2nd approach: Gain Ratio

- **Problem of Information gain approach**
  - Biased towards tests with many outcomes (attributes having a large number of values)
  - E.g: attribute acting as unique identifier
    - Produce a **large number of partitions** (1 tuple per partition)
    - Each resulting partition D is **pure** Info(D)=0
    - The **information gain** is **maximized**
- **Extension to Information Gain**
  - C4.5, a successor of ID3 uses an extension to information gain known as **gain ratio**
  - Overcomes the bias of Information gain
  - Applies a kind of normalization to information gain using a **split information** value

2^{nd} approach: Gain Ratio

- The **split information value** represents the potential information generated by splitting the training data set **D** into **v** partitions, corresponding to v outcomes on attribute **A**

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

  - → **High splitInfo**: partitions have more or less the same size (uniform)
  - → **Low split Info**: few partitions hold most of the tuples (peaks)
- The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

- The attribute with the maximum gain ratio is selected as the splitting attribute

Gain Ratio: Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Using attribute income

1st partition (low) **D1** has **4 tuples**

2nd partition (medium) **D2** has **6 tuples**

3rd partition (high) **D3** has **4 tuples**

$$Gain(income) = 0.029$$

$$GainRatio(income) = \frac{0.029}{0.926} = 0.031$$

$$SplitInfo_{income}(D) = -\frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{6}{14}\log_2\left(\frac{6}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right)$$

$$= 0.926$$

3rd approach: Gini Index

- The Gini Index (used in CART) measures the impurity of a data partition **D**

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

  - → **m**: the number of classes
  - → $\mathbf{p_i}$: the probability that a tuple in D belongs to class Ci
- The Gini Index considers a **binary split** for each attribute **A**, say D_1 and D_2. The **Gini index** of D given that partitioning is:

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

  - → A weighted sum of the impurity of each partition
- The reduction in impurity is given by

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity is chosen as the splitting attribute

Binary Split: Continuous-Valued Attributes

- **D**: a data patition
- Consider attribute **A** with continuous values
- To determine the best binary split on **A**

What to examine?

- → Examine each possible split point
- → The midpoint between each pair of (sorted) adjacent values is taken as a possible split-point

How to examine?

- → For each split-point, compute the weighted sum of the impurity of each of the two resulting partitions (D1: A<=split-point, D2: A> split-point)

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

- → The split-point that gives the minimum Gini index for attribute A is selected as its splitting subset

Binary Split: Discrete-Valued Attributes

- **D**: a data pation
- Consider attribute **A** with **v** outcomes **$\{a_1,\dots,a_v\}$**
- To determine the best binary split on **A**

What to examine?

- Examine the partitions resulting from all possible subsets of **$\{a_1,\dots,a_v\}$**
- Each subset **S_A** is a binary test of attribute **A** of the form **"$A \in S_A$?"**
- **2^v** possible subsets. We exclude the power set and the empty set, then we have **2^v-2** subsets

How to examine?

- For each subset, compute the weighted sum of the impurity of each of the two resulting partitions

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

- The subset that gives the minimum Gini index for attribute A is selected as its splitting subset

Gini(income)

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Compute the Gini index of the training set D: 9 tuples in class yes and 5 in class no

$$Gini(D) = 1 - \left(\left(\frac{9}{14} \right)^2 + \left(\frac{5}{14} \right)^2 \right) = 0.459$$

Using attribute income: there are three values: **low, medium** and **high**

Choosing the subset {low, medium} results in two partions:

- **D1 (income $\in$ {low, medium}):** 10 tuples
- **D2 (income $\in$ {high}):** 4 tuples

Gini(income)

$$Gini_{income \in \{low, median\}}(D) = \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$
$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right)$$
$$= 0.450$$
$$= Gini_{income \in \{high\}}(D)$$

The Gini Index measures of the remaining partitions are:

$$Gini_{\{low, high\} and \{medium\}}(D) = 0.315$$
$$Gini_{\{medium, high\} and \{low\}}(D) = 0.300$$

Therefore, the best binary split for attribute income is on {medium, high} and {low}

