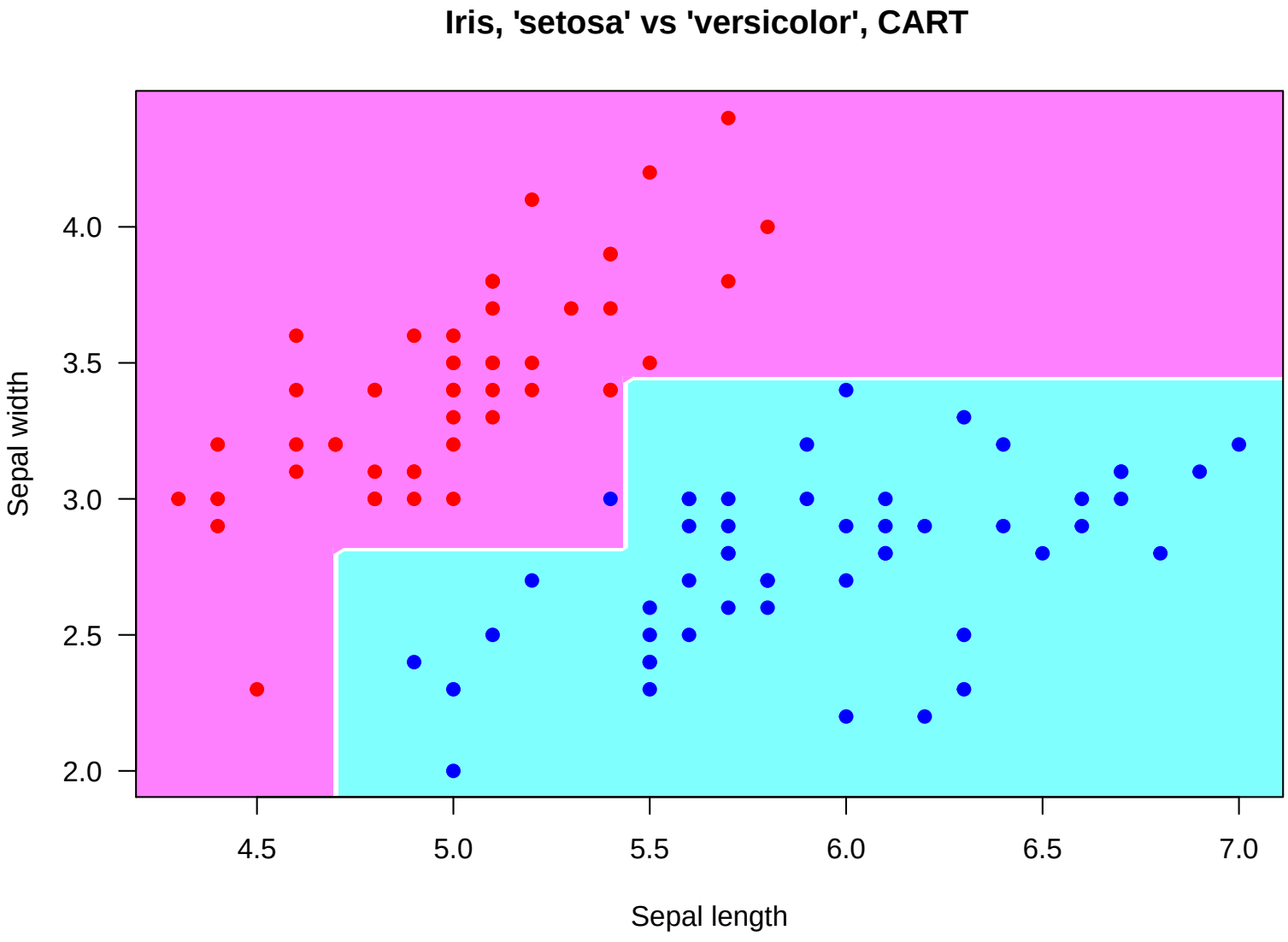
- So g^{agg} stabilizes with increasing B converging to the **bagging estimator** $\mathbb{1}\big(\mathbb{E}_\theta[g(\boldsymbol{x}, \theta, \mathcal{D}_n)]\big)$.

- Thus, B should be chosen as large as possible, regarding computational capabilities.

