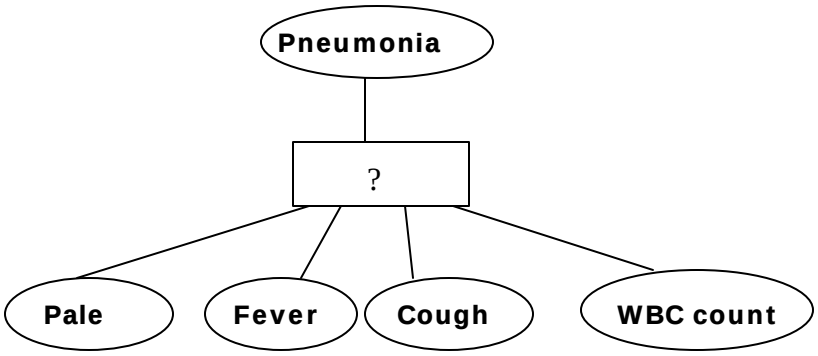
CS 2001 – Lecture 2

Bayesian belief networks

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square
X4-8845

Modeling the uncertainty.

- How to describe, represent the relations in the presence of uncertainty?
- How to manipulate such knowledge to make inferences?



Probability theory

a well-defined coherent theory for representing uncertainty and for reasoning with it

Representation:

Proposition statements – assignment of values to random variables

$Pneumonia = True \qquad WBCcount = high$

Probabilities over statements model the degree of belief in these statements

$P(Pneumonia = True) = 0.001$

$P(WBCcount = high) = 0.005$

$P(Pneumonia = True, Fever = True) = 0.0009$

$P(Pneumonia = False, WBCcount = normal, Cough = False) = 0.97$

Modeling uncertainty with probabilities

- **Full joint distribution:** joint distribution over all random variables defining the domain
  - it is sufficient to represent the complete domain and to do any type of probabilistic reasoning

Problems:

- **Space complexity.** To store full joint distribution requires to remember $O(d^n)$ numbers.

  n – number of random variables, d – number of values
- **Inference complexity.** To compute some queries requires $O(d^n)$ steps.
- **Acquisition problem.** Who is going to define all of the probability entries?

Modeling uncertainty with probabilities

- **Knowledge based system era** (70s – early 80's)
  - Extensional non-probabilistic models
  - Probability techniques avoided because of space, time and acquisition bottlenecks in defining full joint distributions
  - Negative effect on the advancement of KB systems and AI in 80s in general

- Breakthrough (late 80s, beginning of 90s)
  - **Bayesian belief networks**
    - Give solutions to the space, acquisition bottlenecks
    - Significant improvements in the time cost of inferences

Bayesian belief networks (BBNs)

Bayesian belief networks.

- Represent the full joint distribution more compactly with smaller number of parameters.
- Take advantage of conditional and marginal independences among components in the distribution

