Artificial Intelligence
——Decision Trees 2

Yanghui Rao
Assistant Prof., Ph.D
School of Data and Computer Science,
Sun Yat-sen University
raoyangh@mail.sysu.edu.cn

Information Gain (ID3)

- Class label: buy_computer="yes/no"
- Compute the mutual information (互信息) between D and each attribute A
- $H(D)$=0.940
- $H(D|A="age")$=0.694
- $g(D,A="age")$=0.246
- $g(D,A="income")$=0.029
- $g(D,A="student")$=0.151
- $g(D,A="credit\_rating")$=0.048

"age"这个属性的条件熵最小（等价于信息增益最大），因而首先被选出作为根节点

$$g(D,A) = H(D) - H(D|A)$$

Information Gain (ID3)

userID	age	income	student	credit_rating	buys_computer
u1	<=30	high	no	fair	no
u2	<=30	high	no	excellent	no
u3	31…40	high	no	fair	yes
u4	>40	medium	no	fair	yes
u5	>40	low	yes	fair	yes
u6	>40	low	yes	excellent	no
u7	31…40	low	yes	excellent	yes
u8	<=30	medium	no	fair	no
u9	<=30	low	yes	fair	yes
u10	>40	medium	yes	fair	yes
u11	<=30	medium	yes	excellent	yes
u12	31…40	medium	no	excellent	yes
u13	31…40	high	yes	fair	yes
u14	>40	medium	no	excellent	no

Information Gain Ratio (C4.5)

- Information gain (信息增益) measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio (增益率) to overcome the problem (normalization to information gain)
- The attribute with the maximum gain ratio is selected as the splitting attribute

Information Gain Ratio (C4.5)

- $\text{GainRatio}_A(D) = \text{Gain}_A(D) / \text{SplitInfo}_A(D)$

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $\text{GainRatio}_{A=\text{“}income\text{”}}(D) = ?$

Information Gain Ratio (C4.5)

- $\text{GainRatio}_A(D)=\text{Gain}_A(D)/\text{SplitInfo}_A(D)$

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $\text{GainRatio}_{A=\text{“}income\text{”}}(D)=?$

$$SplitInfo_{A=\text{"}income\text{"}}(D)$$

$$= -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right)$$

$$= 0.926$$

- $\text{GainRatio}_{A=\text{“}income\text{”}}(D)=0.029/0.926=0.031$

Gini Index (CART)

- If a data set D contains examples from n classes, gini index, gini(D) is defined as

$$gini(D) = \sum_{j=1}^{n} p_j(1 - p_j) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D.

- If n=2, then $gini(D) = 2p(1-p)$

Gini Index (CART)

- If a data set D is split into two subsets D_1 and D_2 with sizes N_1 and N_2 respectively, the gini index of the split data contains examples from n classes, the gini index $\text{gini}_{\text{split}}(D)$ is defined as

$$gini_{split}(D) = \frac{N_1}{N} gini(D_1) + \frac{N_2}{N} gini(D_2)$$

- The attribute which provides the smallest $\text{gini}_{\text{split}}(D)$ is chosen to split the node (need to enumerate all possible splitting points for each attribute).

Gini Index (CART)

- D has 9 samples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - (\frac{9}{14})^2 - (\frac{5}{14})^2 = 0.459$$

- The attribute *income* partitions D into 10 in D_1: {medium, high} and 4 in D_2

Gini Index (CART)

- D has 9 samples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - (\frac{9}{14})^2 - (\frac{5}{14})^2 = 0.459$$

- The attribute *income* partitions D into 10 in D_1: {medium, high} and 4 in D_2

$$gini_{income \in \{\text{medium,high}\}}(D) = \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2)$$

$$= \frac{10}{14}\left(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2\right) + \frac{4}{14}\left(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2\right)$$

$$= 0.450 = gini_{income \in \{low\}}(D)$$

Decision Tree

- But how can we compute the gini index, information gain of an attribute that is **continuous-valued**?

