

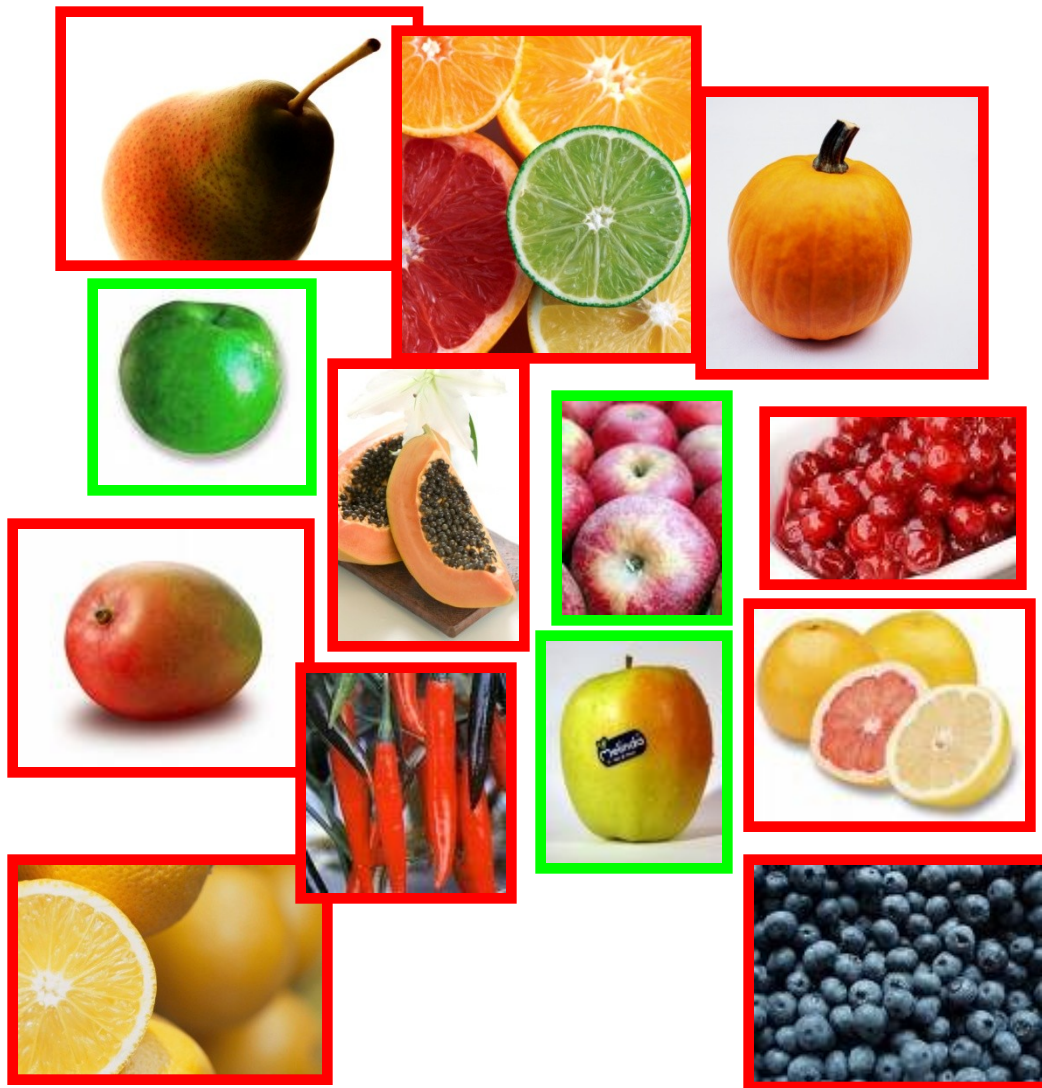


Nearest Neighbor & Bayesian Classifiers

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

- ❑ Inductive-based learning
- ❑ Naïve Bayes Classifiers
- ❑ Bayesian Belief Networks

