CS699
Lecture 4
Classification 1

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **class labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - Predicts categorical class labels (discrete or nominal)
  - Constructs a model based on the training dataset which has known class labels and uses it to classify new data (or determine the class labels of new data)
- Numeric Prediction
  - Models continuous-valued functions, i.e., class attribute is a numeric attribute
  - Can be also used to predict missing values

Prediction Problems: Classification vs. Numeric Prediction

- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

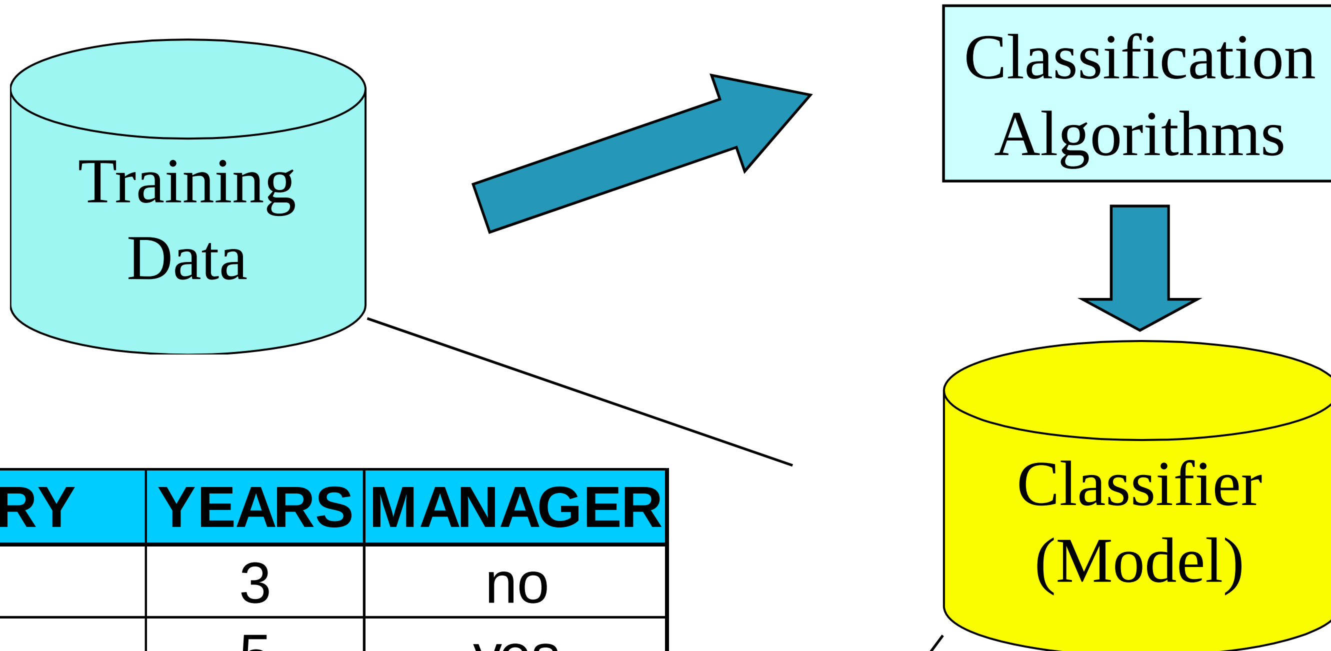
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, mathematical formulae, etc.

Classification—A Two-Step Process

- Model usage: classify future or unknown tuples
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - If the accuracy is acceptable, use the model to classify data tuples with unknown class labels.

    Otherwise, build another model and repeat.