Decision Tree

- But how can we compute the gini index, information gain of an attribute that is **continuous-valued**?
  - Given v values of A, then v-1 possible splits are evaluated. For example, the midpoint between the values a_i and a_{i+1} of A is $(a_i + a_{i+1})/2$

Extracting Classification Rules

- Represent the knowledge in the form of **IF-THEN** rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the prediction of classes
- Rules are easier for humans (可解释性) to understand

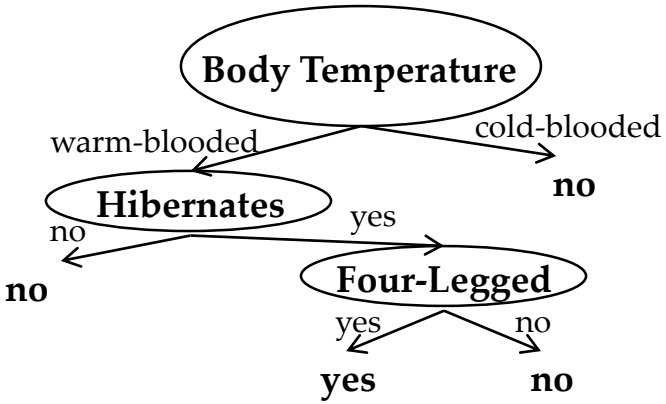
Overfitting

- **Overfitting problems**
  - An induced tree may overfit the training data, in which the performance on the training set does not generalize well to the test examples (*i.e.*, good performance on the training set while poor accuracy for unseen samples)

Overfitting

- The training error is 0, which classifies all warm-blooded vertebrates that do not hibernate as non-mammals

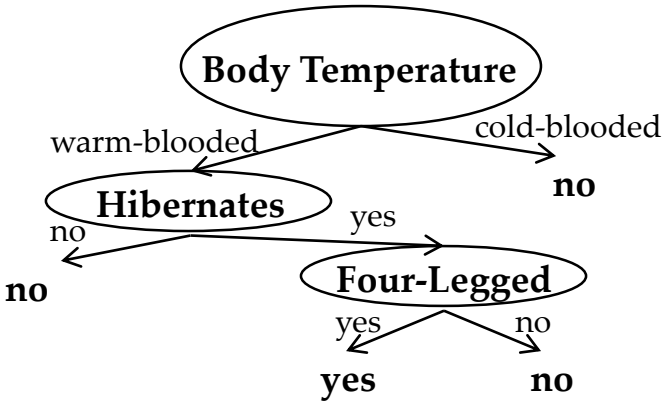
Name	Body Temperature	Four-Legged	Hibernates	Mammals?
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes



Overfitting

- The training error is 0, which classifies all warm-blooded vertebrates that do not hibernate as non-mammals

Name	Body Temperature	Four-Legged	Hibernates	Mammals?
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes



Humans, elephants and dolphins are all misclassified

Overfitting

- Models that make their classification decisions based on a small number of training records are susceptible to overfitting
  - Add more records for training a model

Name	Body Temperature	Four-Legged	Hibernates	Mammals?
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes
human	warm-blooded	no	no	yes
dolphin	warm-blooded	no	no	yes
elephant	warm-blooded	yes	no	yes

Overfitting

Name	Body Temperature	Gives Birth	Four-Legged	Hibernates	Mammals?
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no
whale	warm-blooded	yes	no	no	no
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

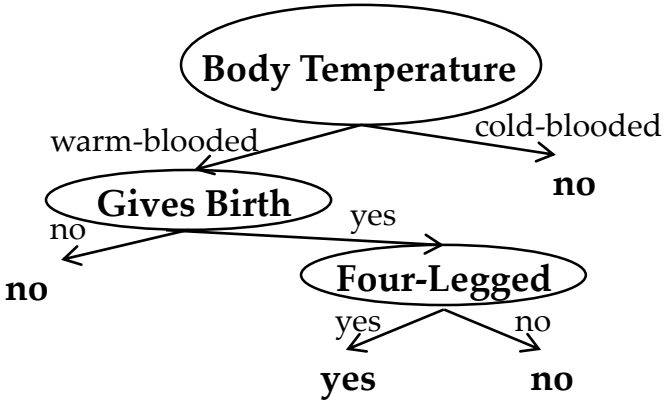
Training set

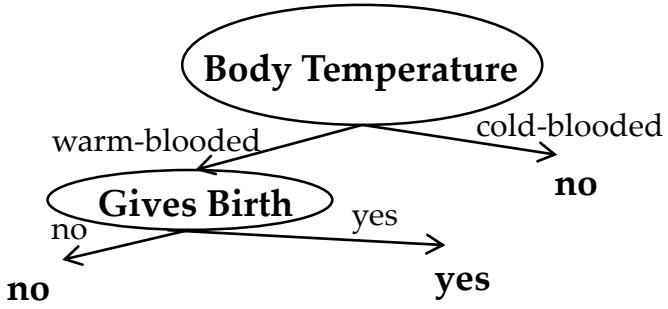
Overfitting

Name	Body Temperature	Gives Birth	Four-Legged	Hibernates	Mammals?
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	warm-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Testing set

