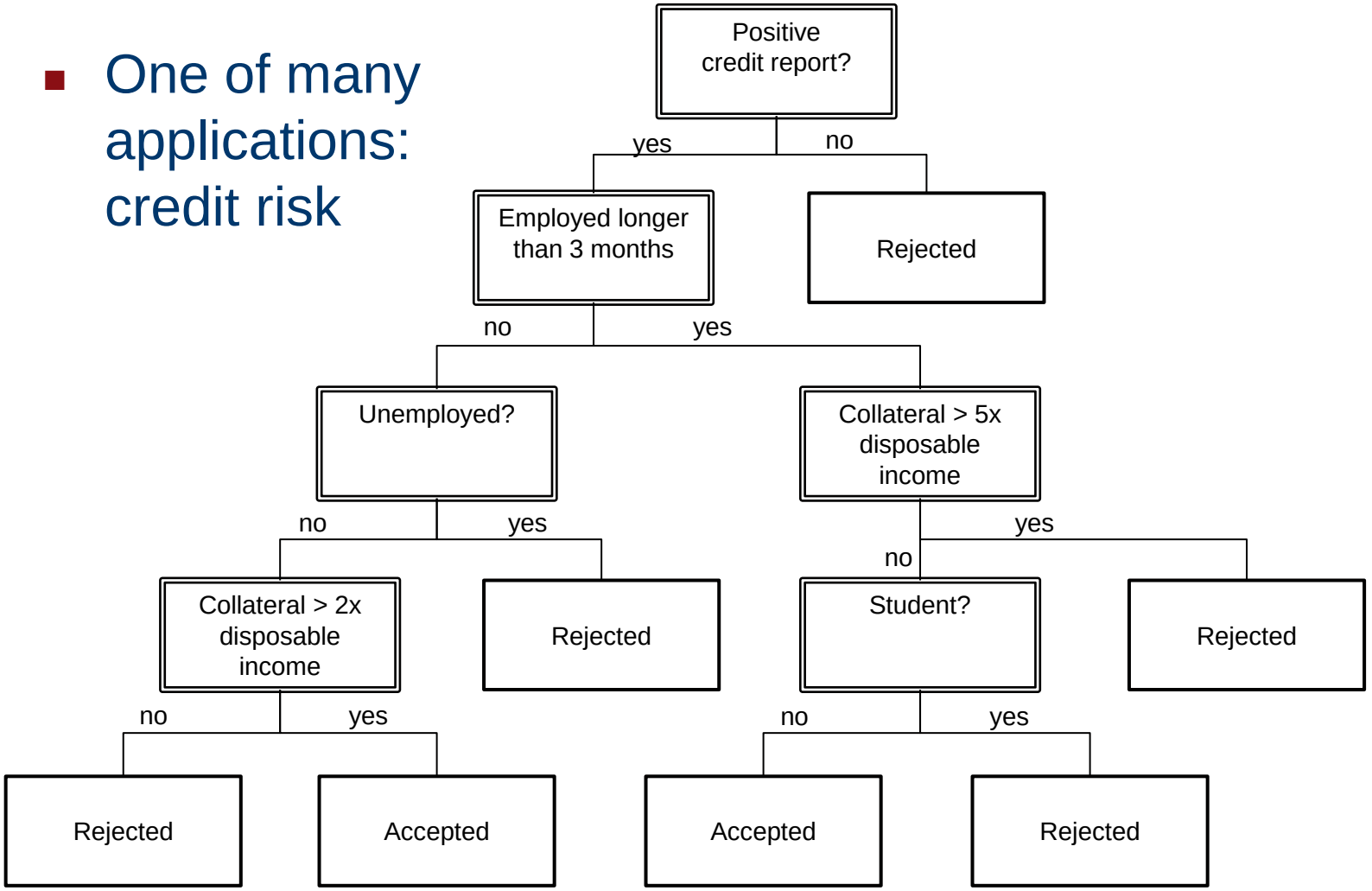
Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen


# Decision Trees

Tobias Scheffer

# Decision Trees

- One of many applications: credit risk



# Decision Trees – Example

- Head tracking and face recognition.

# Overview

- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Decision Trees – Why?

- Simple to interpret.
  - Provides a classification plus a justification for it.
  - “Rejected, because subject has been employed less than 3 months and has collateral < 2 x disposable income“.
- Fast inference:
  - For each instance, only one branch has to be traversed.
  - Frequently used in image processing.
- Simple, efficient, scalable learning algorithm.
- Complex nonlinear models.
- Works for classification and regression problems.

# Classification

- Input: instance $\mathbf{x} \in X$.
  - Instances are represented as a vector of attributes.
  - An instance is an assignment to the attributes.
  - Instances will also be called feature vectors.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$$

e.g. features like color, test results,…

- Output: class $y \in Y$; finite set $Y$.
  - e.g., {accepted, rejected}; {spam, not spam}.
  - The class is also referred to as the target attribute.

# Classifier learning

- Input: training data.
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \rangle$
  - $\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{n1} \end{pmatrix}$
- Output: Classifier.
  - $f : X \to Y$

e.g. a decision tree:
path along the edges to a leaf
provides the classification

# Regression

- Input: instance $\mathbf{x} \in X$.
  - e.g., feature vector

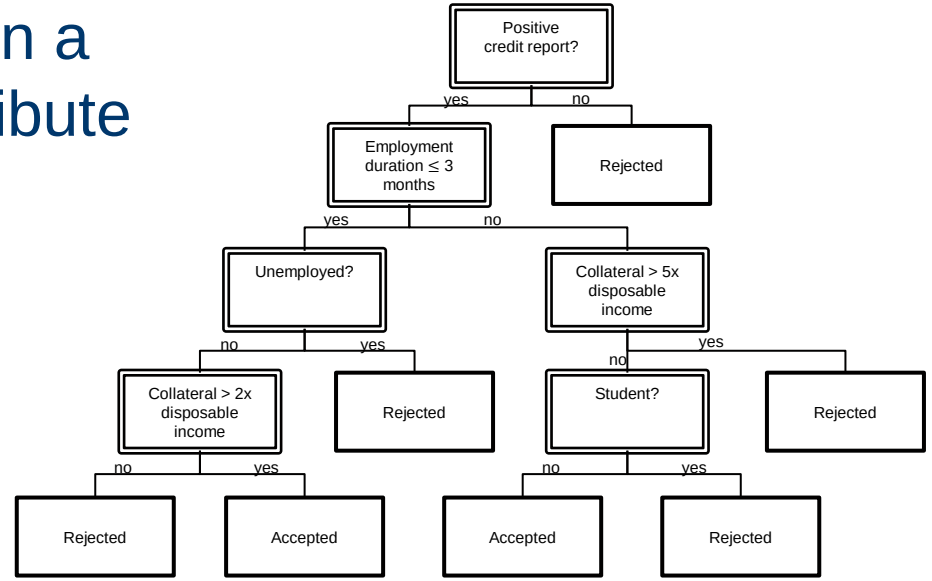
$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$$

- Output: continuous (real) value, $y \in \mathbb{R}$

- Learning problem: training data with continuous target value
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \rangle$
  - e.g. $\langle (\mathbf{x}_1, 3.5\ ), \ldots, (\mathbf{x}_n, -2.8\ ) \rangle$

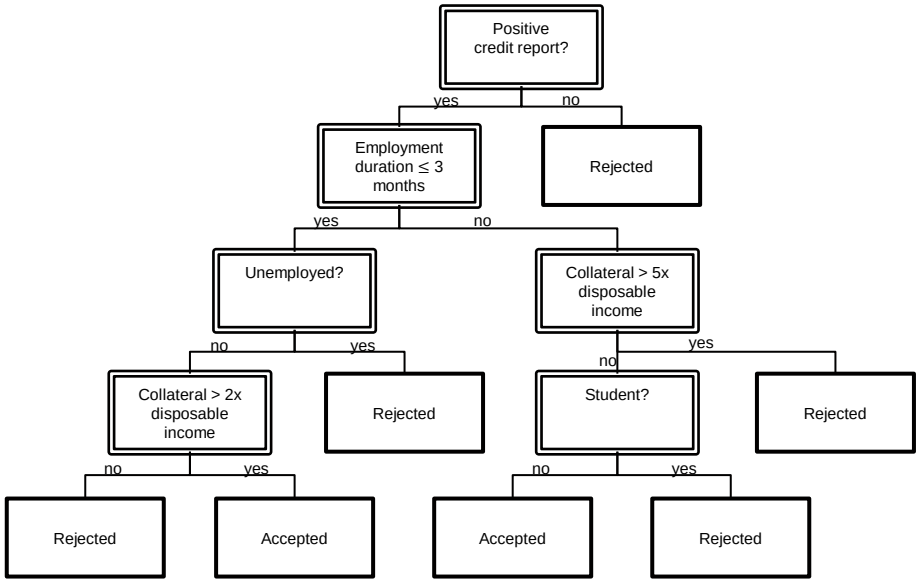
# Decision Trees

- Test nodes:
  - Discrete attributes: node contains an attribute, branches are labeled with values.
  - Continuous attributes; nodes contain “≤” comparisons, branches are labeled “yes” and “no”.
- Terminal nodes contain a value of the target attribute



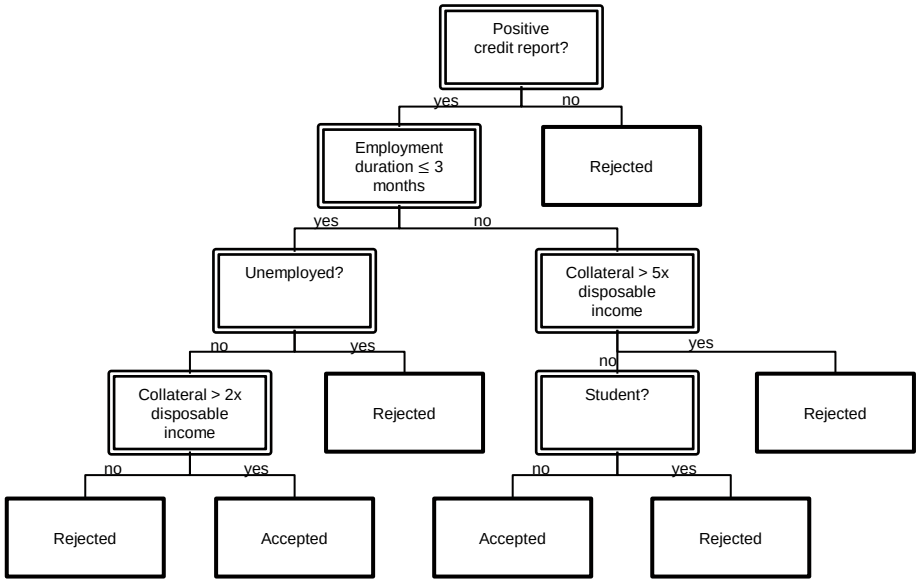
# Decision Trees

- Decision trees can be represented as decision rules, each terminal node corresponds to a rule.
  - E.g., Rejected ← positive credit report ∧ employment duration ≤ 3 months ∧ unemployed.



# Application of Decision Trees

- Recursively descent along branch.
- In each test node, conduct test, choose the appropriate branch.
- In a terminal node, return the value.



# Learning Decision Trees

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

- Find a decision tree that predicts the correct class for the training data.
- Trivial way: create a tree that merely reproduces the training data.

