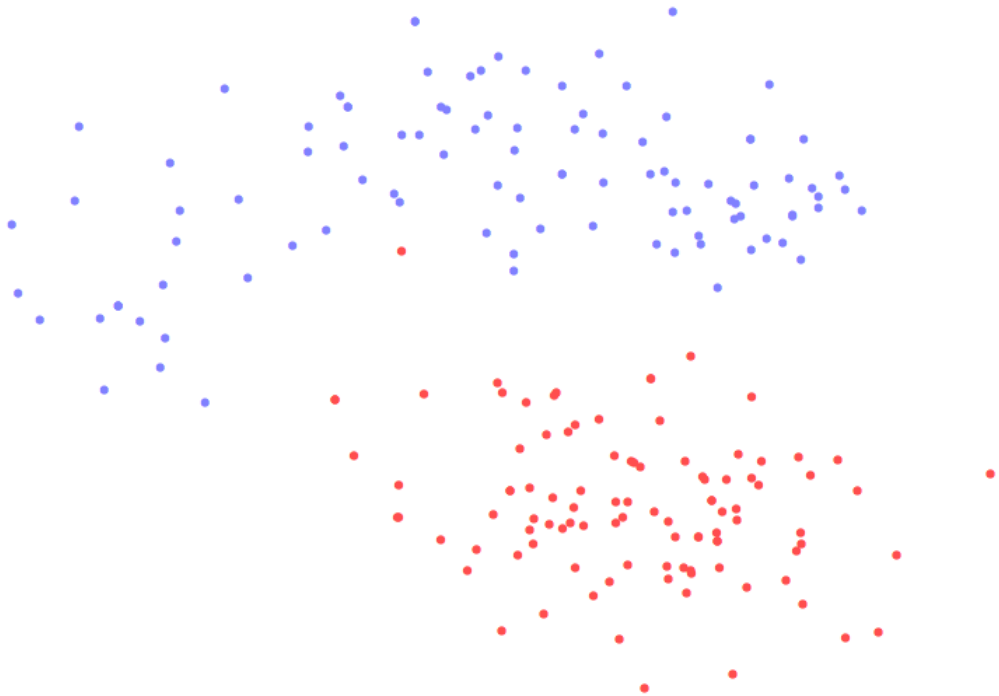
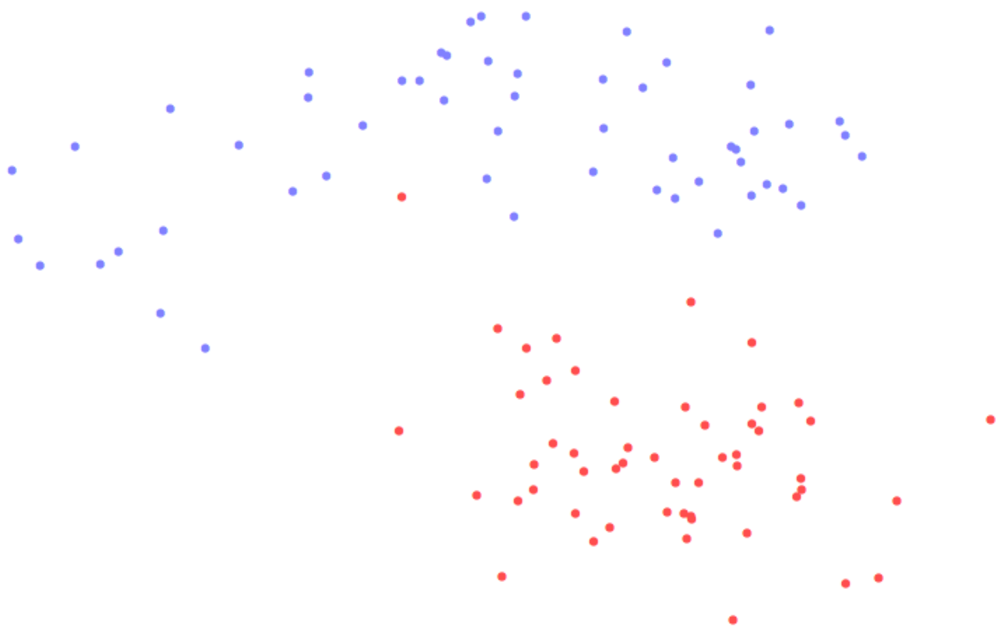
Sample Properties



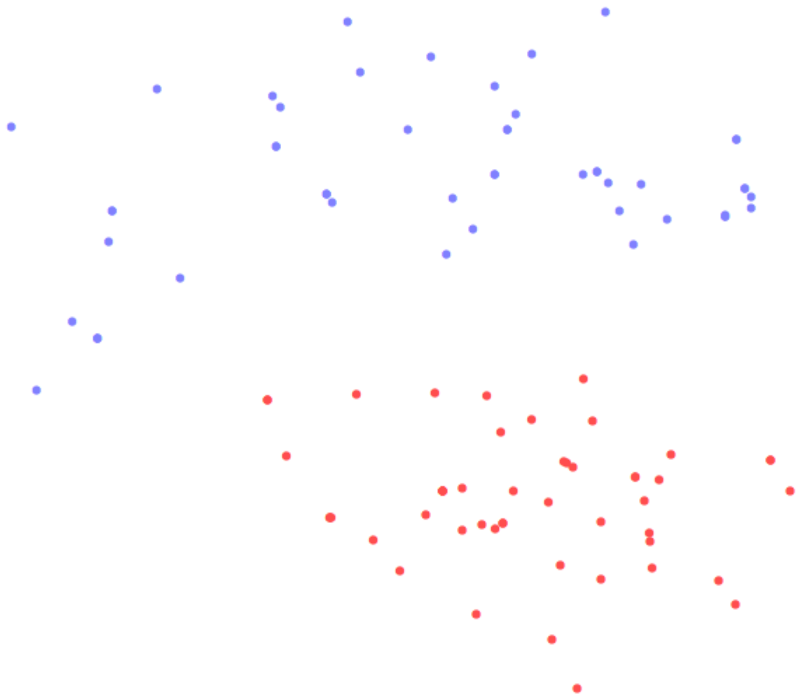
- We observe a sample of n observations from a given data source.
- If we observe a pattern in the data, it can reflect a property of the data source, or it can be a random effect.
- When we train a classifier, it should ideally adapt to properties of the source, but ignore random effects.
- We cannot distinguish the two cases without looking at another sample.

Sample Properties



- We observe a sample of n observations from a given data source.
- If we observe a pattern in the data, it can reflect a property of the data source, or it can be a random effect.
- When we train a classifier, it should ideally adapt to properties of the source, but ignore random effects.
- We cannot distinguish the two cases without looking at another sample.

Sample Properties



- We observe a sample of n observations from a given data source.
- If we observe a pattern in the data, it can reflect a property of the data source, or it can be a random effect.
- When we train a classifier, it should ideally adapt to properties of the source, but ignore random effects.
- We cannot distinguish the two cases without looking at another sample.

TRAINING AND TEST DATA

What happens if we measure the error rate on the training data?

- If the classifier has over-adapted to the idiosyncrasies of the training data, it will perform better on the training data then on new data from the same source.
- Estimates of error rates computed on the training data tend to *underestimate* the actual error rate.

Solution: Data splitting

- *Before* we train the classifier, we split the labeled data into two parts.
- We call these **training data** and **test data**.
- We use the training data to train the classifier.
- We then use the test data to estimate the error rate.

TRAINING AND TEST ERROR

Types of errors

- The error rate (or, more generally, the empirical risk) evaluated on the training data is called the **training error**.
- The error rate or empirical risk evaluated on the test data is the **test error**.

The distinction between these quantities is crucial.