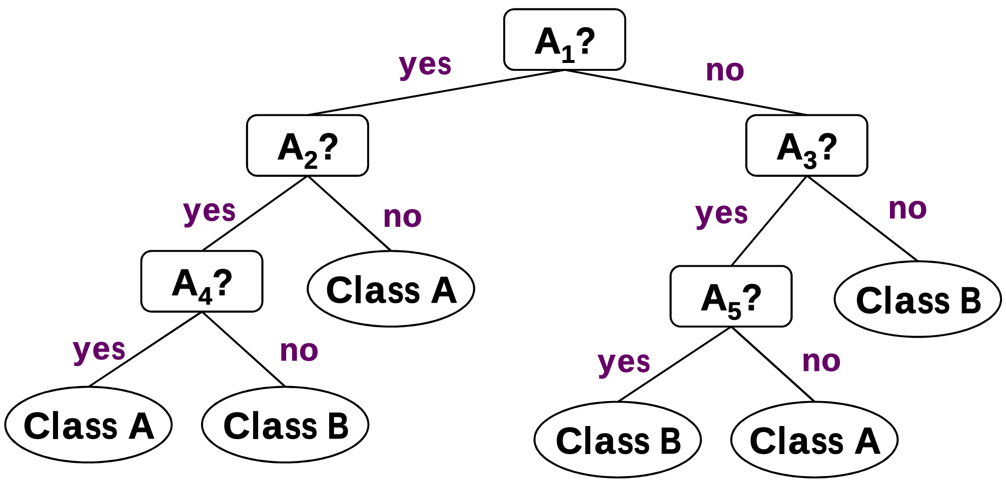
Comparing Attribute Selection Measures

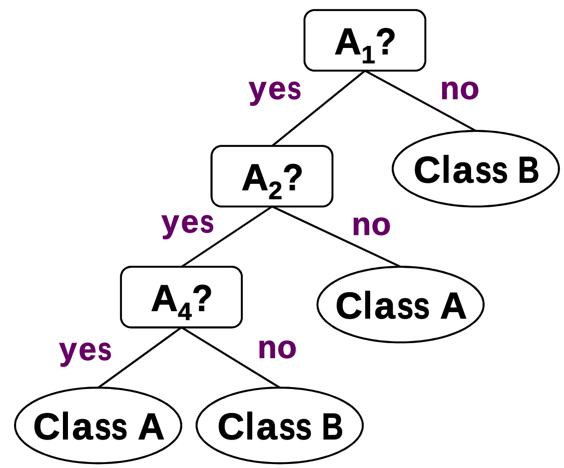
The three measures, in general, return good results but

- **Information Gain**
  - → Biased towards multivalued attributes
- **Gain Ratio**
  - → Tends to prefer unbalanced splits in which one partition is much smaller than the other
- **Gini Index**
  - → Biased towards multivalued attributes
  - → Has difficulties when the number of classes is large
  - → Tends to favor tests that result in equal-sized partitions and purity in both partitions

2.2.3 Tree Pruning

- **Problem: Overfitting**
  - → Many branches of the decision tree will reflect anomalies in the training data due to noise or outliers
  - → Poor accuracy for unseen samples
- **Solution: Pruning**
  - → Remove the least reliable branches





Tree Pruning Approaches

- **1st approach: prepruning**
  - Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - Statistical significance, information gain, Gini index are used to assess the goodness of a split
  - Upon halting, the node becomes a leaf
  - The leaf may hold the most frequent class among the subset tuples
- **Problem**
  - Difficult to choose an appropriate threshold

Tree Pruning Approaches

- **2^{nd} approach: postpruning**
  - Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
  - A subtree at a given node is pruned by replacing it by a leaf
  - The leaf is labeled with the most frequent class
- **Example: cost complexity pruning algorithm**
  - Cost complexity of a tree is a function of the the number of leaves and the error rate (percentage of tupes misclassified by the tree)
  - At each node N compute
    - The cost complexity of the subtree at N
    - The cost complexity of the subtree at N if it were to be pruned
  - If pruning results is smaller cost, then prune the subtree at N
  - Use a set of data different from the training data to decide which is the “best pruned tree”

2.2.4 Scalability and Decision Tree Induction

For scalable classification, propose presorting techniques on disk-resident data sets that are too large to fit in memory.