# Learning Decision Trees

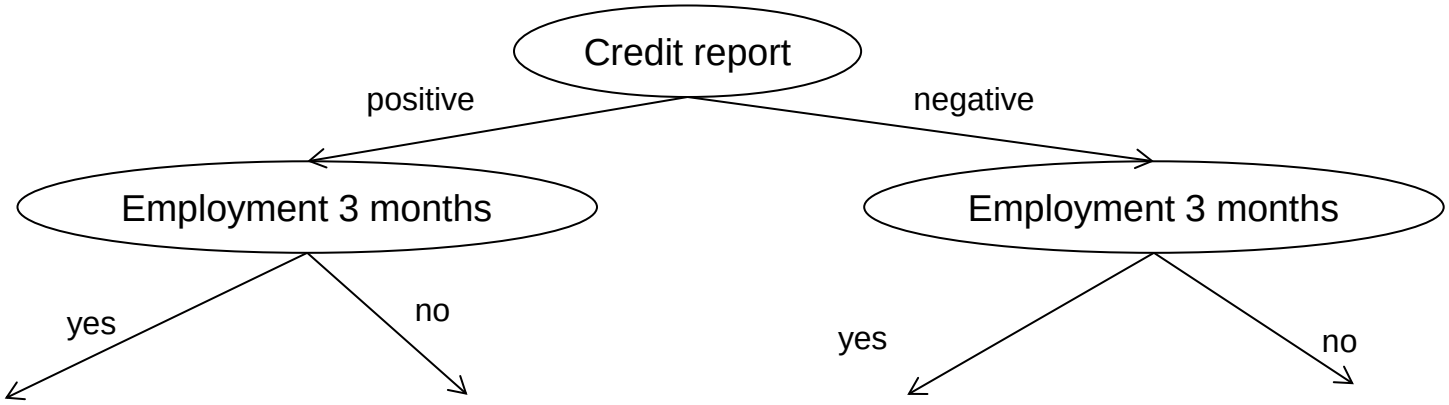
| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

# Learning Decision Trees

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

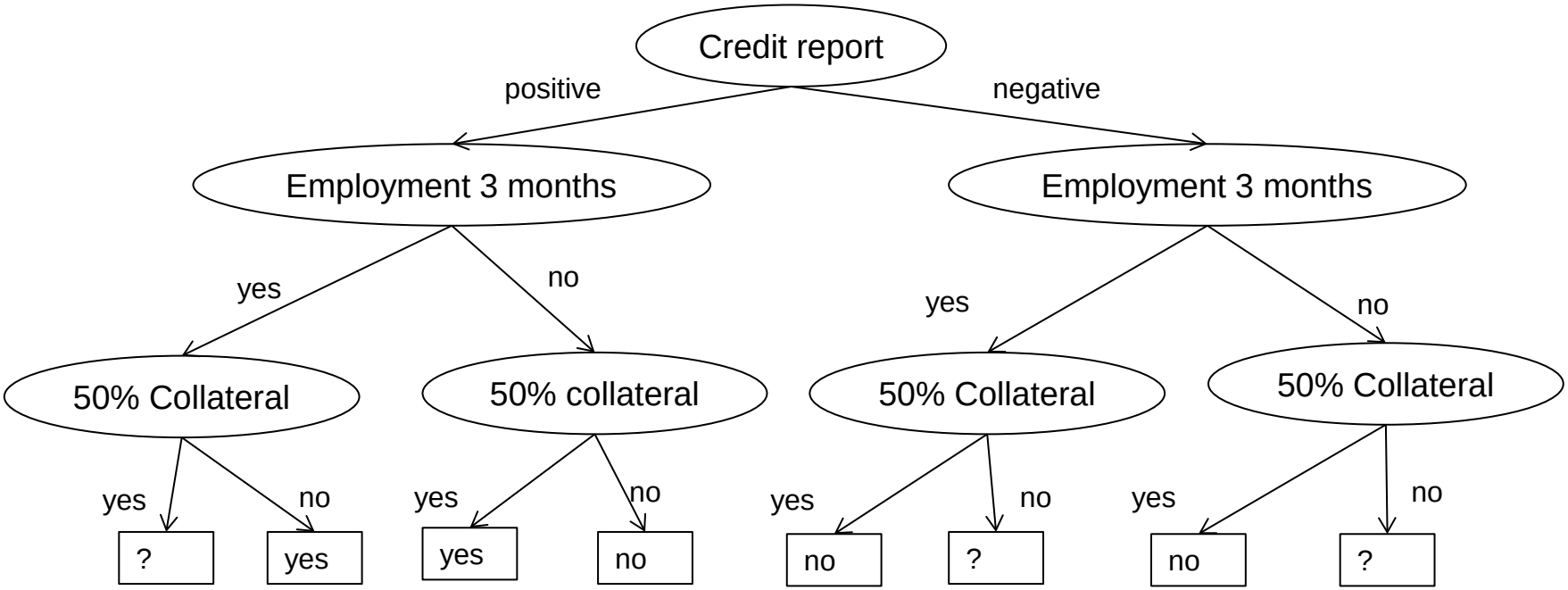
# Learning Decision Trees

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |



# Learning Decision Trees

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |



# Learning of Decision Trees

- Perhaps more elegant: From the trees that are consistent with the training data, choose the smallest (as few nodes as possible).
- Small trees can be good because:
  - they are easier to interpret;
  - There are more training instances per leaf node. Hence, the class decision in each leaf is better substantiated.

# Complete Search for the Smallest Tree

- How many functionally different decision trees are there?
  - Assume $m$ binary attributes and two classes.

- What is the complexity of a complete search for the smallest tree?

# Complete Search for the Smallest Tree

- How many functionally different decision trees are there?
  - Assume $m$ binary attributes and two classes.
  - Tree has $m$ layers of test nodes $\rightarrow 2^m - 1$ test nodes.
  - $2^{(2^m)}$ assignments of classes to leaf nodes.

- What is the complexity of a complete search for the smallest tree?
  - Assignments of $m$ attributes (or node can be missing) to $2^m - 1$ test nodes: $O\left((m+1)^{2^m-1}\right)$.
  - Assignments of class labels to leaf nodes: $O(2^{(2^m)})$

# Overview

- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Learning of Decision Trees

- Greedy algorithm that finds a small tree (instead of the smallest tree) but is polynomial in the number of attributes.
- Idea for Algorithm?

# Greedy Algorithm – Top-Down Construction

1. ID3(L)
    1. If all data in L have same class y, then return leaf node with class y.

ID3(

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |

) = yes

# Greedy Algorithm – Top-Down Construction

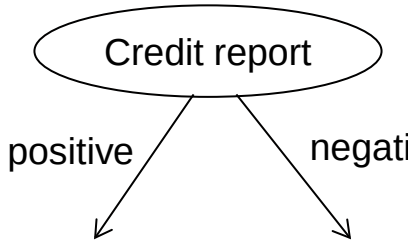
1. ID3(L)
   1. If all data in L have same class y, then return leaf node with class y.
   2. Else
      1. Choose attribute $x_j$ that separates L into subsets $L_1, \ldots, L_k$ with most homogenous class distributions.
      2. Let $L_i = \{(x, y) \in L : x_j = i\}$.
      3. Return test node with attribute $x_j$ and children ID3($L_1$,), …, ID3($L_k$).

ID3(

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

) =

2 x payed back
3 x not payed back




ID3(

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |

)

2 x payed back
1 x not payed back

ID3(

| Loan | Credit report | Employment last 3 months | Collateral > 50% loan | Payed back in full |
|---|---|---|---|---|
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

)

2 x not payed back

# Greedy Algorithm – Top-Down Construction

ID3(L)

1. If all data in L have same class y, then return leaf node with class y.
2. Else
   1. Choose attribute $x_j$ that separates L into subsets $L_1, \ldots, L_k$ with most homogenous class distributions.
   2. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   3. Return test node with attribute $x_j$ and children ID3($L_1$,), …, ID3($L_k$).

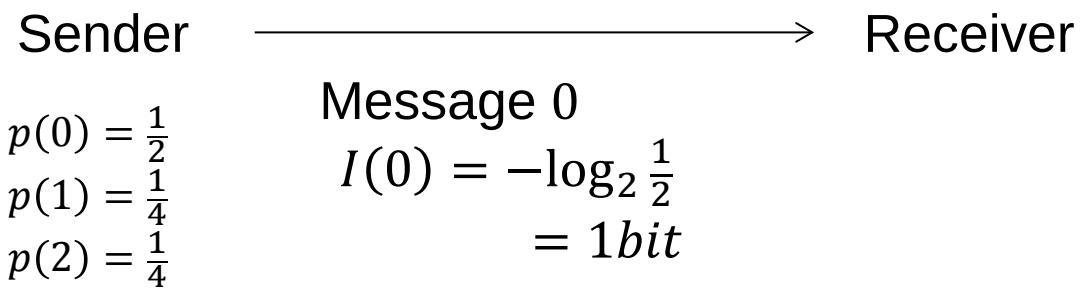
What does it mean that an attribute splits the sample into subsets with homogenous class distributions?

# Information

- Information theory uses models that involve a sender, a channel, a receiver, and a probability distribution over messages.
- Information is a property of messages, measured in units of bit.
- Information in a message (rounded) = number of bits needed to code that message in an optimal code.

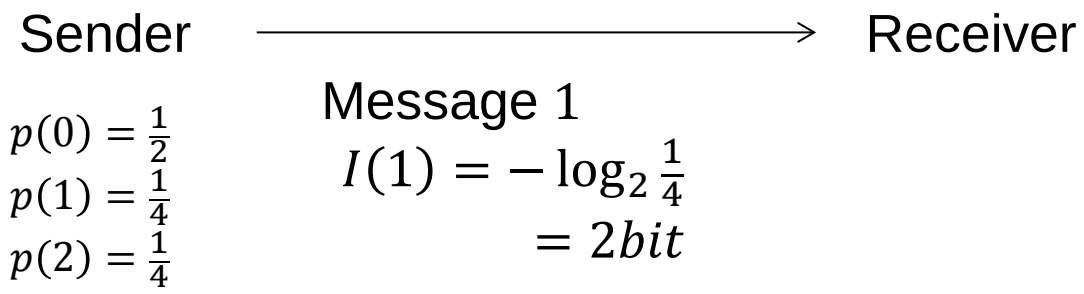
# Information

- Information theory uses models that involve a sender, a channel, a receiver, and a probability distribution over messages.
- The information in a message $y$ that is sent with probability $p(y)$ is $-\log_2 p(y)$.



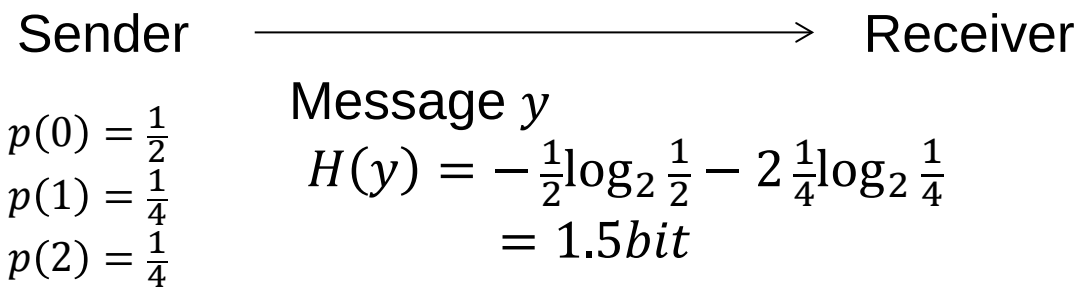
# Information

- Information theory uses models that involve a sender, a channel, a receiver, and a probability distribution over messages.
- The information in a message $y$ that is sent with probability $p(y)$ is $-\log_2 p(y)$.



# Entropy

- Entropy is the expected information of a message.
  - $H(y) = -\sum_{v=1}^{k} p(y = v) \log_2 p(y = v)$
- Entropy quantifies the receiver‘s uncertainty regarding the message
- Empirical entropy $H_L$: use frequencies observed in data $L$ instead of probabilities.



# Entropy of Class Labels in Training Data

- Information / uncertainty of the class labels = expected number of bits needed to send message about class label of an instance to a receiver.

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

$$H_L(y) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97bit$$

# Conditional Entropy

- Information / uncertainty of the class labels under some condition on the features.

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

$$H_L(y|x_1 = n) = -\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{5}\log_2\frac{0}{5} = 0bit$$

# Conditional Entropy

- Information / uncertainty of the class labels under some condition on the features.

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

$$H_L(y|x_1 = n) = -\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{5}\log_2\frac{0}{5} = 0bit$$
$$H_L(y|x_1 = p) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.91bit$$

# Information Gain of an Attribute

- Reduction of entropy by splitting the data along an attribute
  - $G_L(x_j) = H_L(y) - \sum_{v=1}^{k} p_L(x_j = v) H_L(y|x_j = v)$

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

- $G_L(x_1) = H_L(y) - p_L(x_1 = p) H_L(y|x_1 = p) - p_L(x_1 = n) H_L(y|x_1 = n)$
  $= 0.97 - \frac{3}{5} 0.91 - \frac{2}{5} 0 = 0.42 bit$
- Splitting along $x_1$ reduces the uncertainty regarding the class label $y$ by 0.42 bit.

# Information Gain Ratio

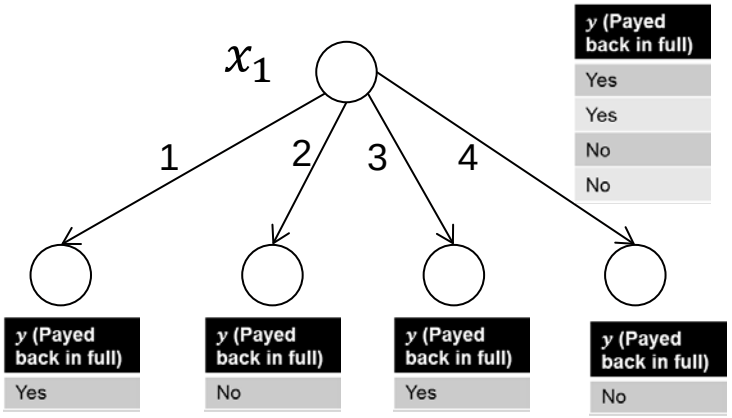
- Motivation:
  - Predicting whether a student will pass an exam.
  - How high is the information gain of the attribute "Matriculation number"?
- Information gain favors attributes with many values.
- Not necessarily an indicator of good generalization.
- Idea: factor the information that is contained in the attribute values into the decision
  - $H_L(x_j) = -\sum_{v=1}^{k} p_L(x_j = v) \log_2 p_L(x_j = v)$

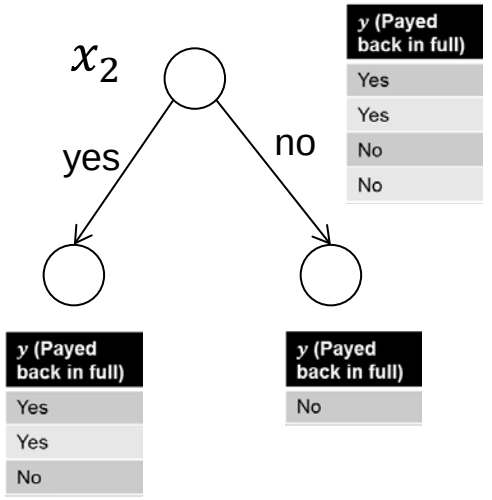
# Information Gain Ratio

- Idea: factor the information that is contained in the attribute values into the decision
  - $H_L(x_j) = -\sum_{v=1}^{k} p_L(x_j = v) \log_2 p_L(x_j = v)$
- Information gain ratio:
  - $GR_L(x_j) = \frac{G_L(x_j)}{H_L(x_j)}$

# Example: Info Gain Ratio

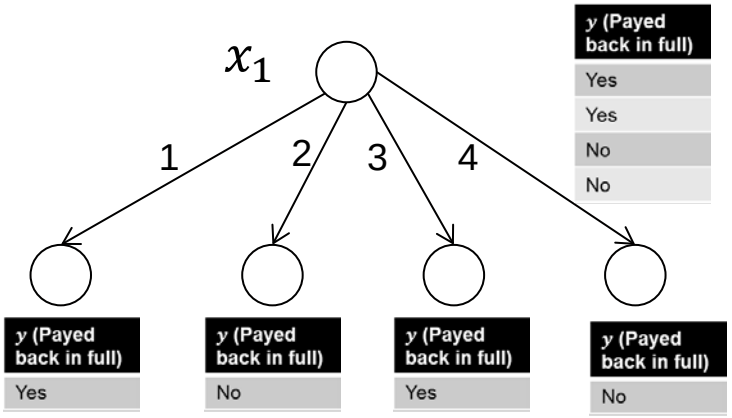
- Which split is better?