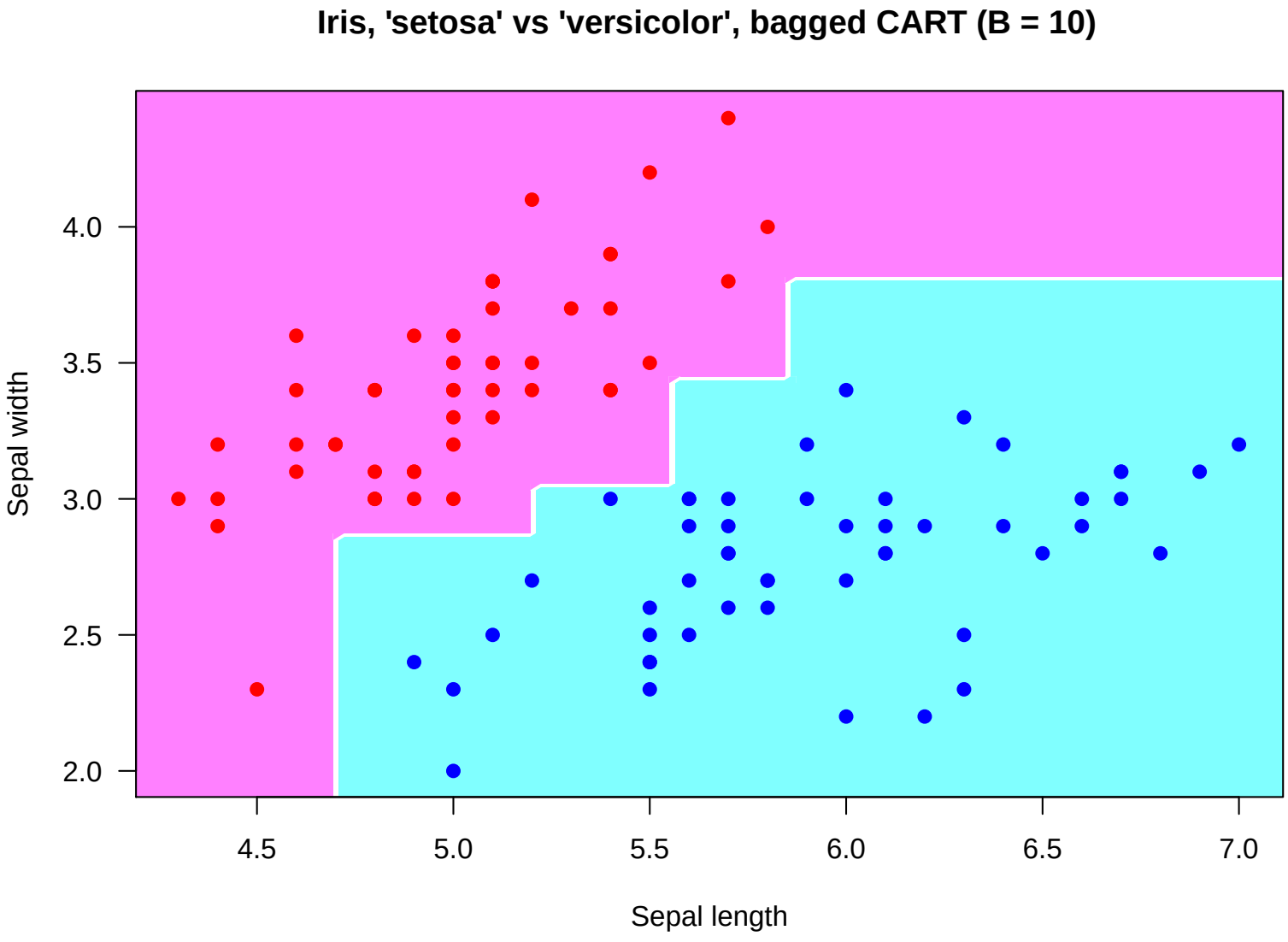
Bagging classification tree: iris data



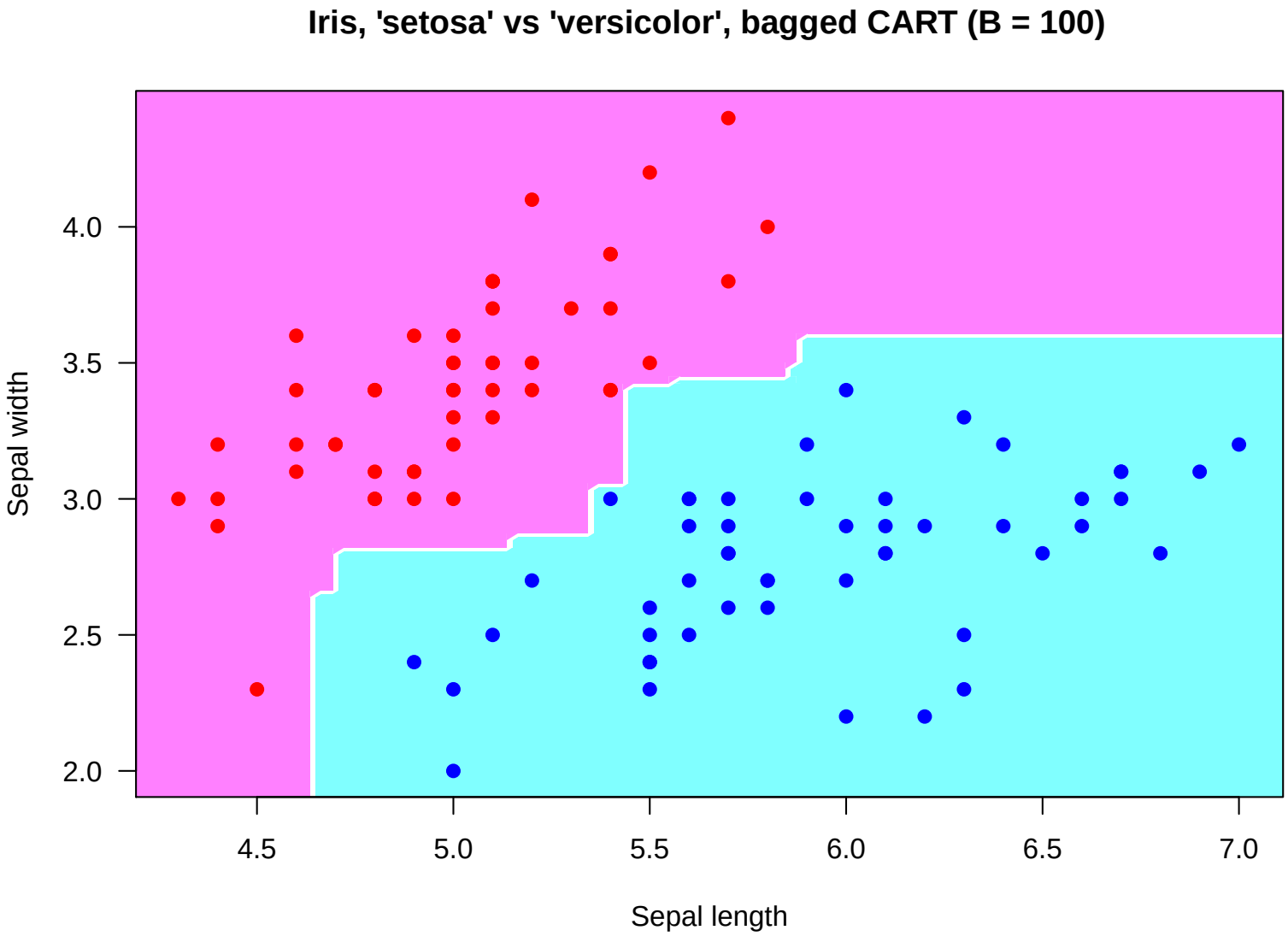
Bagging classification tree: iris data



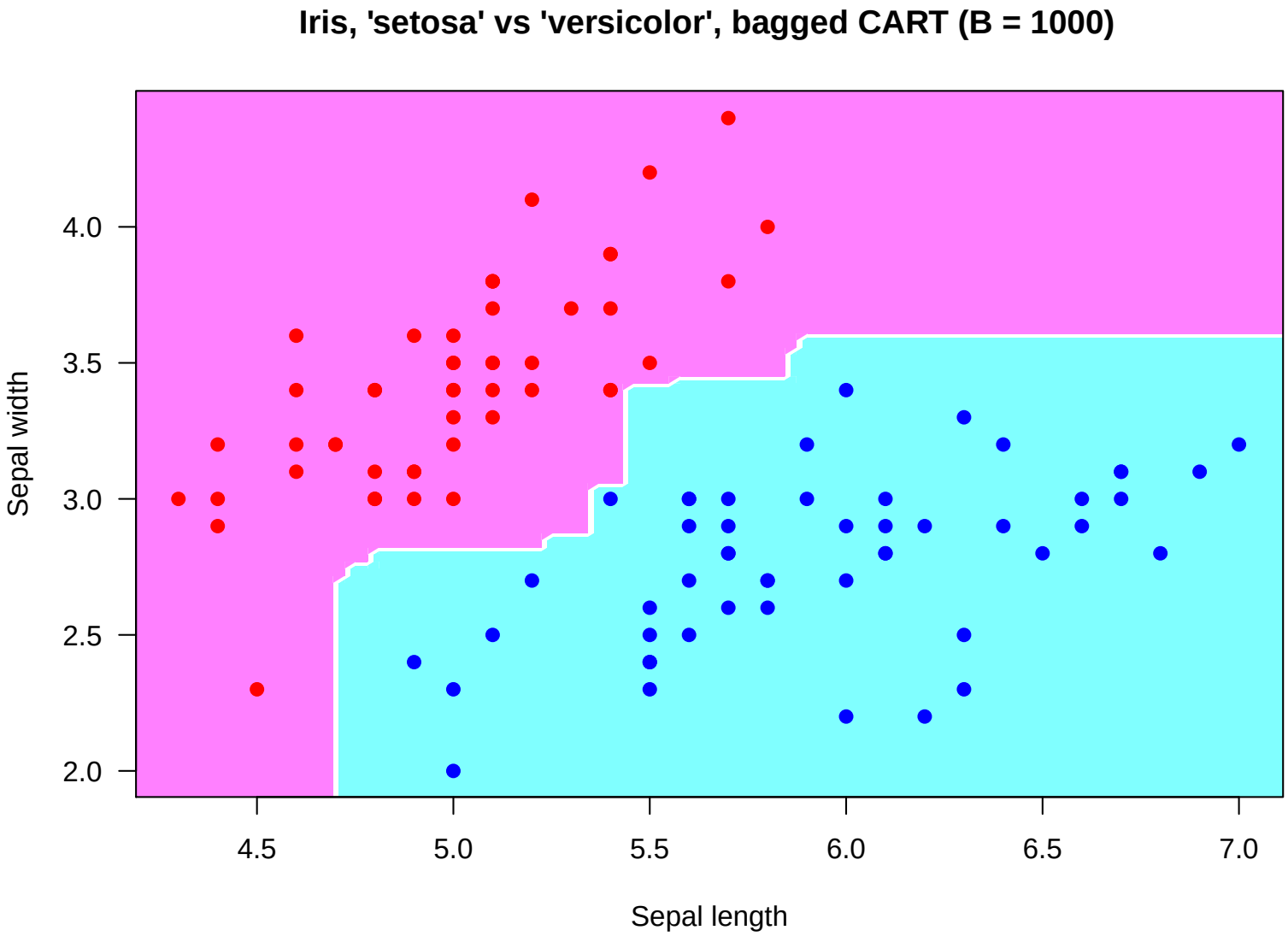
Bagging classification tree: iris data



Bagging classification tree: iris data



Bagging classification tree: iris data



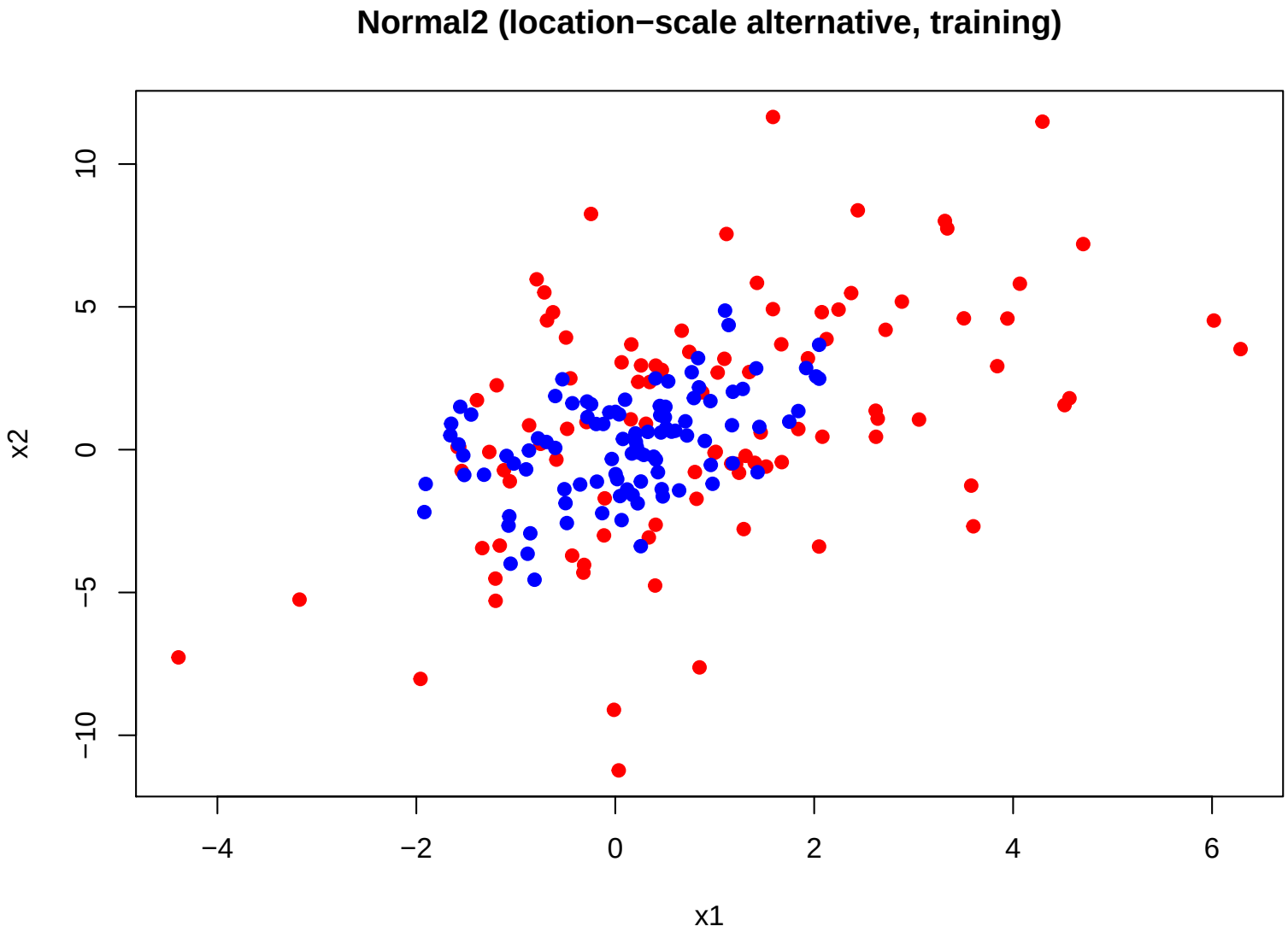
Properties and recommendations

- Bagging a good classifier can make it better, bagging a bad classifier can make it worse.
- Bagging may substantially **improve** performance of (unstable) **order-correct** base classifiers, *i.e.* those for which holds

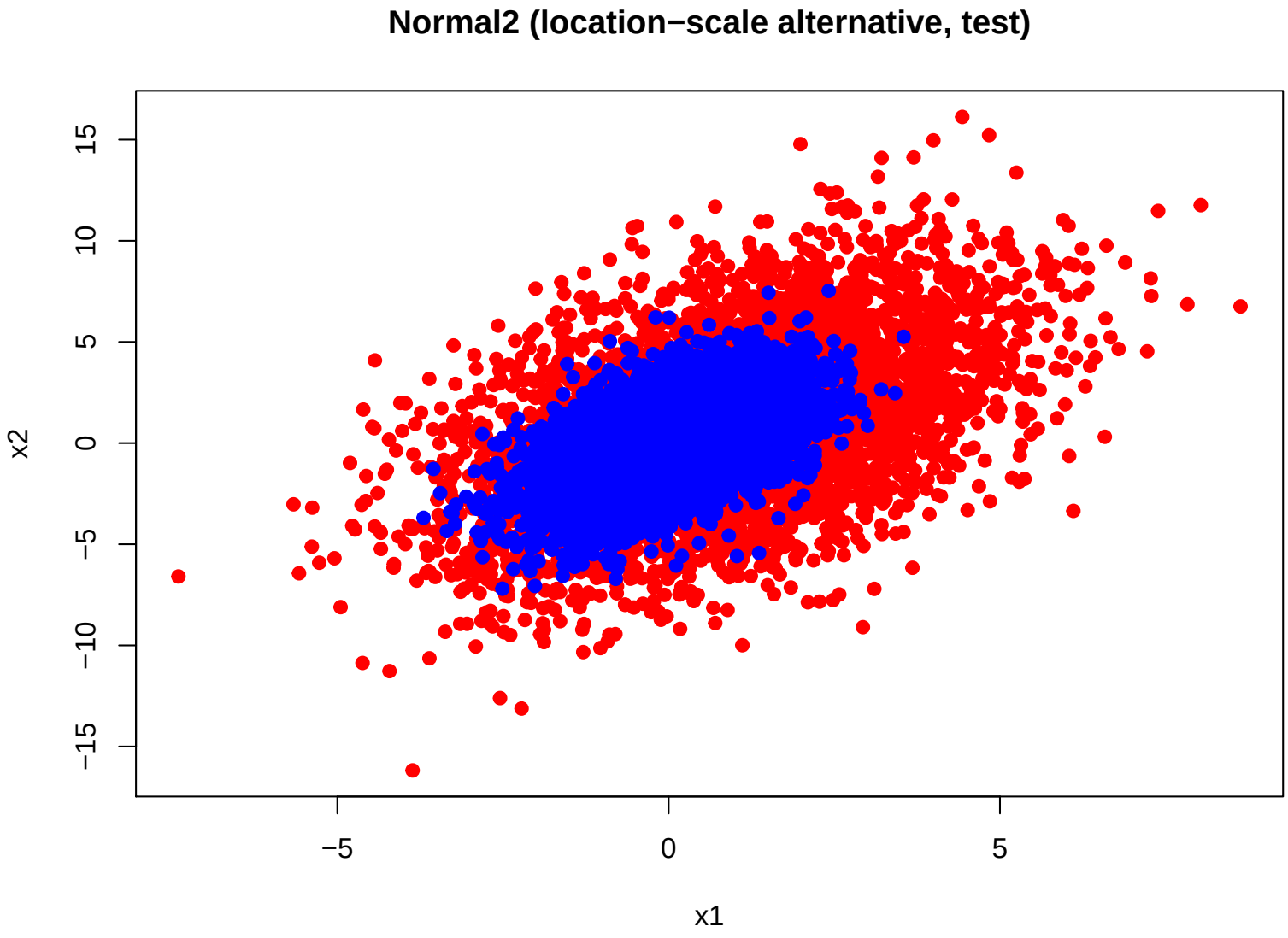
$$\arg\max_y \mathbb{P}\big(g(\boldsymbol{x}) = y\big) = \arg\max_y \mathbb{P}(y \mid \boldsymbol{x})\,.$$

- Significant improvement by bagging is not expected on large data sets because there bootstrap samples are very similar.
- Bagging reduces interpretability of the classifier because any simple structure in the model is lost.

