Generative Models: Learning Joint Probabilities

Machine Learning

A Simple Generative Model I

- Let us assume data are are random samples drawn from (X, Y) or $(X_1, X_2, \ldots, X_d, Y)$
- We know:

$$P(X_1, X_2, \ldots, X_d, Y) = P(X_1, \ldots, X_d|Y)P(Y)$$

where (from the chain rule):

$$P(X_1, X_2, \ldots, X_d|Y) = P(X_1|Y)P(X_2|X_1, Y)\cdots P(X_d|X_{d-1}, \ldots, X$$

- BUT: there is a problem. Estimating $P(X_1, |Y)$ means estimating probability values for every combination of the X's and Ys. If each value of X is a d-dimensional vector of Boolean values, and then for a 2-class problem, there are 2^d such values possible, one each for $Y = 1$ and $Y = 0$. This is a total of 2×2^d (some small saving will be possible since the probabilities need to add to 1)

A Simple Generative Model II

- Maximum likelihood estimates are simply *successes*/*total*. A lot of data instances will be needed to get meaningful estimates of any specific $P(x_1, x_2, \ldots, x_d | Y = 1)$ and $(x_1, x_2, \ldots, x_d | Y = 0)$
- Suppose we assumed that the X_k are *conditionally independent* of each other, given Y. Then:

$$P(X_1, X_2, \ldots, X_d | Y) = P(X_1 | Y) P(X_2 | Y) \cdots P(X_d | Y)$$

Now, for each value of Y, we only need to estimate $2d$ values, which is huge reduction

A Simple Generative Model III

- If the estimates are maximum likelihood estimates. That is:

$$P(X_i = x_{ij}|Y = y_k) = \frac{No.\ of\ instances\ with\ (X_i = x_{ij} \wedge Y = y_k)}{No.\ of\ instances\ with\ Y = y_k}$$

and:

$$P(Y = y_k) = \frac{No.\ of\ instances\ with\ Y = y_k}{Total\ Number\ of\ Instances}$$

(this is an *empirical prior*) (We have already seen how to deal with ML estimation when counts are 0, but we ignore this for now.)

- Once these counts have been obtained from the data, we can compute the values for the joint probabilityteem

A Simple Generative Model IV

- In fact, we can also compute the conditional probability $P(Y|X)$
  For example, for the 2-class problem, we will get $P(Y = 1|X) = \alpha \prod_{i=1}^{d} P(X_i|Y = 1)P(Y = 1)$ and $P(Y = 0|X) = \alpha \prod_{i=1}^{d} P(X_i|Y = 0)P(Y = 0) = \beta$. Since these have to add to 1, we can calculate the value of α and the corresponding $P(Y = \ldots|X)$
  - BUT: what happens if the x_{ij} values are continuous? One common assumption is that the class-conditional values x_{ij} are from a Gaussian distribution.
  - That is, for each value of $Y = y_k$, the X_i is a Gaussian defined specific to the X_i and $Y = y_k$. The parameters of the Gaussian are estimated from data using the usual of estimating means and s.d.'s and the corresponding value of of $P(X_i = x_{ij}|Y = y_k)$ is obtained from $P(X_i = x_{ij}|y_k, \mathcal{N}(\hat{\mu}, \hat{\sigma}))$

A Simple Generative Model V

- This kind of $P(Y|X)$ model is the *Naive Bayes* model. It is an example of a simple Bayesian Network

Aside: When is the Naive Bayes Model Correct?

- The conditional independence assumption is quite a strong constraint. How naive is this in real-life?
- The conditional independence assumption becomes especially problematic if the input data X is a d-dimensional vector in which many of the dimensions are correlated
- Why? Because, in effect, each of the dimensions gets multiply-counted. So, probabilities tend to be skewed to extreme values ($(P(Y = 1|X) = 0.999$ *etc*.)
- Despite this, the simple Bayes model has proved to be surprisingly effective in practice
- Why? Because order is surprisingly noise-resistant. So, although the probability estimates are poor, if $P(Y = \omega_i|X) > P(Y = \omega_j|X)$, then it would unlikely for the simple Bayes probabilities to result in the reverse. This is especially so for 2-class problems
- So, if the object is to *classify* based on probabilities, then it may be good enough to use the simple Bayes model. This is

Bayesian Networks I

- So far, we have used the conditional probability $P(Y|\mathrm{x})$ to calculate the probability of $Y = y_i$ (and perhaps even to classify a data instance, using a decision rule)
- In general, we can use inference on the joint probability to obtain all kinds of probability values, including *conditional probability* values:

$$P(Y|X) = \frac{P(X,Y)}{P(X)} \propto P(X,Y)$$

and *marginal probability* values:

$$P(Y) = \sum_X P(X,Y)$$

Bayesian Networks II

- We saw some of this, with an assumption of class-conditional independence using the simple Bayes model. This is a special case of a more general kind of of factoring of the joint distribution of a set of variables X