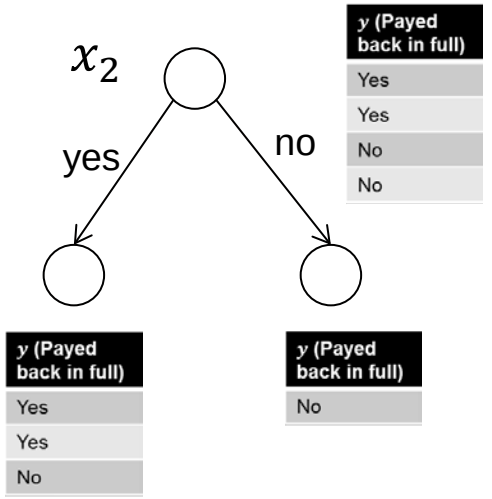


# Example: Info Gain Ratio

- Which split is better?




$$IG_L(x_1) = 1 - 4\tfrac{1}{4}0 = 1$$
$$H_L(x_1) = -4\left(\tfrac{1}{4}\log_2\tfrac{1}{4}\right) = 2$$
$$GR_L(x_1) = \frac{IG_L(x_1)}{H_L(x_1)} = \frac{1}{2}$$

$$IG_L(x_2) = 1 - \tfrac{3}{4}0.92 - \tfrac{1}{4}0 = 0.68$$
$$H_L(x_2) = -\left(\tfrac{3}{4}\log_2\tfrac{3}{4} + \tfrac{1}{4}\log_2\tfrac{1}{4}\right) = 0{,}81$$
$$GR_L(x_2) = \frac{IG_L(x_2)}{H_L(x_2)} = 0.84$$

# Algorithm ID3

- Preconditions:
  - Classification problems
  - All attributes have fixed, discrete ranges of values.
- Idea: recursive algorithm.
  - Choose attribute that conveys highest information regarding the class label (Use information gain, gain ratio, or Gini index).
  - Split the training data according to the attribute.
  - Recursively call algorithm for branches.
  - In each branch, use each attribute only once; if all attributes have been used, return leaf node.

# Learning Decision Trees with ID3

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
    1. For all attributes $x_j \in X$, calculate split criterion $G_L(x_j)$ or $GR_L(x_j)$.
    2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$ or $GR_L(x_j)$.
    3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
    4. Return test node with attribute $x_j$ and children $ID3(L_1, X \setminus x_j)$, …, $ID3(L_k, X \setminus x_j)$.

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
    1. For all $x_j \in X$, calculate $G_L(x_j)$.
    2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
    3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
    4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
    1. For all $x_j \in X$, calculate $G_L(x_j)$.
    2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
    3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
    4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x,y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children $ID3(L_1, X \setminus x_j)$, …, $ID3(L_k, X \setminus x_j)$.

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

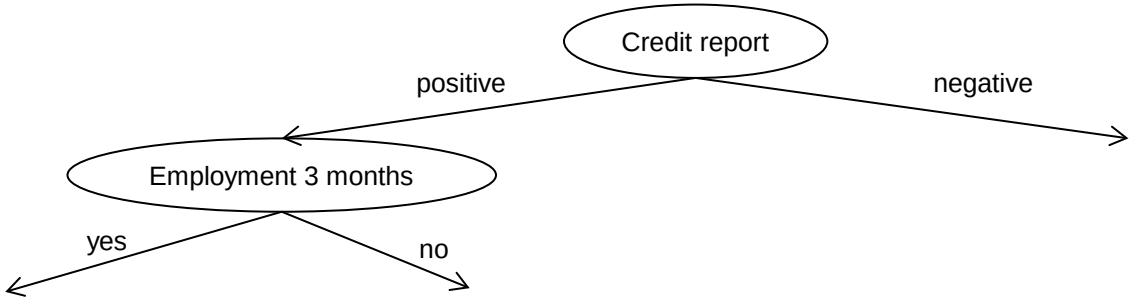
1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

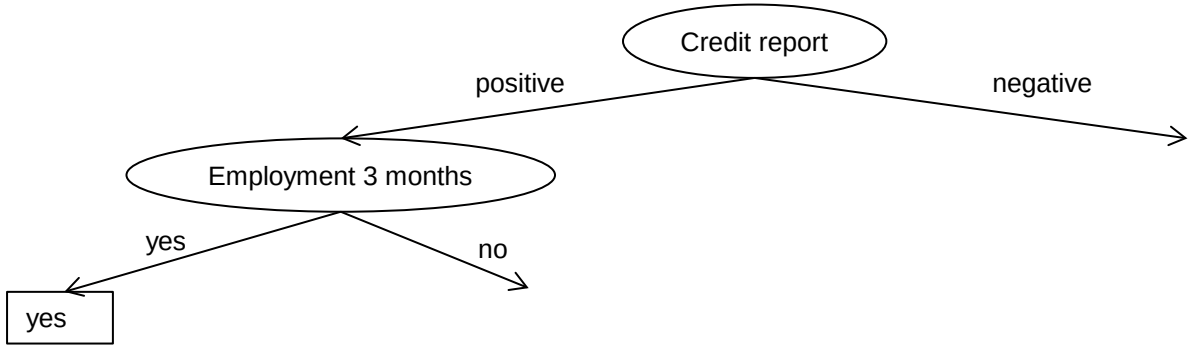


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

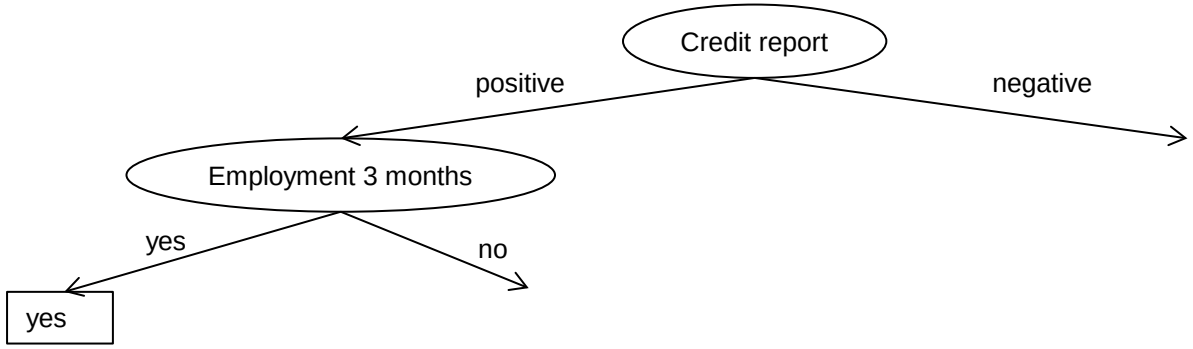


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x,y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

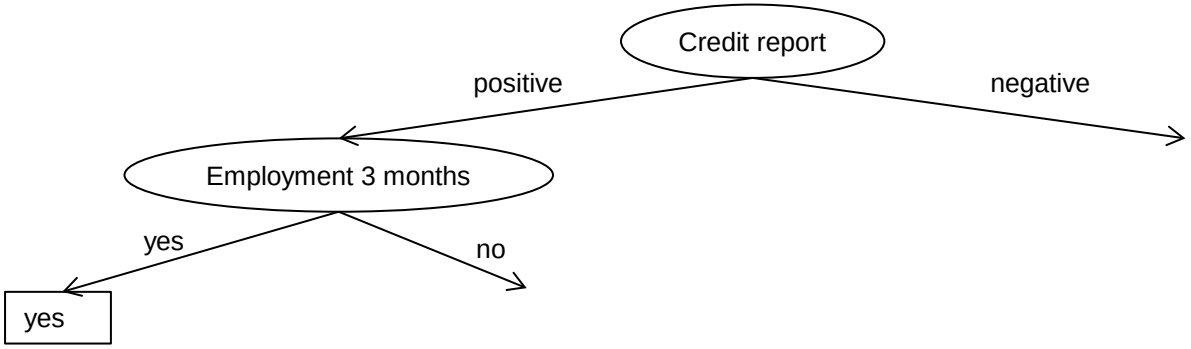


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

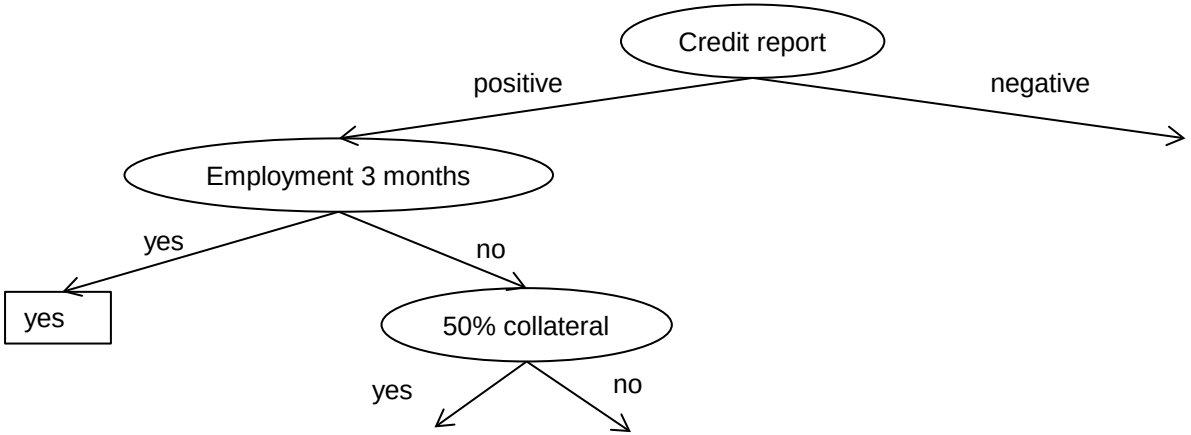


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x,y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