Interpretation

- The training error measures how well the classifier fits the training data.
- The test error estimates how well the classifier predicts.
- Note this is an *estimate* rather than a *measurement*. Measuring the test error would require access to the data distribution. Since we do not have that distribution, we estimate the error from data.

Important

The test data must not be used for training in any way.

Once the training method has used *any* information extracted from the test data, the test error estimate is confounded.

$$\overbrace{(X_1, Y_1), \ldots, (X_n, Y_n)}^{\text{training}}, \quad \overbrace{(X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})}^{\text{test}}$$

$$\underbrace{(X_1, Y_1), \ldots, (X_n, Y_n)}_{\downarrow \; f}$$

Data

- Suppose a data source generates $n + m$ labelled data points.
- We split these into n training and m test points:

$$(X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})$$

- We assume that (X_i, Y_i) is independent of (X_j, Y_j), for $i \neq j$.
- That means the data are i.i.d., since they have the same distribution (the data source).

$$\overbrace{(X_1, Y_1), \ldots, (X_n, Y_n)}^{\text{training}}, \quad \overbrace{(X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})}^{\text{test}}$$

$$\underbrace{(X_1, Y_1), \ldots, (X_n, Y_n)}_{\downarrow \; f}$$

Using the data

- We train a classifier f on the training data. The classifier is obtained from the data by a deterministic procedure. Since the data is random, the classifier is random.

$$(X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})$$

- Since the test data is independent of the training data, the classifier is stochastically independent of the test data.
- That means an estimate of the classifier's error obtained from the test data is unbiased.
- If training uses *any* information from the test data, the classifier and the test data become dependent.
- Typically, the effect of this dependence is that the test error *systematically underestimates* the actual prediction error on data from the source.

TREE CLASSIFIERS

Idea

- Recall: Classifiers classify according to location in $\mathbb{R}^d$
- Linear classifiers: Divide space into two halfspaces
- What if we are less sophisticated and divide space only along axes? We could classify e.g.

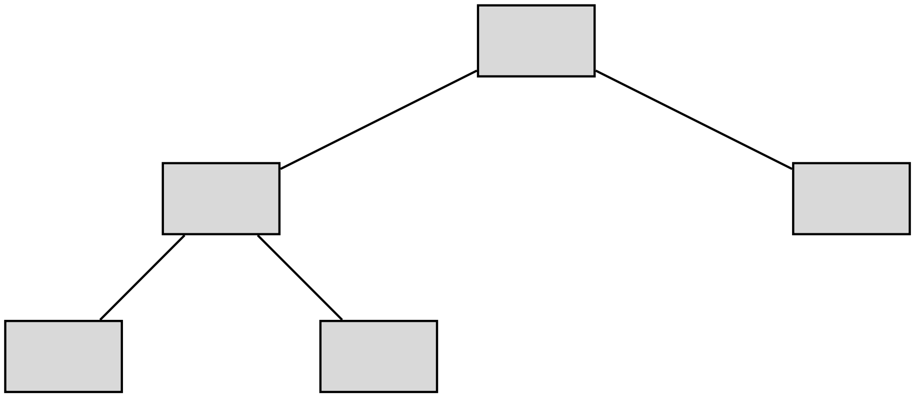
$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \qquad \text{according to} \qquad \mathbf{x} \in \begin{cases} \text{Class +} & \text{if } x_3 > 0.5 \\ \text{Class -} & \text{if } x_3 \leq 0.5 \end{cases}$$

- This decision would correspond to an affine hyperplane perpendicular to the x_3-axis, with offset 0.5.

Tree classifier

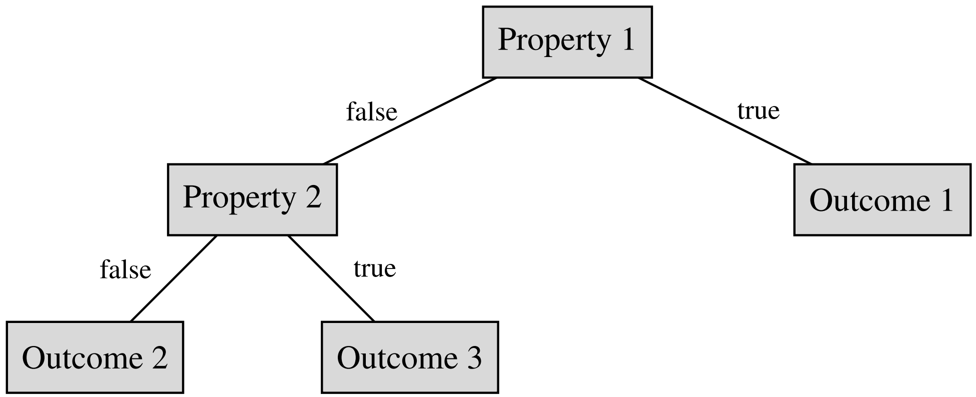
- A tree classifier combines several simple decision rules as the one above into a classifier using a so-called *decision tree*.

TREES



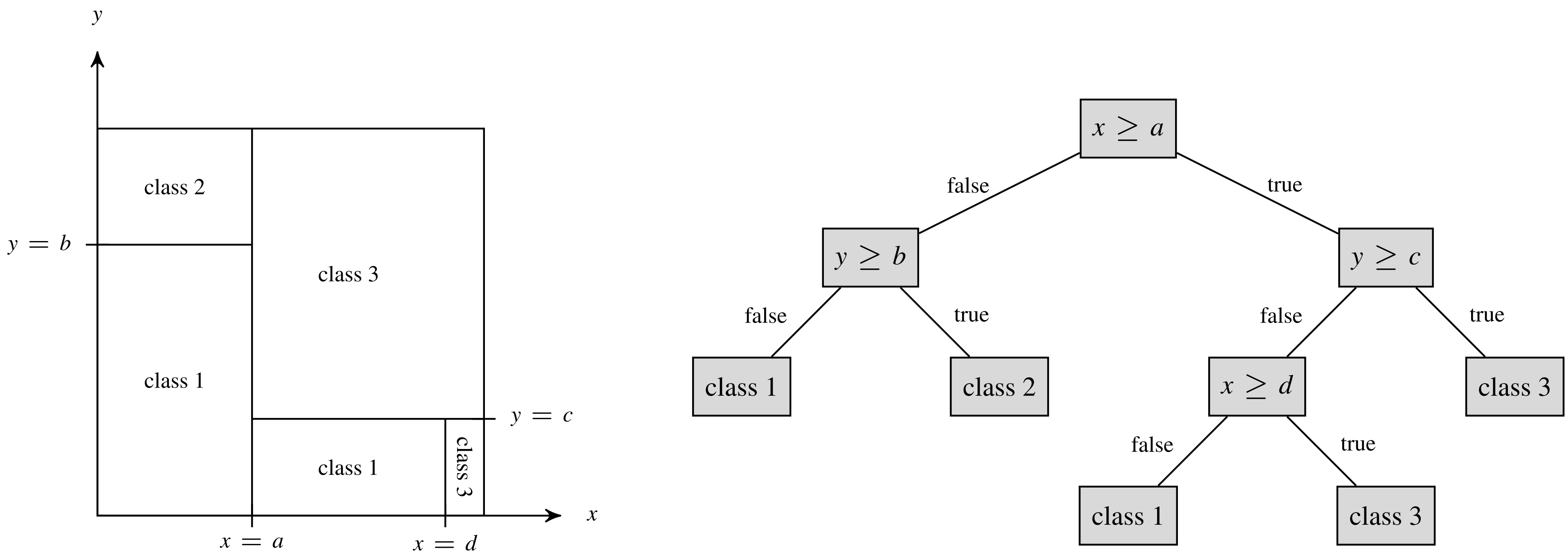
- A **tree** is a diagram consisting of **nodes** (marked as gray boxes above) and **edges** (the connecting lines).
- The topmost node is called the **root**. Each (except the root) is connected to exactly one node above it, called its **parent**.
- Nodes can be connected other nodes below them, called their **children**.
- Nodes at the bottom (those with no children) are called **leaves**.
- If each node has either two or no children, the tree is called a **binary tree**.