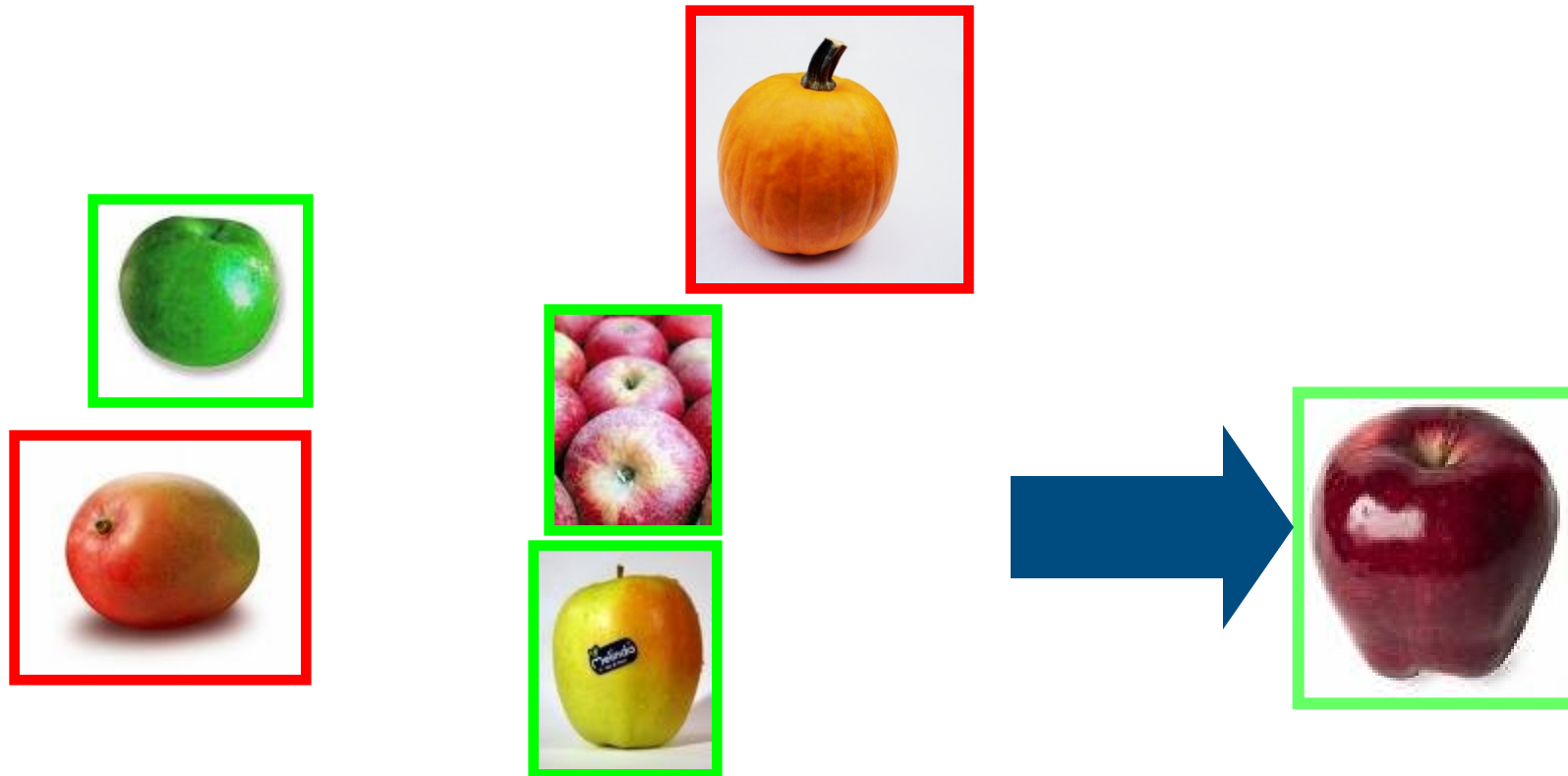
Inductive-based learning



Is this an apple?

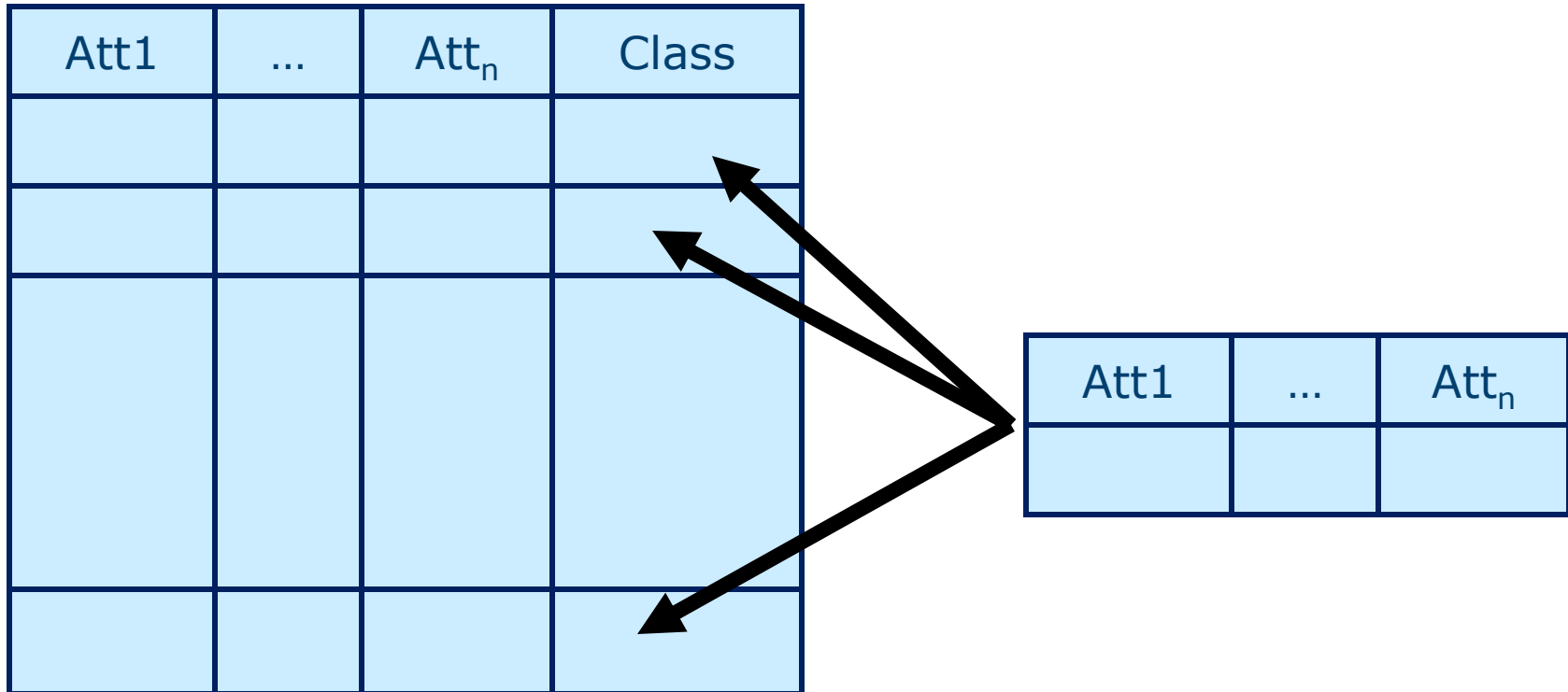


- ❑ To decide whether an unseen fruit is an apple, consider the k fruits that are more similar to the unknown fruit
- ❑ Then, classify the unknown fruit using the most frequent class



- ❑ If $k=5$, these 5 fruits are the most similar ones to the unclassified example
- ❑ Since, the majority of the fruits are apples, we decide that the unknown fruit is an apple

- ❑ Store the training records
- ❑ Use the training records to predict the class label of unseen cases