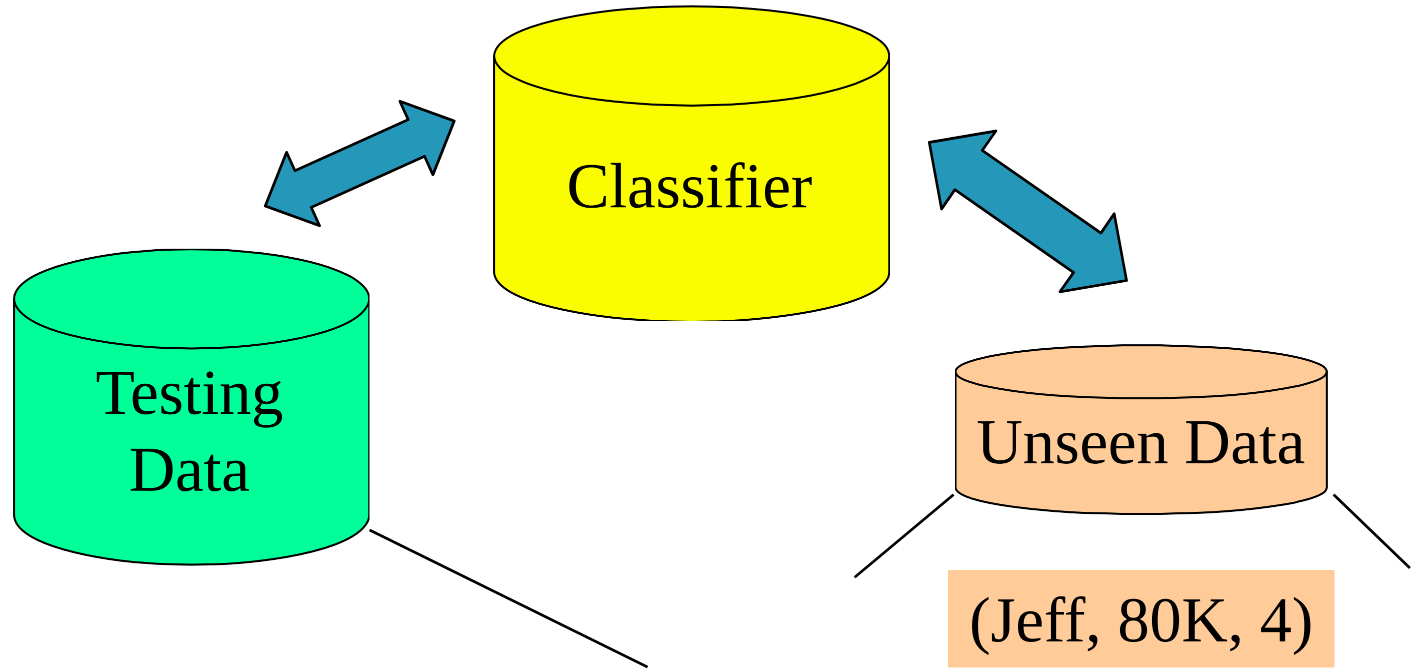
Process (1): Model Construction



NAME	SALARY	YEARS	MANAGER
Mike	85K	3	no
Mary	105K	5	yes
Bill	75K	7	yes
Jim	90K	5	no
Dave	85K	6	yes
Anne	80K	3	no

IF salary > 100K
OR years > 5
THEN manager = ‘yes’

Process (2): Using the Model in Prediction



NAME	SALARY	YEARS	MANAGER
Tom	90K	2	no
Merlisa	75K	4	no
George	95K	6	yes
Joseph	120K	5	yes

Manager?

No

Example (training) Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31..40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31..40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31..40	medium	no	excellent	yes
31..40	high	yes	fair	yes
>40	medium	no	excellent	no

1R Classifier

- Simple and cheap
- Idea: Make rules based on a single attribute
- Compute the error rate for each attribute
- Choose the one with the smallest error rate

1R Classifier

- Algorithm

For each attribute

For each value of the attribute

count the frequency of each class

find the most frequent class

make rule: assign that class to this attribute-value

Compute the error rate of the rules (of this attribute)

Choose the rules with the smallest error rate

1R Classifier

- Example dataset

outlook	temperature	humidity	windy	play
sunny	hot	high	F	N
sunny	hot	high	T	N
overcast	hot	high	F	Y
rainy	mild	high	F	Y
rainy	cool	normal	F	Y
rainy	cool	normal	T	N
overcast	cool	normal	T	Y
sunny	mild	high	F	N
sunny	cool	normal	F	Y
rainy	mild	normal	F	Y
sunny	mild	normal	T	Y
overcast	mild	high	T	Y
overcast	hot	normal	F	Y
rainy	mild	high	T	N

1R Classifier

- Make rules for each attribute and calculate error rate

Attribute outlook

sunny → no (3/5), error rate = 2/5

overcast → yes (4/4), error rate = 0

rainy → yes (3/5), error rate = 2/5

total error = 4/14

Attribute temperature

hot → no (2/4), error rate = 2/4 (arbitrary tie breaking)

mild → yes (4/6), error rate = 2/6

cool → yes (3/4), error rate = 1/4

total error = 5/14

1R Classifier

Attribute humidity

high → no (4/7), error rate = 3/7

normal → yes (6/7), error rate = 1/7

total error = 4/14

Attribute windy

false → yes (6/8), error rate = 2/8

true → no (3/6), error rate = 3/6 (arbitrary tie breaking)

total error = 5/14

1R Classifier

- *outlook* and *humidity* have the same error rate of 4/14. If we (randomly) choose *outlook*, the rules are

outlook: sunny → no
overcast → yes
rainy → yes

Bayesian Classification

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem

- Based on Bayes' rule (or Bayes' theorem) of conditional probability:

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$, where

- P(H) (*prior probability*), the initial probability
  - E.g., X will buy a computer, regardless of age, income, …
- P(X) (*evidence*)*:* probability that a sample data is observed
- P(X|H) (*likelihood*), the probability of observing the sample X, given that the hypothesis holds
  - E.g., Given that X buys a computer, the prob. that X is 31..40, medium income, …

Bayesian Theorem

- A simple example of using Bayes' theorem for inference: If a person has muscle pain, what is the probability that the person has flu?