- **A and B are independent**

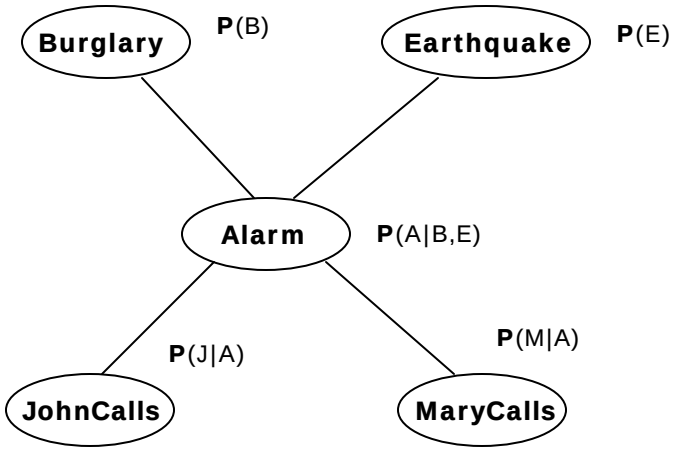
$$P(A, B) = P(A)P(B)$$

- **A and B are conditionally independent given C**

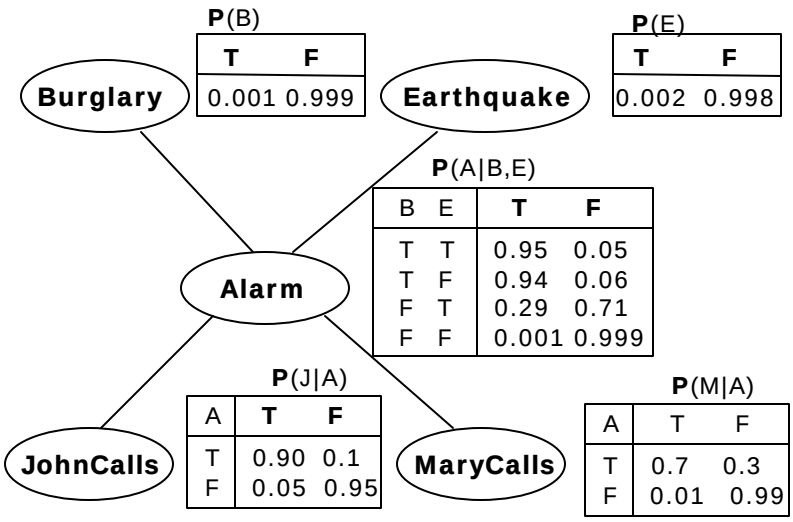
$$P(A, B \mid C) = P(A \mid C)P(B \mid C)$$
$$P(A \mid C, B) = P(A \mid C)$$

Bayesian belief network.

1. **Graph** represents marginal and conditional independences variables
2. **Parameters** defining local conditional distributions relating variables and their parents



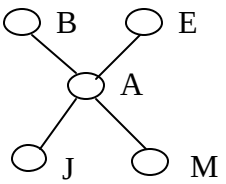
Bayesian belief network.



Full joint distribution in BBNs

Full joint distribution is defined in terms of local conditional distributions (obtained via the chain rule):

$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$



Example:

Assume the following assignment of values to random variables

$B=T, E=T, A=T, J=T, M=F$

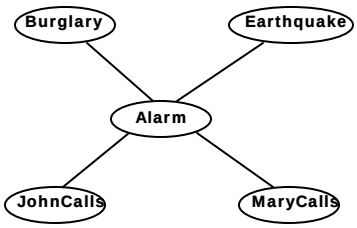
Then its probability is:

$$P(B=T, E=T, A=T, J=T, M=F) =$$
$$P(B=T)P(E=T)P(A=T \mid B=T, E=T)P(J=T \mid A=T)P(M=F \mid A=T)$$

Parameter complexity problem

- In the BBN the full joint distribution is expressed as a product of conditionals (of smaller) complexity

$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$



Parameters:

full joint: $2^5 = 32$

BBN: $2^3 + 2(2^2) + 2(2) = 20$

Parameters to be defined:

full joint: $2^5 - 1 = 31$

BBN: $2^2 + 2(2) + 2(1) = 10$

Inference in Bayesian networks

- BBN models compactly the full joint distribution by taking advantage of existing independences between variables
- Simplifies the acquisition of a probabilistic model
- But we are interested in solving various **inference tasks**:
  - **Diagnostic task. (from effect to cause)**

$$\mathbf{P}(Burglary \mid JohnCalls = T)$$

  - **Prediction task. (from cause to effect)**

$$\mathbf{P}(JohnCalls \mid Burglary = T)$$

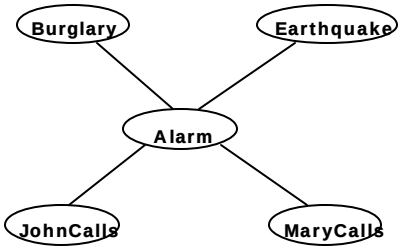
  - **Other probabilistic queries (**queries on joint distributions).

$$\mathbf{P}(Alarm)$$

- **Question:** Can we take advantage of independences to construct special algorithms and speeding up the inference?

Inference in Bayesian network

- **Bad news:**
  - Exact inference problem in BBNs is NP-hard (G. Cooper)
  - Approximate inference is NP-hard (Dagum, Luby)
- **But** very often we can achieve significant improvements
- Assume our Alarm network



- Assume we want to compute: $P(J = T)$

Inference in Bayesian networks

Computing: $P(J=T)$

Approach 1. Blind approach.