- ❑ Instance-based methods are the simplest form of learning
- ❑ Training instances are searched for instance that most closely resembles new instance
- ❑ The instances themselves represent the knowledge
- ❑ Similarity (distance) function defines what's "learned"
- ❑ "lazy evaluation", until a new instance must be classified
- ❑ Instance-based learning is lazy learning

- ❑ Rote-learner
 - ▶ Memorizes entire training data
 - ▶ Performs classification only if attributes of record match one of the training examples exactly

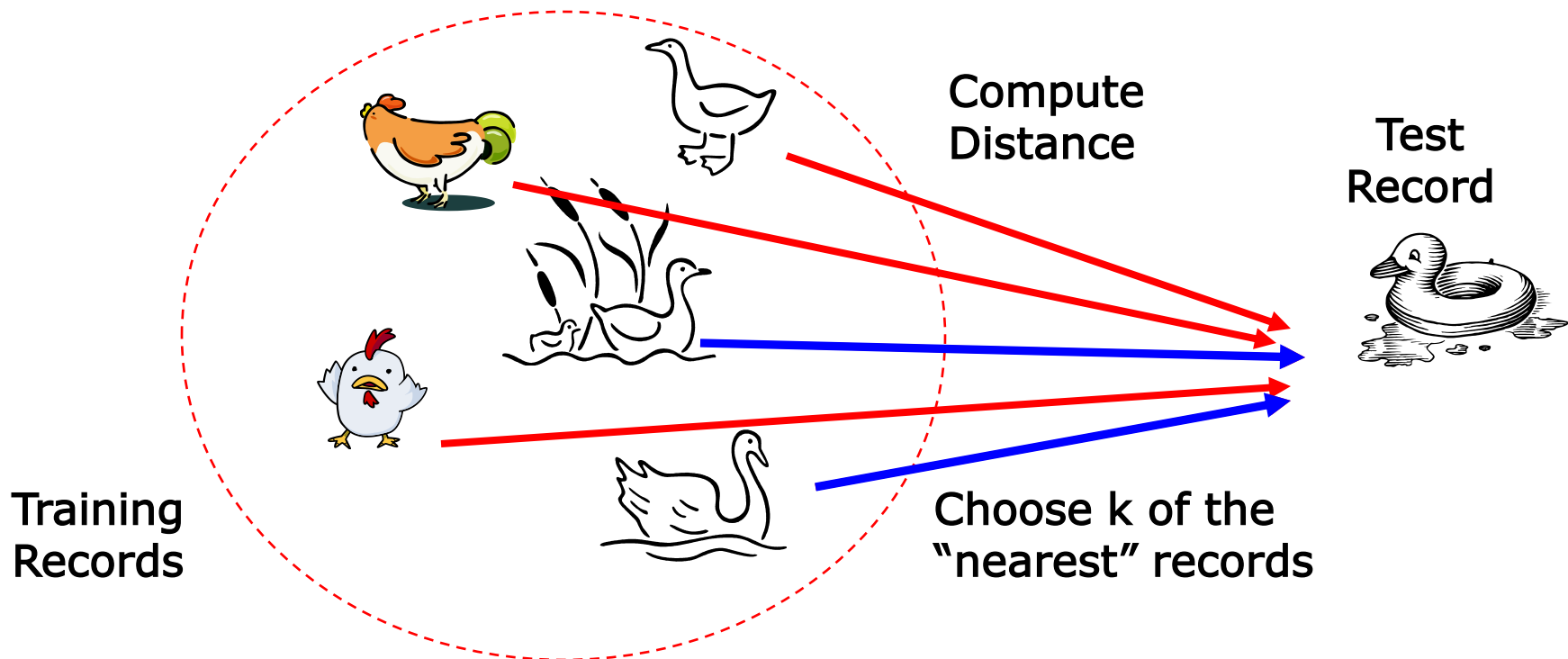
- ❑ Nearest neighbor
 - ▶ Uses k “closest” points (nearest neighbors) for performing classification

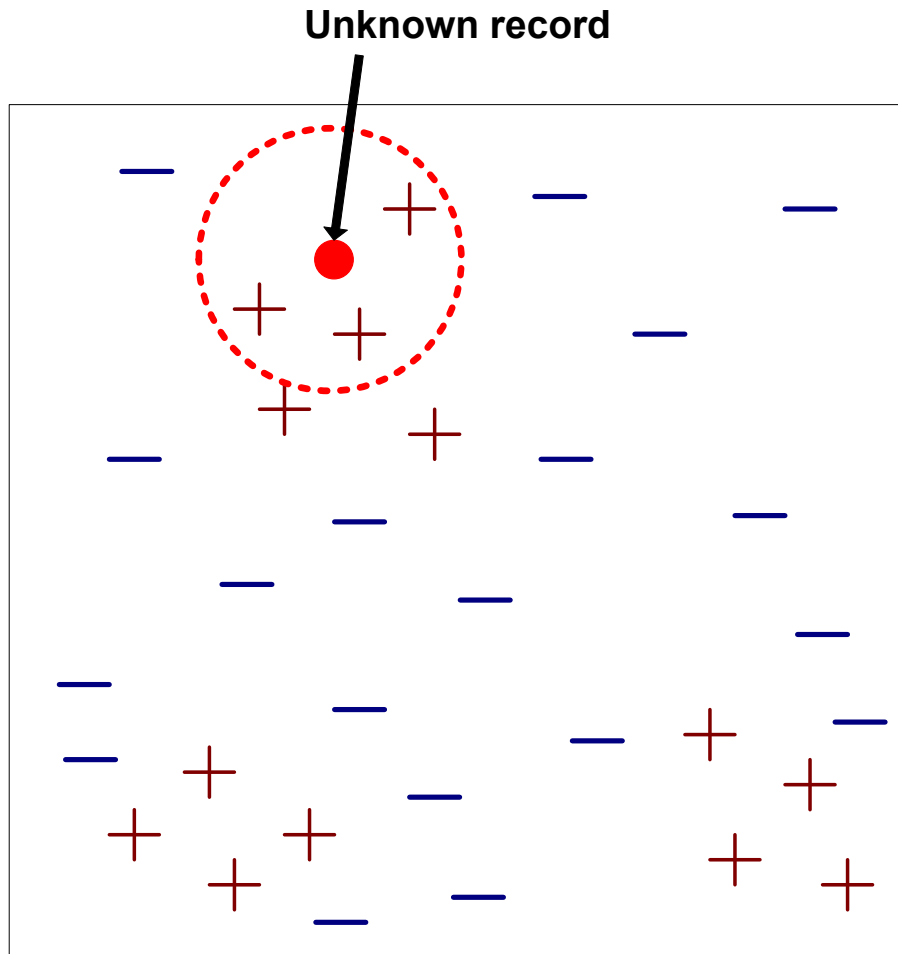
- ❑ Rote Learning is basically memorization
 - ▶ Saving knowledge so it can be used again
 - ▶ Retrieval is the only problem
 - ▶ No repeated computation, inference or query is necessary