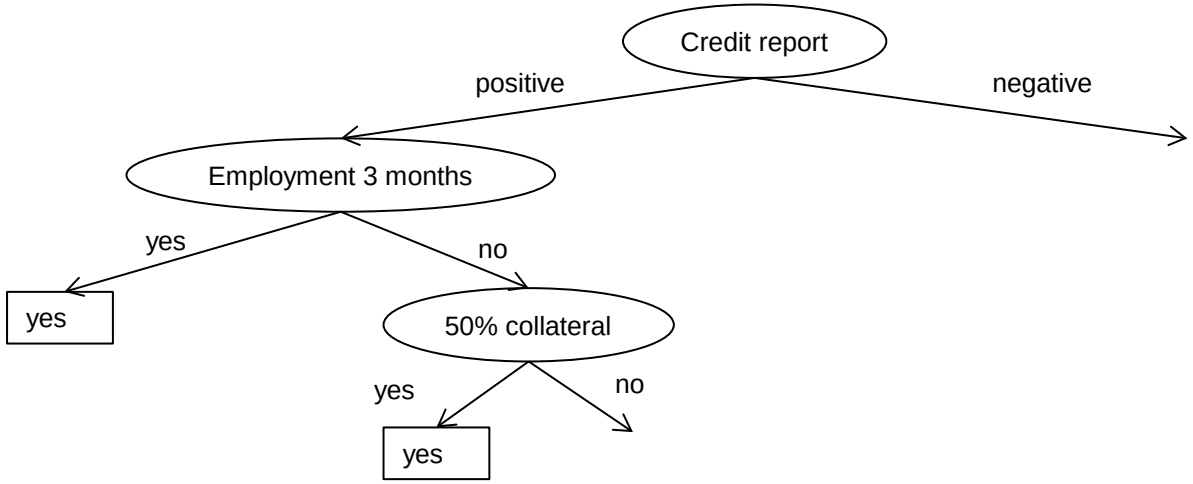


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

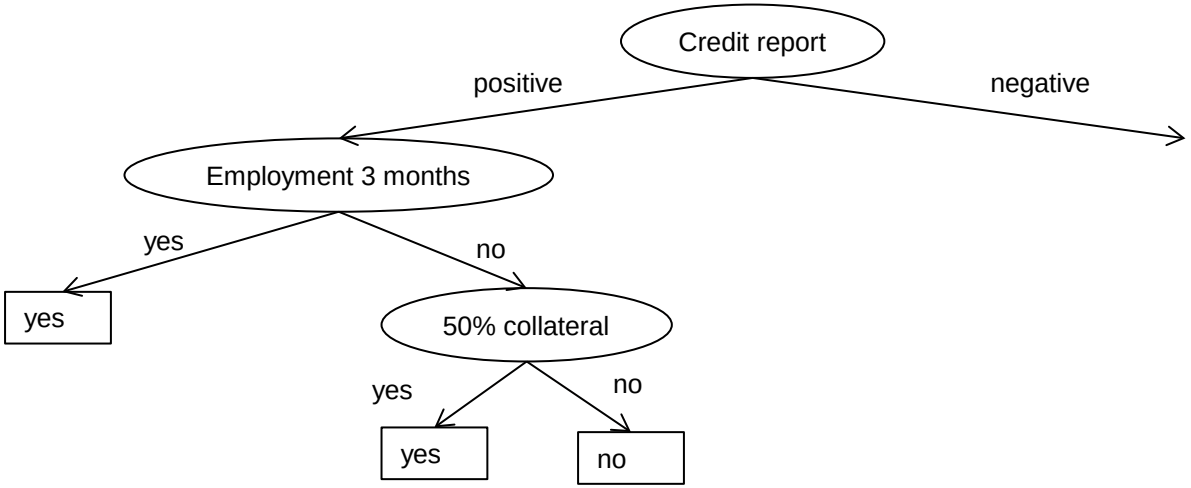


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

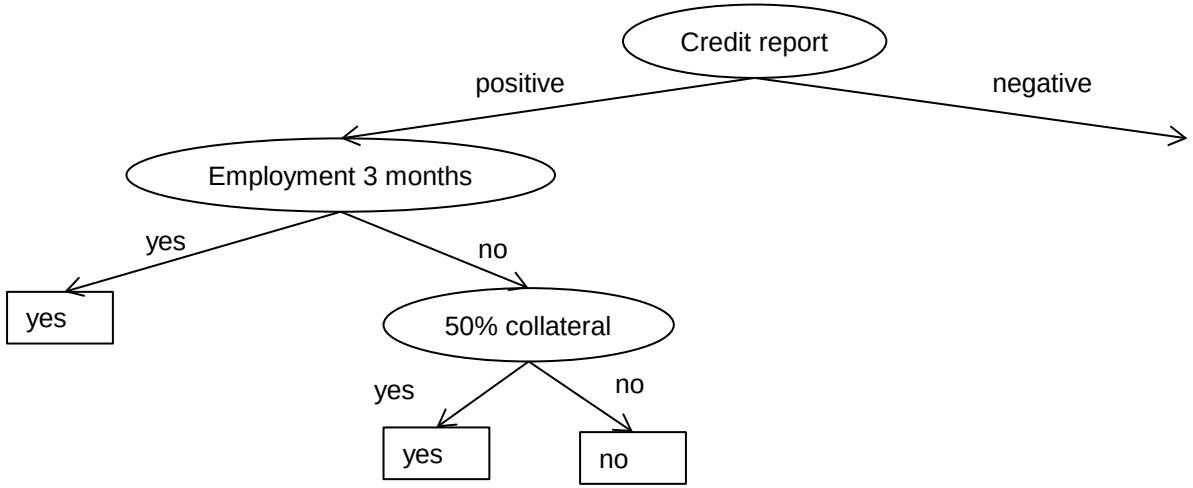


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x,y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

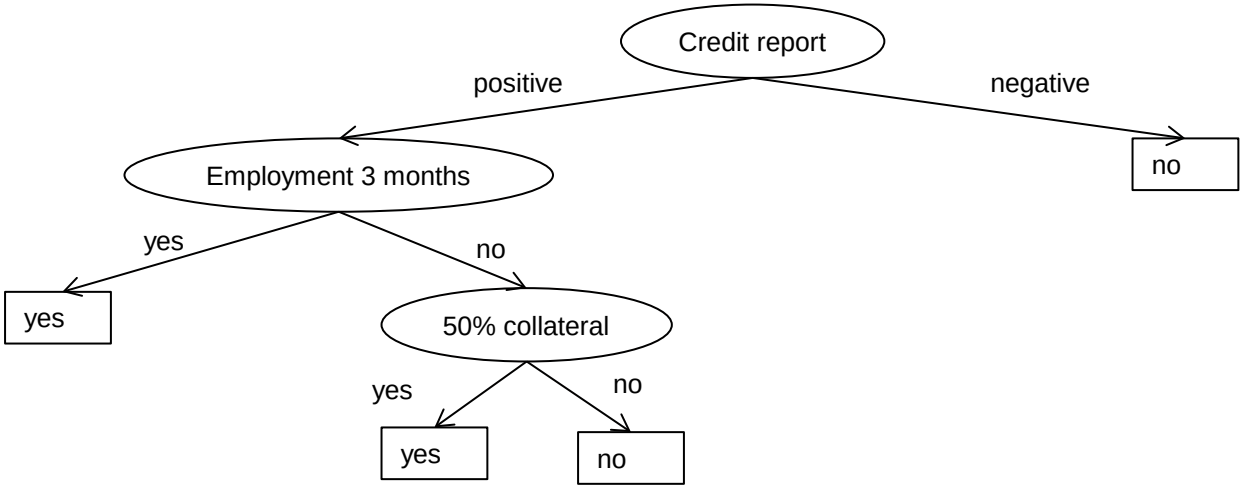


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={},
   then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children
      ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).

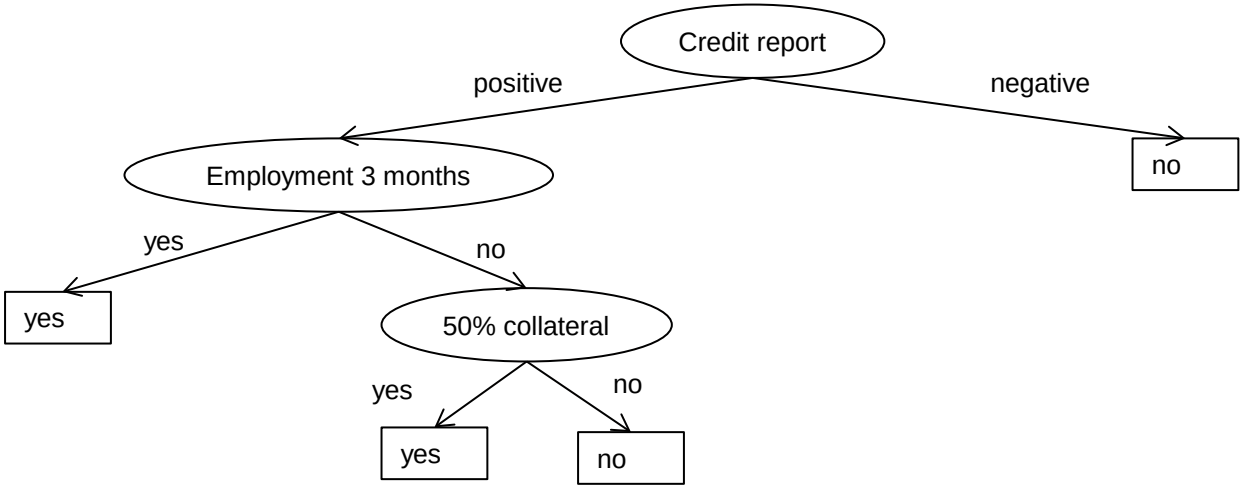


# ID3: Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $x_3$ (Collateral > 50% loan) | $y$ (Payed back in full) |
|---|---|---|---|---|
| 1 | Positive | Yes | No | Yes |
| 2 | Positive | No | Yes | Yes |
| 3 | Positive | No | No | No |
| 4 | Negative | No | Yes | No |
| 5 | Negative | Yes | No | No |

ID3(L, X)

1. If all data in L have same class y or X={}, then return leaf node with majority class y.
2. Else
   1. For all $x_j \in X$, calculate $G_L(x_j)$.
   2. Choose attribute $x_j \in X$ with highest $G_L(x_j)$.
   3. Let $L_i = \{(x, y) \in L : x_j = i\}$.
   4. Return test node with attribute $x_j$ and children ID3($L_1$, $X \setminus x_j$), …, ID3($L_k$, $X \setminus x_j$).



# Overview

- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Continuous Attributes

- ID3 constructs a branch for every value of the selected attribute.
- This only works for discrete attributes.
- Idea: Choose attribute value pair, use test $x_j \leq v$.
  - $G_L(x_j \leq v)$
    $= H_L(y) - p_L(x_j \leq v)H_L(y|x_j \leq v) - p_L(x_j > v)H_L(y|x_j > v)$

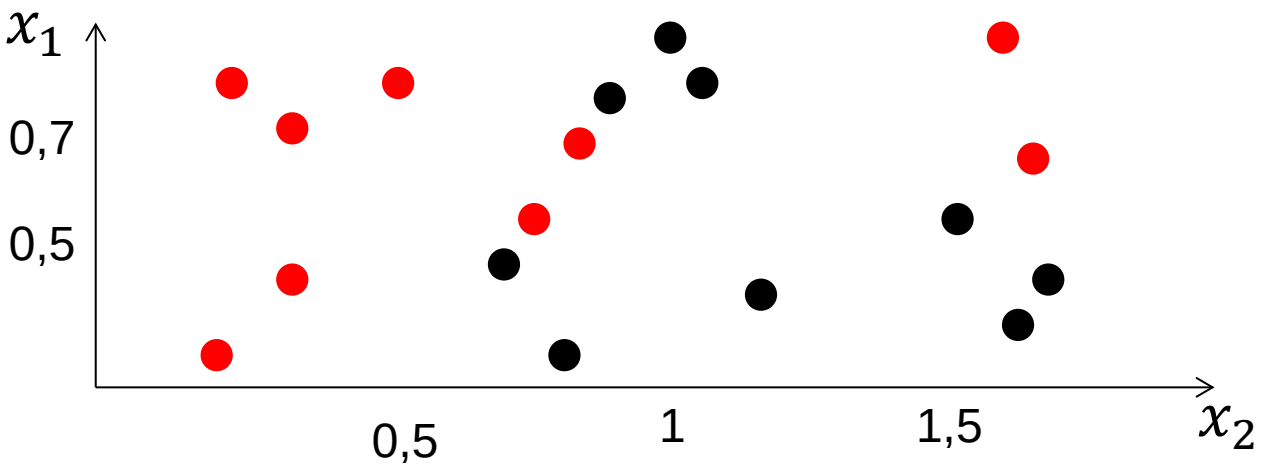
- Problem: there are infinitely many values.
- Idea: use only values that occur for that attribute in the training data.

# Learning Decision Trees with C4.5

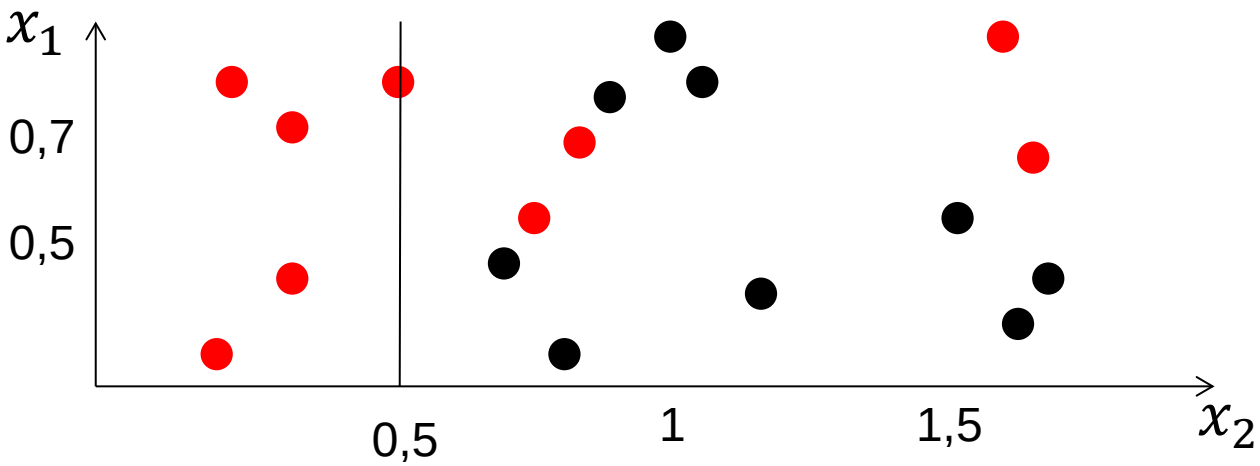
C4.5(L)

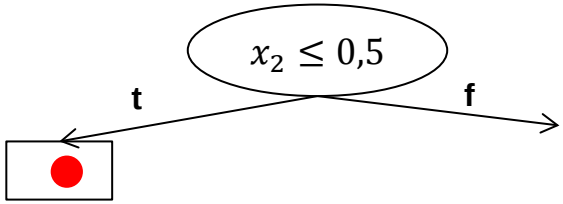
1. If all data in L have same class y or are identical, then return leaf node with majority class y.
2. Else
   1. For all discrete attributes $x_j \in X$: calculate $G_L(x_j)$.
   2. For all continuous attributes $x_j \in X$ and all values v that occur for $x_j$ in L: calculate $G_L(x_j \leq v)$.
   3. If discrete attribute has highest $G_L(x_j)$:
      1. Let $L_i = \{(x,y) \in L : x_j = i\}$.
      2. Return test node with attribute $x_j$ and children $C4.5(L_1)$, …, $C4.5(L_k)$.
   4. If continuous attribute has highest $G_L(x_j \leq v)$:
      1. Let $L_{\leq} = \{(x,y) \in L : x_j \leq v\}$, $L_{>} = \{(x,y) \in L : x_j > v\}$
      2. Return test node with test $x_j \leq v$ and children $C4.5(L_{\leq})$, $C4.5(L_{>})$.