Bayesian Theorem

- F: A patent has flu, M: A patient has muscle pain
- We know from historical data;

  P(M|F) = 0.75 (if a person catches flu, he/she has muscle pain 75% of the time)

  P(F) = 0.00002 (the probability that a person has flu)

  P(M) = 0.005 (the probability that a person has muscle pain)
- The probability that a person has flu if that person has muscle pain:

$$P(F|M) = \frac{P(M|F)P(F)}{P(M)}$$ = 0.75 * 0.00002 / 0.005 = 0.003

Bayesian Theorem

- Given a data sample X, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = (likelihood x prior)/evidence

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-(attribute value) vector **X** = (x_1, x_2, …, x_n), here x_i is an attribute value of A_i.
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori
- In other words, compute $P(C_1|\mathbf{X})$, $P(C_2|\mathbf{X})$, …, $P(C_m|\mathbf{X})$ and predict **X** belongs to the class with the highest probability.
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

- $P(C_i|\mathbf{X})$ can be computed using Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- *Buys_computer* dataset example: Given that we know properties of a (new) customer (i.e., an evidence X), the probability that the customer will buy a computer (C_1: *buys_computer* = *yes*) or the probability that the customer will not buy a computer (C_2 : *buys_computer* = *no*)

Towards Naïve Bayesian Classifier

- Class label, *buys_computer*, has two values, Y and N:

$$P(class=Y \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid class=Y)P(class=Y)}{P(\mathbf{X})}$$

$$P(class=N \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid class=N)P(class=N)}{P(\mathbf{X})}$$

- And, assign the class with higher probability to X (or we predict the class label of X will be the class with higher probability).

Towards Naïve Bayesian Classifier

- In the equations, P(**X**) is the same for all classes. So we compute and compare only the numerators.

$$P(\mathbf{X}|C_i)P(C_i)$$

- For class label with two values, Y and N, we compute and compare the following two:

$$P(\mathbf{X}|class=Y)P(class=Y)$$

$$P(\mathbf{X}|class=N)P(class=N)$$

Derivation of Naïve Bayes Classifier

- We need to compute $P(X|C_i)$ and $P(C_i)$.
- $P(C_i)$ can be easily estimated from the training dataset

- For the Buys_computer dataset, there are two classes, yes and no. Let yes be C_1 and no be C_2.
- We can estimate $P(C_i)$ as follows:

  - $P(C_1)$ = (# yes tuples) / (total # tuples)
  - $P(C_2)$ = (# no tuples) / (total # tuples)

Derivation of Naïve Bayes Classifier

- Computation of $P(X|C_i)$ is not easy.
- It can be simplified with *class-conditional independence assumption* (a naïve assumption)
- *class-conditional independence* assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):
- This greatly reduces the computation cost

Derivation of Naïve Bayes Classifier

- Based on this assumption, we compute $P(x_k|C_i)$ for each attribute x_i, and multiply them all to obtain $P(X|C_i)$ (this is possible because we assumed all attributes are independent of each other).

- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

(Example follows in the next slides)

Naïve Bayesian Classifier: Training Dataset

Class:

C_1:buys_computer = 'yes'

C_2:buys_computer = 'no'

Data sample

X = (age >40,

Income = high,

Student = no

Credit_rating = excellent)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

- Class prior probabilities are:
  $P(C_1)$ = P(buys_computer = "yes") = 9/14 = 0.643
  $P(C_2)$ = P(buys_computer = "no") = 5/14= 0.357

- Next, we compute $P(X|C_i)$ for each class
  P(age = ">40" | buys_computer = "yes") = 3/9 = 0.333
  P(age = ">40" | buys_computer = "no") = 2/5 = 0.4

  P(income = "high" | buys_computer = "yes") = 2/9 = 0.222
  P(income = "high" | buys_computer = "no") = 2/5 = 0.4

  P(student = "no" | buys_computer = "yes) = 3/9 = 0.333
  P(student = "no" | buys_computer = "no") = 4/5 = 0.8

  P(credit_rating = "excellent" | buys_computer = "yes") = 3/9 = 0.333
  P(credit_rating = "excellent" | buys_computer = "no") = 3/5 = 0.6

Naïve Bayesian Classifier: An Example

P(X|C_i) :

P(X|buys_computer = "yes") = 0.333 x 0.222 x 0.333 x 0.333 = 0.008

P(X|buys_computer = "no") = 0.4 x 0.4 x 0.8 x 0.6 = 0.077

P(X|C_i)*P(C_i) :

P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.008 * 0.643
= 0.005

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.077 * 0.357
= 0.027

Therefore, the model predicts that X belongs to class ("buys_computer = no")

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables:
    - patient_profile: age, height, weight, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