- ❑ A simple example of rote learning is caching
 - ▶ Store computed values (or large piece of data)
 - ▶ Recall this information when required by computation
 - ▶ Significant time savings can be achieved
 - ▶ Many AI programs (as well as more general ones) have used caching very effectively

□ Basic idea:

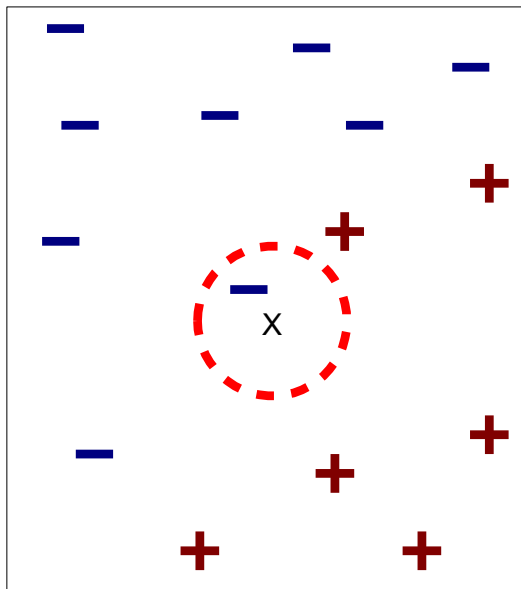
- ▶ If it walks like a duck, quacks like a duck, then it's probably a duck



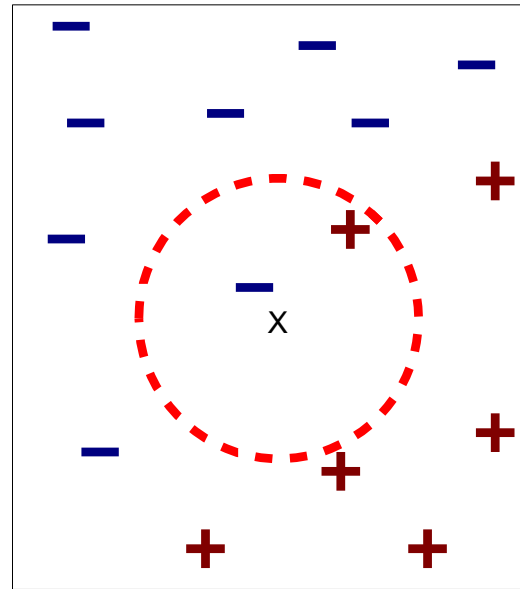


- Requires three things
 - ▶ The set of stored records
 - ▶ Distance Metric to compute distance between records
 - ▶ The value of k , the number of nearest neighbors to retrieve

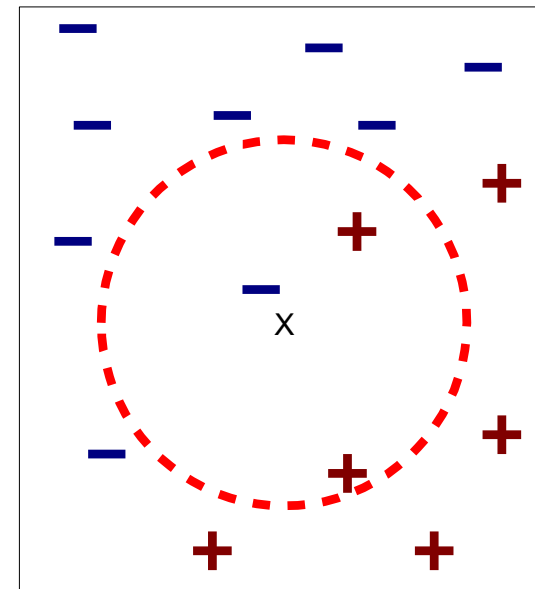
- To classify an unknown record:
 - ▶ Compute distance to other training records
 - ▶ Identify k nearest neighbors
 - ▶ Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

- ❑ To compute the similarity between two points, use for instance the Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- ❑ Determine the class from nearest neighbor list
 - ▶ take the majority vote of class labels among the k-nearest neighbors
 - ▶ Weigh the vote according to distance, e.g., the weight factor, $w = 1/d^2$
- ❑ Taking the square root is not required when comparing distances
- ❑ Another popular metric is *city-block (Manhattan)* metric, distance is the sum of absolute differences

- Different attributes are measured on different scales
- Attributes need to be *normalized*:

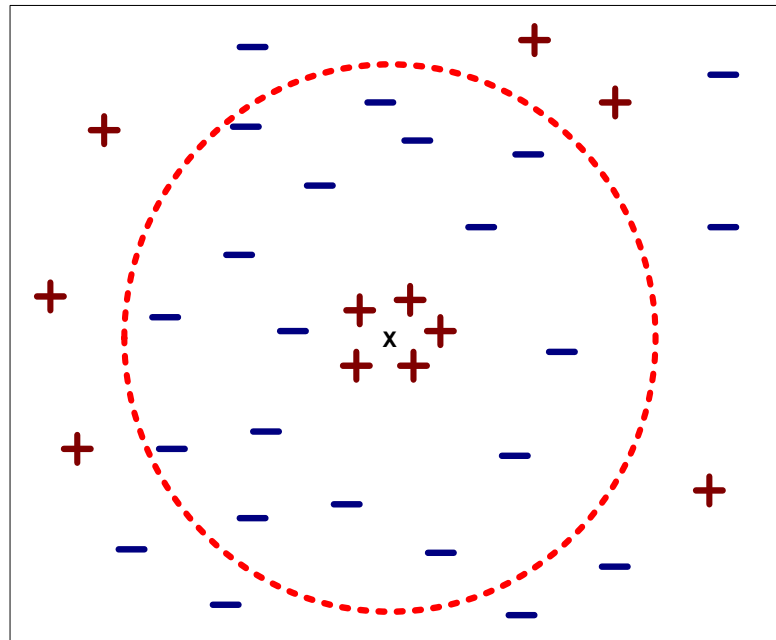
$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

$$a_i = \frac{v_i - Avg(v_i)}{StDev(v_i)}$$

v_i : the actual value of attribute i

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

- ❑ Choosing the value of k :
 - ▶ If k is too small, sensitive to noise points
 - ▶ If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification: Scaling Issues

- ❑ Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- ❑ Example:
 - ▶ height of a person may vary from 1.5m to 1.8m
 - ▶ weight of a person may vary from 90lb to 300lb
 - ▶ income of a person may vary from \$10K to \$1M

- ❑ Problem with Euclidean measure:
 - ▶ High dimensional data, **curse of dimensionality**
 - ▶ Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1

$d = 1.4142$

VS

1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1

$d = 1.4142$

- Solution: Normalize the vectors to unit length

- k-NN classifiers are lazy learners
 - ▶ It does not build models explicitly
 - ▶ Unlike eager learners such as decision tree induction and rule-based systems
 - ▶ Classifying unknown records are relatively expensive

- ❑ Simplest case: one numeric attribute
 - ▶ Distance is the difference between the two attribute values involved (or a function thereof)

- ❑ Several numeric attributes: normally, Euclidean distance is used and attributes are normalized

- ❑ Nominal attributes: distance is set to 1 if values are different, 0 if they are equal

- ❑ Are all attributes equally important?
 - ▶ Weighting the attributes might be necessary

- ❑ Often very accurate
- ❑ But slow, since simple version scans entire training data to derive a prediction
- ❑ Assumes all attributes are equally important, thus may need attribute selection or weights
- ❑ Possible remedies against noisy instances:
 - ▶ Take a majority vote over the k nearest neighbors
 - ▶ Removing noisy instances from dataset (difficult!)
- ❑ Statisticians have used k -NN since early 1950s
 - ▶ If $n \rightarrow \infty$ and $k/n \rightarrow 0$, error approaches minimum

- ❑ Also uses and lazy evaluation and analyzes similar instances
- ❑ But instances are not “points in a Euclidean space”
- ❑ Methodology
 - ▶ Instances represented by rich symbolic descriptions (e.g., function graphs)
 - ▶ Multiple retrieved cases may be combined
 - ▶ Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- ❑ Research issues
 - ▶ Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

- ❑ Instance-based learning is based on lazy evaluation
- ❑ Decision-tree and Bayesian classification are based on eager evaluation
- ❑ Key differences
 - ▶ Lazy method may consider query instance x_q when deciding how to generalize beyond the training data D
 - ▶ Eager method cannot since they have already chosen global approximation when seeing the query

❑ Efficiency

- ▶ Lazy learning needs less time for training but more time predicting

❑ Accuracy

- ▶ Lazy methods effectively use a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
- ▶ Eager methods must commit to a single hypothesis that covers the entire instance space

- ❑ PEBLS: Parallel Exemplar-Based Learning System (Cost & Salzberg)
- ❑ Works with both continuous and nominal features
- ❑ For nominal features, distance between two nominal values is computed using **modified value difference metric**
- ❑ Each record is assigned a weight factor
- ❑ Number of nearest neighbor, $k = 1$

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Distance between nominal attribute values:

$$d(\text{Single}, \text{Married}) = |2/4 - 0/4| + |2/4 - 4/4| = 1$$

$$d(\text{Single}, \text{Divorced}) = |2/4 - 1/2| + |2/4 - 1/2| = 0$$

$$d(\text{Married}, \text{Divorced}) = |0/4 - 1/2| + |4/4 - 1/2| = 1$$

$$d(\text{Refund}=\text{Yes}, \text{Refund}=\text{No}) = |0/3 - 3/7| + |3/3 - 4/7| = 6/7$$

$$d(V_1, V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|$$

Class	Marital Status		
	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

Class	Refund	
	Yes	No
Yes	0	3
No	3	4

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
X	Yes	Single	125K	No
Y	No	Married	100K	No

Distance between record X and record Y:

$$\Delta(X, Y) = w_X w_Y \sum_{i=1}^d d(X_i, Y_i)^2$$

where: $w_X = \frac{\text{Number of times X is used for prediction}}{\text{Number of times X predicts correctly}}$

$w_X \cong 1$ if X makes accurate prediction most of the time

$w_X > 1$ if X is not reliable for making predictions

Naïve Bayes classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C|A) = \frac{P(A \wedge C)}{P(A)}$$

$$P(A|C) = \frac{P(A \wedge C)}{P(C)}$$

- Bayes theorem,

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

- A priori probability of C: $P(C)$
Probability of event before evidence is seen
- A posteriori probability of C: $P(C|A)$
Probability of event after evidence is seen

- ❑ What's the probability of the class given an instance?
- ❑ Evidence E = instance, represented as a tuple of attributes $\langle e_1, \dots, e_n \rangle$
- ❑ Event H = class value for instance
- ❑ We are looking for the class value with the highest probability for E
- ❑ I.e., we are looking for the hypothesis that has the highest probability to explain the evidence E

$$class = \arg \max_{h \in H} P(H|E)$$

- Given the hypothesis H and the example E described by n attributes, Bayes Theorem says that