# C4.5: Example

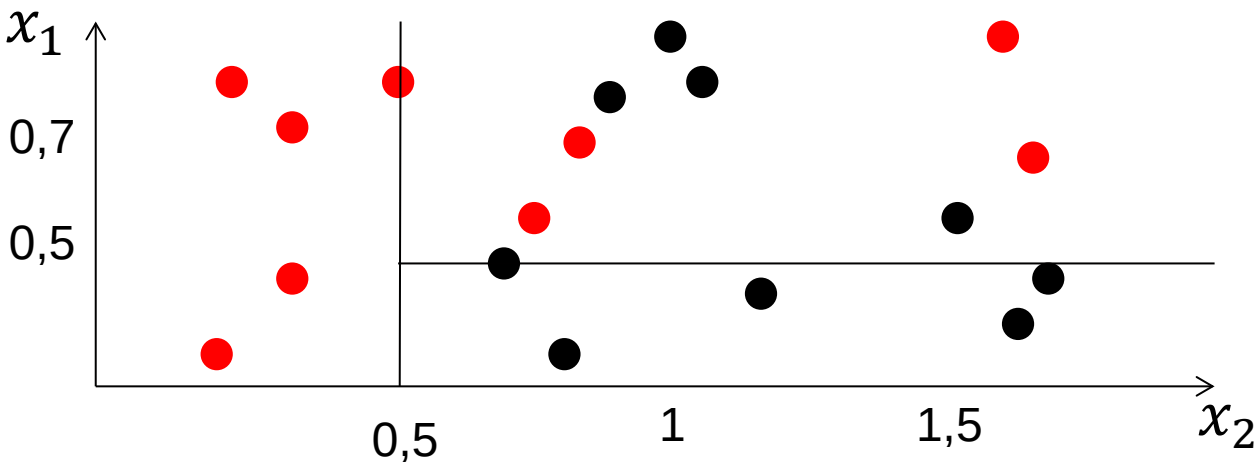


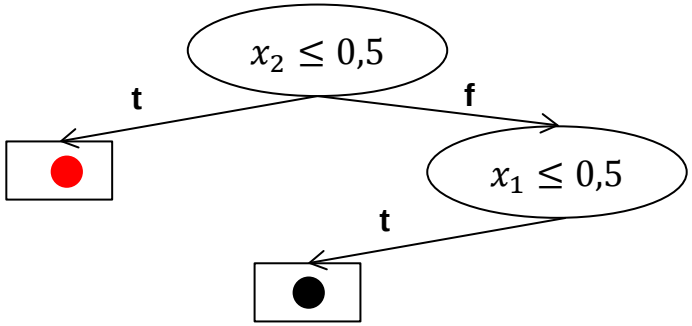
# C4.5: Example



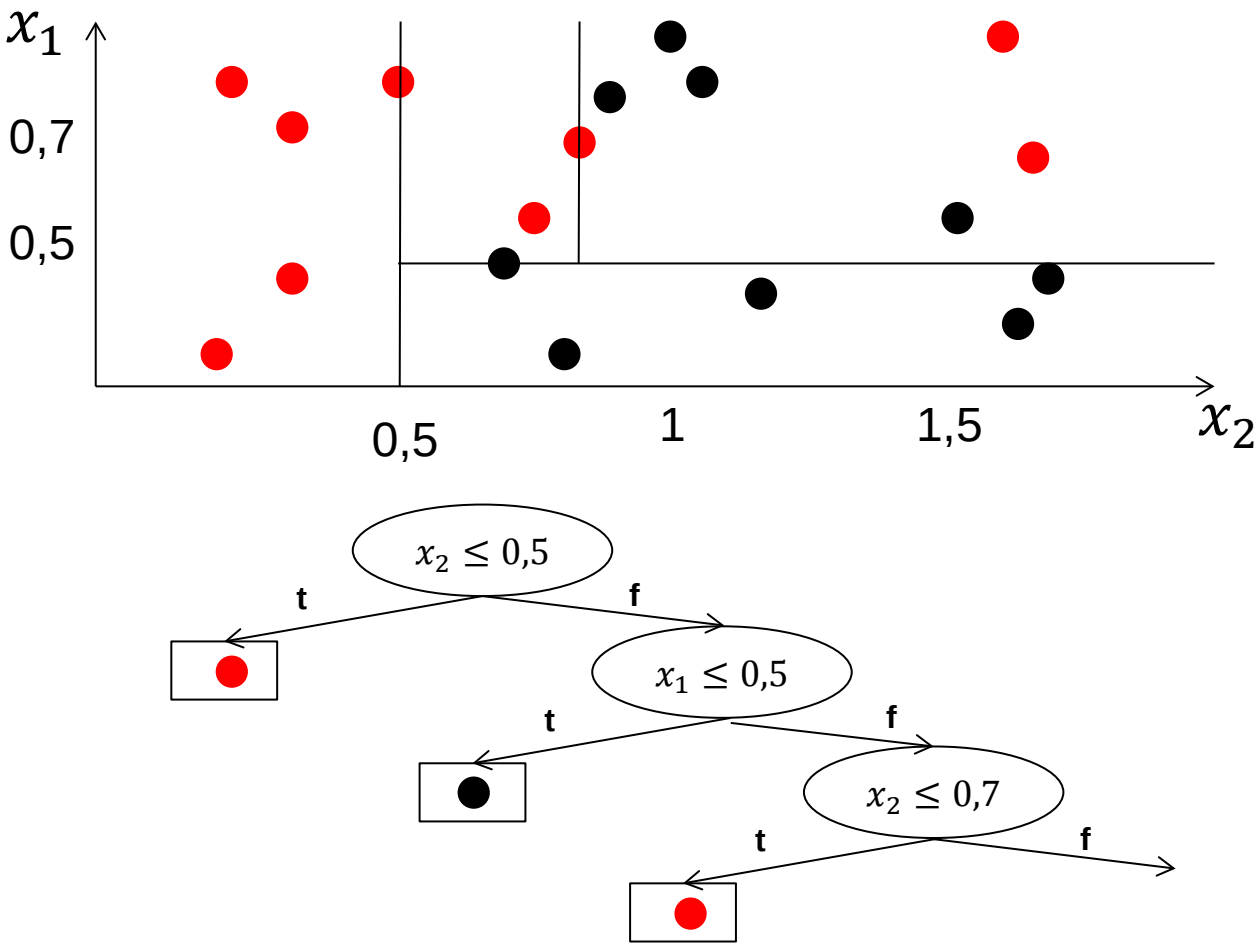



# C4.5: Example

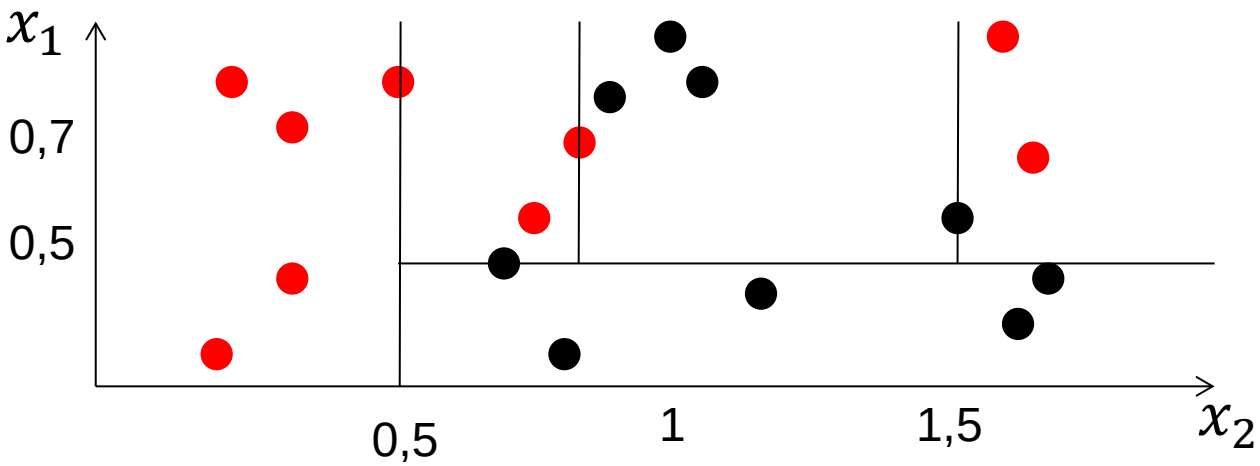


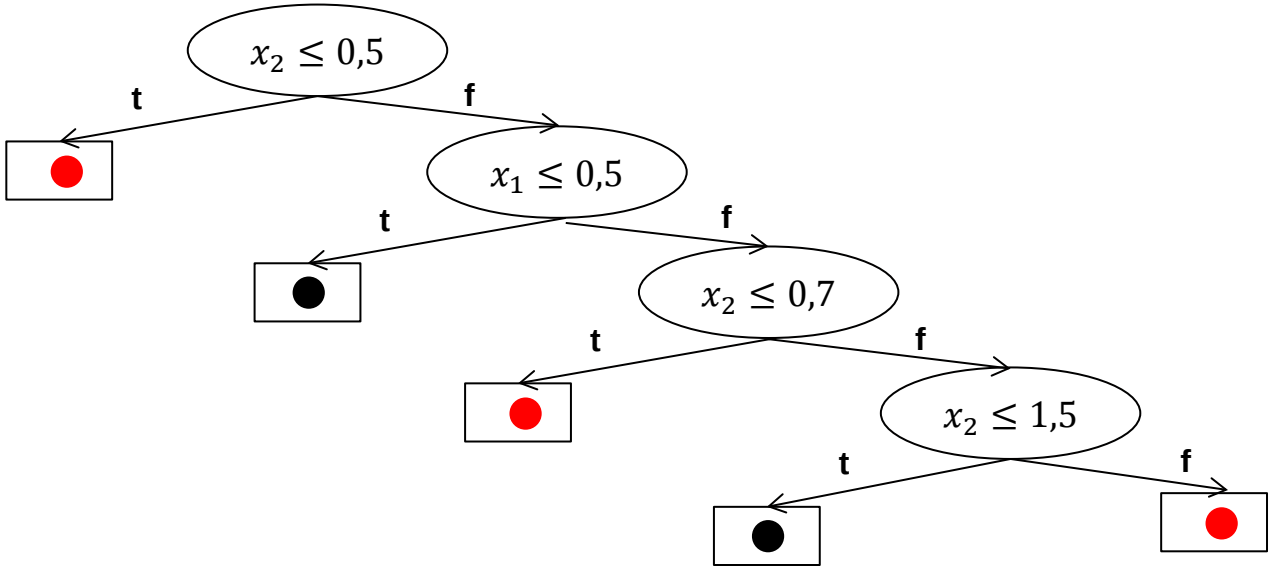



# C4.5: Example



# C4.5: Example



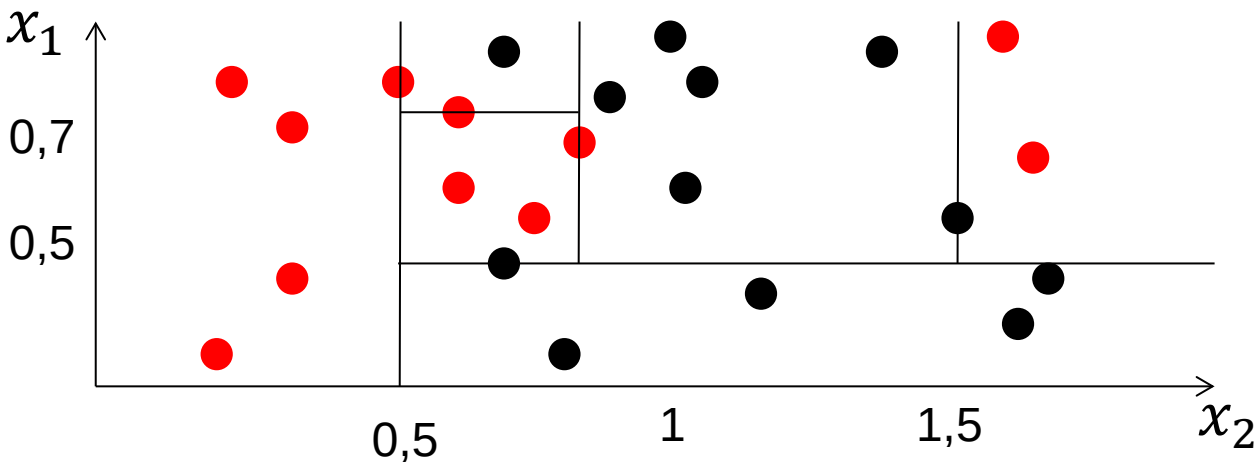



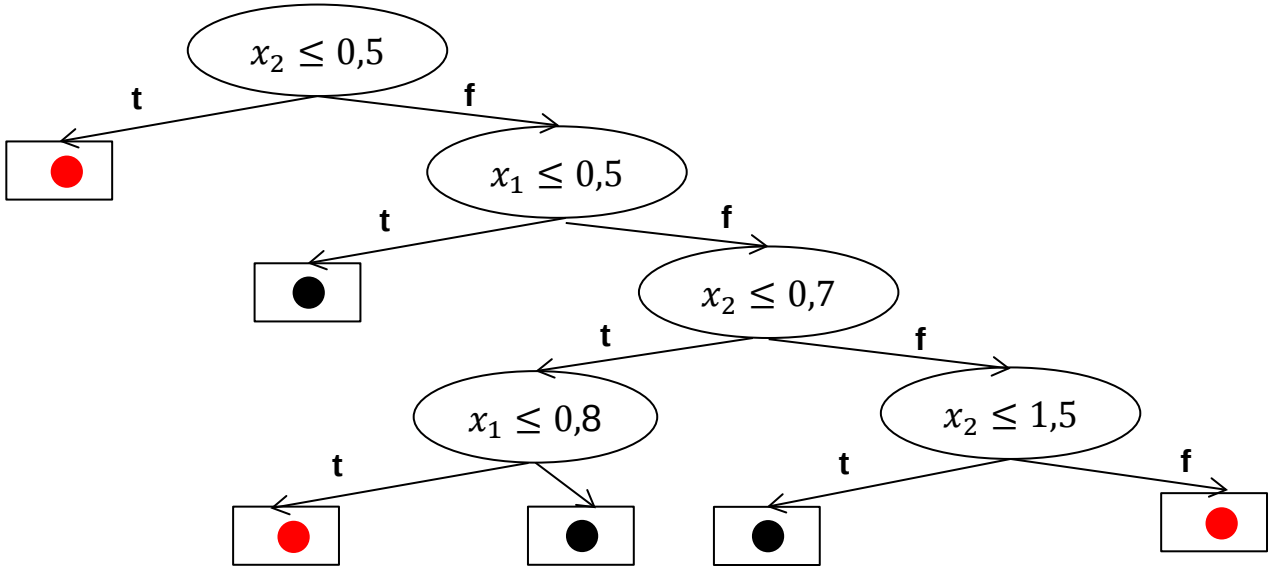
# Pruning

- Leaf nodes that only are supported by a single (or very few) instances, often do not provide a good classification.
- Pruning:
  - Remove test nodes whose leaves have less than $\tau$ instances.
  - Collect in new leaf node that is labeled with the majority class

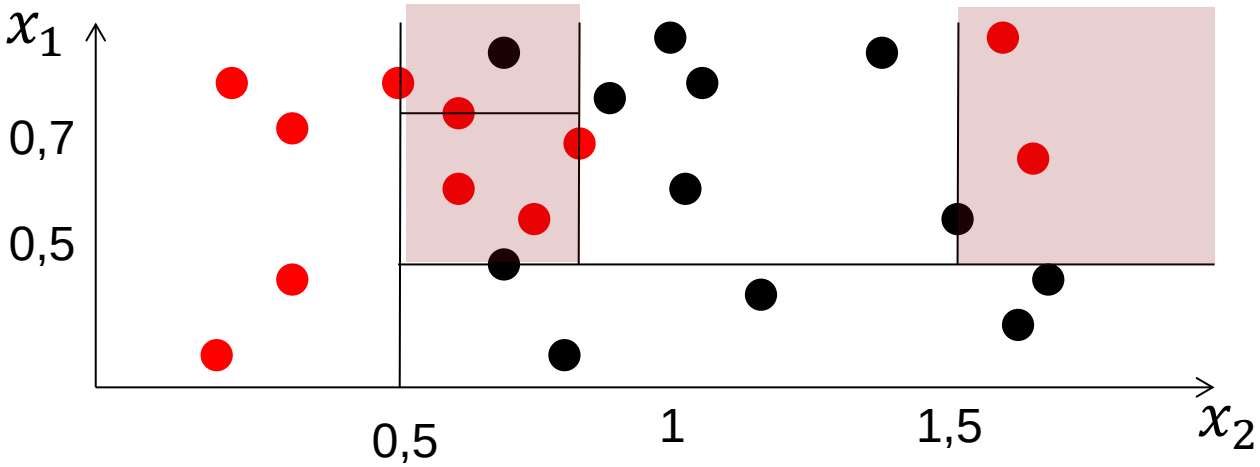
- Pruning parameter $\tau$ is a regularization parameter that has to be tuned (e.g., by cross validation).

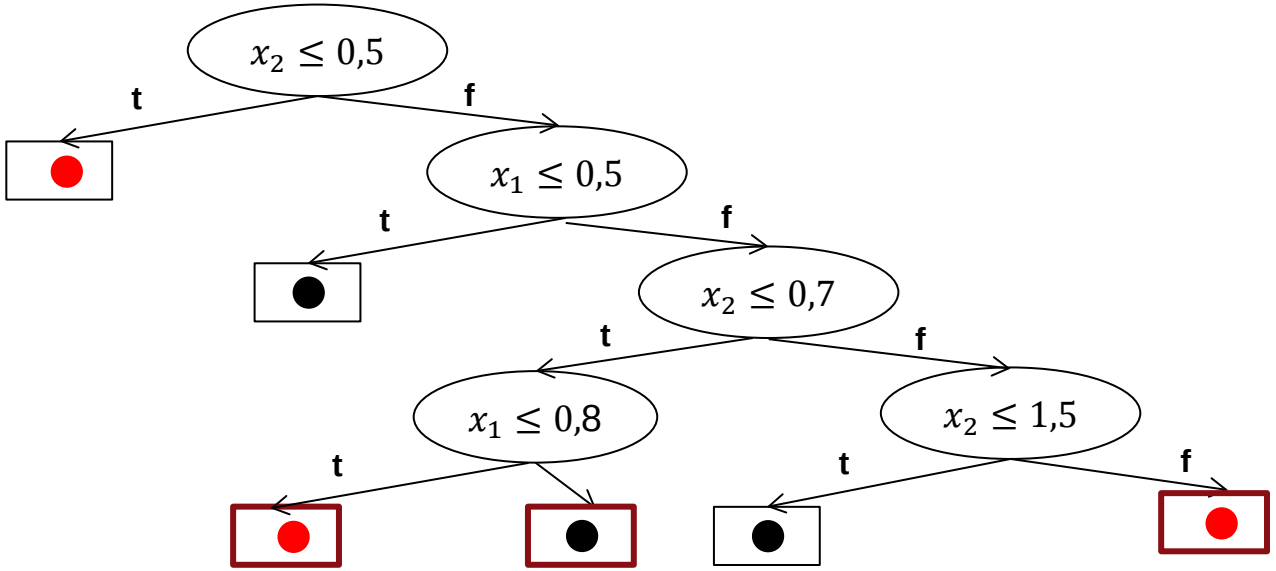
# Pruning with Threshold 5: Example



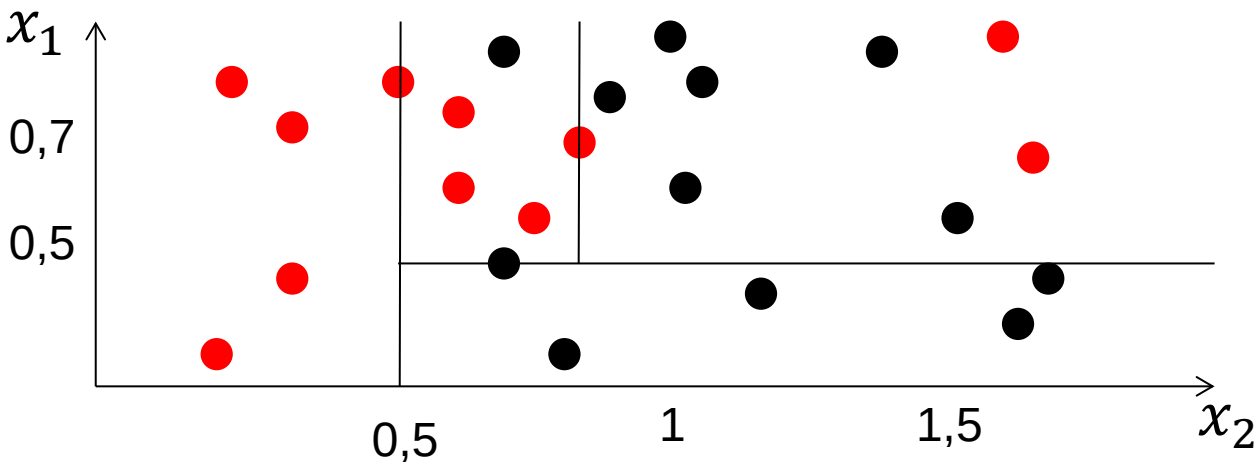



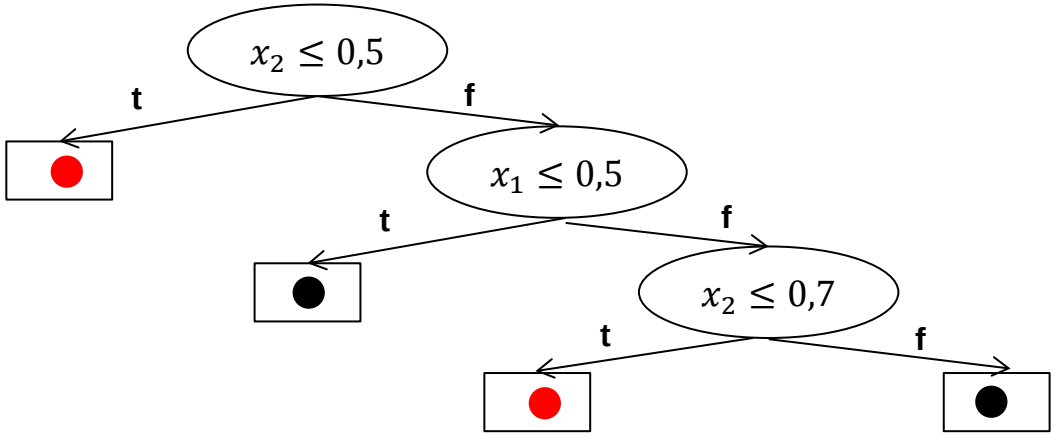
# Pruning with Threshold 5: Example



# Pruning with Threshold 5: Example



# Reduced Error Pruning

- Split training data into a training set and a pruning validation set.
- Construct a decision tree using the training set (for instance, with C4.5).
- Starting from the bottom layer, iterate over all test nodes whose children are leaf nodes.
  - If removing the test node and replacing it with a leaf node that predicts the majority class reduces the error rate on the pruning set, then do it!

# Overview

- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Regression

- Input: instance $\mathbf{x} \in X$.
  - e.g., feature vector

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$$

- Output: continuous (real) value, $y \in \mathbb{R}$
- Learning problem: training data with continuous target value
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \rangle$
  - e.g. $\langle (\mathbf{x}_1, 3.5\ ), \ldots, (\mathbf{x}_n, -2.8\ ) \rangle$

# Regression

- Input: instance $\mathbf{x} \in X$.
  - e.g., feature vector

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$$

- Output: continuous (real) value, $y \in \mathbb{R}$
- Learning goal:
  - Low quadratic error rate + simple model.
  - $SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2$; $MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$
- Split criterion for regression?

# Regression Trees

- Variance of the target attribute on sample $L$:
  - $Var(L) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$
- Variance = MSE of predicting the mean value.
- Splitting criterion: variance reduction of $[x_j \leq v]$:
  - $R_L[x_j \leq v]$
    $= Var(L) - \frac{n_{[x_j \leq v]}}{n} Var\left(L_{[x_j \leq v]}\right) - \frac{n_{[x_j > v]}}{n} Var\left(L_{[x_j > v]}\right)$
- Stopping criterion:
  - Do not create a new test node if $nVar(L) \leq \tau$.