Overfitting





The training error is 0
The error rate on the testing set is 30%

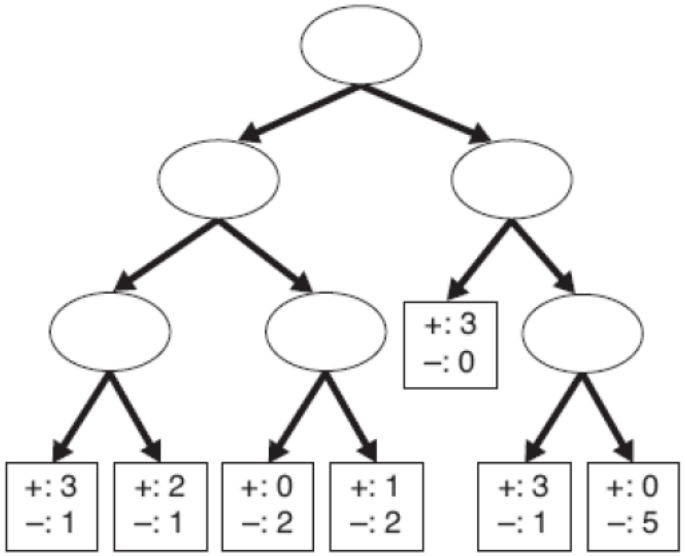
The training error rate is 20%
The error rate on the testing set is 10%

Overfitting

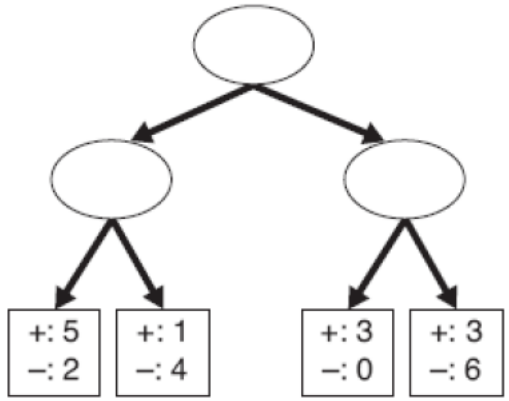
- The training error can be reduced by increasing the model complexity
  - When the tree becomes too large, the training error rate continues to decrease
- While the test (generalization) error can be large because the model may accidentally fit some of the noise points in the training data
  - Too many branches, some may reflect anomalies due to noise or outliers

Incorporating model complexity

- The generalization error is estimated as the sum of
  - Training error
  - A penalty term for model complexity
- The training error for T_L is $e(T_L)=4/24=0.167$
- The training error for T_R is $e(T_R)=6/24=0.25$



Decision Tree, T_L



Decision Tree, T_R

Incorporating model complexity

- In the case of a decision tree, let
  - L be the number of leaf nodes.
  - n_l be the l-th leaf node.
  - $m(n_l)$ be the number of training records classified by n_l.
  - $r(n_l)$ be the number of misclassified records by n_l.
  - $\zeta(n_l)$ be a penalty term associated with the node n_l.
- The resulting error e_c of the decision tree can be estimated as follows:

$$e_c = \frac{\sum_{l=1}^{L} \left(r(n_l) + \zeta(n_l) \right)}{\sum_{l=1}^{L} m(n_l)}$$

Incorporating model complexity

- We consider the previous two decision trees T_L and T_R.
- We assume that the penalty term is equal to 0.5 for each leaf node.
- The error estimate for T_L is

$$e_c(T_L) = \frac{4 + 7 \times 0.5}{24} = \frac{7.5}{24} = 0.3125$$

- The error estimate for T_R is

$$e_c(T_R) = \frac{6 + 4 \times 0.5}{24} = \frac{8}{24} = 0.3333$$

Incorporating model complexity

- Based on this penalty term, T_L is better than T_R.
- For a binary tree, a penalty term of 0.5 means that a node should always be expanded into its two child nodes if it improves the classification of at least one training record.
- This is because expanding a node, which is the same as adding 0.5 to the overall error, is less costly than committing one training error.

Incorporating model complexity

- Suppose the penalty term is equal to 1 for all the leaf nodes.
- The error estimate for T_L becomes 0.458.
- The error estimate for T_R becomes 0.417.
- Based on this penalty term, T_R is better than T_L.
- A penalty term of 1 means that a node should not be expanded unless it reduces the misclassification error by more than one training record.

Tree Pruning

- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Approaches to Train Final Trees

- Separate training and testing sets
- Use cross validation, *e.g.*, *k*-fold cross validation
  - Partition data set into *k* parts
  - Training on random (*k*-1) parts, testing on 1 part
  - Repeat *k* times

Summary on Decision Tree

- Deal with one attribute each time
- Continuous random variables should be split to discrete random variables