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

- Naïve assumption: attributes are statistically independent
- Evidence splits into parts that are **independent**

$$P(H|E) = \frac{P(e_1|H) \cdots P(e_n|H)P(H)}{P(E)}$$

- **Training:** the class probability $P(H)$ and the conditional probability $P(e_i|H)$ for each attribute value e_i and each class value H
- **Testing:** given an instance E , the class is computed as

$$\begin{aligned} \text{class} &= \arg \max_{h \in H} P(H|E) \\ &= \arg \max_{h \in H} \frac{P(e_1|H) \cdots P(e_n|H)P(H)}{P(E)} \\ &= \arg \max_{h \in H} P(e_1|H) \cdots P(e_n|H)P(H) \end{aligned}$$

- ❑ “Opposite” of OneRule: use all the attributes
- ❑ Two assumptions
 - ▶ Attributes are equally important
 - ▶ Attribute are statistically independent
- ❑ I.e., knowing the value of one attribute says nothing about the value of another (if the class is known)
- ❑ Independence assumption is almost never correct!
- ❑ But this scheme works well in practice

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Outlook	Temperature		Humidity			Windy		Play					
	Yes	No	Yes	No	Yes	No	Yes	No					
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

What is the value of "play"?

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$\begin{aligned}
 P(\text{yes}|E) &= P(\text{Sunny}|\text{yes})P(\text{Cool}|\text{yes})P(\text{High}|\text{yes})P(\text{True}|\text{yes}) \\
 &= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} \\
 &= 0.0053
 \end{aligned}$$

$$\begin{aligned}
 P(\text{no}|E) &= P(\text{Sunny}|\text{no})P(\text{Cool}|\text{no})P(\text{High}|\text{no})P(\text{True}|\text{no}) \\
 &= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} \\
 &= 0.0206
 \end{aligned}$$

Conversion into a probability by normalization:

$$P(\text{yes}) = 0.0053 / (0.0053 + 0.0206) = 0.205$$

$$P(\text{no}) = 0.0206 / (0.0053 + 0.0206) = 0.795$$

- ❑ What if an attribute value does not occur with every class value?

- ❑ For instance, “Humidity = high” for class “yes”
 - ▶ Probability will be zero!

$$P(\textit{Humidity} = \textit{high}|\textit{yes}) = 0$$

- ▶ A posteriori probability will also be zero!
(No matter how likely the other values are!)

$$P(\textit{yes}|\langle \dots, \textit{Humidity} = \textit{high}, \dots \rangle) = 0$$

- ❑ Remedy: add 1 to the count for every attribute value-class combination (Laplace estimator)
- ❑ Result: probabilities will never be zero!
(also: stabilizes probability estimates)

- n number of examples with class H
- n_c number of examples with class H and attribute e
- The m -estimate for $P(e|H)$ is computed as,

$$P(e|h) = \frac{n_c + mp}{n + m}$$

- Where m is a parameter known as the equivalent sample size
- p is a user specified parameter
 - ▶ If n is zero, then $P(e|H) = p$, thus p is the prior probability of observing e
 - ▶ $p = 1/k$ given k values of the attribute considered

- ❑ **Training**: instance is not included in frequency count for attribute value-class combination
- ❑ **Testing**: attribute will be omitted from calculation
- ❑ **Example**:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Likelihood of "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Likelihood of "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

$P(\text{"yes"}) = 0.0238 / (0.0238 + 0.0343) = 41\%$

$P(\text{"no"}) = 0.0343 / (0.0238 + 0.0343) = 59\%$

- Usual assumption: attributes have a normal or Gaussian probability distribution (given the class)
- The probability density function for the normal distribution is defined by two parameters:

- ▶ Sample mean,

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

- ▶ Standard deviation,

$$\sigma = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2$$

- ▶ Then the density function $f(x)$ is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

	Outlook		Temperature		Humidity			Windy		Play	
	Yes	No	Yes	No	Yes	No		Yes	No	Yes	No
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	$\mu = 73$	$\mu = 75$	$\mu = 79$	$\mu = 86$	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	$\sigma = 6.2$	$\sigma = 7.9$	$\sigma = 10.2$	$\sigma = 9.7$	True	3/9	3/5		
Rainy	3/9	2/5									

□ Example density value:

$$f(\text{temperature} = 66 | \text{yes}) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2 \times 6.2^2}} = 0.0340$$

- A new day,

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

- Missing values during training are not included in calculation of mean and standard deviation

Likelihood of "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

$P(\text{"yes"}) = 0.000036 / (0.000036 + 0.000136) = 20.9\%$

$P(\text{"no"}) = 0.000136 / (0.000036 + 0.000136) = 79.1\%$

- ❑ Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- ❑ Why? Because classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class
- ❑ However: adding too many redundant attributes will cause problems (e.g. identical attributes)
- ❑ Also, many numeric attributes are not normally distributed