- Sum out all uninstantiated variables from the full joint,
- Express the joint distribution as a product of conditionals

$$P(J=T)=$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(B=b,E=e,A=a,J=T,M=m)$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(A=a\mid B=b,E=e)P(B=b)P(E=e)$$

Computational cost:

Number of additions: **15**

Number of products: 16*4=**64**

Inference in Bayesian networks

Approach 2. Interleave sums and products

- Combines sums and product in a smart way (multiplications by constants can be taken out of the sum)

$$P(J=T)=$$

$$=\sum_{b\in T,F}\sum_{e\in T,F}\sum_{a\in T,F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(A=a\mid B=b,E=e)P(B=b)P(E=e)$$

$$=\sum_{b\in T,F}\sum_{a\in T.F}\sum_{m\in T,F}P(J=T\mid A=a)P(M=m\mid A=a)P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]$$

$$=\sum_{a\in T,F}P(J=T\mid A=a)[\sum_{m\in T,F}P(M=m\mid A=a)][\sum_{b\in T,F}P(B=b)[\sum_{e\in T,F}P(A=a\mid B=b,E=e)P(E=e)]$$

Computational cost:

Number of additions: 1+ 2*(1)+2*(1+2*(1))=**9**

Number of products: 2*(2+2*(0)+2*(1 + 2*(1)))=**16**

Inference in Bayesian networks

- The smart interleaving of sums and products can help us to speed up the computation of joint probability queries
- What if we want to compute: $P(B = T, J = T)$

$$P(B = T, J = T) =$$

$$= \sum_{a \in T,F} P(J=T \mid A=a)[\sum_{m \in T,F} P(M=m \mid A=a)][P(B=T)[\sum_{e \in T,F} P(A=a \mid B=T, E=e)P(E=e)]$$

Are there any similarities from the previous query ?

$$P(J = T) = \quad ?$$

Inference in Bayesian networks

- The smart interleaving of sums and products can help us to speed up the computation of joint probability queries
- What if we want to compute: $P(B = T, J = T)$

$$P(B = T, J = T) =$$

$$= \sum_{a \in T,F} P(J=T \mid A=a)[\sum_{m \in T,F} P(M=m \mid A=a)][P(B=T)[\sum_{e \in T,F} P(A=a \mid B=T, E=e)P(E=e)]$$

$$P(J = T) =$$

$$= \sum_{a \in T,F} P(J=T \mid A=a)[\sum_{m \in T,F} P(M=m \mid A=a)][\sum_{b \in T,F} P(B=b)[\sum_{e \in T,F} P(A=a \mid B=b, E=e)P(E=e)]]$$

- A lot of shared computation
  - Smart cashing of results can save the time for more queries

Inference in Bayesian networks

- The smart interleaving of sums and products can help us to speed up the computation of joint probability queries
- What if we want to compute: $P(B = T, J = T)$

$$P(B = T, J = T) =$$
$$= \sum_{a \in T,F} P(J = T \mid A = a)[\boxed{\sum_{m \in T,F} P(M = m \mid A = a)}][\boxed{P(B = T)[\sum_{e \in T,F} P(A = a \mid B = T, E = e) P(E = e)]}]$$

$$P(J = T) =$$
$$= \sum_{a \in T,F} P(J = T \mid A = a)[\boxed{\sum_{m \in T,F} P(M = m \mid A = a)}][\sum_{b \in T,F} \boxed{P(B = b)[\sum_{e \in T,F} P(A = a \mid B = b, E = e) P(E = e)]}]$$

- A lot of shared computation
  - Smart cashing of results can save the time if more queries

Inference in Bayesian networks

- When cashing of results becomes handy?
- What if we want to compute a diagnostic query:

$$P(B = T \mid J = T) = \frac{P(B = T, J = T)}{P(J = T)}$$

- Exactly probabilities we have just compared !!
- There are other queries when cashing and ordering of sums and products can be shared and saves computation

$$\mathbf{P}(B \mid J = T) = \frac{\mathbf{P}(B, J = T)}{P(J = T)} = a\,\mathbf{P}(B, J = T)$$

- General technique: Variable elimination

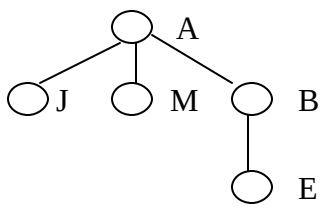
Inference in Bayesian networks

- General idea of variable elimination

$$P(True) = 1 =$$

$$= \sum_{a \in T,F} [\underbrace{\sum_{j \in T,F} P(J=j \mid A=a)}_{f_J(a)}][\underbrace{\sum_{m \in T,F} P(M=m \mid A=a)}_{f_M(a)}][\underbrace{\sum_{b \in T,F} P(B=b)[\underbrace{\sum_{e \in T,F} P(A=a \mid B=b, E=e)P(E=e)}_{f_E(a,b)}]}_{f_B(a)}]$$