DECISION TREES



- Binary trees can be used as decision diagrams.
- Each inner node (a node that is not a leaf) represents a property.
- The two children of the node represent the cases “property is false” or “property is true”.
- Each leaf represents an outcome. That means: An outcome is a combination of true and false properties.
- Such a tree is called a **decision tree**.

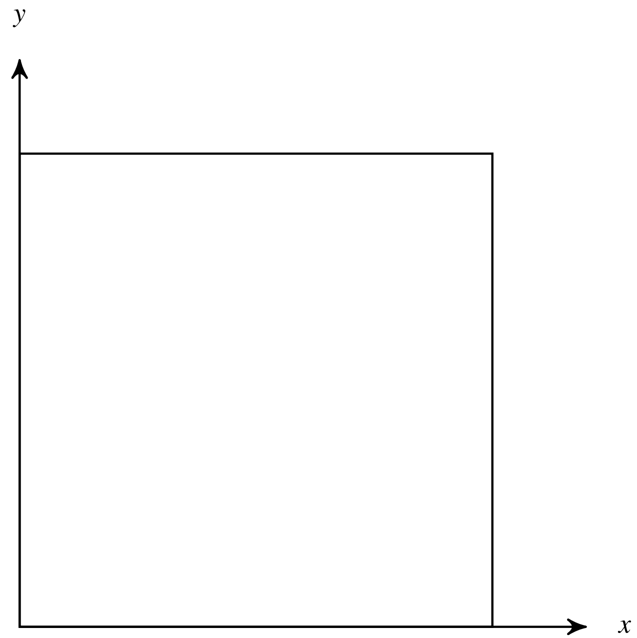
TREE CLASSIFIER



- A **tree classifier** in $\mathbb{R}^d$ is a decision tree.
- Each property at an inner node corresponds to a decision of the form $x_j \geq c$, where $j \in \{1, \ldots, d\}$ is one of the coordinates, and $c \in \mathbb{R}$ is a constant.
- Each leaf corresponds to a class.
- We classify a data point $\mathbf{x} \in \mathbb{R}^d$ by starting at the root, and following the decisions through the diagram until we reach a leaf. We then assign $\mathbf{x}$ to the class inscribed at that leaf.

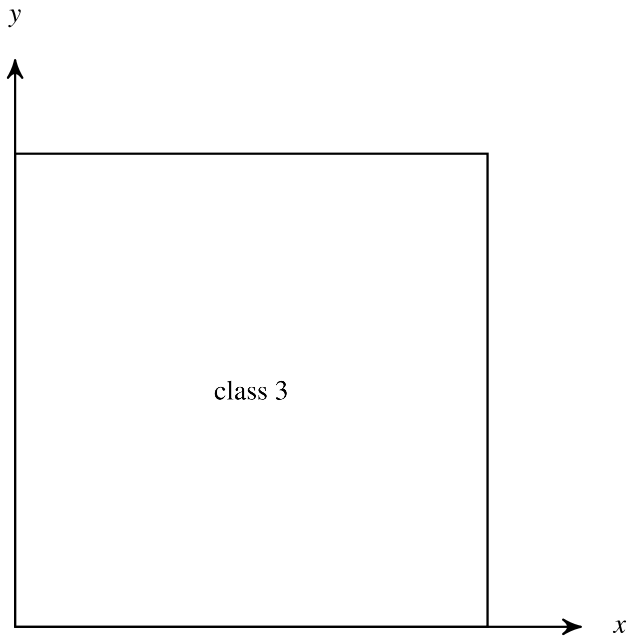
Growing a Tree

Example: Data in quadratic domain, three classes, class 3 is the largest class.



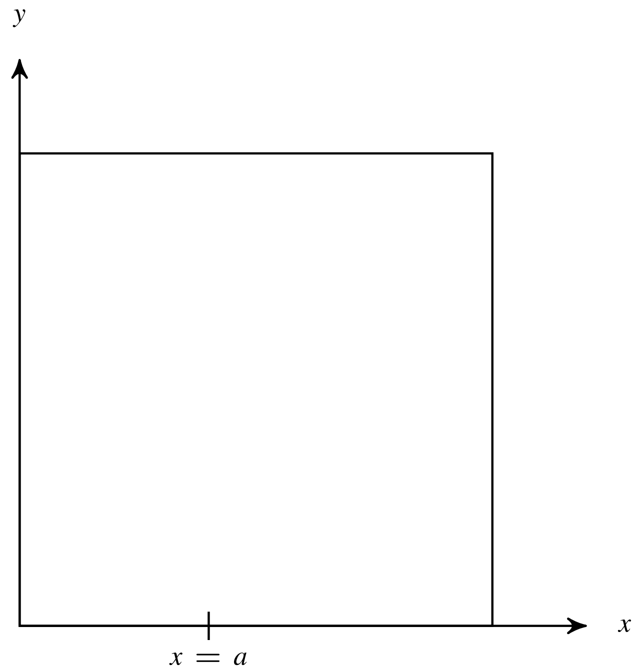
Growing a Tree

Example: Data in quadratic domain, three classes, class 3 is the largest class.



GROWING A TREE

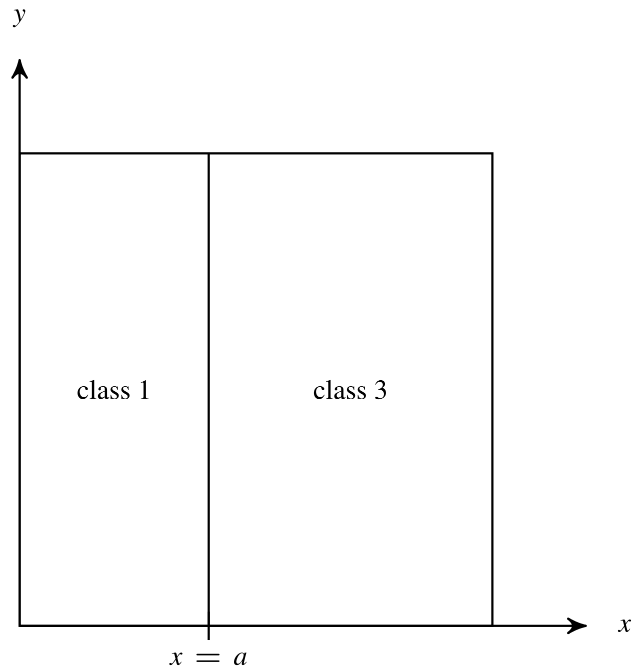
Example: Data in quadratic domain, three classes, class 3 is the largest class.