Bayesian Belief Networks

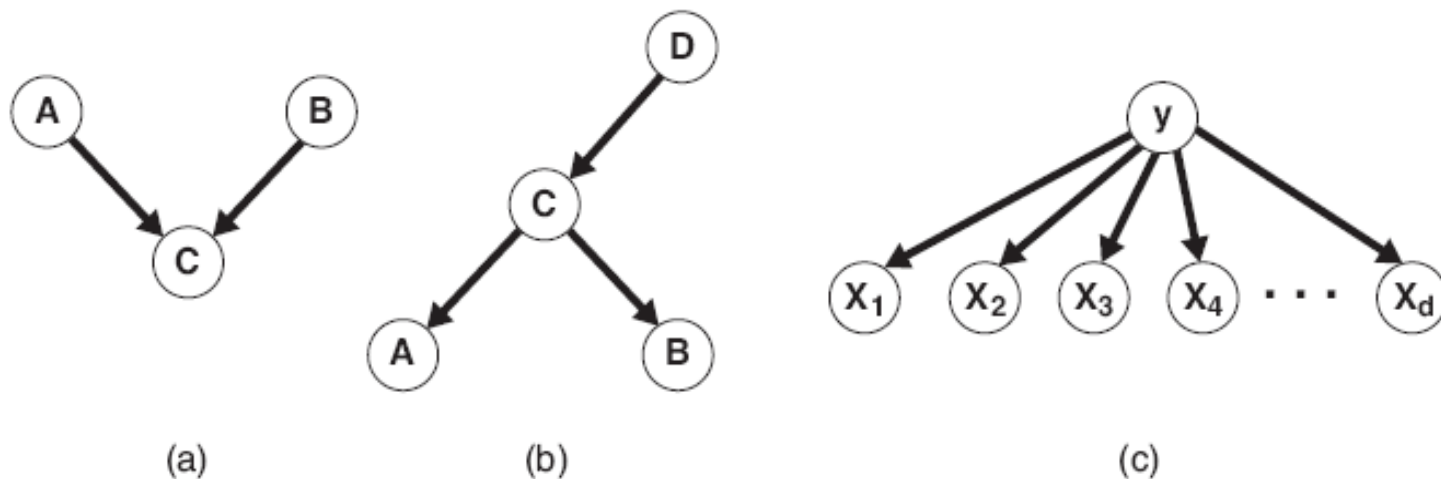
- ❑ The conditional independence assumption,
 - ▶ makes computation possible
 - ▶ yields optimal classifiers when satisfied
 - ▶ but is seldom satisfied in practice, as attributes (variables) are often correlated

- ❑ Bayesian Belief Networks (BBN) allows us to specify which pair of attributes are conditionally independent

- ❑ They provide a graphical representation of probabilistic relationships among a set of random variables

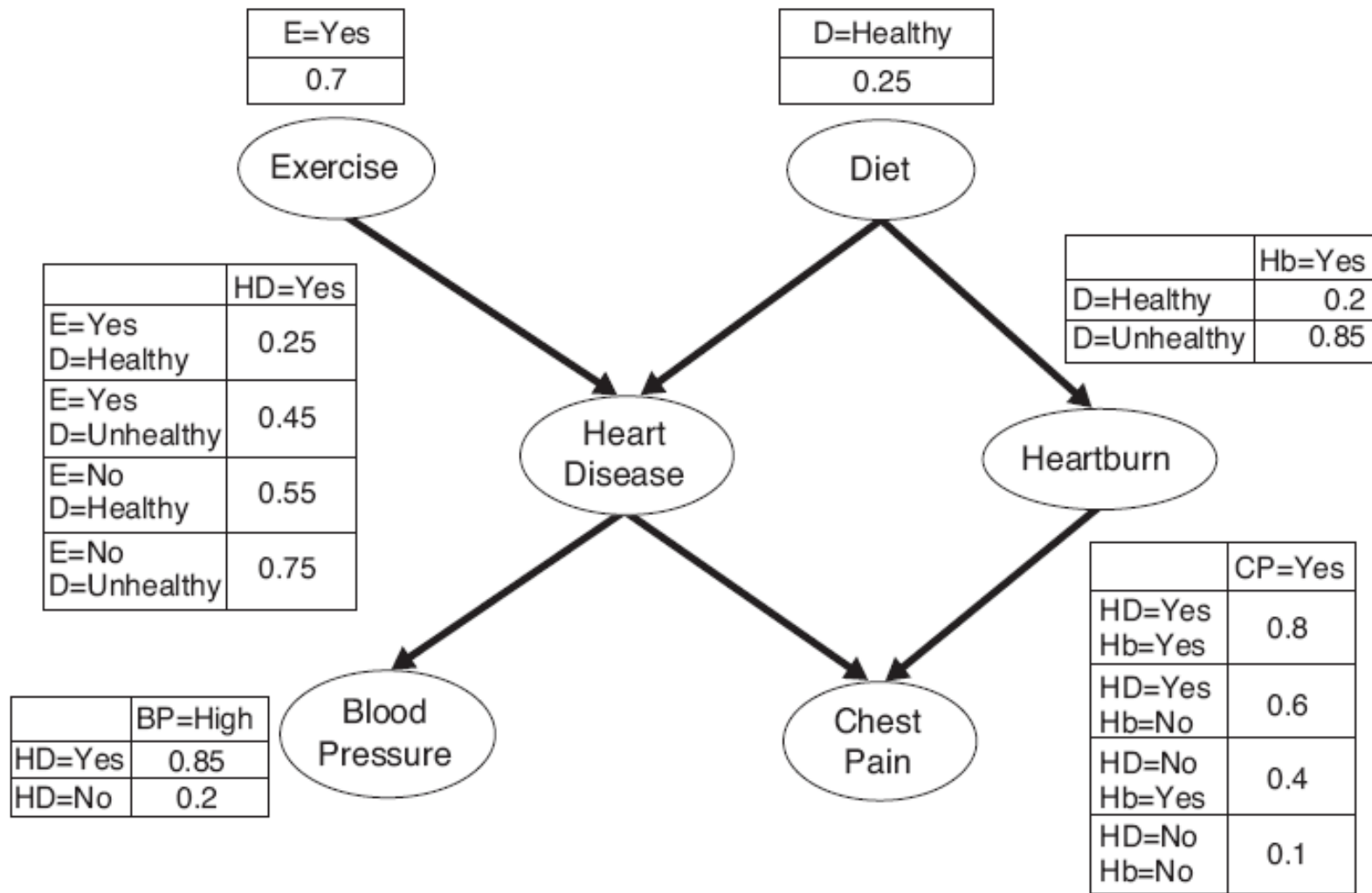
- ❑ Describe the probability distribution governing a set of variables by specifying
 - ▶ Conditional independence assumptions that apply on subsets of the variables
 - ▶ A set of conditional probabilities