# Learning Regression Trees with CART

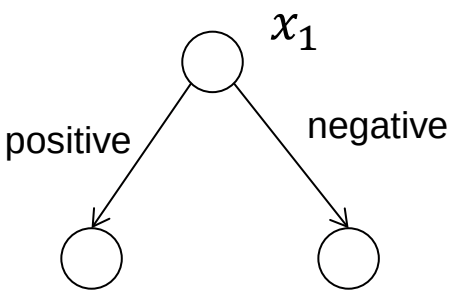
CART(L)

1. If $\sum_{i=1}^{n}(y_i - \bar{y})^2 < \tau$, then return leaf node with prediction $\bar{y}$.
2. Else
   1. For all discrete attributes $x_j \in X$: calculate $R_L(x_j)$.
   2. For all continuous attributes $x_j \in X$ and all values v that occur for $x_j$ in L: calculate $R_L(x_j \leq v)$.
   3. If discrete attribute has highest $R_L(x_j)$:
      1. Let $L_i = \{(x,y) \in L: x_j = i\}$.
      2. Return test node with attribute $x_j$ and children $CART(L_1)$, …, $CART(L_k)$.
   4. If continuous attribute has highest $R_L(x_j \leq v)$:
      1. Let $L_{\leq} = \{(x,y) \in L: x_j \leq v\}$, $L_{>} = \{(x,y) \in L: x_j > v\}$
      2. Return test node with test $x_j \leq v$ and children $CART(L_{\leq})$, $CART(L_{>})$.

# CART- Example

| Loan | $x_1$ (Credit report) | $x_2$ (Employment last 3 months) | $y$ (recovery rate) |
|---|---|---|---|
| 1 | Positive | Yes | 0.8 |
| 2 | Positive | No | 0.9 |
| 3 | Positive | No | 0.4 |
| 4 | Negative | No | 0.1 |
| 5 | Negative | Yes | 0.2 |

- Which split is better?
- $Var(L) = \frac{1}{5}\left((0.8 - 0.48)^2 + \cdots + (0.2 - 0.48)^2\right) = 0.1148$



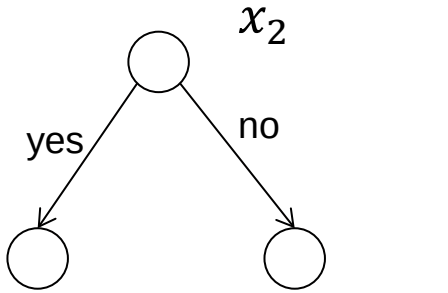

| $y$ (recovery rate) |
|---|
| 0.8 |
| 0.9 |
| 0.4 |

| $y$ (recovery rate) |
|---|
| 0.1 |
| 0.2 |

$$Var_{pos} = \frac{1}{3}\left((0.8 - 0.7)^2 + (0.9 - 0.7)^2 + (0.4 - 0.7)^2\right) = 0.0466$$

$$Var_{neg} = \frac{1}{2}\left((0.1 - 0.15)^2 + (0.2 - 0.15)^2\right) = 0.0025$$

$$0.1148 - \frac{3}{5} \times 0.0466 - \frac{2}{5} \times 0.0025 = \mathbf{0.085}$$




| $y$ (recovery rate) |
|---|
| 0.8 |
| 0.2 |

| $y$ (recovery rate) |
|---|
| 0.9 |
| 0.4 |
| 0.1 |

$$Var_{yes} = \frac{1}{2}\left((0.8 - 0.5)^2 + (0.2 - 0.5)^2\right) =$$

$$Var_{no} = \frac{1}{3}\left((0.9 - 0.46)^2 + (0.4 - 0.46)^2 + (0.1 - 0.46)^2\right) =$$

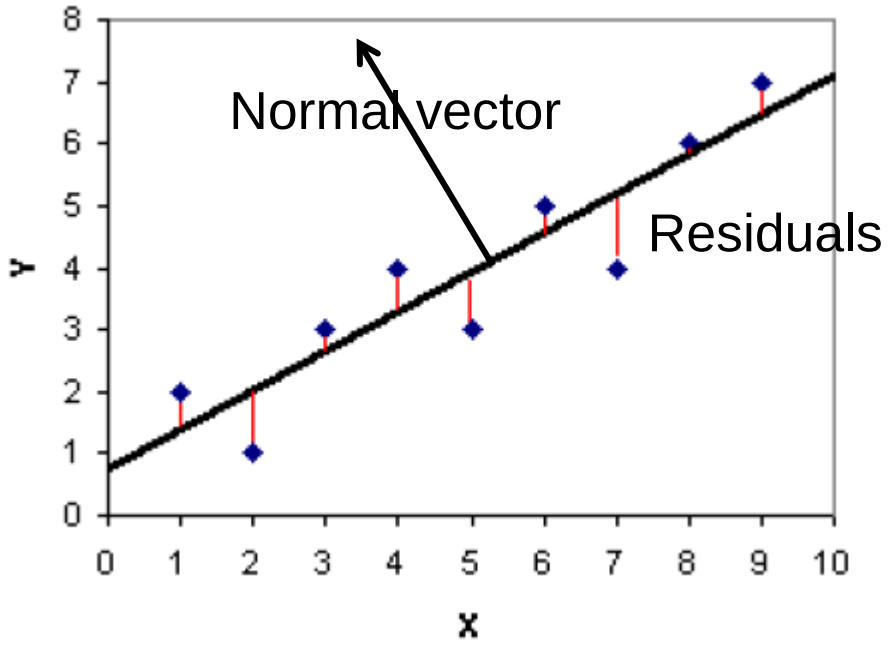
$$0.1148 - \frac{2}{5} \times \qquad - \frac{3}{5} \times \qquad =$$

# Model Trees

- Nodes in regression trees are labeled with the mean value of their training instances.
- Idea: Instead train local linear regression model using the instances that fall into the leaf.

# Linear Regression

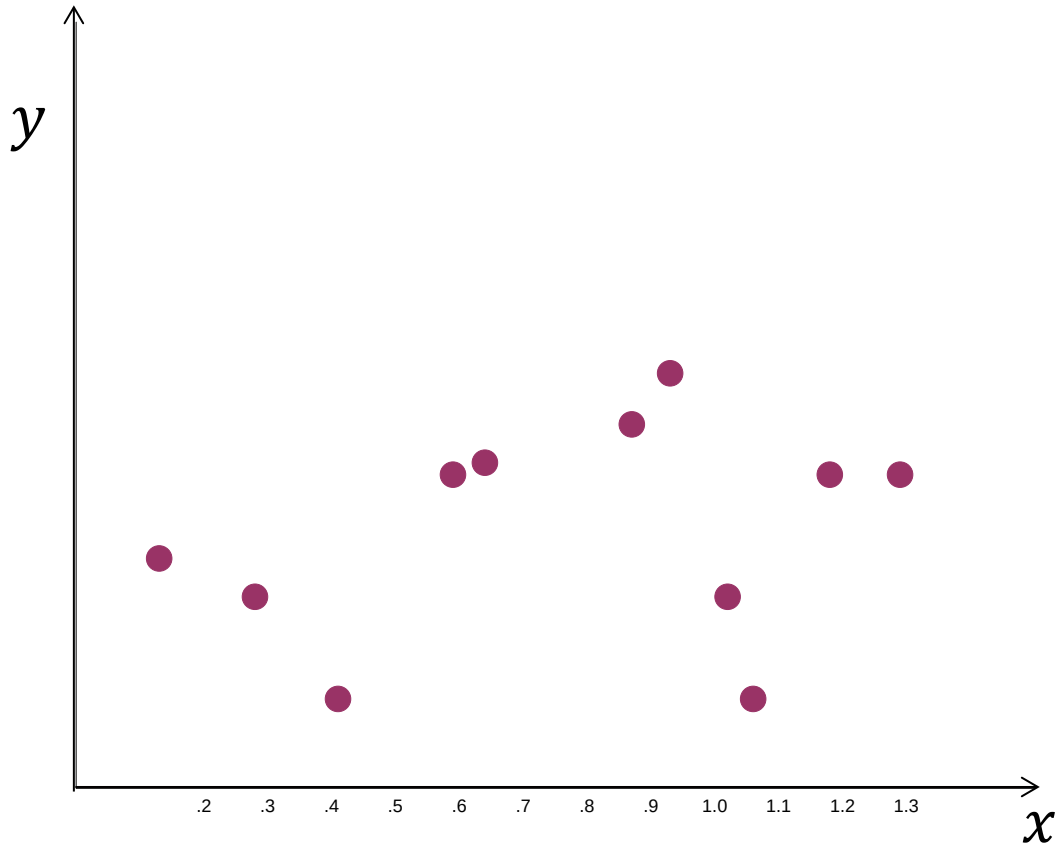
- Input: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \rangle$
- Linear regression model $f(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} + \theta_0$
- Will be discussed in later lectue.




Points along the Regression plane are perpendicular to the normal vector.

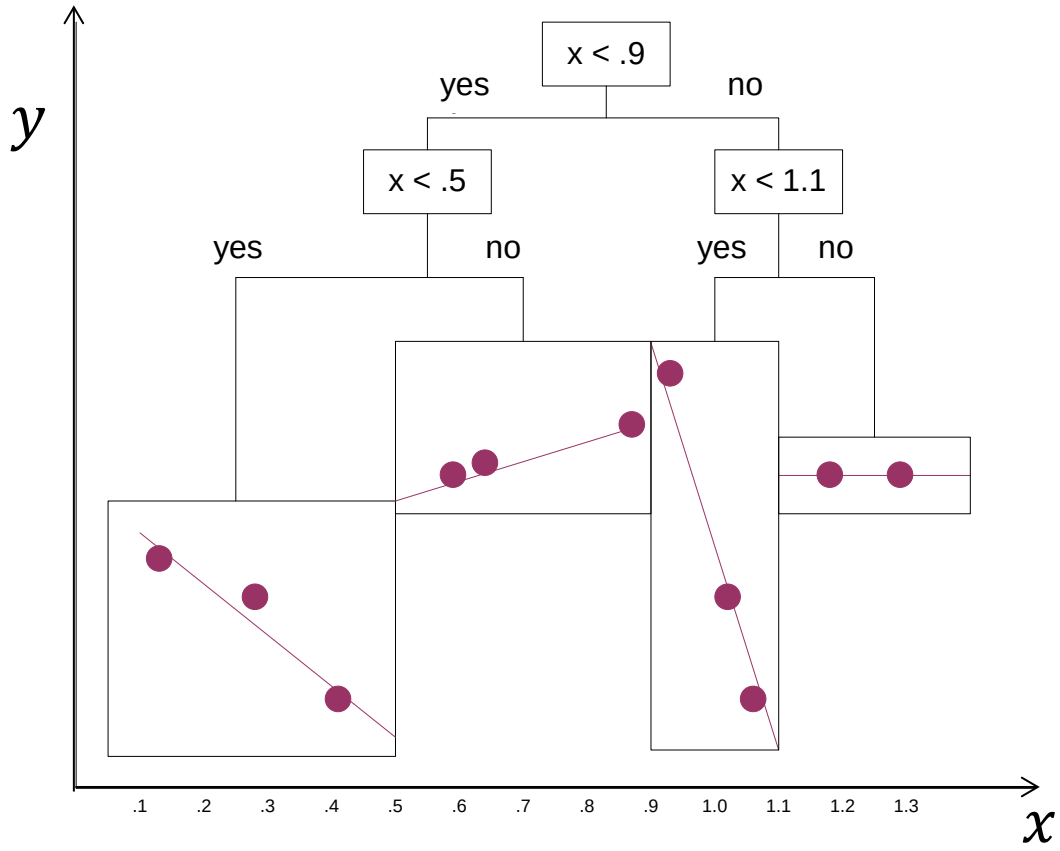
# Model Trees

- Decision trees but with a linear regression model at each leaf node.



# Model Trees

- Decision trees but with a linear regression model at each leaf node.



# Overview

- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Ensemble Learning

- Assume that we have $3$ models $f_i(\mathbf{x})$.
- Each model has an error probability of $0.1$.
- Let $f(\mathbf{x})$ be the majority vote among the $f_i(\mathbf{x})$:
  - $f(\mathbf{x}) = \arg\max_y |\{i: f_i(\mathbf{x}) = y\}|$.
- What is the error probability of $f(\mathbf{x})$?

# Ensemble Learning

- Assume that we have $3$ models $f_i(\mathbf{x})$.
- Each model has an error probability of $0.1$.
- Let $f(\mathbf{x})$ be the majority vote among the $f_i(\mathbf{x})$:
  - $f(\mathbf{x}) = \arg\max_y |\{i: f_i(\mathbf{x}) = y\}|$.
- What is the error probability of $f(\mathbf{x})$?
  - Depends on how correlated the models are.
  - If they always make identical prediction, the error rate of the ensemble is 0.1 as well.

# Ensemble Learning

- Assume that we have $3$ models $f_i(\mathbf{x})$.
- Each model has an error probability of $0.1$.
- Let $f(\mathbf{x})$ be the majority vote among the $f_i(\mathbf{x})$:
  - $f(\mathbf{x}) = \arg\max_y |\{i: f_i(\mathbf{x}) = y\}|$.
- What is the error probability of $f(\mathbf{x})$ if the models are independent?
  - 2 out of 3 models have to err.
  - There are 3 subsets of 2 models.
  - $R \leq 3 \times 0.1^2 = 0.03$.
- Ensemble is much better than individual models.

# Ensemble Learning

- Assume that we have $n$ models $f_i(\mathbf{x})$.
- Each model has an error probability of $0.5 - \varepsilon$.
- Let $f(\mathbf{x})$ be the majority vote among the $f_i(\mathbf{x})$:
  - $f(\mathbf{x}) = \arg\max_y |\{i: f_i(\mathbf{x}) = y\}|$.
- Ensemble $f(\mathbf{x})$ may be much better than individual models if
  - Each model‘s error probability is below 0.5
  - The models are sufficiently uncorrelated.