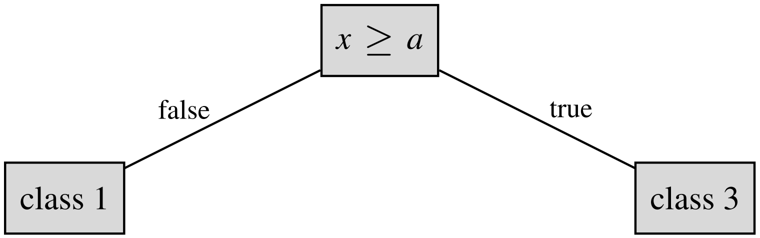


$x \geq a$

Growing a Tree

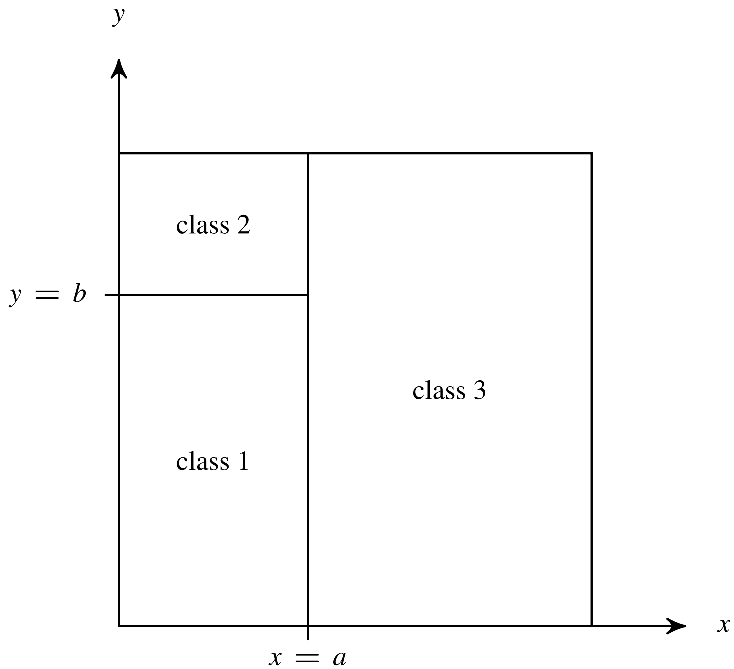
Example: Data in quadratic domain, three classes, class 3 is the largest class.

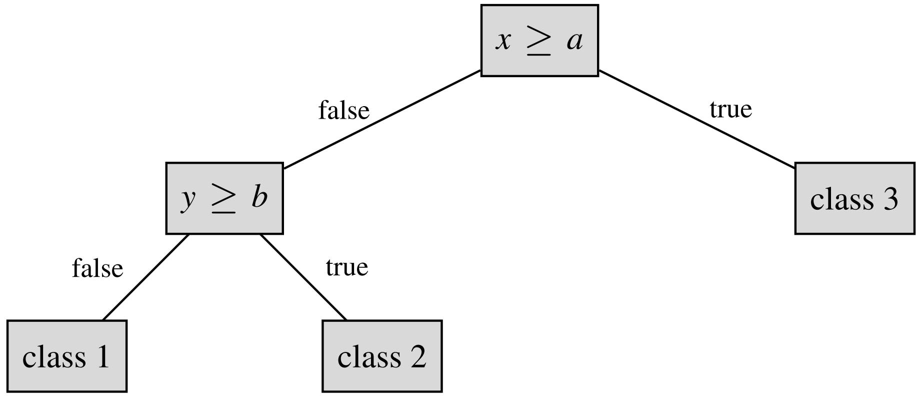




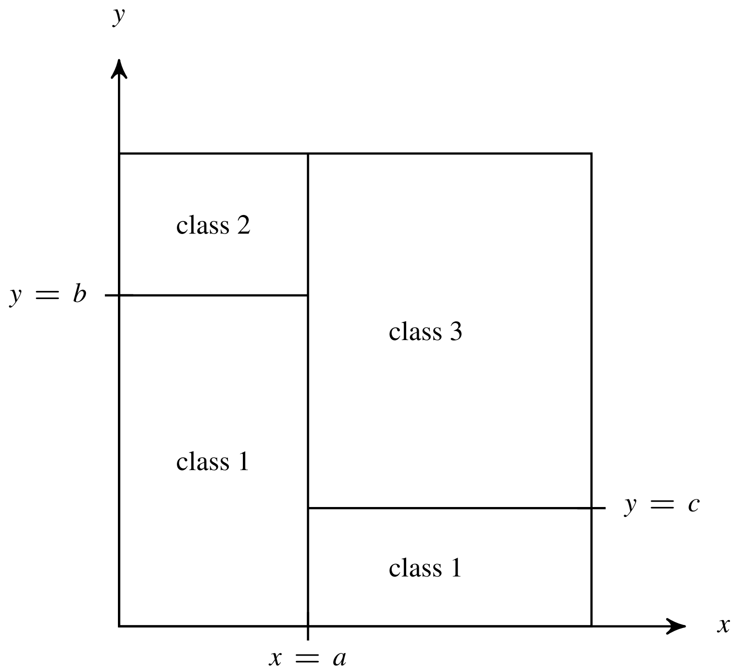
GROWING A TREE

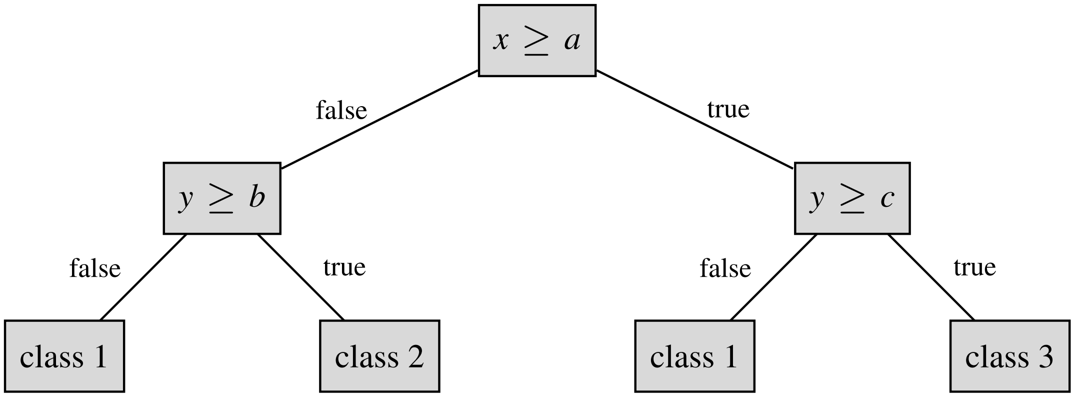
Example: Data in quadratic domain, three classes, class 3 is the largest class.





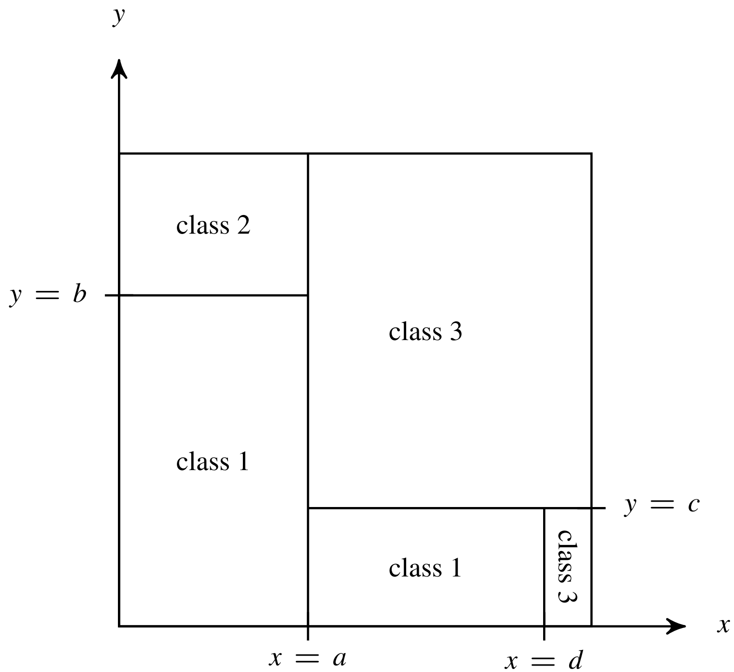
Example: Data in quadratic domain, three classes, class 3 is the largest class.