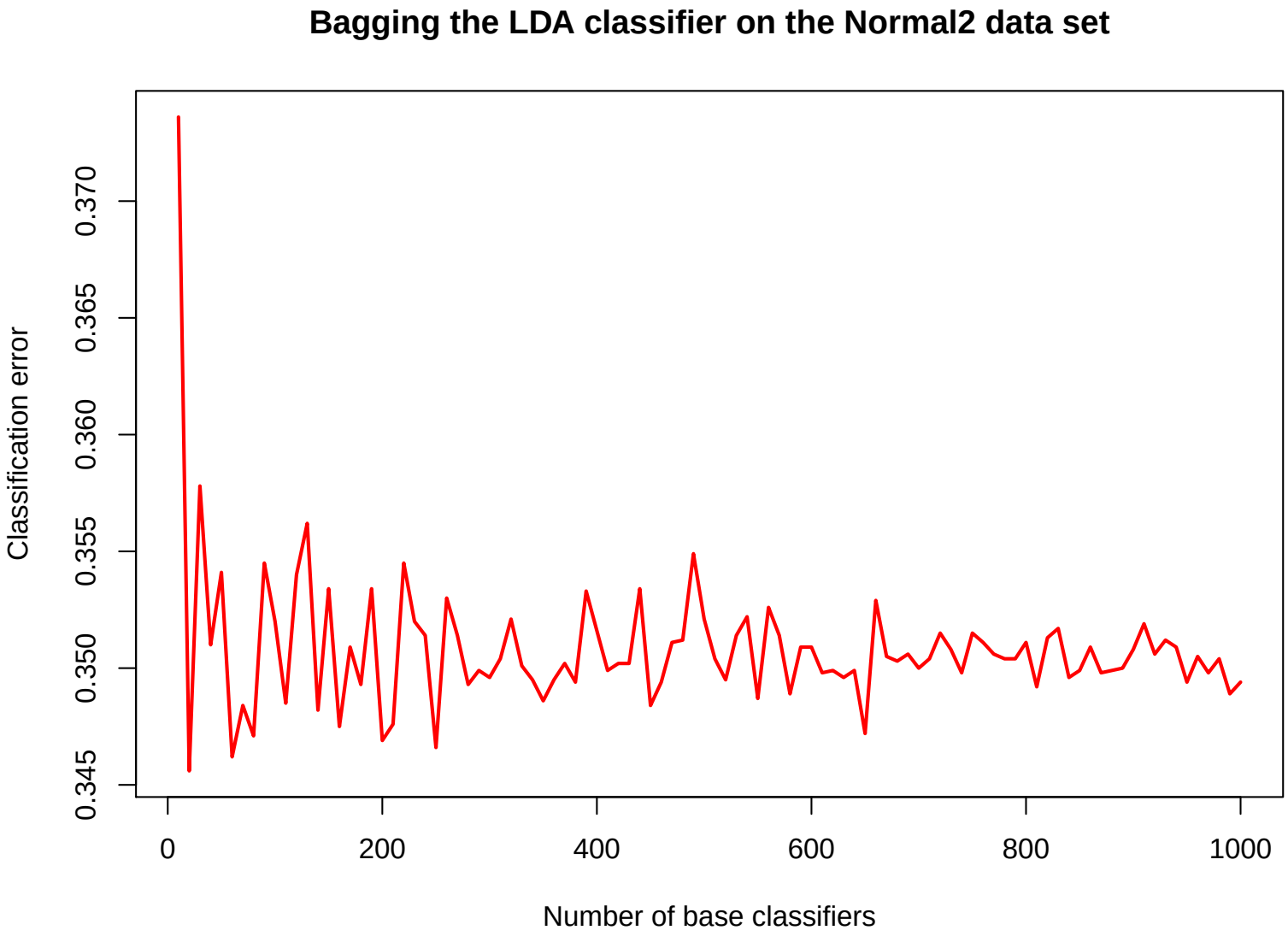
Normal2 data



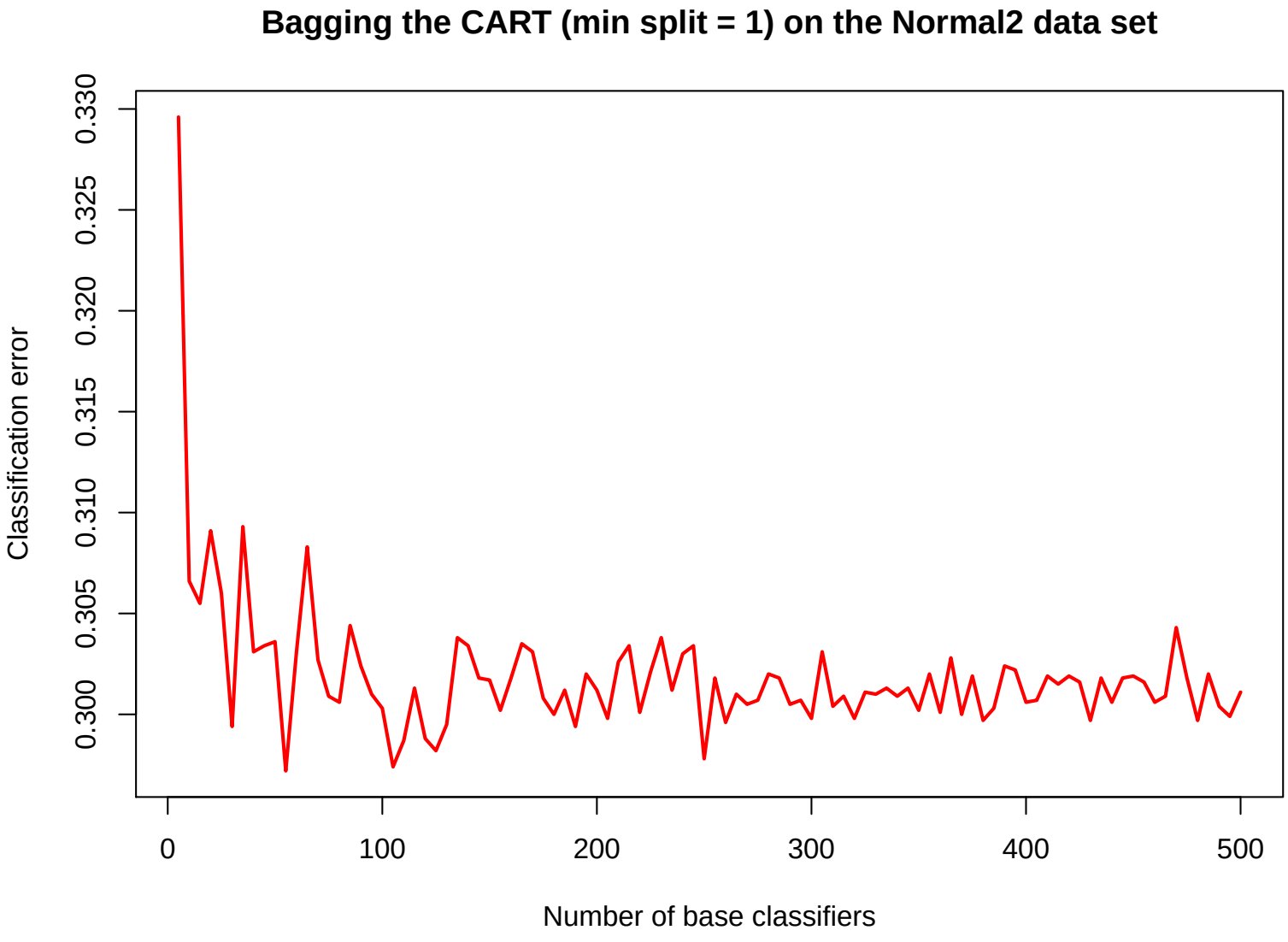
Normal2 data



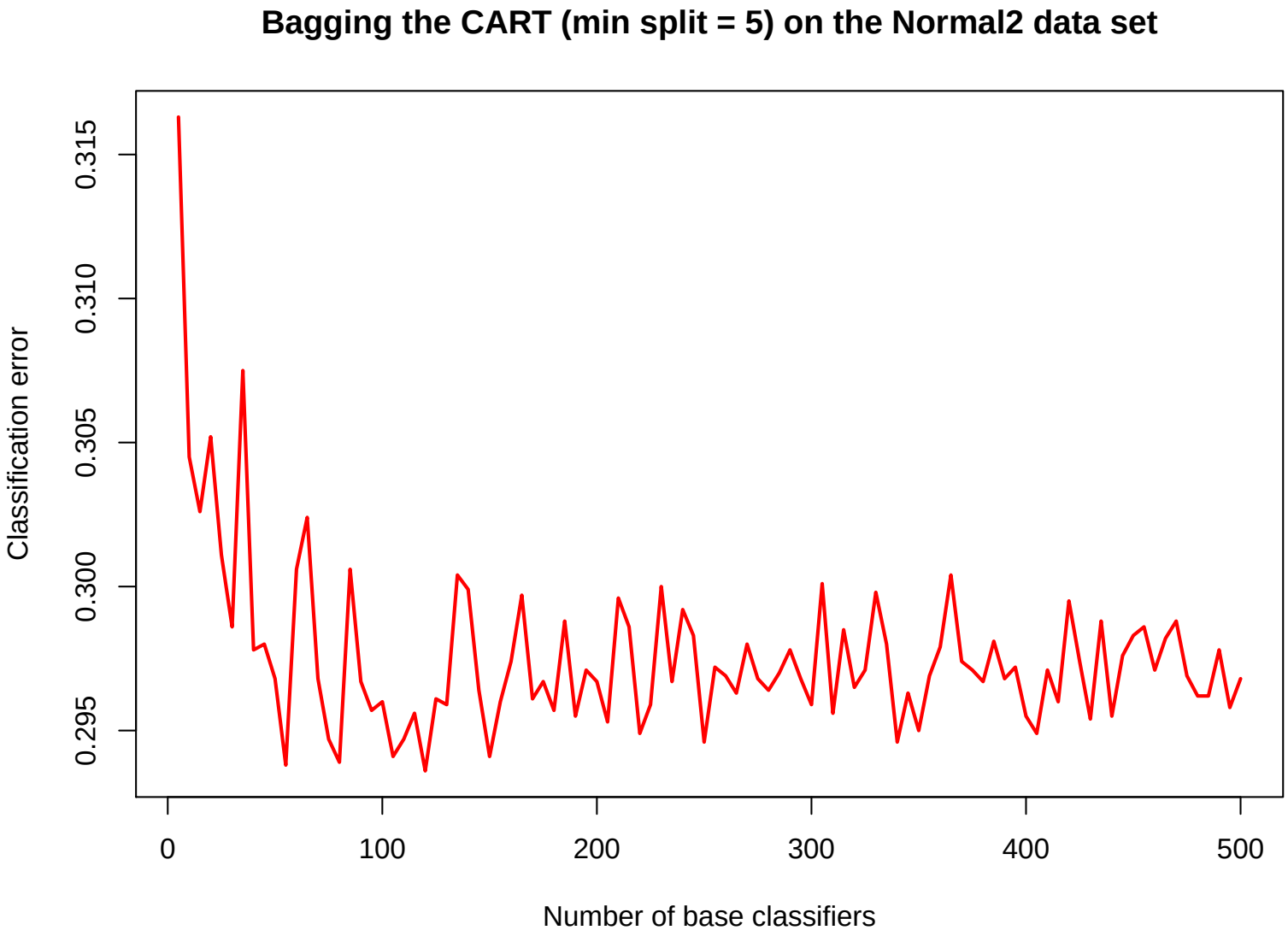
Bagging LDA: Normal2 data



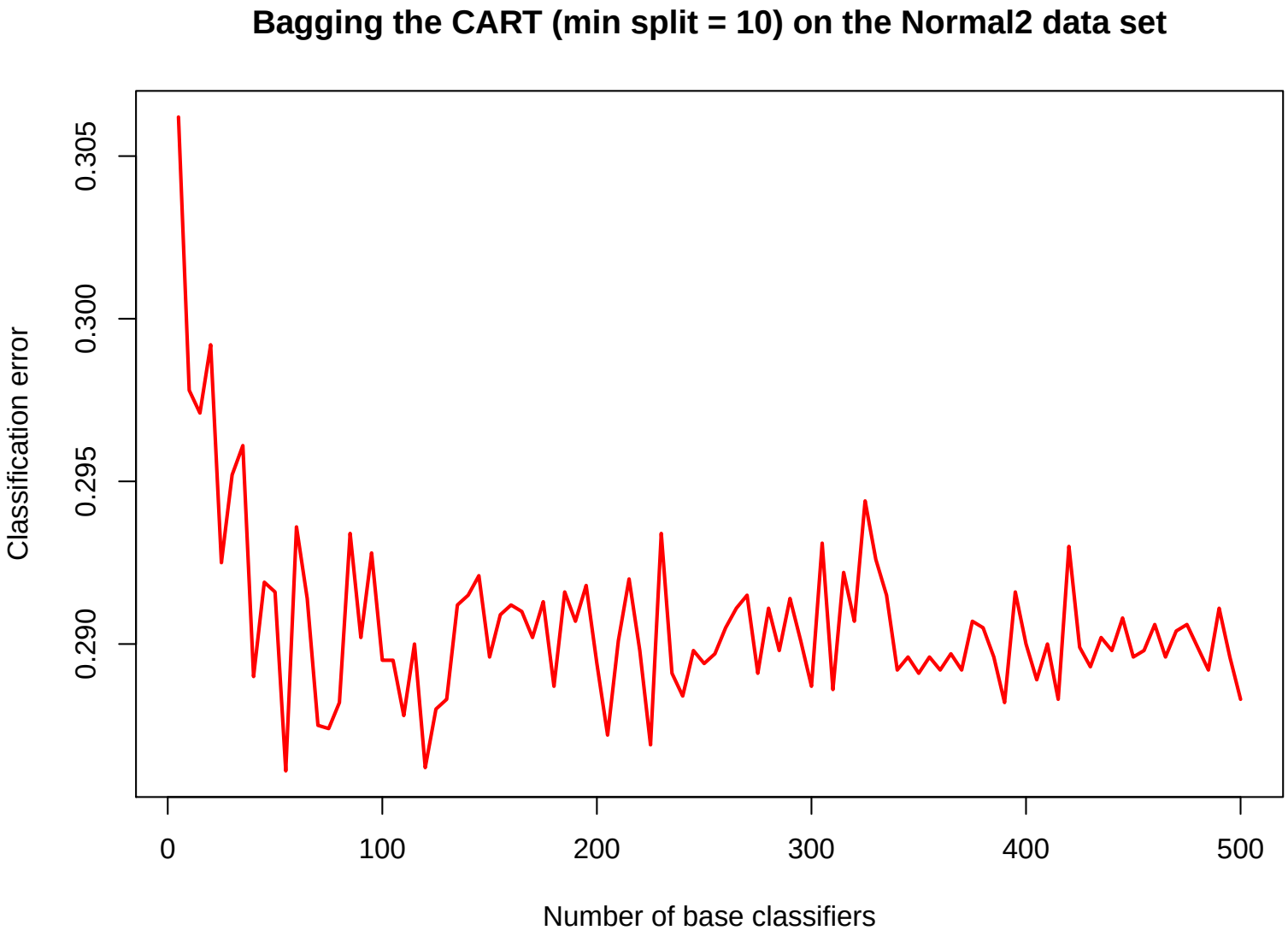
Bagging classification tree: Normal2 data



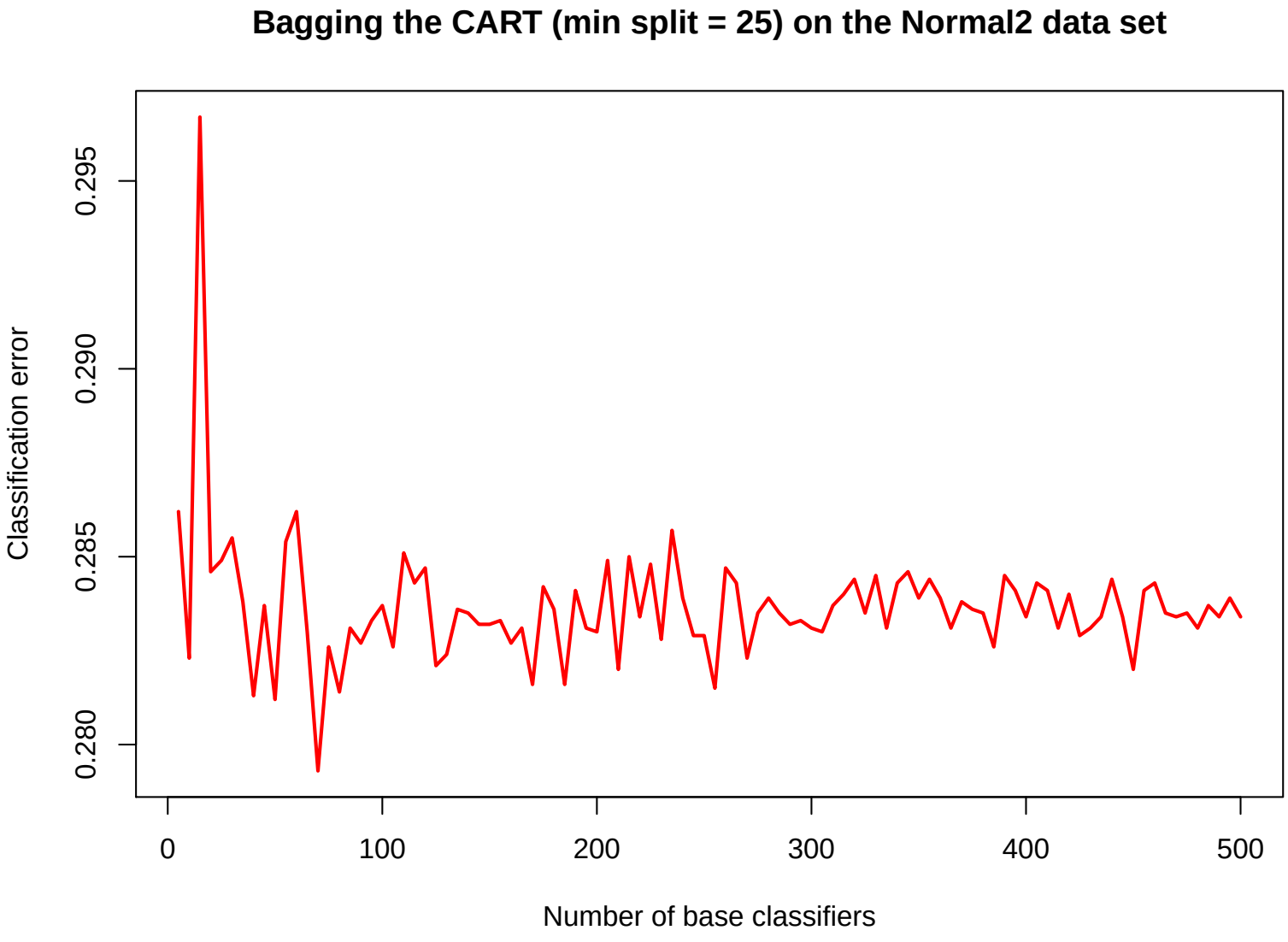
Bagging classification tree: Normal2 data



Bagging classification tree: Normal2 data



Bagging classification tree: Normal2 data



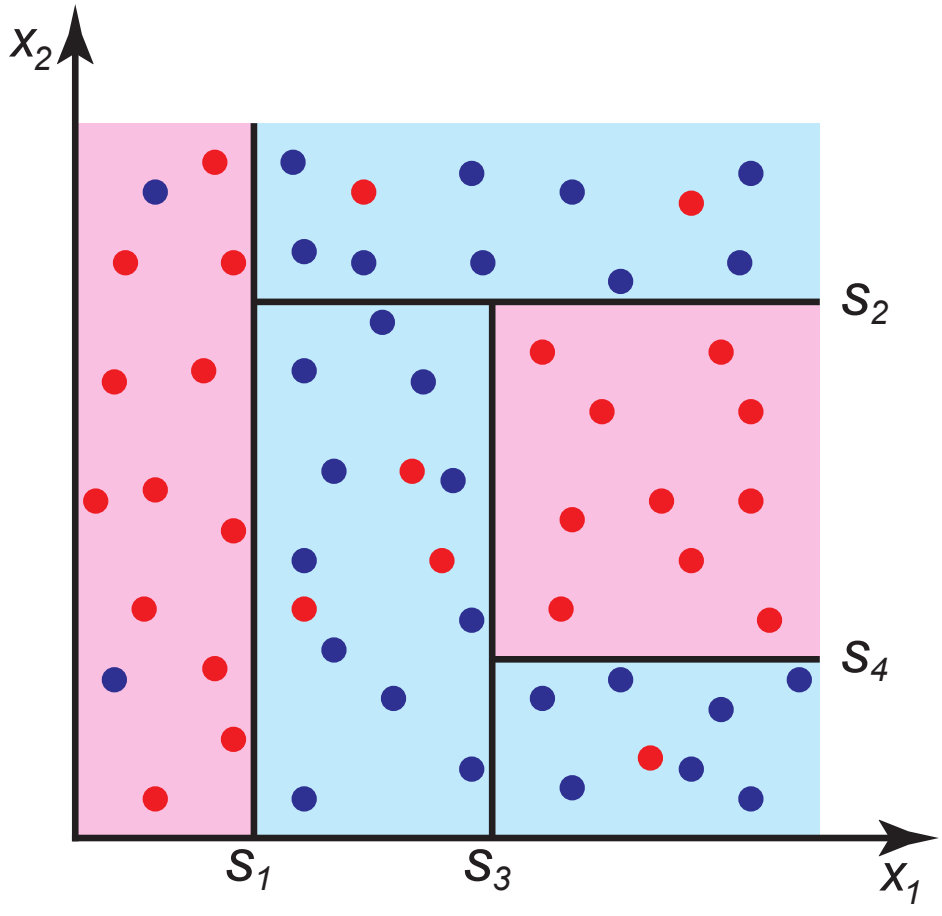
Contents

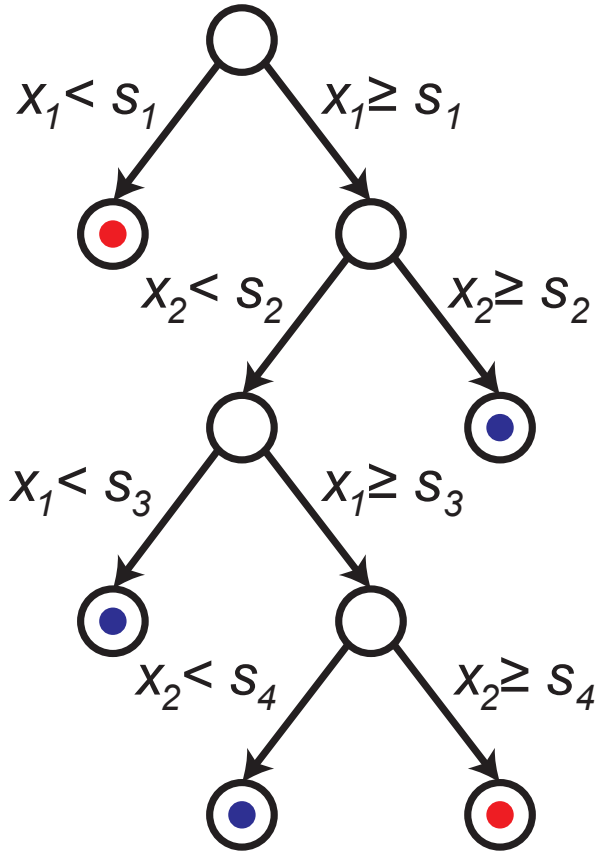
The key idea

- Random forest is a **collection of trees**.
- Random forests have been introduced by **Léo Breiman** in the early 2000s. The following web-page is dedicated to random forests: `http://www.stat.berkeley.edu/~breiman/RandomForests/`
- Random forests can be seen as a **modified version of bagging** as they also aggregate trees taught on the bootstrap samples.
- Let $T_k(\boldsymbol{x})$, $k = 1, ..., B$ be tree-similar classifiers ($T_k : \mathbb{R}^d \to \{0, 1\}$). The **random forest** classifier assigns new observation $\boldsymbol{x}$ by aggregating these:

$$T^{RF}(\boldsymbol{x}) = \mathbb{1}\left(\frac{1}{B}\sum_{k=1}^{B} T_k(\boldsymbol{x}) > 0.5\right).$$

The key idea





- **To reduce correlation between trees** Breiman proposes to **select** the best variable when splitting each node among m **randomly chosen variables** out of all d variables.

Random forests (algorithm)

Training

Input:

- Training sample $\big((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\big) = \mathcal{D}_n \in \mathbb{R}^d \times \{0, 1\}$.
- Number of trees B; minimum number of observations for a node to be split n_{min}; impurity criterion Q.
- number of variables to use when splitting $m \in \{1, ..., d\}$.

For $k = 1, ..., B$

1. Draw a sample $\mathcal{D}_n^k$ from $\mathcal{D}_n$ using bootstrap.
2. Learn the classification tree T_k on $\mathcal{D}_n^k$; each time when splitting a node, search optimal variable among m variables randomly chosen out of all d variables.

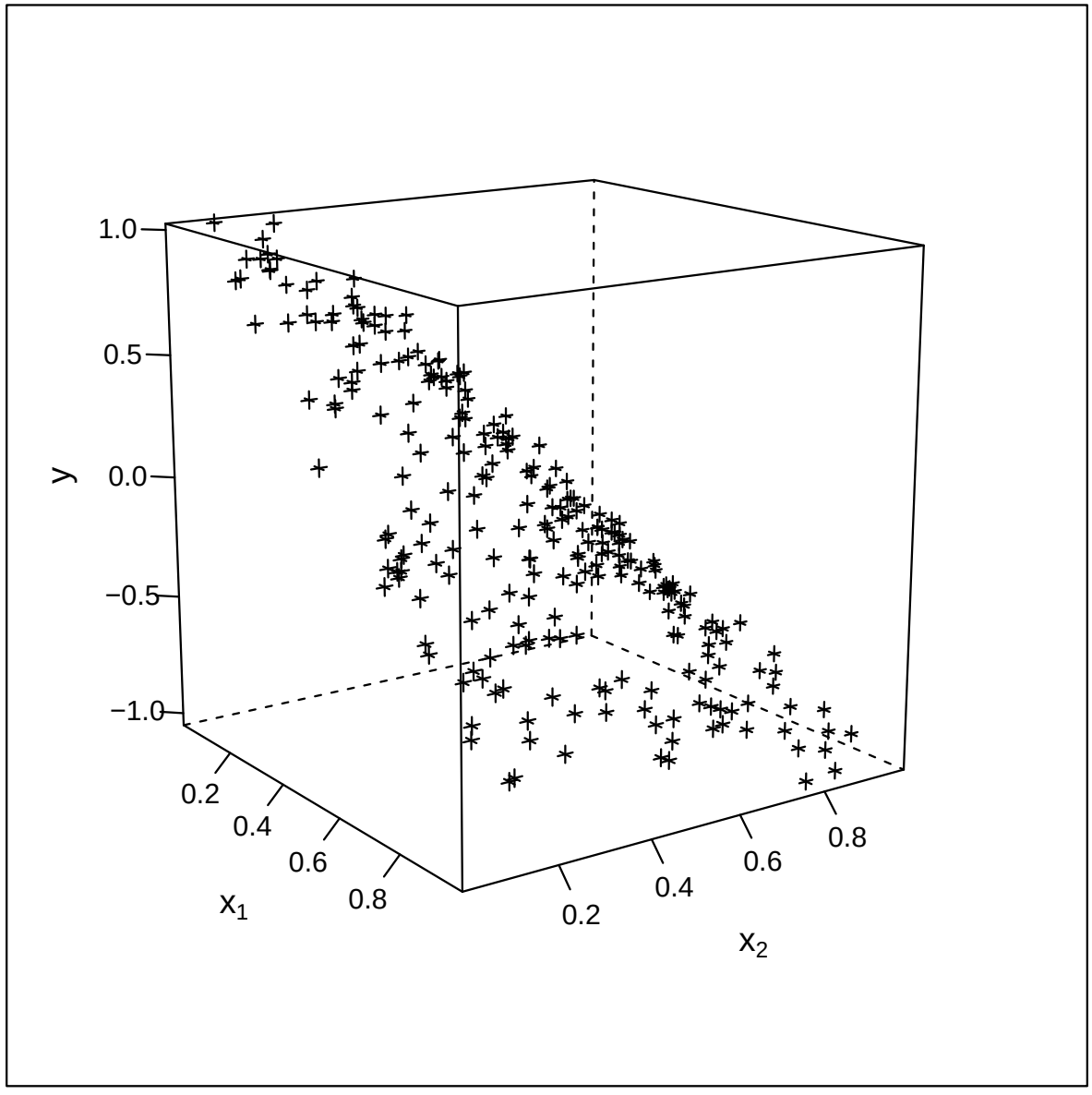
Output: The aggregated classifier $T^{RF}(\cdot) = \mathbb{1}\Big(\frac{1}{B}\sum_{k=1}^{B} T_k(\cdot) > 0.5\Big)$.

Classification

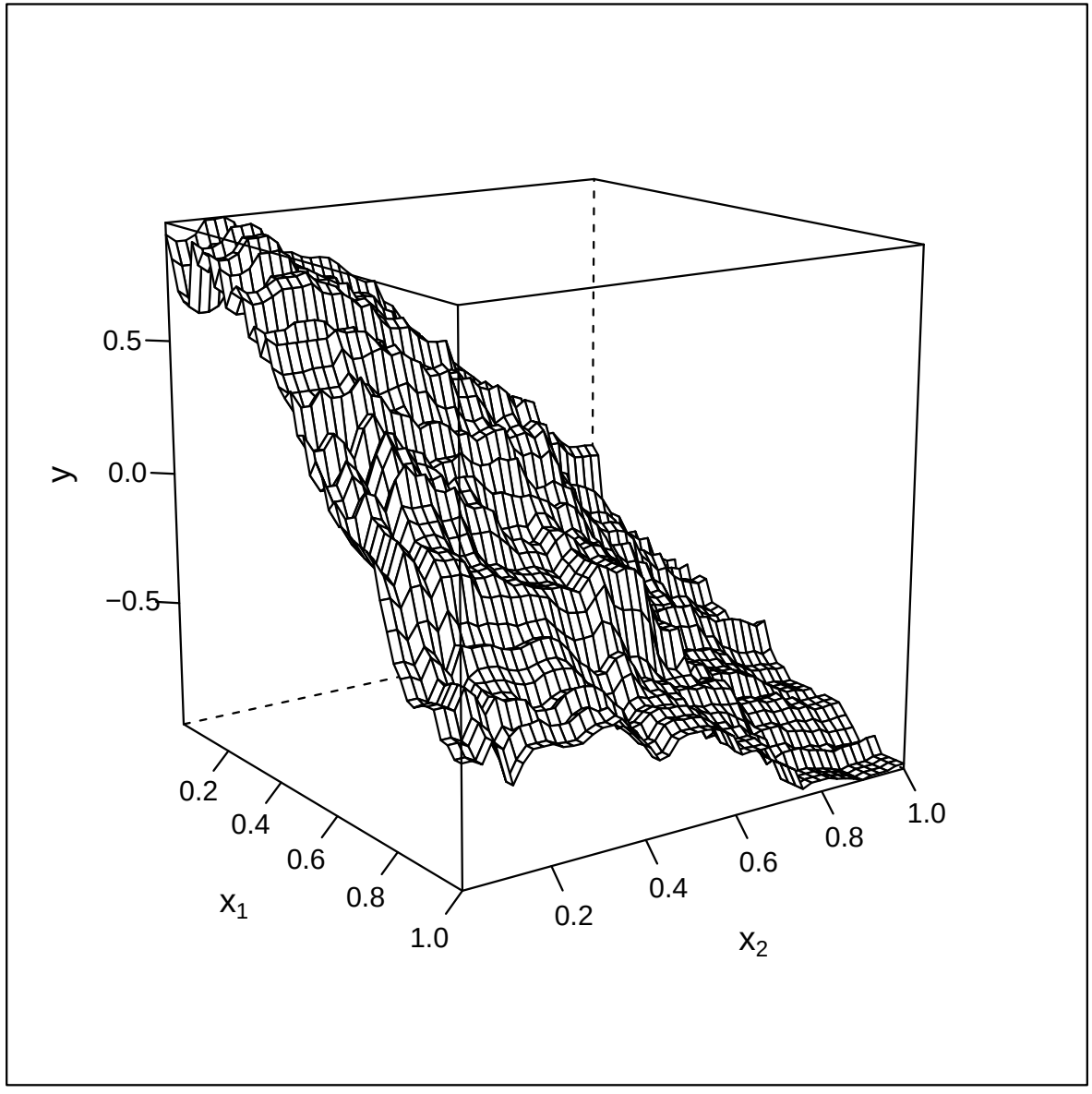
- Classify the new observation $\boldsymbol{x}$ as

$$T^{RF}(\boldsymbol{x}) = \mathbb{1}\Big(\frac{1}{B}\sum_{k=1}^{B} T_k(\boldsymbol{x}) > 0.5\Big).$$