Variable order:



Results cashed in the tree structure

Inference in Bayesian network

- **Exact inference algorithms:**
  - Symbolic inference (D'Ambrosio)
  - Recursive decomposition/variable elimination (Cooper, Dechter)
  - Message-passing algorithm (Pearl)
  - Clustering and joint-tree approach (Lauritzen, Spiegelhalter)
  - Arc reversal (Olmsted, Schachter)

- **Approximate inference algorithms:**
  - **Monte Carlo methods** :
    - Rejection sampling, Likelihood sampling
  - Variational methods

Learning Bayesian belief networks

- **Why learning?**
  - "subjective" estimates of conditional probability parameters by a human
    - need to adapt parameters in the light of observed data
  - large databases available
    - uncover important probabilistic dependencies from data and use them in inference tasks

Learning of BBN

Two learning tasks:

- Learning of the network structure
- Learning of parameters of conditional probabilities

- **Variables**:
  - **Observable** – values present in every data sample
  - **Hidden** – values are never in the sample
  - **Missing values** – values sometimes present, sometimes not
- **<u>Here</u>:**
  - Learning parameters of the fixed BBN structure
  - All variables are observable

Learning of BBNs.

Data: $D = \{D_1, D_2, .., D_n\}$
$D_i = \mathbf{x}_i$ a vector of variable values

Random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ with:

- **Continuous values**
- **Discrete values** ⇦

E.g. ***blood pressure*** with numerical values
or ***chest pain*** with discrete values
[no-pain, mild, moderate, strong]

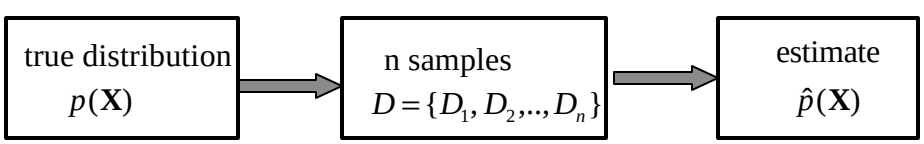
Underlying true probability distribution:

$$p(\mathbf{X})$$

Learning of BBNs.

Data: $D = \{D_1, D_2, .., D_n\}$
$D_i = \mathbf{x}_i$ a vector of attribute values

Objective: try to estimate the underlying true probability distribution over variables $\mathbf{X}$, $p(\mathbf{X})$, using examples in D



Standard (iid) assumptions: Samples

- **are independent of each other**
- **come from the same (identical) distribution (fixed $p(\mathbf{X})$)**

Learning via parameter estimation

- For the fixed BBN we have:
  - **a model of the distribution** over variables in $\boldsymbol{X}$ with parameters $\mathbf{T}$
  - Parameters = parameters of local conditional probabilities

- **Objective:** find parameters $\mathbf{T}$ that fit the data the best

$$\tilde{p}(\mathbf{X} \mid \mathbf{T}) \approx p(\mathbf{X})$$

- There are various criteria for defining the best set of parameters

Parameter estimation. Criteria.

- **Maximum likelihood (ML)**

  maximize $P(D \mid \Theta, x)$

  x - represents prior (background) knowledge (structure of the BBN)

- **Maximum a posteriori probability (MAP)**

  maximize $P(\Theta \mid D, x)$

$$P(\Theta \mid D, x) = \frac{P(D \mid \Theta, x) P(\Theta \mid x)}{P(D \mid x)}$$

Parameter estimation. Example.

Coin example: we have a coin that can be biased

Outcomes: two possible values -- head or tail

Data: D a sequence of outcomes x_i such that

- **head** $x_i = 1$
- **tail** $x_i = 0$

Model: probability of a head q

probability of a tail $(1-q)$

Objective:

We would like to estimate the probability of a **head** $\hat{q}$

Probability of an outcome x_i

$$P(x_i \mid q) = q^{x_i}(1-q)^{(1-x_i)}$$ **Bernoulli distribution**

Maximum likelihood (ML) estimate.