- **SLIQ** (EDBT'96 — Mehta et al.)
  - → Builds an index for each attribute and only class list and the current attribute list reside in memory
- **SPRINT** (VLDB'96 — J. Shafer et al.)
  - → Constructs an attribute list data structure
- **PUBLIC** (VLDB'98 — Rastogi & Shim)
  - → Integrates tree splitting and tree pruning: stop growing the tree earlier
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - → Builds an AVC-list (attribute, value, class label)
- **BOAT** (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - → Uses bootstrapping to create several small samples

Summary of Section 2.2

- Decision Trees have relatively **faster learning** speed than other methods
- Conversable to simple and **easy to understand** classification rules
- Information Gain, Ratio Gain and Gini Index are the most common methods of **attribute selection**
- **Tree pruning** is necessary to remove unreliable branches
- **Scalability** is an issue for large datasets

Chapter 2: Classification & Prediction

- **2.1 Basic Concepts of Classification and Prediction**
- **2.2 Decision Tree Induction**
  - 2.2.1 The Algorithm
  - 2.2.2 Attribute Selection Measures
  - 2.2.3 Tree Pruning
  - 2.2.4 Scalability and Decision Tree Induction
- **2.3 Bayes Classification Methods**
  - 2.3.1 Naïve Bayesian Classification
  - 2.3.2 Note on Bayesian Belief Networks
- **2.4 Rule Based Classification**
- **2.5 Lazy Learners**
- **2.6 Prediction**
- **2.7 How to Evaluate and Improve Classification**

2.3 Bayes Classification Methods

- **What are Bayesian Classifiers?**
  - → Statistical classifiers
  - → Predict class membership probabilities: probability of a given tuple belonging to a particular class
  - → Based on Bayes' Theorem
- **Characteristics?**
  - → Comparable performance with decision tree and selected neural network classifiers
- **Bayesian Classifiers**
  - → Naïve Bayesian Classifiers
    - Assume independency between the effect of a given attribute on a given class and the other values of other attributes
  - → Bayesian Belief Networks
    - Graphical models
    - Allow the representation of dependencies among subsets of attributes

Bayes' Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered **"evidence"**
- **H** is some **hypothesis** that X belongs to a specified class C

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

- **P(H|X)** is the posterior probability of **H** conditioned on **X**

Example: **predict** whether a costumer **will buy a computer** or **not**

→ Costumers are described by two attributes: **age** and **income**

→ **X** is a 35 years-old costumer with an income of 40k

→ **H** is the hypothesis that the costumer will buy a computer

→ **P(H|X)** reflects the probability that costumer **X will buy a computer** given that **we know** the costumers' **age** and **income**

Bayes' Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered **"evidence"**
- **H** is some **hypothesis** that X belongs to a specified class C

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

- **P(X|H)** is the posterior probability of **X** conditioned on **H**

Example: **predict** whether a costumer **will buy a computer** or **not**

- → Costumers are described by two attributes: **age** and **income**
- → **X** is a 35 years-old costumer with an income of 40k
- → **H** is the hypothesis that the costumer will buy a computer
- → **P(X|H)** reflects the probability that costumer **X**, is 35 years-old and earns 40k, given that we know that the costumer will buy a computer

Bayes' Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered **"evidence"**
- **H** is some **hypothesis** that X belongs to a specified class C

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

- **P(H)** is the **prior** probability of **H**

Example: **predict** whether a costumer **will buy a computer** or **not**

- → **H** is the hypothesis that the costumer will buy a computer
- → The prior probability of **H** is the probability that a costumer will buy a computer, regardless of age, income, or any other information for that matter
- → The posterior probability **P(H|X)** is **based on more information** than the prior probability **P(H)** which is **independent** from X

Bayes' Theorem In the Classification Context

- **X** is a data tuple. In Bayesian term it is considered **"evidence"**
- **H** is some **hypothesis** that X belongs to a specified class C

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

- **P(X)** is the **prior** probability of **X**

  Example: **predict** whether a costumer **will buy a computer** or **not**
  - → Costumers are described by two attributes: **age** and **income**
  - → **X** is a 35 years-old costumer with an income of 40k
  - → **P(X)** is the probability that a person from our set of costumers is 35 years-old and earns 40k

Naïve Bayesian Classification

D: A training set of tuples and their associated class labels

Each tuple is represented by n-dimensional vector **$X(x_1,\ldots,x_n)$**, n measurements of n attributes $A_1,\ldots,A_n$