# Ensemble Learning

- Ensemble methods differ in how they try to obtain uncorrelated classifiers from given training data.
- Bagging: sample random subsets of training instances.
- Random forests: sample random subsets of training instances and random subsets of features.
- Boosting: iteratively sample subsets of the training instances such that instances that are misclassified by the current ensemble receive a higher weight.

# Bootstrapping

- The bootstrapping procedure draws $n$ instances from a set of $n$ instances *with replacement*.
- Instances can be drawn multiple times.

1. Input: sample L of size n.
2. For $i$=1…k
   1. Draw n instances uniformly with replacement from L into set $L_i$.
   2. Learn model $f_i$ on sample $L_i$.
3. Return models $(f_1, \ldots, f_k)$.

# Bagging – Bootstrap Aggregating

- Input: sample L of size n.

1. For $i$=1…k
   1. Draw n instances uniformly with replacement from L into set $L_i$.
   2. Learn model $f_i$ on sample $L_i$.
2. For classification:
   1. Let $f(\mathbf{x})$ be the majority vote among $(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$.
3. For regression:
   1. Let $f(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f_i(\mathbf{x})$.

# Bagging – Bootstrap Aggregating

- Models are somewhat uncorrelated because they have been trained on different subsets of the data.
- But all models use the same attributes and share a good part of the training data.

- Idea: also vary the attributes.

# Random Forests

- Input: sample L of size n, attributes X.

1. For $i$=1…k
   1. Draw n instances uniformly with replacement from L into set $L_i$.
   2. Draw m attributes from all attributes X into $X_i$.
   3. Learn model $f_i$ on sample $L_i$ using attributes $X_i$.
2. For classification:
   1. Let $f(\mathbf{x})$ be the majority vote among $(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$.
3. For regression:
   1. Let $f(\mathbf{x}) = \frac{1}{\mathrm{k}} \sum_{i=1}^{k} f_i(\mathbf{x})$.

# Random Forests

- Number of sampled attributes is a hyperparameter.
- Decision trees are sometimes trained with depth-bound.
- Decision trees are usually trained without pruning.
- The number of trees is a hyperparameter; often, large, fixed values are used (e.g., 1000 trees).

# Overview

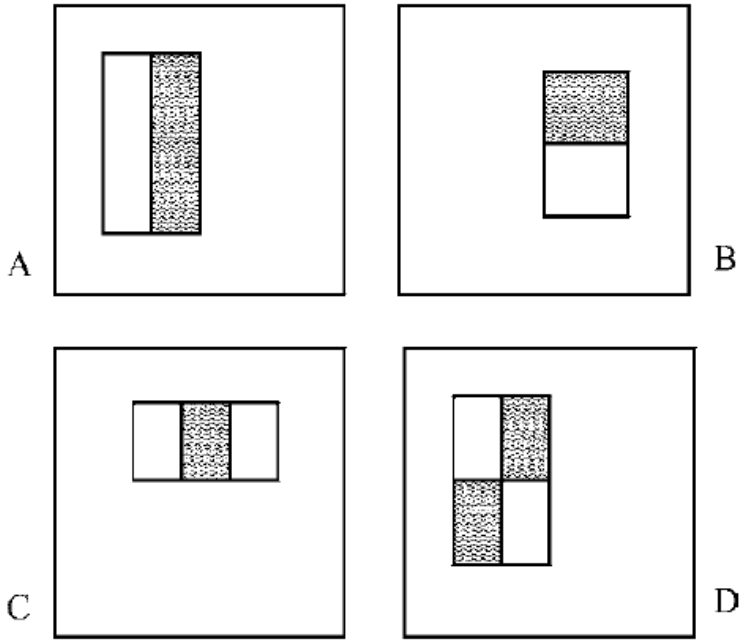
- Decision trees.
- Classification with categorical features: ID3.
- Classification with continuous features: C4.5.
- Regression: CART, model trees.
- Bagging.
- Random forests.
- Example: Viola-Jones algorithm for face detection.

# Face Detection Training Data

- Rectangular boxes that contain aligned images of faces (positive examples) and not faces (negative examples) in a fixed resolution.
- A typical resolution is 24 x 24 pixels.

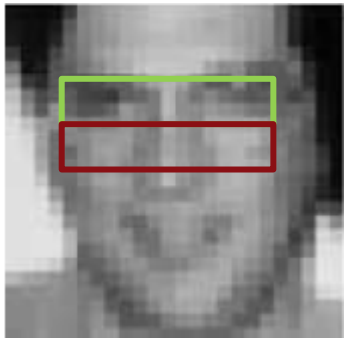
# Haar Features

- Subtract white rectangle of an image from black rectangle.
  - Size, type, position, polarity are parameters.
  - A and C detect horizontal lines, B vertical lines, D detects diagonal patterns.



# Haar Features

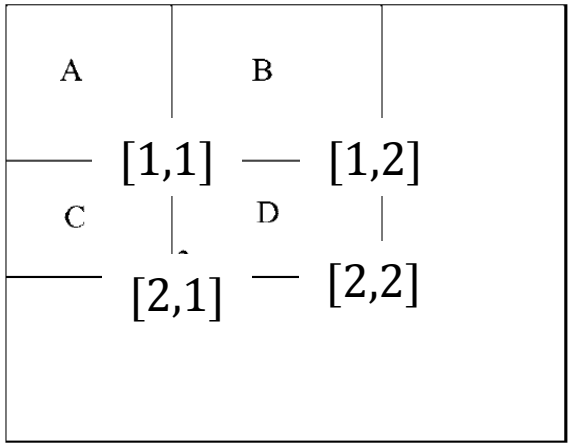
- Subtract white rectangle of an image from black rectangle.
  - Size, type, position, polarity are parameters.
  - A and C detect horizontal lines, B vertical lines, D detects diagonal patterns.



$$\sum_{\substack{\text{pixels } i \text{ in green} \\ \text{rectangle}}} \text{grey value}(i) - \sum_{\substack{\text{pixels } i \text{ in red} \\ \text{rectangle}}} \text{grey value}(i)$$

# Haar Features: Efficient Calculation

- Integral image: $I_{int}(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$.
- Can be computed in one pass over the image.
- All Haar features can be calculated in constant time from the integral image
  - $B - A = I_{int}([1,2]) - 2I_{int}([1,1])$
  - $A - C = I_{int}([1,1]) - (I_{int}([2,1]) - I_{int}([1,1]))$



# Decision Stumps with Haar Features

- Decision stumps = decision trees of depth 1.

1. `Input: Training data L.`
2. `For all available features (type, size, position, polarity, threshold):`
    1. `calculate split criterion (reduction of error rate).`
3. `Create test node with best Haar feature.`

# Ensemble of Decision Stumps

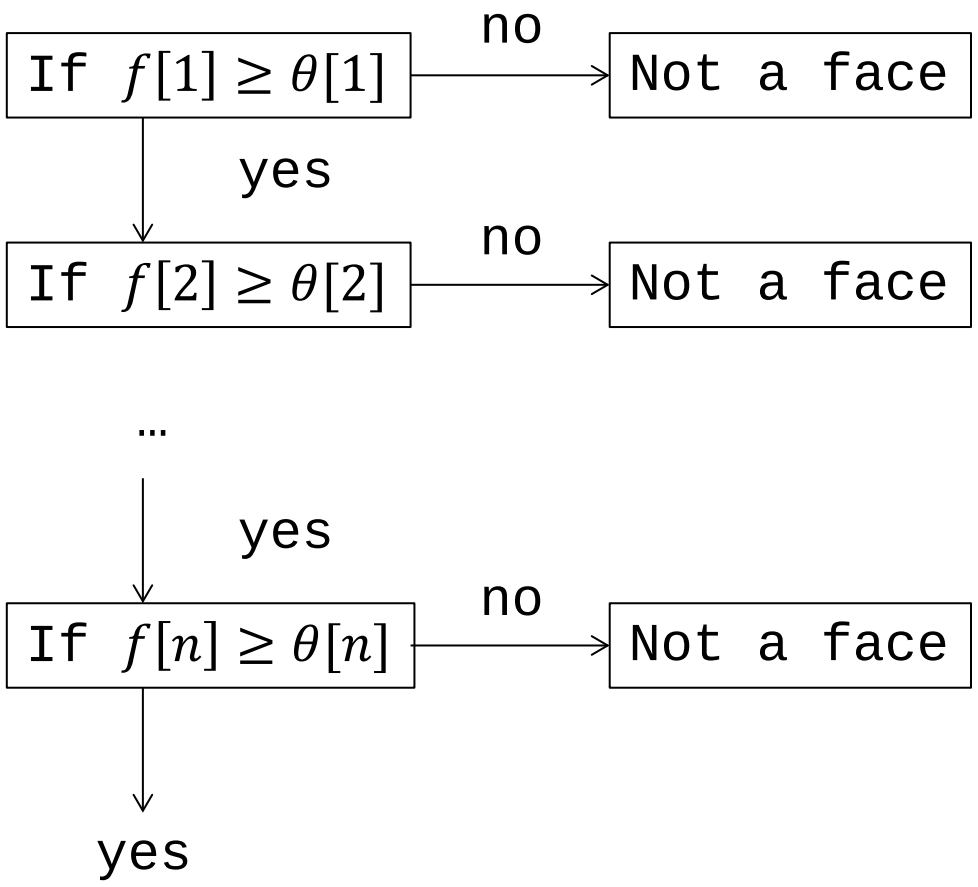
- Use AdaBoost (a variant of boosting) to create an ensemble of decision stumps of prescribed size.
- The ensemble is a weighted combination of features that have a continuous value.
- The weighted combination is compared against a threshold.
- Lowering this threshold increases the recognition rate and the false-positive rate
- Increasing this threshold decreases the recognition rate and the false-positive rate

# Training a Recognition Cascade

- Input: Training data L, number of layers n, desired recall for each layer, r[l], ensemble size for each layer, s[l].
- For layer l=1...n
  - Train an ensemble (weighted combination of features) f[l] of decision stumps of size s[l].
  - Adjust the threshold $\theta$[l] such that a recall of r[l] is obtained.
  - Conbinue training on the next layer with those instances that are classified as a face by the ensemble f[l].

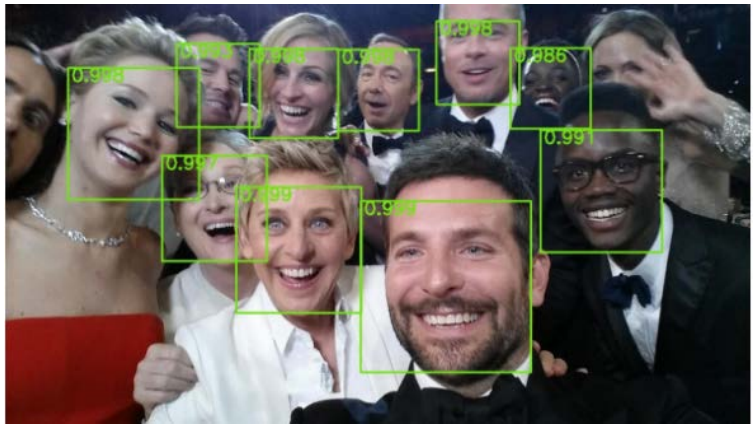
# Face Detection Cascade

- First layer is an ensemble of two Haar features that has a recall of 100% and removes 50% of the non-faces.
- Subsequent layers have larger ensembles



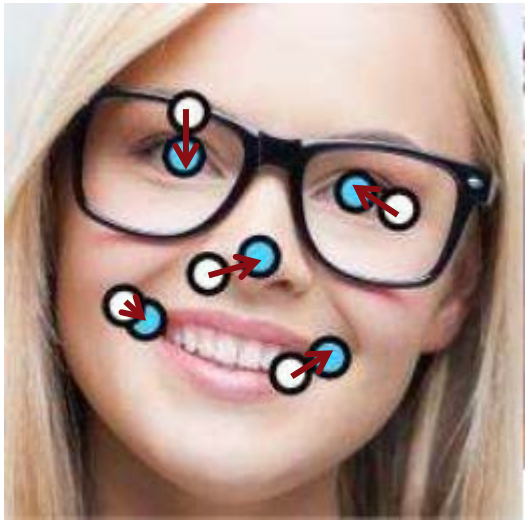
# Scanning Images for Faces

1. For all sizes of the bounding box:
    1. For all positions of the bounding box in the image:
        1. Resample the image window under the bounding box to the fixed resolution of the cascade (24 x 24 pixels)
        2. Apply the face detection cascade.
2. Return all bounding boxes which were classified as face by the cascade.



# Shape Regression with LBF Features

- Random forest is applied iteratively to predict position updates which move a face of landmarks to their correct position.



# Summary of Decision Trees

- Classification
  - Discrete attributes: ID3.
  - Continuous attributes: C4.5.
- Regression:
  - CART: mean value in each leaf.
  - Model trees: linear regression model in each leaf.
- Decision trees are comprehensible models, tests provide the reason for each classification result.
- Applying a decision tree is fast; popular in image processing tasks (face detection, face alignment).