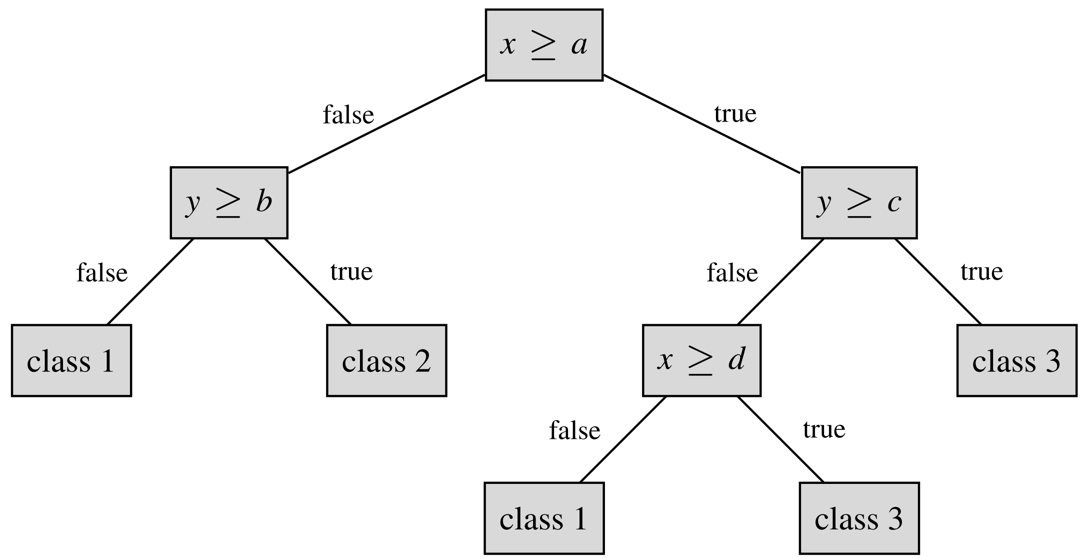




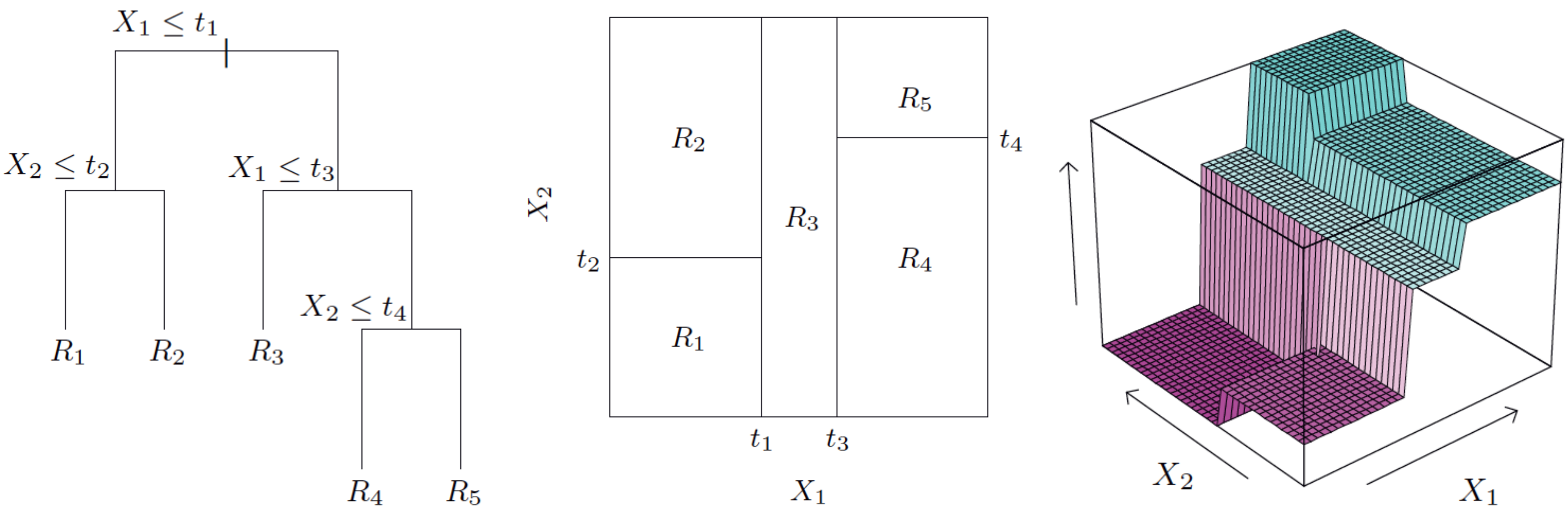
Growing a Tree

Example: Data in quadratic domain, three classes, class 3 is the largest class.





TREES



- Each leaf of the tree corresponds to a region R_m of $\mathbb{R}^d$.
- Each inner node represents the union of all regions represented by nodes below it.
- Classes $k \in \{1, \ldots, K\}$ (not restricted to two classes).
- The tree represents a function $f : \mathbb{R}^d \to \{1, \ldots, K\}$. If we plot that function, it is piece-wise constant on rectangular regions in $\mathbb{R}^2$, or on "box-shaped" regions in $\mathbb{R}^d$.

Figures: Hastie, Tibshirani, Friedman, "The elements of statistical learning"

Approach

The basic strategy is very simple:

- At each step, decide where to place the next split.
- That replaces one of the regions represented by the tree by two new regions.
- Assign each new region by majority vote among the training data points in that region.

Where do we split?

We have to decide:

- Which region should be split.
- Along which axis.
- At which split point.

Idea: Find the split that results in the largest reduction in training error.

Finding a split point

Cost of a split

- Suppose we split region $\mathcal{R}_m$ along axis j at value t.
- That results in two new regions, say $\mathcal{R}_m^1$ and $\mathcal{R}_m^2$.
- In the tree, that means we replace the node $\mathcal{R}_m$ by the criterion $x_n \geq t$, and add $\mathcal{R}_m^1$ and $\mathcal{R}_m^2$ as child nodes.
- We define the **cost** of this split as

$$\text{cost}(m, j, t) := \quad \# \text{ of misclassified points in } \mathcal{R}_m^1 \quad + \quad \# \text{ of misclassified points in } \mathcal{R}_m^2$$

  (That means: We assign $\mathcal{R}_m^1$ and $\mathcal{R}_m^2$ class labels by majority vote, and check how many training points are misclassified by these class labels.)

Training a tree classifier

- For each region m and each axis j, find the split point t that minimizes $\text{cost}(m, j, t)$,

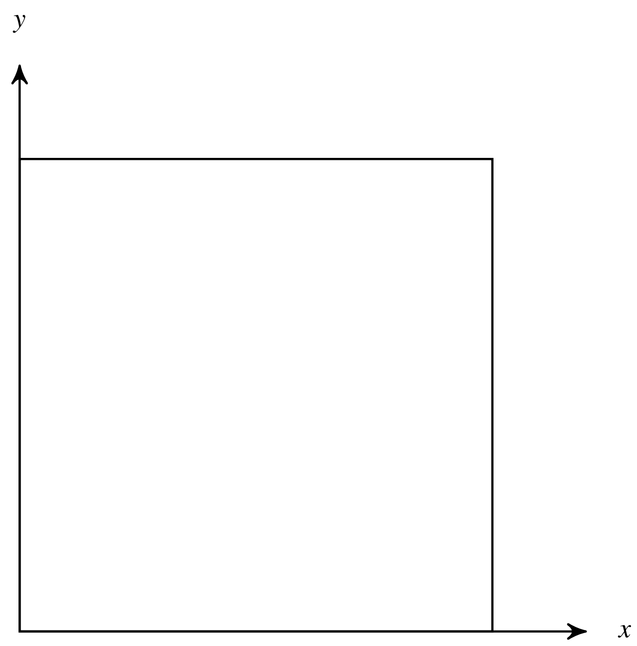
$$t_{mj} := \arg\min_{t \in \mathbb{R}} \text{cost}(m, j, t) \ .$$

- From the list of all such points t_{mj}, pick the one with the smallest cost.
- Perform that split.