Classes: suppose there are **m** classes **$C_1,\ldots,C_m$**

Principle

- Given a tuple **X**, the classifier will predict that X belongs to the class having the **highest posterior probability** conditioned on X
- Predict that tuple X belongs to the class C_i if and only if

$$P(C_i \mid X) > P(C_j \mid X) \quad \text{for} \quad 1 \leq j \leq m,\ j \neq i$$

- Maximize **$P(C_i|X)$**: find the **maximum posteriori hypothesis**

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

- P(X) is **constant** for all classes, thus, **maximize $P(X|C_i)P(C_i)$**

Naïve Bayesian Classification

- To maximize **P(X|Ci)P(Ci)**, we need to know class prior probabilities
  - If the probabilities are not known, assume that **$P(C_1)=P(C_2)=\ldots=P(C_m)$ $\Rightarrow$ maximize $P(X|C_i)$**
  - Class prior probabilities can be estimated by **$P(C_i)=|C_{i,D}|/|D|$**
- Assume **Class Conditional Independence** to reduce computational cost of **$P(X|C_i)$**
  - given **$X(x_1,\ldots,x_n)$**, $P(X|C_i)$ is:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

$$= P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

  - The probabilities $P(x_1|C_i)$, …$P(x_n|C_i)$ can be estimated from the training tuples

Estimating $P(x_i | C_i)$

- **Categorical Attributes**
  - Recall that $\mathbf{x_k}$ refers to the value of attribute A_k for tuple X
  - **X** is of the form $\mathbf{X(x_1,\ldots,x_n)}$
  - $\mathbf{P(x_k|C_i)}$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_{i,D}|$, the number of tuples of class C_i in D
  - **Example**
    - **8** costumers in class $\mathbf{C_{yes}}$ (costumer will buy a computer)
    - **3** costumers among the 8 costumers **have high income**
    - $\mathbf{P(income=high|C_{yes})}$ the probability of a costumer having a high income knowing that he belongs to class C_{yes} is **3/8**
- **Continuous-Valued Attributes**
  - A continuous-valued attribute is assumed to have a **Gaussian (Normal)** distribution with mean μ and standard deviation σ

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

Estimating $P(x_i | C_i)$

- **Continuous-Valued Attributes**
  - → The probability $P(x_k | C_i)$ is given by:

$$P(x_k \mid C_i) = g(x_k, \mu_{Ci}, \sigma_{Ci})$$

  - → Estimate μ_{Ci} and σ_{Ci} the mean and standard variation of the values of attribute A_k for training tuples of class C_i
  - → **Example**
    - **X** a 35 years-old costumer with an income of 40k (age, income)
    - Assume the age attribute is continuous-valued
    - Consider class C_{yes} (the costumer will buy a computer)
    - We find that in D, the costumers who will buy a computer are 38±12 years of age ⇒ $\mu_{Cyes}=38$ and $\sigma_{Cyes}=12$

$$P(age = 35 \mid C_{yes}) = g(35, 38, 12)$$

Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Tuple to classify is

X (age=youth, income=medium, student=yes, credit=fair)

Maximize $P(X \mid C_i)P(C_i)$, for $i=1,2$

Example

Given **X (age=youth, income=medium, student=yes, credit=fair)**

Maximize **$P(X|C_i)P(C_i)$,** for i=1,2

First step: Compute $P(C_i)$. The prior probability of each class can be computed based on the training tuples:

P(buys_computer=yes)=9/14=0.643

P(buys_computer=no)=5/14=0.357

Second step: compute $P(X|C_i)$ using the following conditional prob.

P(age=youth|buys_computer=yes)=0.222

P(age=youth|buys_computer=no)=3/5=0.666

P(income=medium|buys_computer=yes)=0.444

P(income=medium|buys_computer=no)=2/5=0.400

P(student=yes|buys_computer=yes)=6/9=0.667

P(tudent=yes|buys_computer=no)=1/5=0.200

P(credit_rating=fair|buys_computer=yes)=6/9=0.667

P(credit_rating=fair|buys_computer=no)=2/5=0.400

Example

$$P(X \mid buys\_computer=yes) = P(age=youth \mid buys\_computer=yes) \times P(income=medium \mid buys\_computer=yes) \times P(student=yes \mid buys\_computer=yes) \times P(credit\_rating=fair \mid buys\_computer=yes) = 0.044$$

$$P(X \mid buys\_computer=no) = P(age=youth \mid buys\_computer=no) \times P(income=medium \mid buys\_computer=no) \times P(student=yes \mid buys\_computer=no) \times P(credit\_rating=fair \mid buys\_computer=no) = 0.019$$