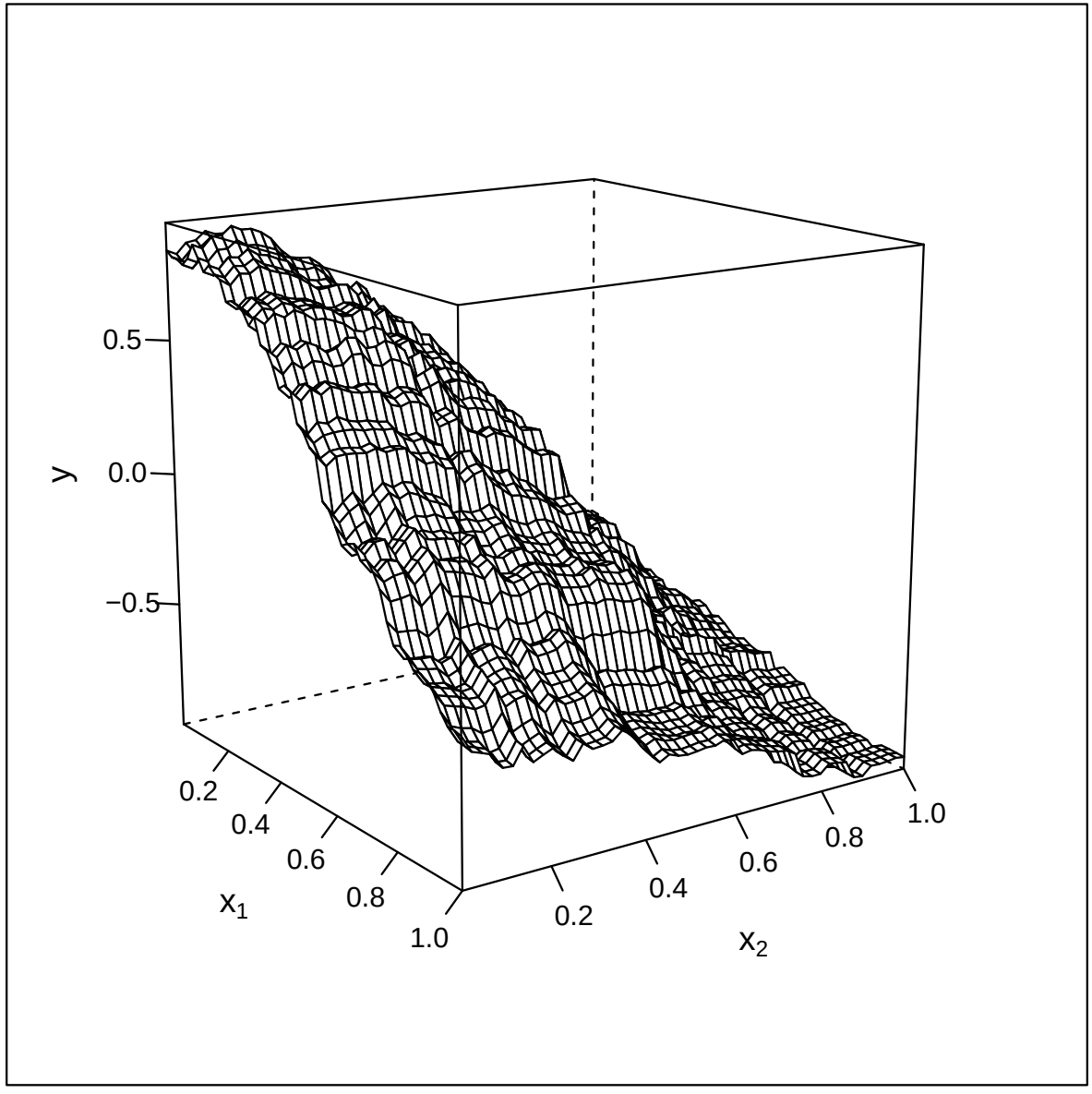
Random forest (regression): illustration



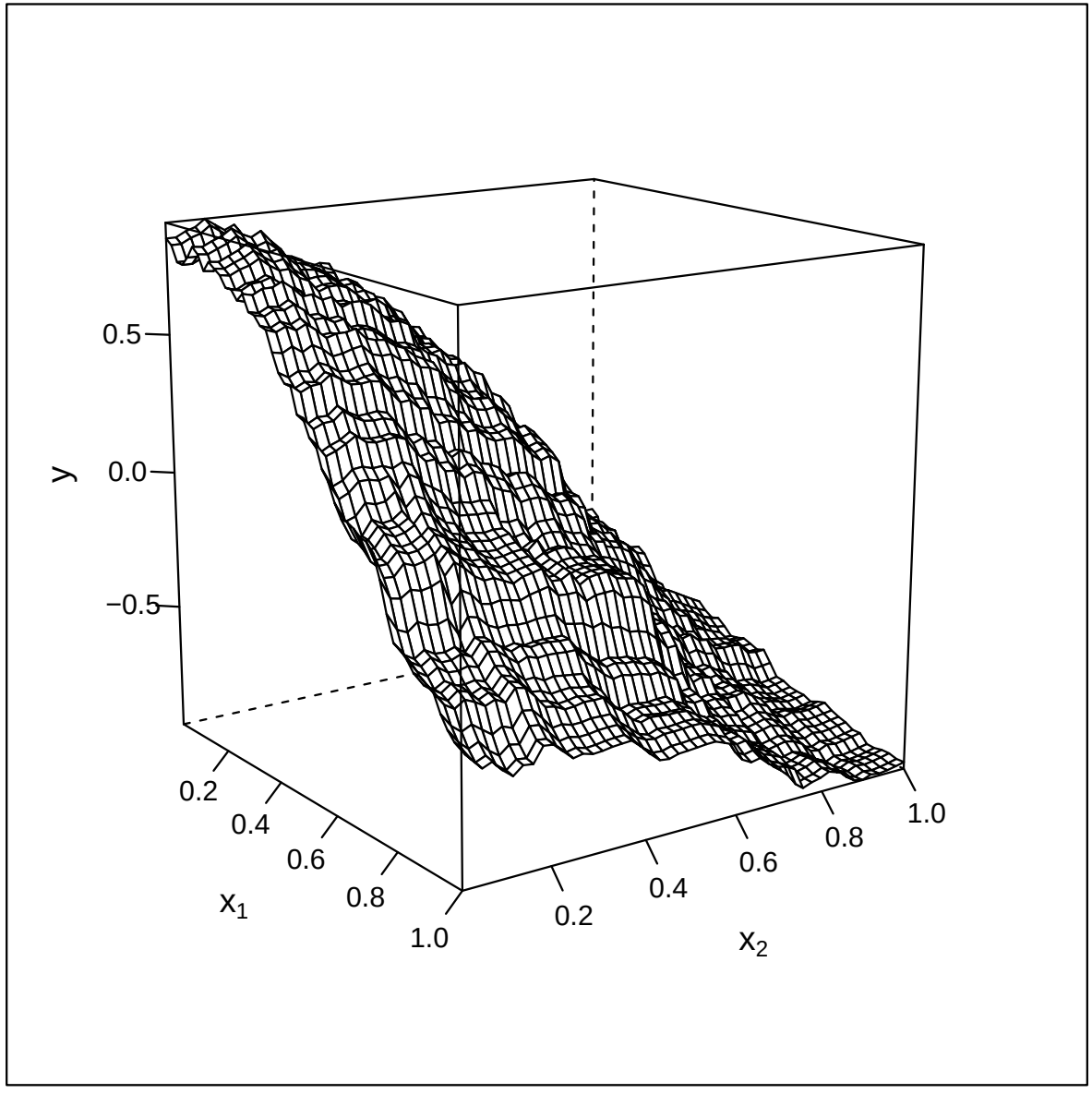
Random forest (regression): illustration



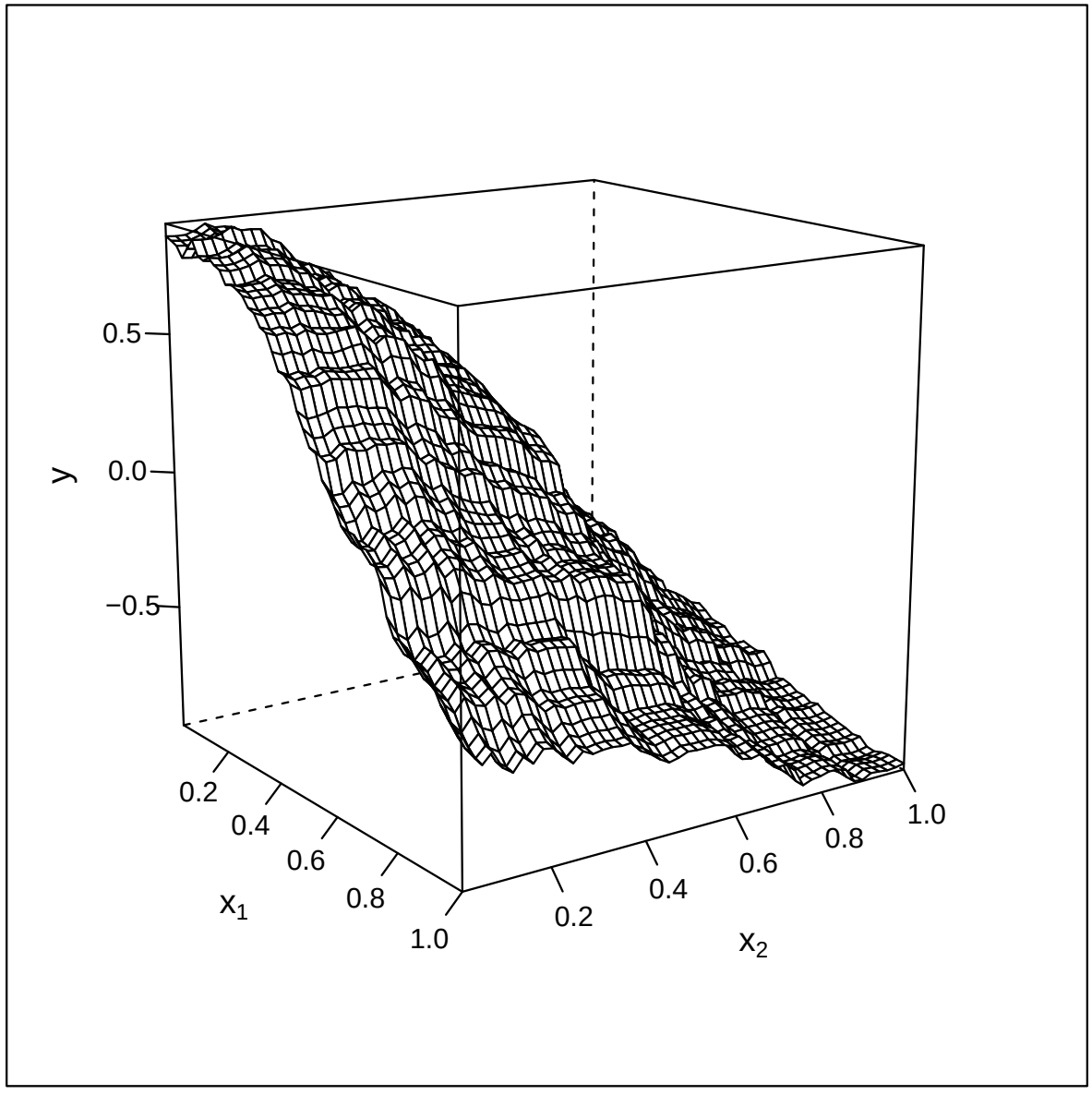
Random forest (regression): illustration