We keep doing so until the number of regions m reaches some specified, maximal value M.

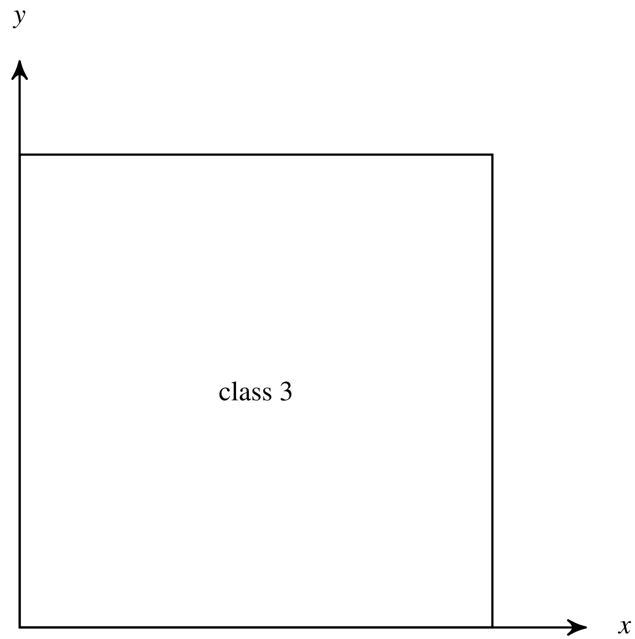
TRAINING

Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.



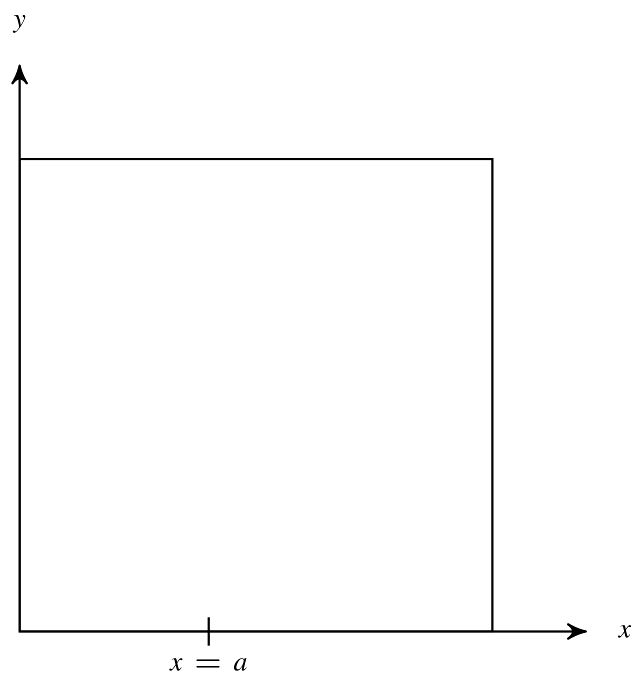
TRAINING

Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.



TRAINING

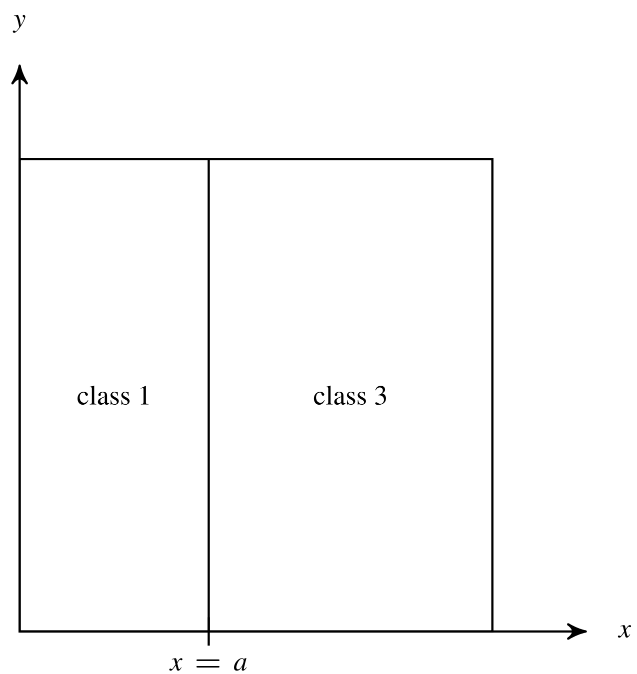
Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.

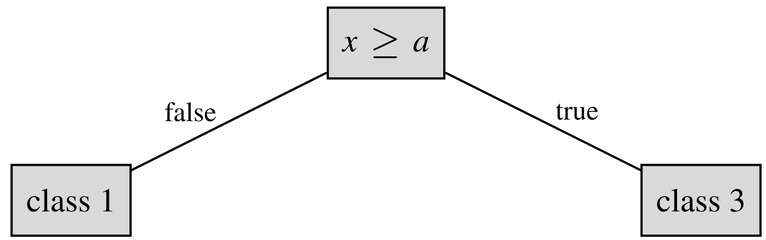


$$\boxed{x \geq a}$$

TRAINING

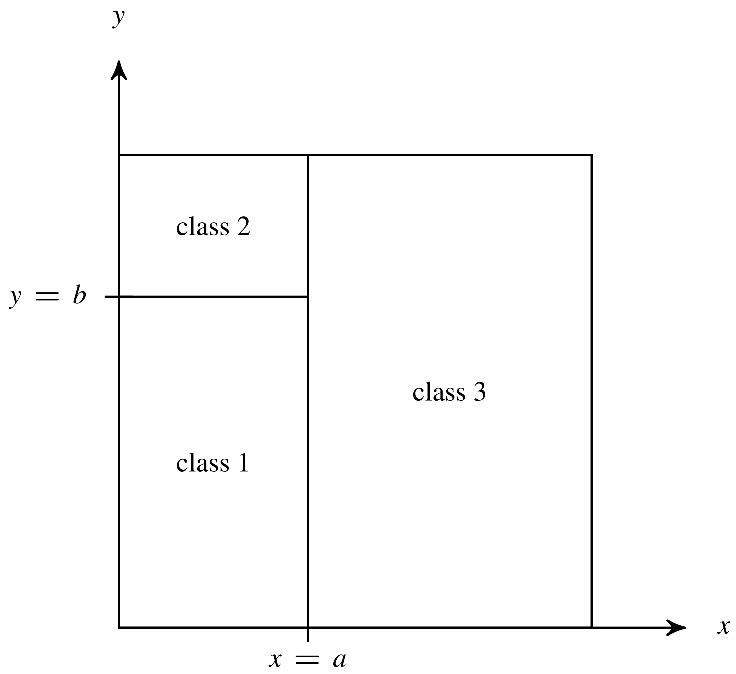
Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.