Third step: compute **$P(X \mid C_i)P(C_i)$** for each class

$P(X \mid buys\_computer=yes)P(buys\_computer=yes) = 0.044 \times 0.643 = \mathbf{0.028}$

$P(X \mid buys\_computer=no)P(buys\_computer=no) = 0.019 \times 0.357 = \mathbf{0.007}$

The naïve Bayesian Classifier predicts buys_computer=yes for tuple X

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- **Use Laplacian correction** (or Laplacian estimator)
  - → Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

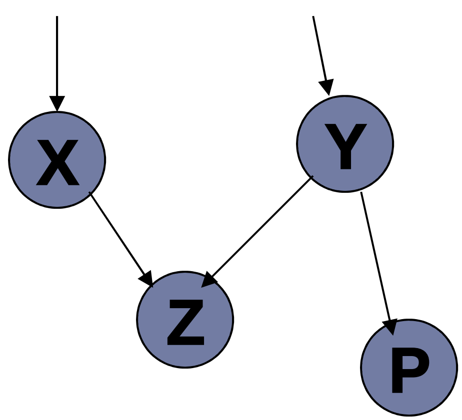
    Prob(income = high) = 11/1003
  - → The "corrected" prob. estimates are close to their "uncorrected" counterparts

Summary of Section 2.3

- 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
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

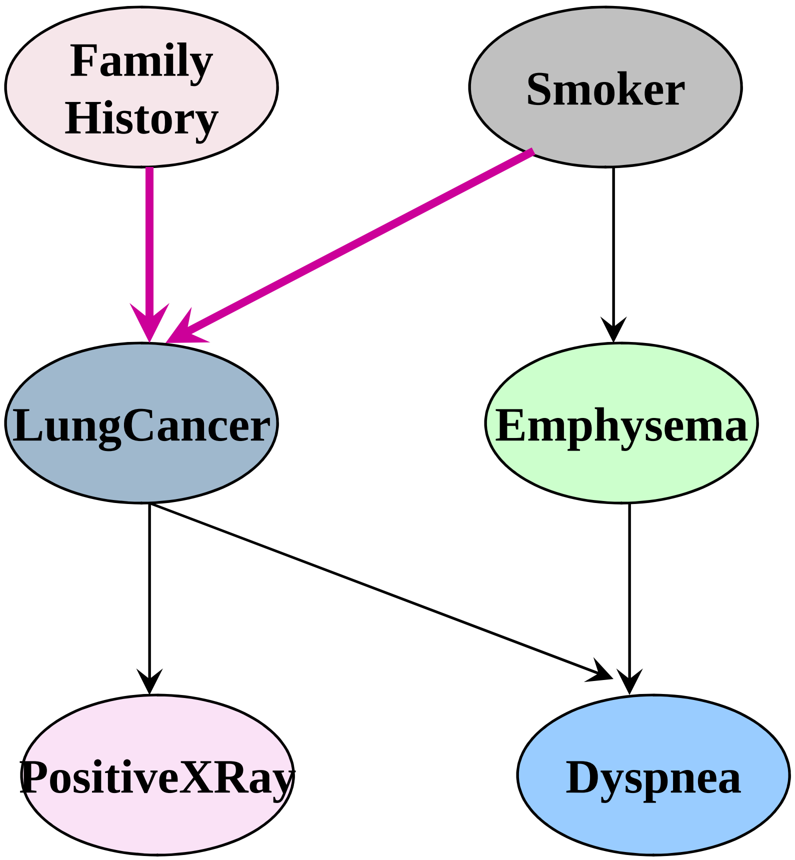
2.3.2 Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - → Represents <u>dependency</u> among the variables
  - → Gives a specification of joint probability distribution



- → Nodes: random variables
- → Links: dependency
- → X and Y are the parents of Z, and Y is the parent of P
- → No dependency between Z and P
- → Has no loops or cycles

Example



Bayesian Belief Networks

The **conditional probability table** (**CPT**) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose

Summary of Section 2.3

- Bayesian Classifiers are **statistical classifiers**
- They provide **good accuracy**
- Naïve Bayesian classifier assumes **independency** between attributes
- **Causal relations** are captured by Bayesian Belief Networks