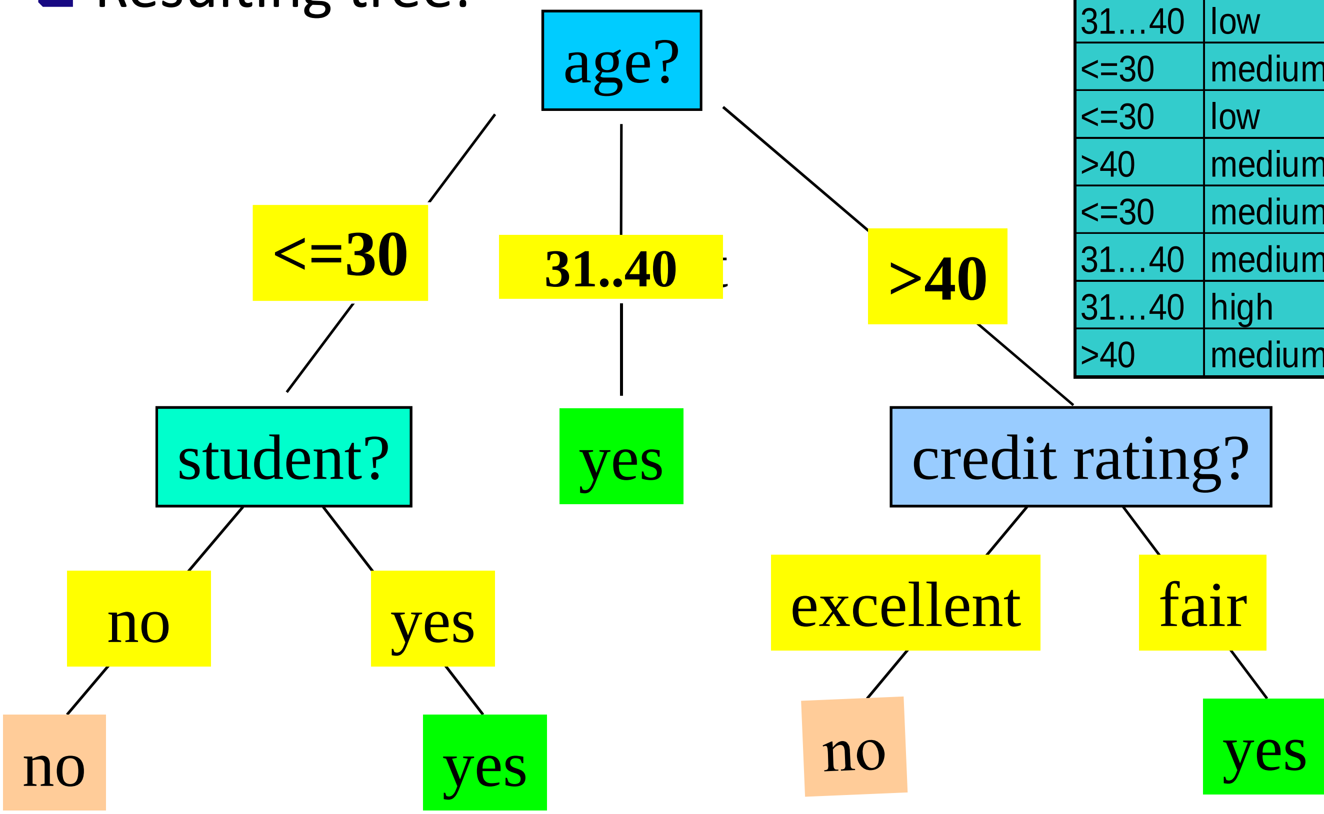
Decision Tree

- A popular classifier
- A model is built as a decision tree

- Internal node represents a test on an attribute
- Branch represents outcome of test
- Leaf node has a class label
- See example in the next slide

Decision Tree Induction Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down, recursive manner
  - Attributes are categorical (if continuous-valued, they are discretized first)
  - At start, all training samples are considered to be at the root.

Algorithm for Decision Tree Induction

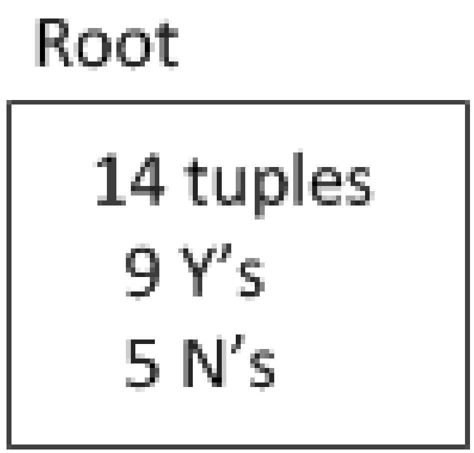
- Basic algorithm (continued)
  - A test attribute is selected on the basis of a heuristic or statistical measure (e.g., information gain)
  - samples are partitioned based on the selected attribute – children nodes are created
  - The same process is repeated at each child node (recursively)

Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

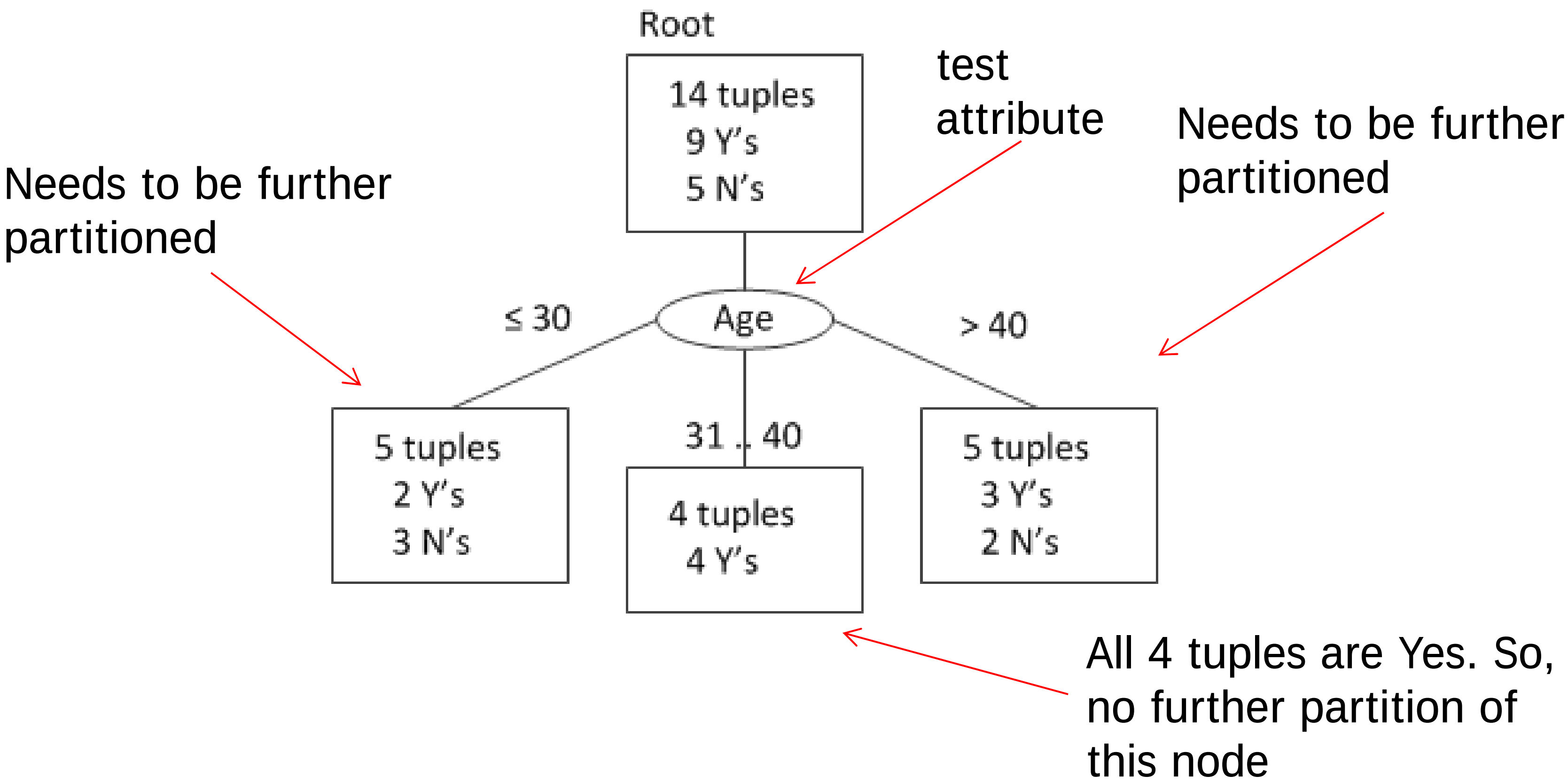
Algorithm for Decision Tree Induction

- Initially all samples are in the same partition, associated with the root node.