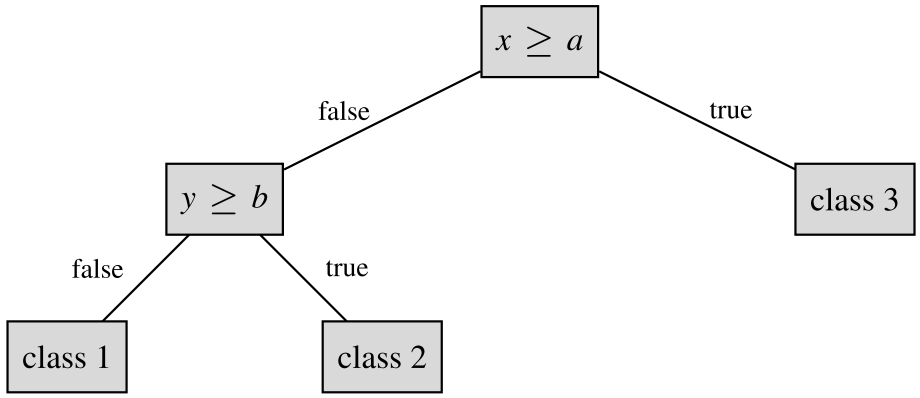




TRAINING

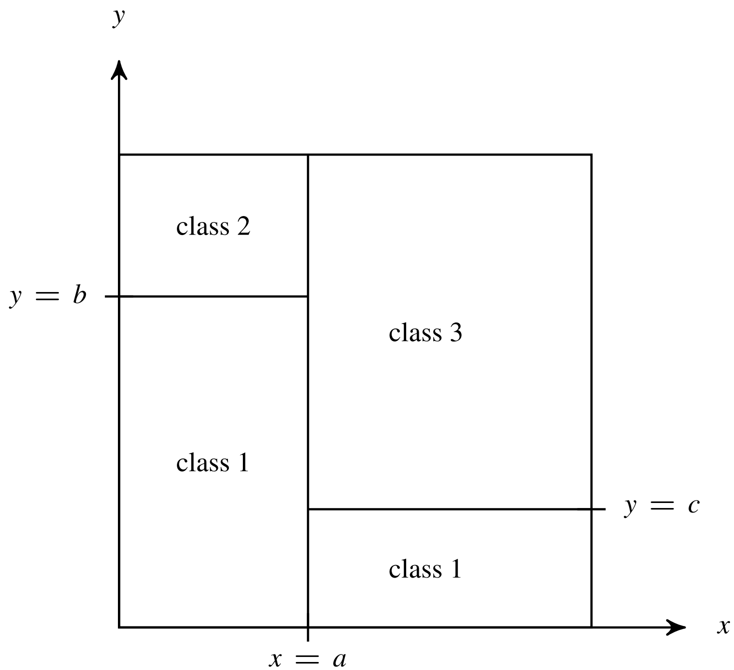
Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.

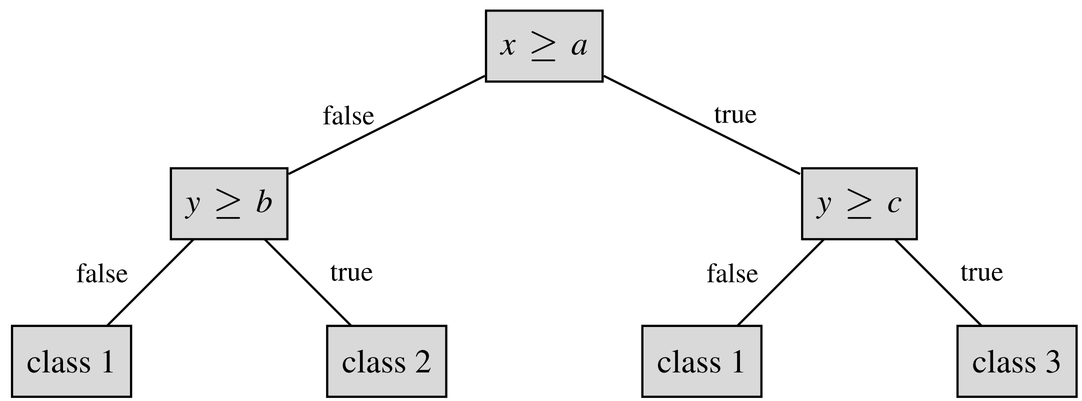




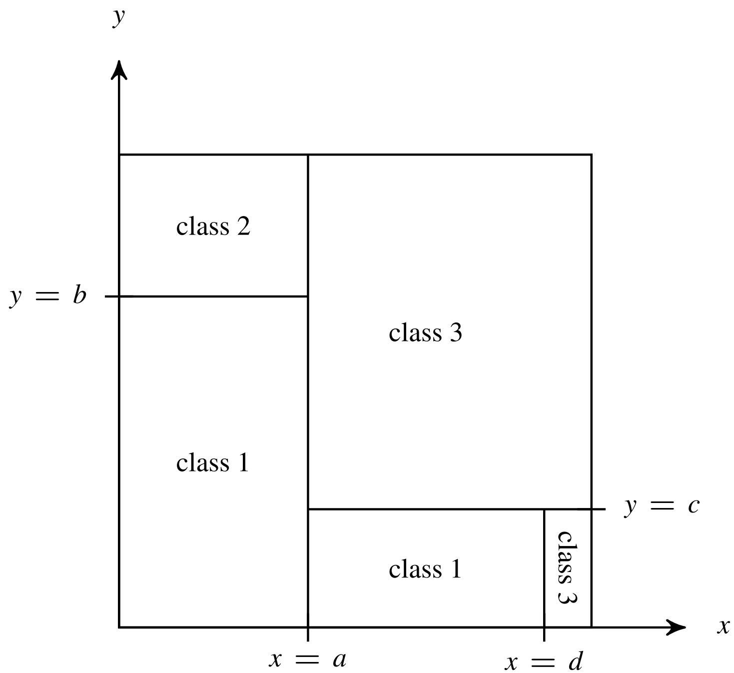
TRAINING

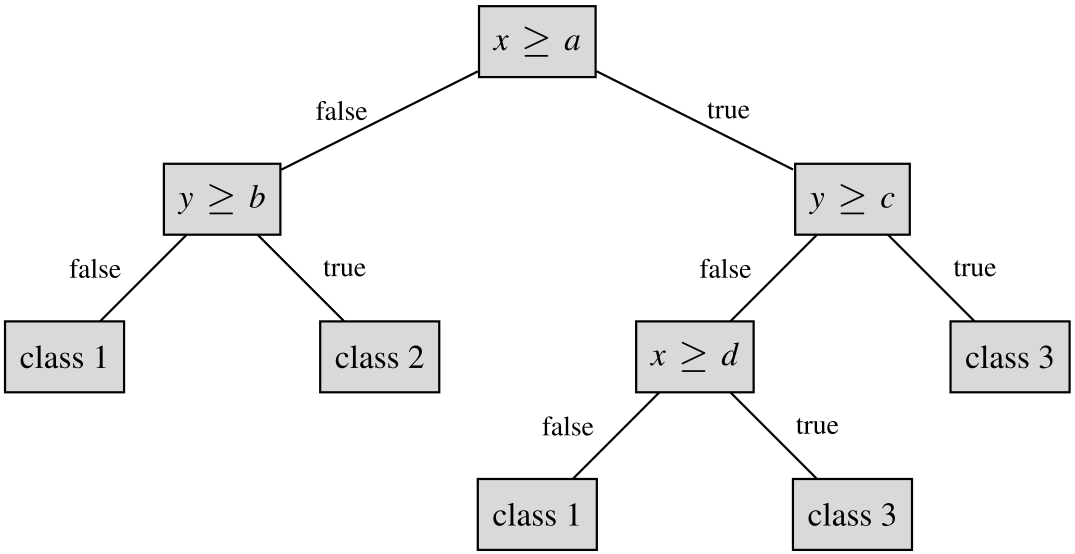
Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.





Example: Data in quadratic domain, three classes, class 3 is the largest class. We specify the maximum number of regions as $M = 5$.