- ❑ Two key elements
 - ▶ A direct acyclic graph, encoding the dependence relationships among variables
 - ▶ A probability table associating each node to its immediate parents node



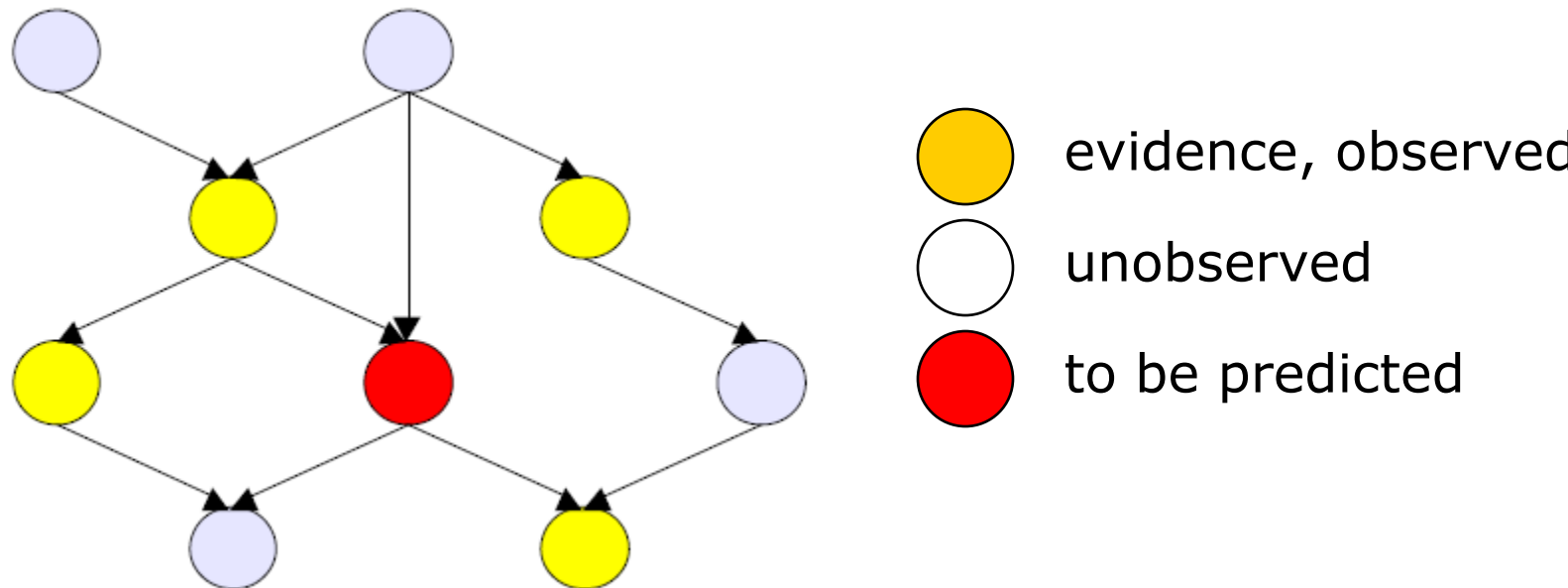
- a) Variable A and B are independent, but each one of them has an influence on the variable C
- b) A is conditionally independent of both B and D, given C
- c) The configuration of the typical naïve Bayes classifier

- ❑ The network topology imposes conditions regarding the variable conditional independence
- ❑ Each node is associated with a probability table
 - ▶ If a node X does not have any parents, then the table contains only the prior probability $P(X)$
 - ▶ If a node X has only one any parent Y , then the table contains only the conditional probability $P(X|Y)$
 - ▶ If a node X has multiple parents, Y_1, \dots, Y_k the table contains the conditional probability $P(X|Y_1 \dots Y_k)$



- ❑ Let $T = \{X_1, X_2, \dots, X_d\}$ denote a total order of the variables
- ❑ **for** $j = 1$ **to** d **do**
 - ▶ Let $X_{T(j)}$ denote the j th highest order variable in T
 - ▶ Let $\pi(X_{T(j)}) = \{X_{T(1)}, \dots, X_{T(j-1)}\}$ denote the variables preceding $X_{T(j)}$
 - ▶ Remove the variables from $\pi(X_{T(j)})$ that do not affect $X_{T(j)}$
 - ▶ Create an arc between $X_{T(j)}$ and the remaining variables in $\pi(X_{T(j)})$
- ❑ **done**

- ❑ Guarantees a topology that does not contain cycles
- ❑ Different orderings generate different networks
- ❑ In principles $d!$ possible networks to test



- In general the inference is NP-complete but there are approximating methods, e.g. Monte-Carlo

- ❑ Bayesian belief network allows a subset of the variables conditionally independent
- ❑ A graphical model of causal relationships
- ❑ Several cases of learning Bayesian belief networks
 - ▶ Given both network structure and all the variables is easy
 - ▶ Given network structure but only some variables
 - ▶ When the network structure is not known in advance

- ❑ BBN provides an approach for capturing prior knowledge of a particular domain using a graphical model
- ❑ Building the network is time consuming, but adding variables to a network is straightforward
- ❑ They can encode causal dependencies among variables
- ❑ They are well suited to dealing with incomplete data
- ❑ They are robust to overfitting

Summary

- ❑ Instance-based methods
 - ▶ Simplest form of learning
 - ▶ Training instances are searched for instance that most closely resembles new instance
 - ▶ The instances themselves represent the knowledge
 - ▶ Similarity (distance) function defines what's "learned"

- ❑ Naïve Bayes Classifiers
 - ▶ Compute the probability of a class label by computing its probability $P(H)$ and the conditional probability $P(e_i|H)$ for each attribute value e_i and each class value H
 - ▶ Assume that attributes are independent
 - ▶ Assume a normal distribution for numerical attributes

- ❑ Bayesian Belief Networks
 - ▶ Introduces graph structure to model dependencies among attributes
 - ▶ Structure + Conditional probabilities model the data
 - ▶ What structure?