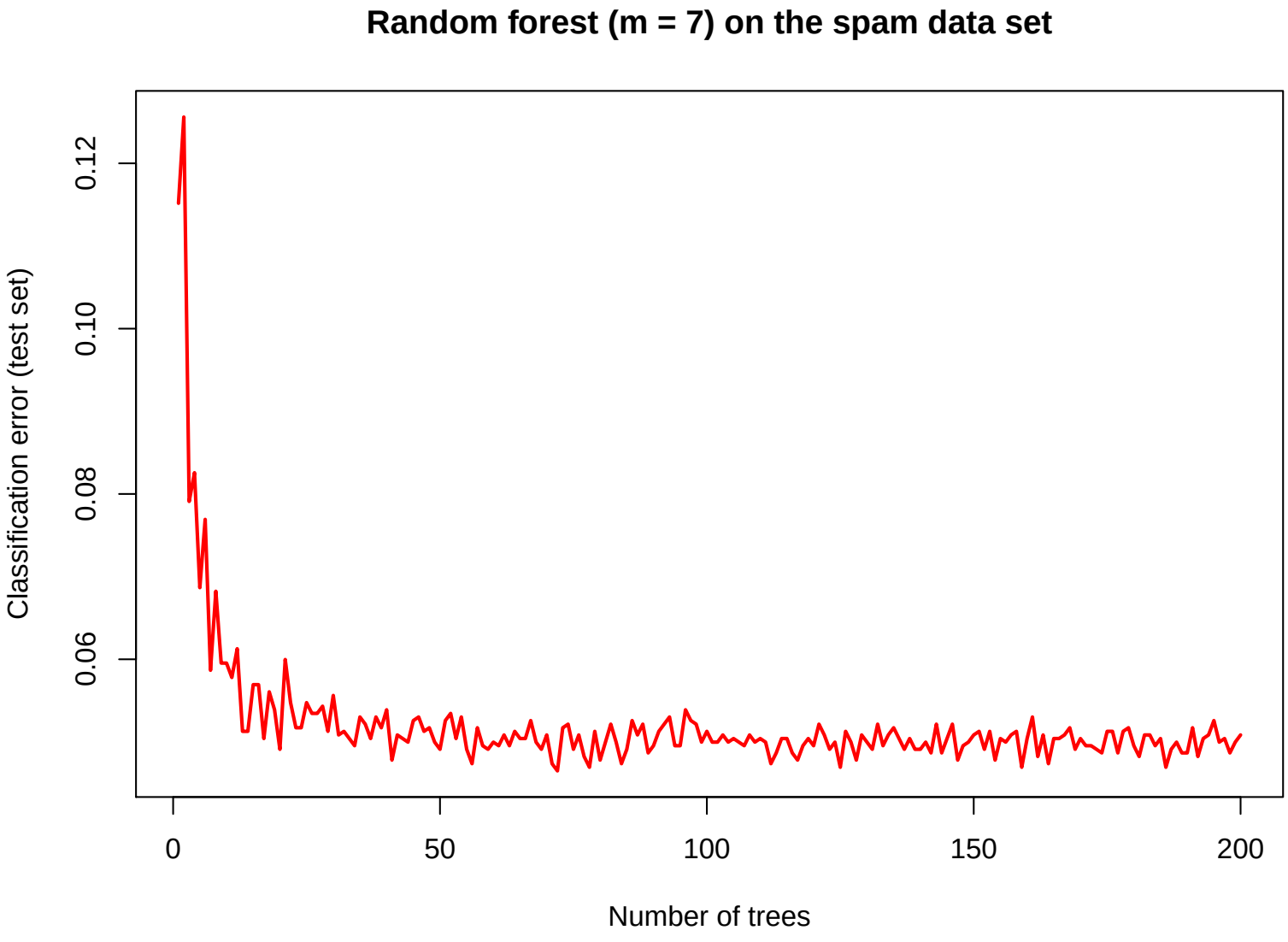
Random forest (regression): illustration



Random forest (regression): illustration



Random forests: spam data



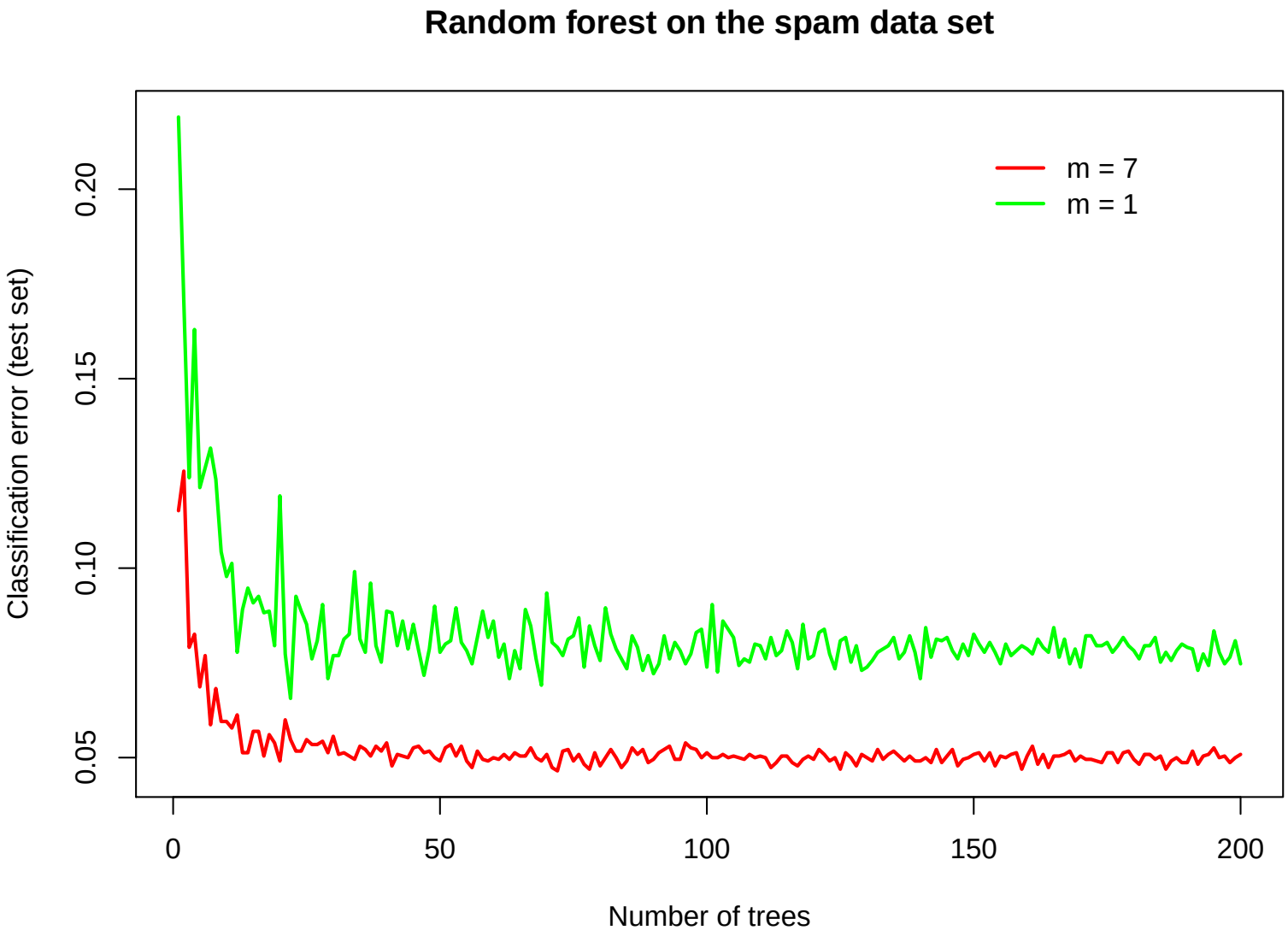
Properties

- There are **two** introduced **sources of randomness**:
  - B **bootstrap samples**,
  - m **variables randomly chosen** out of d when splitting each tree node.
- The method is simple, implementations are available in numerous software.
- The classifier is known for relatively **high speed** of training and classification.
- The classifier is known for its relatively **precise prediction on complex data**, *i.e.* those including many variables, missing entries, *etc*.
- Classifier has **limited sensibility w.r.t. to the choice of parameters**: B, m, n_{min}.

Choice of number m of variables for a node

- Parameter m is related to the **dependence between** single **trees**.
- The **lower** is m:
  - to larger extent the variables at which to split each node **are chosen randomly**,
    thus the more **different** are single **trees**,
    thus the more **independent** are single **trees**,
  - the **lower** is the **prediction accuracy of** each single **tree**,
    and thus of the entire forest as well.
- The **higher** is m: *vice versa*.
- It is recommended to check the performance of the random forests for **different choices of** m.
- The **inventors recommend** $m = \lfloor \sqrt{d} \rfloor$
  (the default value in `R`-package `randomForest`).

Random forests: spam data



Performance and interpretation

- It is desirable to measure the performance of the random forests, as of any other classification technique, in terms of the error probability:

$$R(X, Y) = \mathbb{P}\left(T^{RF}(X) \neq Y\right).$$

- As usual, the error can be measured:
  - For a **probability distribution**: by a **simulation study**, *i.e.* train T^{RF} and measure its classification error for a number of simulated data sets.
  - For **given data**: by **splitting data** into training and test subsets, by **iterating this splitting** if data are small, or by **cross-validation**.
- Random forests offer an additional possibility to directly estimate classification error exploiting the **out-of-bag (OOB)** principle.
- The same idea can be extended from sample points to variables allowing to measure **variable importance**.

Out-of-bag error

- For each pair $(\boldsymbol{x}_i, y_i)$ from $\mathcal{D}_n$, let I_i be the **set of indices of trees** whose bootstrap samples $\mathcal{D}_n^*$ **do not contain this observation**.

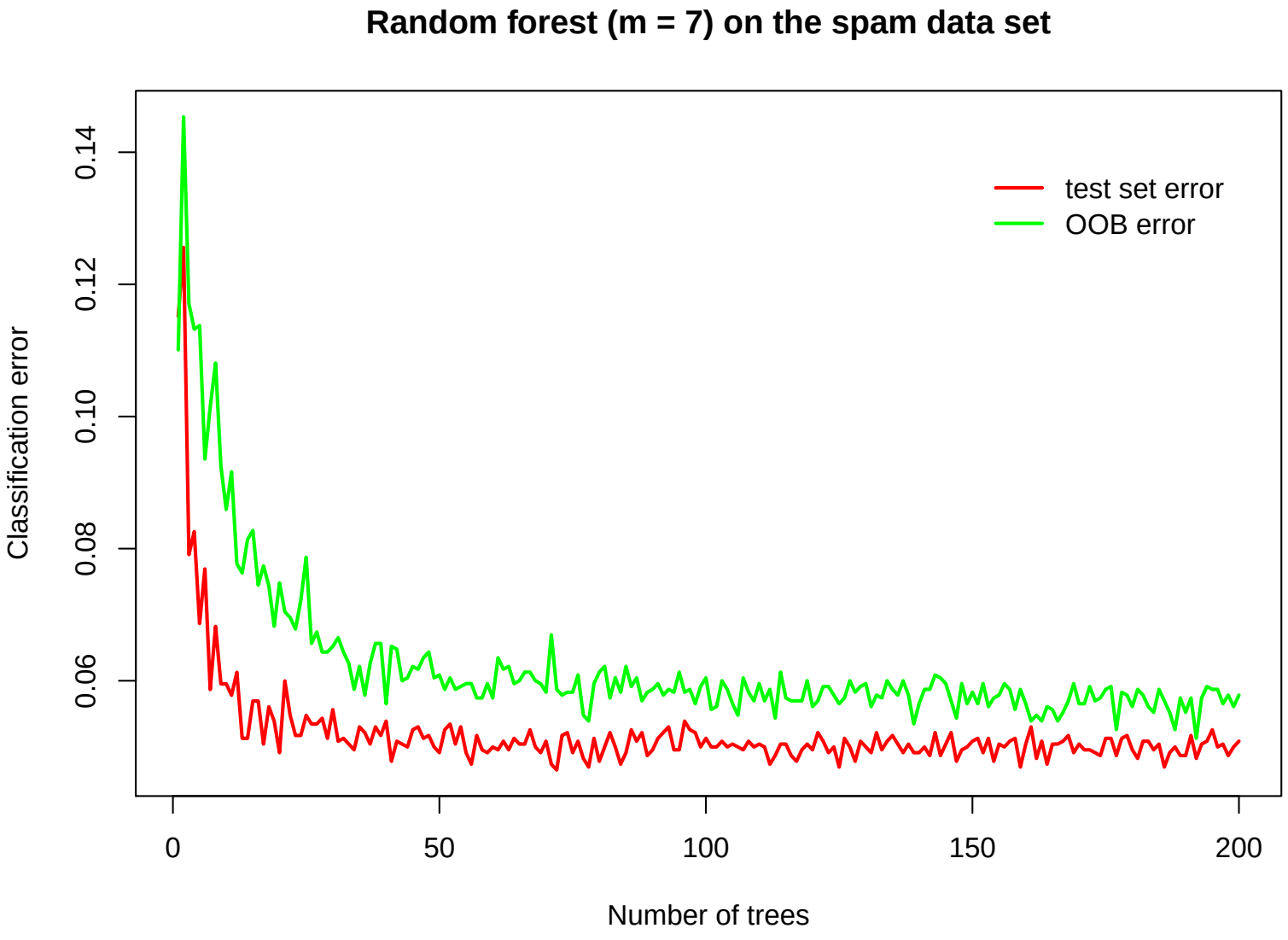
- By these trees, observation $\boldsymbol{x}_i$ is then classified as