EXAMPLE: SPAM FILTERING

Data

- 4601 email messages
- Classes: email, spam

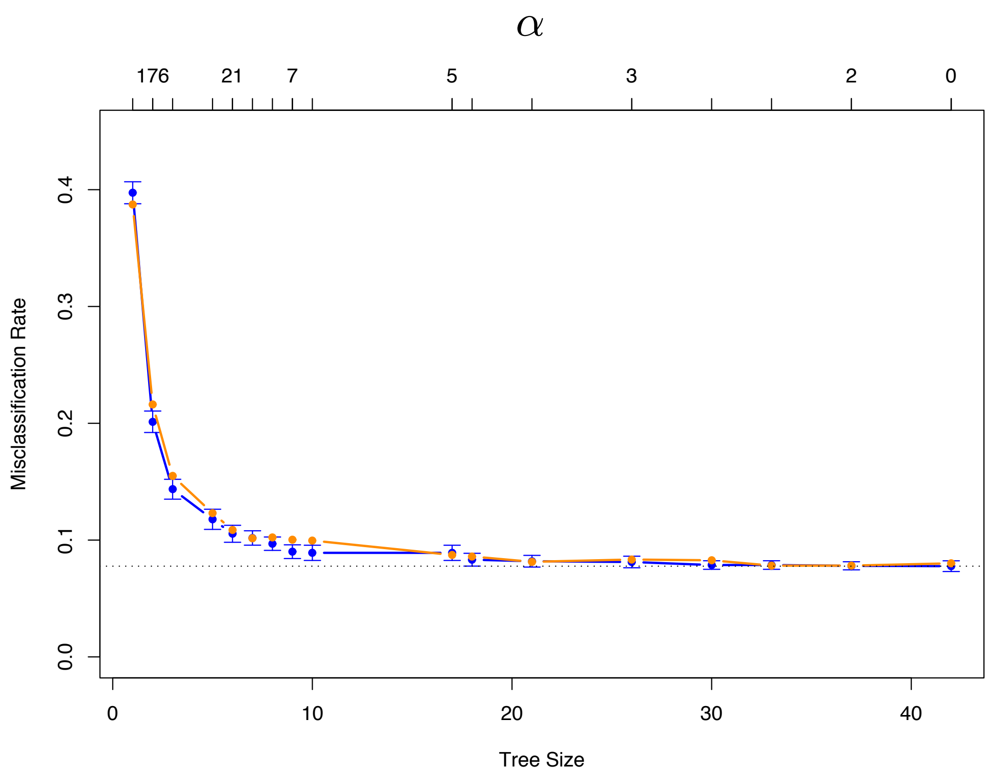
	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

Tree classifier

- 17 nodes
- Performance:

	Predicted	
True	Email	Spam
Email	57.3%	4.0%
Spam	5.3%	33.4%

Figures: Hastie, Tibshirani, Friedman, "The elements of statistical learning"

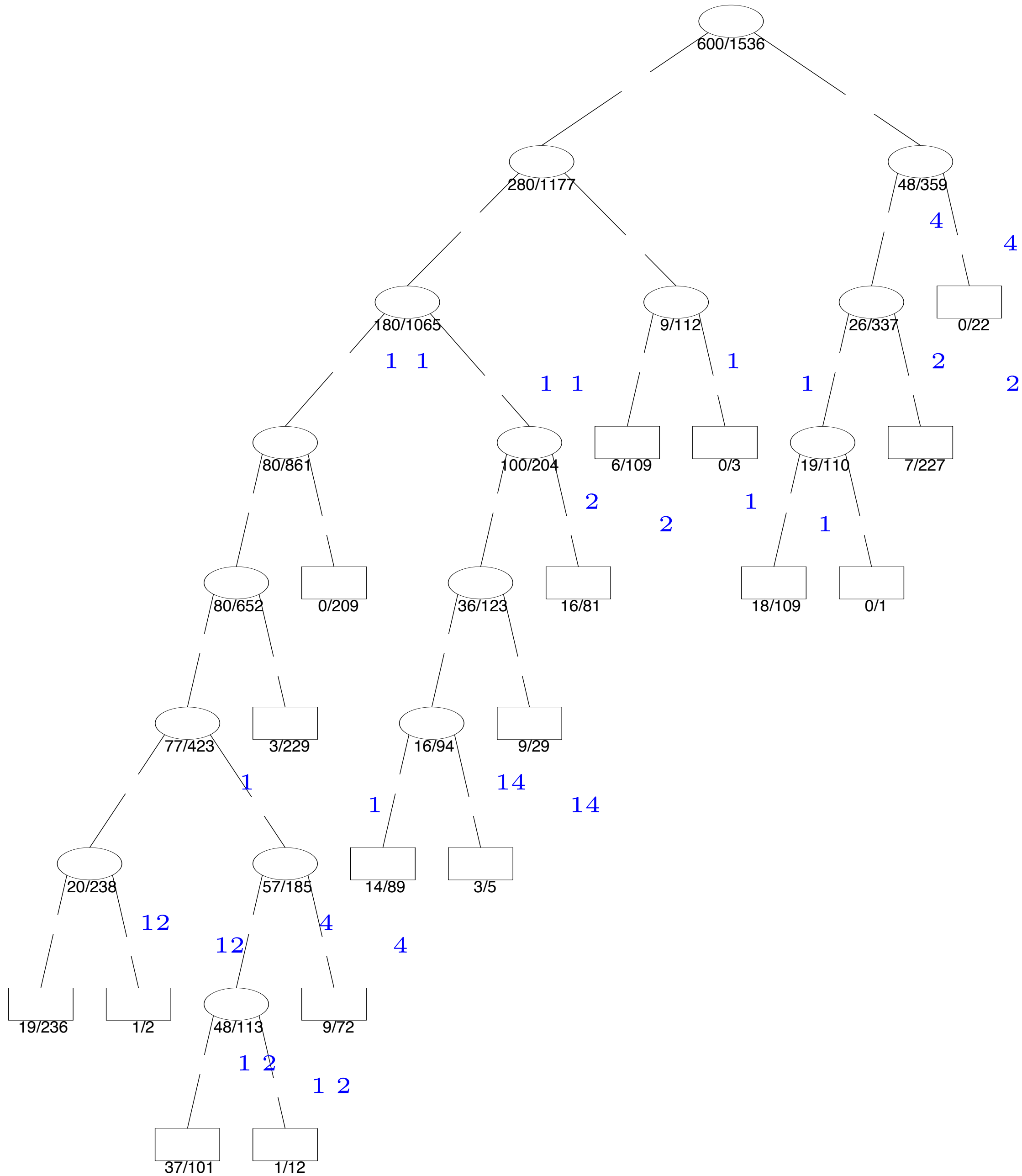


Tree Size

- Complete tree of height D defines 2^D regions.
- D too small: Insufficient accuracy. D too large: Overfitting.
- D can be determined by cross validation or more sophisticated methods ("complexity pruning" etc), which we will not discuss here.

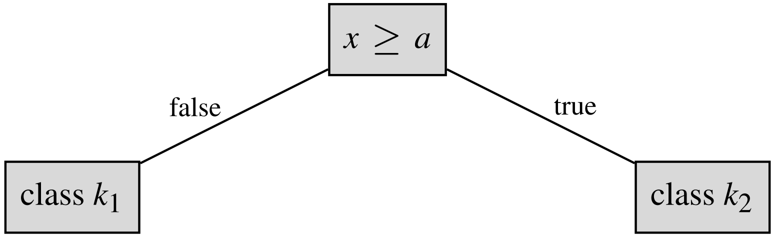
Figures: Hastie, Tibshirani, Friedman, “The elements of statistical learning”

SPAM FILTERING: TREE



Figures: Hastie, Tibshirani, Friedman, "The elements of statistical learning"

DECISION STUMPS



- The simplest possible tree classifier is a tree of depth 1. Such a classifier is called a **decision stump**.
- A decision stump is parameterized by a pair (j, t_j) of an axis j and a splitting point t_j.
- Splits $\mathbb{R}^d$ into two regions.
- Decision boundary is an affine hyperplane which is perpendicular to axis j and intersects the axis at t_j.
- Decision stumps are often used in so-called *ensemble methods*. These are algorithms that combine many poor classifiers into a good classifier. We will discuss ensemble methods later.