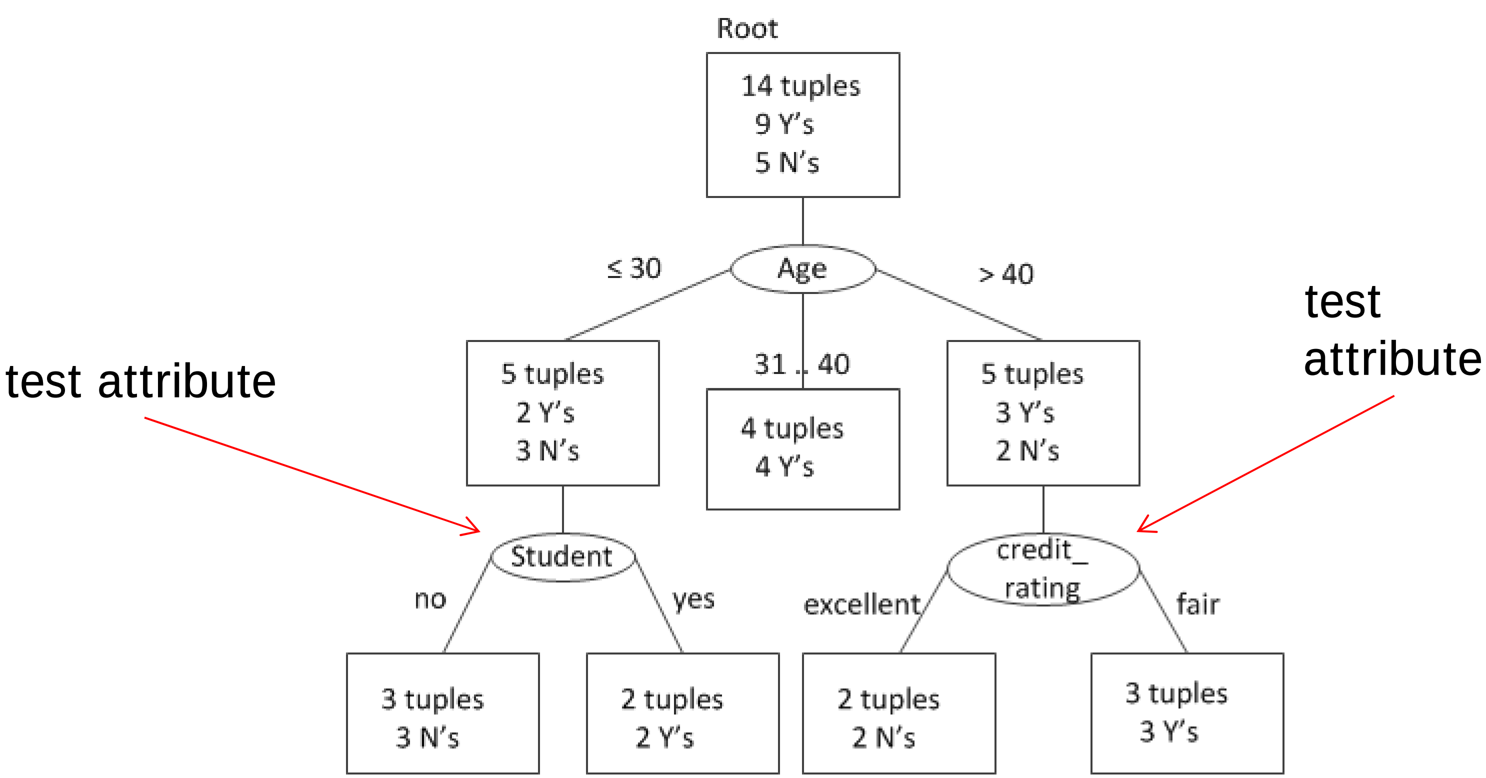
Algorithm for Decision Tree Induction

- Age is chosen as the test attribute, and samples are partitioned into three nodes based on the values of Age, i.e., "≤ 30", "31..40", and ">40."



Algorithm for Decision Tree Induction

- The same process is repeated (recursively) on the left node and on the right node.

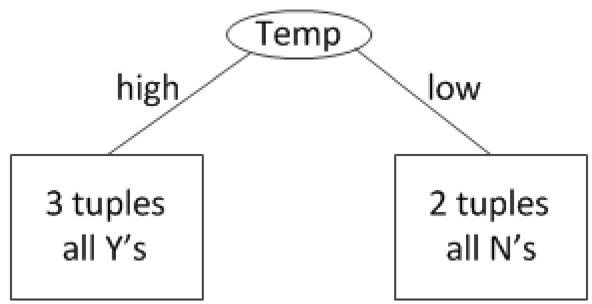


Attribute Selection Measure:

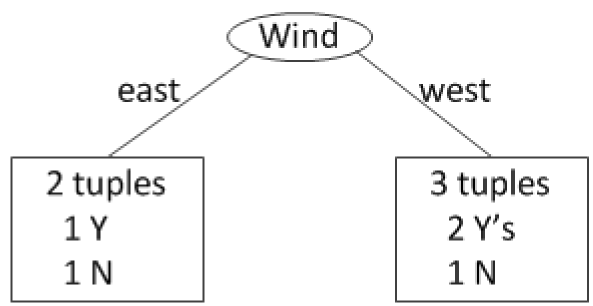
- Consider the following dataset:

Temp	Wind	Class
high	east	Y
low	west	N
low	east	N
high	west	Y
high	west	Y

Split by *Temp*



Split by *Wind*



- Which split is better?

Attribute Selection Measure:

- Test attribute is selected based on “purity measure.”
- Purity measures: information gain, gain ratio, Gini index

- Information gain is based on *info*.
- *Info* represents how pure a dataset is with regard to class labels.
- Suppose a dataset has 10 tuples with two class values, Y and N.
- If the dataset has all Y’s or all N’s, then the dataset is purest and the value of *info* is 0.
- If the dataset has 5 Y’s and 5 N’s, this is the extreme impure case, and the value of *info* is 1.