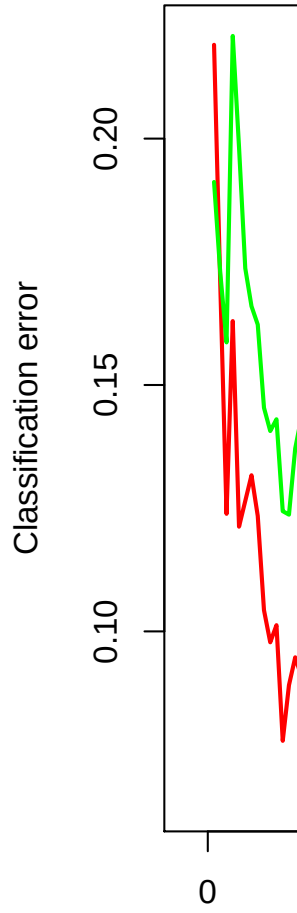
$$\hat{y}_i = \frac{1}{\#I_i} \sum_{k \in I_i} T_k(\boldsymbol{x}_i)\,.$$

- Averaging over all observations $\boldsymbol{x}_i$, $i = 1, ..., n$ from $\mathcal{D}_n$ gives the out-of-bag estimate of the error rate:

$$R_{OOB} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left(\hat{y}_i \neq y_i\right).$$

Random forests: spam data





Importance of a variable

- For a bootstrap sample $\mathcal{D}_n^k$ associated with the kth tree, let $\mathcal{D}_n^{k-}$ be the **subset of training sample not contained in** $\mathcal{D}_n^k$, *i.e.* it holds $\mathcal{D}_n^k \cup \mathcal{D}_n^{k-} = \mathcal{D}_n$ and $\mathcal{D}_n^k \cap \mathcal{D}_n^{k-} = \emptyset$.
- Then, let $R_{OOB(k)}$ be the classification error estimated on this subset $\mathcal{D}_n^{k-}$:

$$R_{OOB(k)} = \frac{1}{\#\mathcal{D}_n^{k-}} \sum_{\boldsymbol{x} \in \mathcal{D}_n^{-k}} \mathbb{1}\big(T_k(\boldsymbol{x}) \neq y_i\big)\,.$$

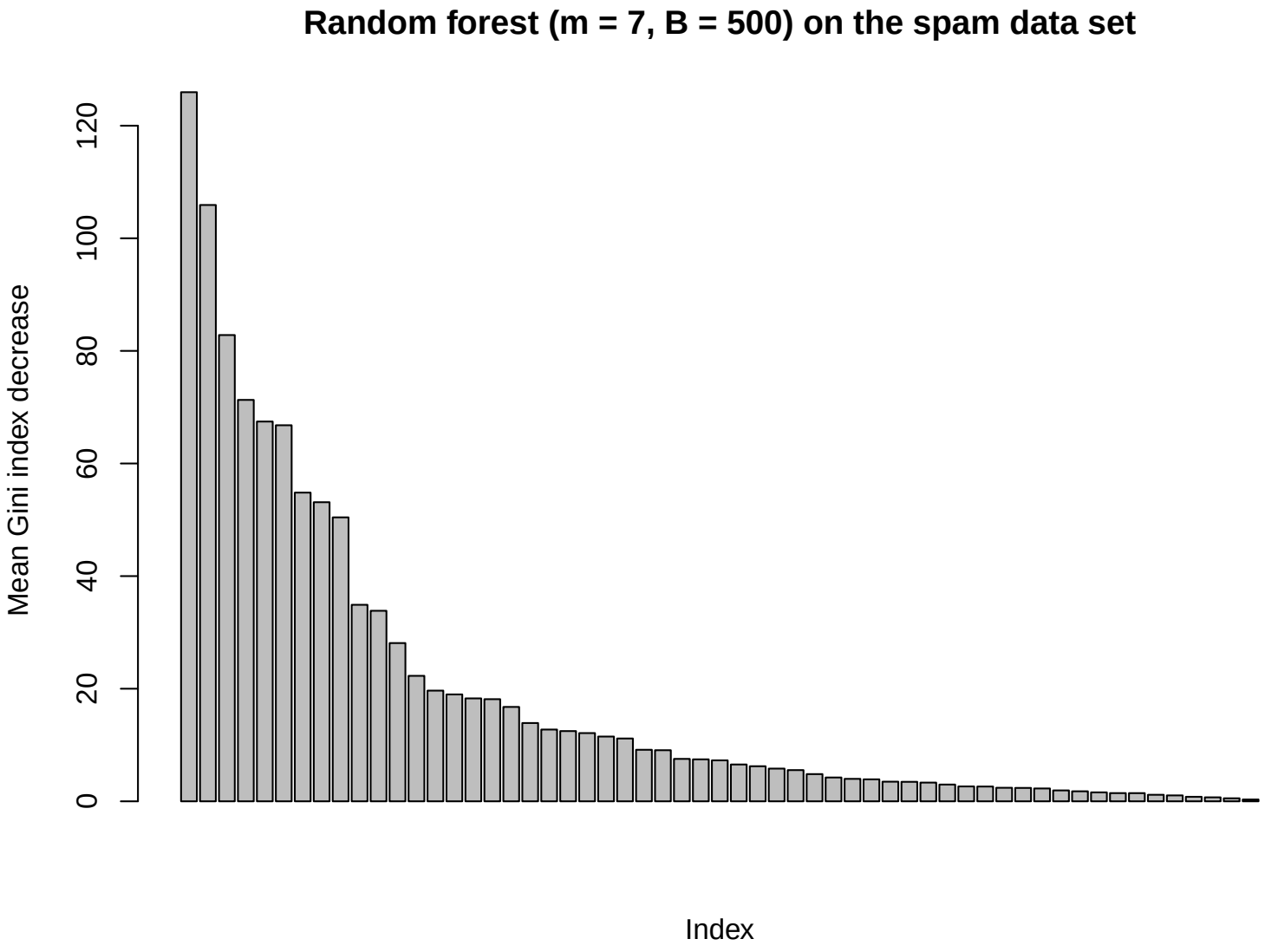
- Further, let $\mathcal{D}_n^{k-}(j)$ be the same subset $\mathcal{D}_n^{k-}$ where the **values of variable** $j \in \{1, ..., d\}$ have been **randomly perturbed**, and measure the error from above on this perturbed subset:

$$R_{OOB(k,j)} = \frac{1}{\#\mathcal{D}_n^{k-}(j)} \sum_{\boldsymbol{x} \in \mathcal{D}_n^{-k}(j)} \mathbb{1}\big(T_k(\boldsymbol{x}) \neq y_i\big)\,.$$

- The **importance of variable** j can thus be measured (by averaging over all B trees) as:

$$Imp(X_j) = \frac{1}{B} \sum_{k=1}^{B} (R_{OOB(k,j)} - R_{OOB(k)})\,.$$

Random forests: spam data



Consistency of the purely random forest classifier

Define the **purely random tree classifier** T^{pr} as follows:

- The support of X (and thus the root node of T^{pr}) is $[0,1]^d$.
- At each step, the leaf is chosen uniformly at random among all existing leaves.
- At each node, the split variable j is chosen uniformly at random among $1, ..., d$.
- The selected cell is split at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell.
- The procedure is repeated k times where $k \geq 1$ is fixed in advance.
- The only data driven element is the class label of the leaf, chosen due to the majority of the observations contained in it.

Theorem (Biau, Devroye, Lugosi, 2008)

Assume that the distribution of X is supported on $[0,1]^d$. Then the purely random forest classifier $T_B^{prRF} = \mathbb{1}\big(\frac{1}{B}\sum_{k=1}^{B} T^{pr}(\cdot, \mathcal{D}_n)\big)$ (as well as $\lim_{B\to\infty} T_B^{prRF}$) is consistent whenever $k \to \infty$ and $\frac{k}{n} \to 0$ as $k \to \infty$.

Consistency of the scale-invariant random forest classifier

Define the **scale-invariant random tree classifier** T^{si} as follows:

- Take the **purely random tree classifier**.
- Let the root node be the entire space $\mathbb{R}^d$.
- Define the node-cutting procedure as follows:
  if the cell (node) m contains n_m points $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_m}$, then the random index I is chosen uniformly from the set $\{0, 1, ..., n_m\}$, and the cut is performed in the chosen variable between the points $\boldsymbol{x}_I$ and $\boldsymbol{x}_{I+1}$.

Theorem (Biau, Devroye, Lugosi, 2008)

Assume that the distribution of X has non-atomic marginals in $\mathbb{R}^d$. Then the scale-invariant random forest classifier $T_B^{siRF} = \mathbb{1}\big(\frac{1}{B}\sum_{k=1}^{B} T^{si}(\cdot, \mathcal{D}_n)\big)$ (as well as $\lim_{B\to\infty} T_B^{siRF}$) is consistent whenever $k \to \infty$ and $\frac{k}{n} \to 0$ as $k \to \infty$.

Consistency of bagging

Remind:

- Bagging classifier:

$$g_B^{agg}(\boldsymbol{x}) = \mathbb{1}\Big(\frac{1}{B}\sum_{k=1}^{B} g_k(\boldsymbol{x}, \theta_k, \mathcal{D}_n) > 0.5\Big)\,.$$

- Averaged classifier (the limit of the bagging classifier):

$$\lim_{B\to\infty} g_B^{agg}(\boldsymbol{x}) = \mathbb{1}\big(\mathbb{E}_\theta[g(\boldsymbol{x}, \theta, \mathcal{D}_n)]\big)\,.$$

with θ being a random variable delivering a bootstrap sample of size $Bin(n, q_n)$ (without replacement), and $q_n \in [0, 1]$.

Theorem (Biau, Devroye, Lugosi, 2008)

Assume that the classifier g is consistent for a certain distribution (X, Y). Then the bagging classifier g_B^{agg} and its limit $\mathbb{1}\big(\mathbb{E}_\theta[g(\boldsymbol{x}, \theta, \mathcal{D}_n)]\big)$ are also consistent if $nq_n \to \infty$ as $n \to \infty$.

Thank you for your attention!

Thank you for your attention!

And some references

- Biau, G., Devroye, L., and Lugosi, G. (2008).
  Consistency of random forests and other averaging classifiers.
  Journal of Machine Learning Research, 9, 2015–2033.

- Breiman, L. (1996).
  Bagging predictors.
  Machine Learning, 24, 123–140.

- Breiman, L. (2001).
  Random forests.
  Machine Learning, 45, 5–32.

- Hastie, T., Tibshirani, R., and Friedman, J. (2009).
  The Elements of Statistics Learning: Data Mining, Inference, and Prediction (Second Edition).
  Springer.

- Zhou, Z.-H. (2012).
  Ensemble Methods: Foundations and Algorithms.
  CRC Press.