Likelihood of data:

$$P(D \mid q, x) = \prod_{i=1}^{n} q^{x_i}(1-q)^{(1-x_i)}$$

Maximum likelihood estimate

$$q_{ML} = \arg\max_q P(D \mid q, x)$$

Optimize log-likelihood

$$l(D,q) = \log P(D \mid q, x) = \log \prod_{i=1}^{n} q^{x_i}(1-q)^{(1-x_i)} =$$

$$\sum_{i=1}^{n} x_i \log q + (1-x_i)\log(1-q) = \log q \underbrace{\sum_{i=1}^{n} x_i}_{} + \log(1-q)\underbrace{\sum_{i=1}^{n}(1-x_i)}_{}$$

N_1 - number of heads seen N_2 - number of tails seen

Maximum likelihood (ML) estimate.

Optimize log-likelihood

$$l(D,q) = N_1 \log q + N_2 \log(1-q)$$

Set derivative to zero

$$\frac{\partial l(D,q)}{\partial q} = \frac{N_1}{q} - \frac{N_2}{(1-q)} = 0$$

Solving $$q = \frac{N_1}{N_1 + N_2}$$

ML Solution: $$q_{ML} = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

Maximum a posteriori estimate

Maximum a posteriori estimate

– Selects the mode of the posterior distribution

$$q_{MAP} = \arg\max_q \; p(q \mid D,x)$$

$$p(q \mid D,x) = \frac{P(D \mid q,x)\, p(q \mid x)}{P(D \mid x)}$$ **(via Bayes rule)**

$P(D \mid q,x)$ - is the likelihood of data

$$P(D \mid q,x) = \prod_{i=1}^{n} q^{x_i} (1-q)^{(1-x_i)} = q^{N_1} (1-q)^{N_2}$$

$p(q \mid x)$ - is the prior probability on q

How to choose the prior probability?

Prior distribution

Choice of prior: Beta distribution

$$p(q \mid x) = Beta(q \mid a_1, a_2) = \frac{\Gamma(a_1 + a_2)}{\Gamma(a_1)\Gamma(a_2)} q^{a_1 - 1}(1-q)^{a_2 - 1}$$

Why?

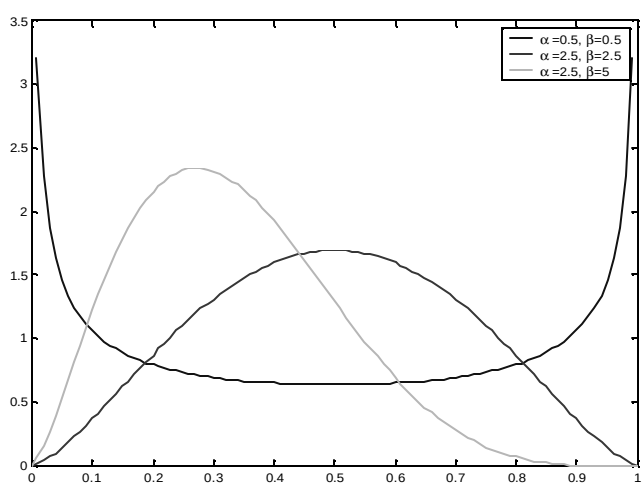
Beta distribution "**fits**" binomial sampling - **conjugate choices**

$$P(D \mid q, x) = q^{N_1}(1-q)^{N_2}$$

$$p(q \mid D, x) = \frac{P(D \mid q, x) Beta(q \mid a_1, a_2)}{P(D \mid x)} = Beta(q \mid a_1 + N_1, a_2 + N_2)$$

MAP Solution: $$q_{MAP} = \frac{a_1 + N_1 - 1}{a_1 + a_2 + N_1 + N_2 - 2}$$

Beta distribution



Bayesian learning

- **Both ML or MAP pick one parameter value**
  - Is it always the best solution?
  - **Assume:** there are two very different parameter settings that are close in terms of probability (ML or MAP). Using only one of them may introduce a strong bias, if we use them e.g. for predictions.
- **Bayesian approach**
  - Remedies the limitation of one choice
  - Considers all parameter settings and averages the result

$$P(\Delta \mid D,x) = \int_q P(\Delta \mid q,x)\, p(q \mid D,x)\, dq$$

- **Example:** predict the result of the next coin flip
  - Choose outcome 1 if $P(x=1 \mid D,x)$ is higher