Attribute Selection Measure: Info Examples

- Notation: $I(x_1, x_2, \ldots, x_m)$, where
  m is the number of distinct class values,
  x_i is the number of tuples belonging to the i-th class
  $|D| = x_1 + x_2 + \ldots + x_m$

- Computation of *info*:

$$I(x_1, x_2, \ldots, x_m) = -\sum_{i=1}^{m} p_i \log_2(p_i), \textit{ where } p_i = \frac{x_i}{|D|}$$

- *Info* is also referred to as *entropy*

Attribute Selection Measure: Info Examples

- Computing log base-2

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2} = \frac{\log_{10} x}{0.301}$$

$$\log_2 0.6 = \frac{\log_{10} 0.6}{\log_{10} 2} = \frac{-0.2218}{0.301} = -0.7369$$

Attribute Selection Measure: Info Examples

- A dataset has 6 tuples with 5 Y's and 1 N:

$$I(5,1) = -\frac{5}{6}\log_2(\frac{5}{6}) - \frac{1}{6}\log_2(\frac{1}{6}) = 0.650$$

- A dataset has 10 tuples with all Y's

$$I(10,0) = -\frac{10}{10}\log_2(\frac{10}{10}) - \frac{0}{10}\log_2(\frac{0}{10}) = 0$$

- A dataset has 10 tuples with 5 Y's and 5 N's

$$I(5,5) = -\frac{5}{10}\log_2(\frac{5}{10}) - \frac{5}{10}\log_2(\frac{5}{10}) = 1$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Compute the information gain of each attribute
- Select the attribute with the highest information gain
- Informal description of *information gain*:
  - *Info*(*D*): The amount of information we need to classify a sample in *D*.

    Example: we need 0.9
  - $Info_A(D)$: After splitting *D* on an attribute *A*, the amount of information needed to classify a sample.

    Example: after splitting on A, now we need only 0.3
  - $Gain(A) = Info(D) – Info_A(D)$.

    Example: the gain is 0.9 – 0.3 = 0.6

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:
  - m: # classes

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed to classify D, after using attribute A to split D into v partitions:
  - v: # distinct values of A

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by splitting by A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Let *D* be the Buys_computer dataset

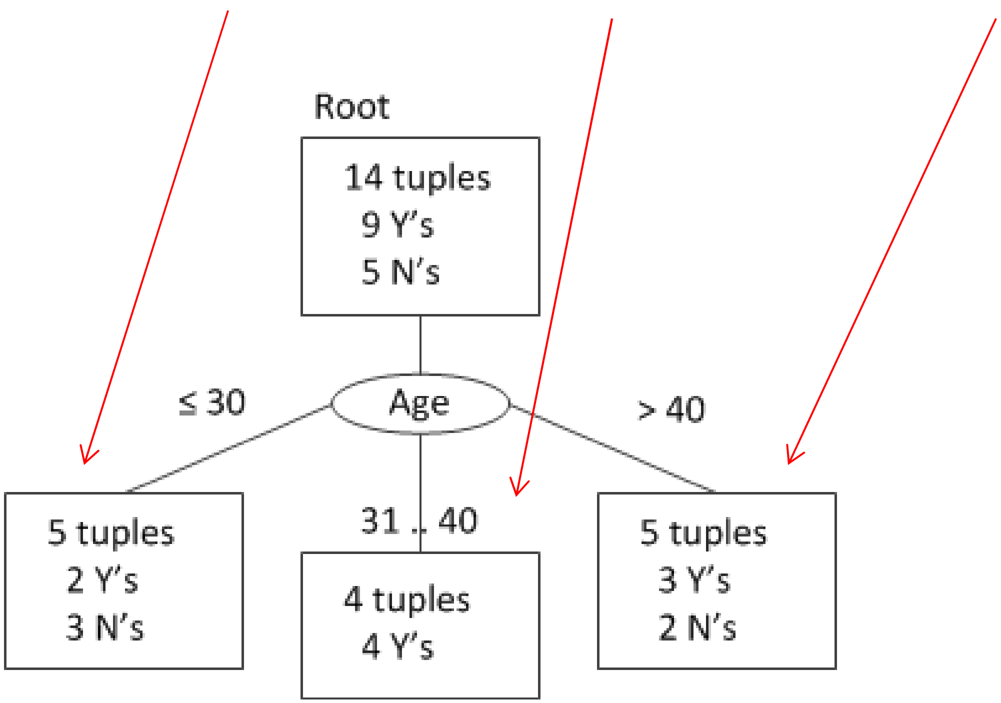
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- There are 14 tuples in *D*
- 9 Yes's and 5 No's

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- If we split D based on Age, three partitions are created.
- Let's call them Node-1 (≤ 30), Node-2 (31..40), and Node-3(> 40).