Bayesian learning

- **Predictive probability of an outcome** $x=1$ **in the next trial**

$$P(x=1 \mid D,x)$$

$$P(x=1 \mid D,x) = \int_0^1 P(x=1 \mid q,x)\, p(q \mid D,x)\, dq$$

$$= \int_0^1 q\, p(q \mid D,x)\, dq = E(q)$$

- **Equivalent to the expected value of the parameter**
  - expectation is taken with regard to the posterior distribution

$$p(q \mid D,x) = Beta(q \mid a_1 + N_1, a_2 + N_2)$$

Bayesian learning, expectation

The result is:

$$E(q) = \int_0^1 q\,Beta(q \mid h_1, h_2)\,dq = \frac{h_1}{h_1 + h_2}$$

Bayesian estimate of the parameter

$$E(q) = \frac{a_1 + N_1}{a_1 + N_1 + a_2 + N_2}$$

Predictive probability of event x=1

$$E(q) = P(x=1 \mid q, x) = \frac{a_1 + N_1}{a_1 + N_1 + a_2 + N_2}$$

Example. Multi-way coin toss

Multi-way coin toss (roll of dice)

- **Data:** a set of N outcomes (multi-set)

  N_i - a number of times an outcome i has been seen

Model parameters: $? = (q_1, q_2, \ldots q_k)$ **s.t.** $\sum_{i=1}^{k} q_i = 1$

q_i - probability of an outcome i

Probability of data (likelihood)

$$P(N_1, N_2, \ldots N_k \mid ?, x) = \frac{N!}{N_1! N_2! \ldots N_k!} q_1^{N_1} q_2^{N_2} \ldots q_k^{N_k}$$

Multinomial distribution

ML estimate:

$$q_{i,ML} = \frac{N_i}{N}$$

MAP estimate

Choice of prior: Dirichlet distribution

$$Dir(?\,|\,a_1,..,a_k) = \frac{\Gamma(\sum_{i=1}^{k} a_i)}{\prod_{i=1}^{k}\Gamma(a_i)} q_1^{a_1-1} q_2^{a_2-1} \ldots q_k^{a_k-1}$$

Dirichlet is the conjugate choice for multinomial

$$P(D\,|\,?,x) = P(N_1, N_2, \ldots N_k\,|\,?,x) = \frac{N!}{N_1!N_2!\ldots N_k!} q_1^{N_1} q_2^{N_2} \ldots q_k^{N_k}$$

Posterior distribution

$$p(?\,|\,D,x) = \frac{P(D\,|\,?,x)Dir(?\,|\,a_1,a_2,..a_k)}{P(D\,|\,x)} = Dir(?\,|\,a_1+N_1,..,a_k+N_k)$$

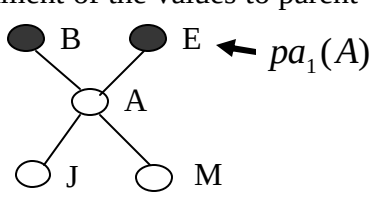
MAP estimate: $$q_{i,MAP} = \frac{a_i + N_i - 1}{\sum_{i=1,..k}(a_i + N_i) - k}$$

Learning of parameters of BBNs

- We need to estimate parameters of conditional distributions

$$\mathbf{P}(X_i \mid pa(X_i))$$

- **Idea:**
  - Fix an assignment of the values to parent variables of X_i



  - Pull out of the dataset D only data that are in agreement with this assignment -- $pa_1(A)$
  - Learn the parameters of the conditional $\mathbf{P}(A \mid pa_1(A))$
  - If discrete-valued variable A this is like a multiway coin toss

Estimates of parameters of BBN

- **Two assumptions to make this possible:**
  - **Sample independence**

$$P(D \mid \mathbf{T}, x) = \prod_{u=1}^{n} p(D_u \mid \mathbf{T}, x)$$

  - **Parameter independence**

$$P(\mathbf{T} \mid D, x) = \prod_{i=1}^{d} \prod_{j=1}^{q_i} p(q_{ij} \mid D, x)$$

Parameters of each node-parents conditional can be optimized independently

ML Course

CS2750 Machine Learning, Spring 2003

Instructor: **Milos Hauskrecht**

web page: http://www.cs.pitt.edu/~milos/courses/cs2750/

- Covers modern machine learning techniques, including learning of BBNs, their structures and parameters in different settings, as well as, many other learning frameworks, such as neural networks, support vector machines etc.