Attribute Selection Measure: Information Gain (ID3/C4.5)

- Info of Node-1 is: $I(2,3) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0.971$

- Info of Node-2 is: $I(4,0) = 0$

- Info of Node-3 is: $I(3,2) = I(2,3) = 0.971$

- After splitting by Age, the amount of information needed to classify *a* tuple is computed as the weighted average of above three:

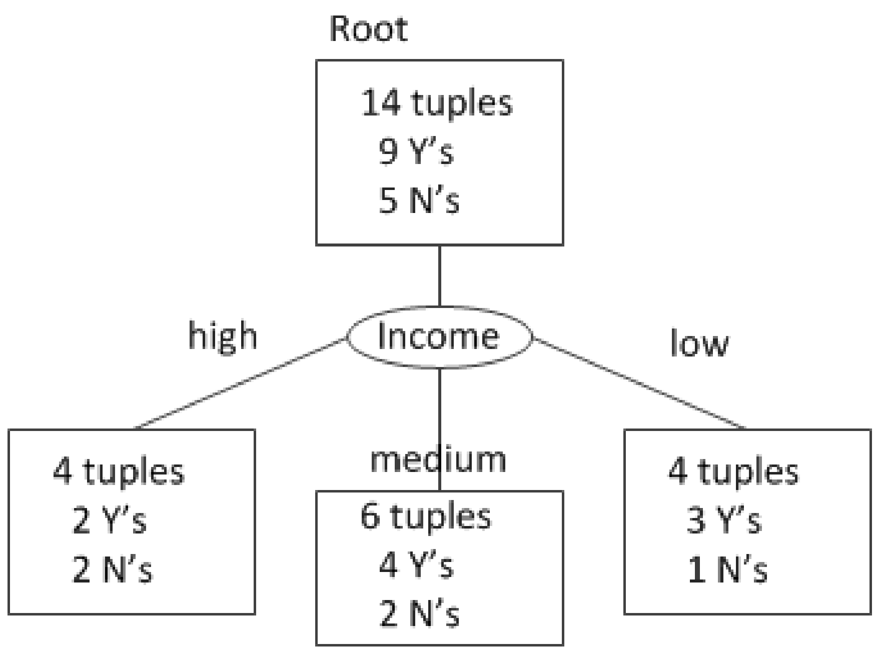
$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

- So, information gain obtained by splitting by Age is:

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Information Gain – Another Example

- If we split by income



Information Gain – Another Example

- high :4 tuples; 2 yes's and 2 no's
- medium: 6 tuples; 4 yes's and 2 no's
- low: 4 tuples; 3 yes's and 1 no

$$Info_{income}(D) = \frac{4}{14} I(2,2) + \frac{6}{14} I(4,2) + \frac{4}{14} I(3,1) = 0.911$$

$$I(2,2) = -\frac{2}{4}\log_2(\frac{2}{4}) - \frac{2}{4}\log_2(\frac{2}{4}) = 1.0 \quad I(4,2) = -\frac{4}{6}\log_2(\frac{4}{6}) - \frac{2}{6}\log_2(\frac{2}{6}) = 0.918$$

$$I(3,1) = -\frac{3}{4}\log_2(\frac{3}{4}) - \frac{1}{4}\log_2(\frac{1}{4}) = 0.811$$

$$Gain(income) = Info(D) - Info_{income}(D) = 0.940 - 0.911 = 0.029$$

Attribute Selection: Information Gain

- We can compute *Gain*(*student*) and *Gain*(*credit_rating*) in the same way.
- Then, we have:

  Gain(*Age*) = 0.246

  Gain(*income*) = 0.029

  Gain(*student*) = 0.151

  Gain(*credit_rating*) = 0.048
- We select *Age* as the test attribute because it has the highest information gain.

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem.

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

Here, attribute *A* has *v* distinct values, $|D_j|$ is the number of tuples with the *j*-th value.

- $GainRatio(A) = Gain(A)/SplitInfo_A(D)$
- The attribute with the maximum gain ratio is selected as the splitting attribute.

Other Attribute Selection Measures

- GINI index
- CHAID
- C-SEPG-statisticMDL (Minimal Description Length) principle
- Multivariate splits (partition based on multiple variable combinations)
  - Example: CART
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

- Overfitting:
  - A model reflects every details of the training dataset, even an anomaly or noise.
  - A model becomes complex.
  - Accuracy on the training dataset is high.
  - Accuracy on the test dataset becomes low.
  - So, an overfitted model does not generalize well.

Overfitting and Tree Pruning

- Overfitting of decision tree:
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- 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 (more common): *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”

References

- Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012
- http://www.cs.illinois.edu/~hanj/bk3/
- Ian H. Witten, E. Frank, and M.A. Hall, "Data Mining Practical Machine Learning Tools and Techniques," Third Ed., 2011, Morgan Kaufmann