Factored Joint Distributions using Bayesian Networks

- A way of representing and reasoning with the full joint probability distribution
  - Can answer any kind of probabilistic query. For eg. $P(A = yes|B = yes)$ or $P(B = no|A = yes)$
- Capture independence and conditional independence where they exist. This is *domain knowledge*.
  - For eg. *we know* for this problem, $P(A|C) = P(A)$
- Among variables where dependencies exist, encode the relevant portion of the full joint distribution
  - For eg. $P(A = yes|B = yes) = 0.80$
- Use a graphical representation, making it easier to visualise, investigate complexity and study inference algorithms

Bayesian Network: What it Is

- A Bayesian Network is a Directed Acyclic Graph (DAG) in which
  1. Each node denotes some random variable X
  2. Each node has a conditional probability distribution $\mathrm{P}(X|$ $Parents(X))$

- The intuitive meaning of an arc from node X to node Y is that X *directly influences* Y

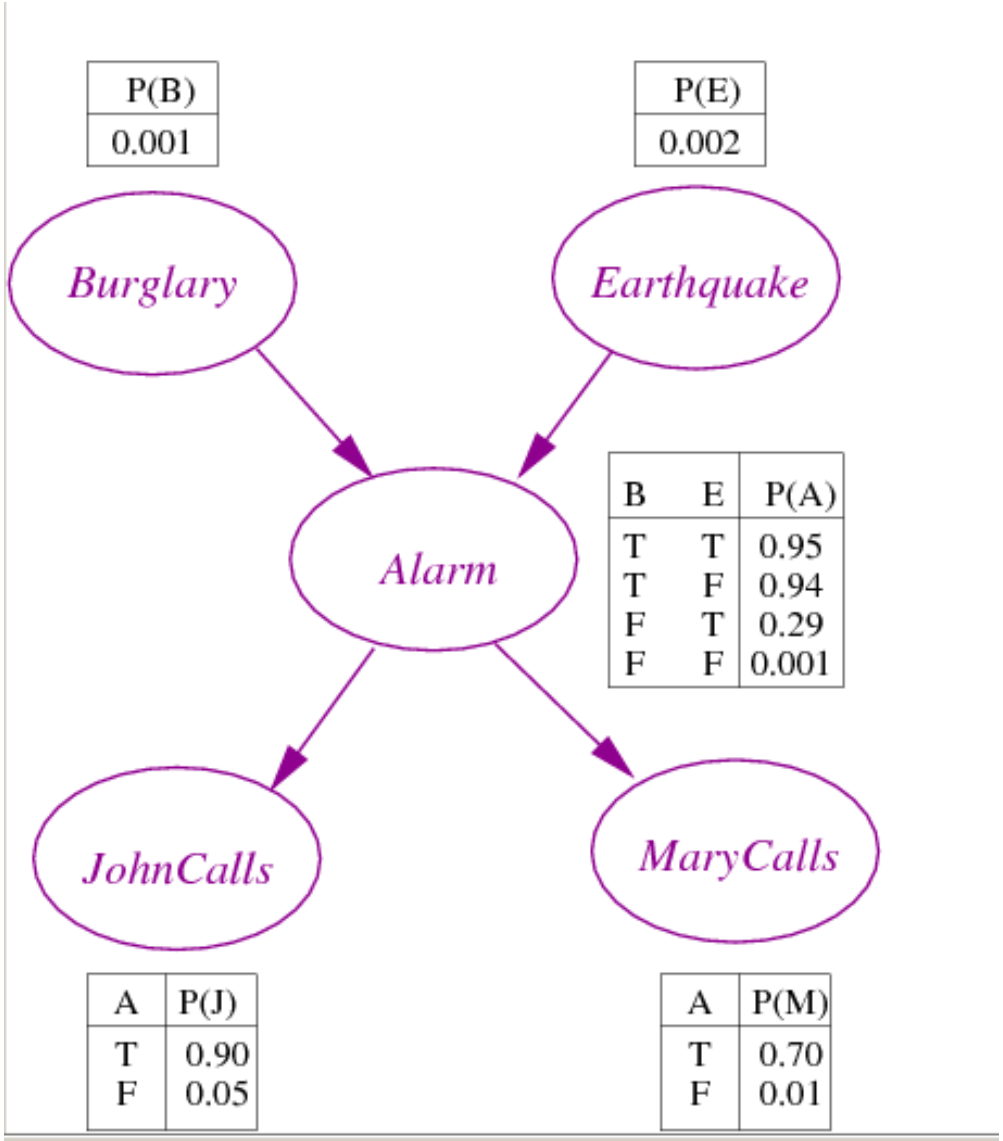
Aside: Property of DAGs

- An *ancestral ordering* of the n nodes in a directed graph G is an ordering $[v_1, v_2, \ldots, v_n]$ such that the ancestors of a node v_i appear before v_i in the ordering
- Key property: DAGs always have at least one ancestral ordering.

Bayesian Network: Additional Terminology

- If X and its parents are discrete, we can represent the distribution $\mathrm{P}(X|Parents(X))$ by a *conditional probability table* (CPT)
- The CPT specifies the probability of each value of X given each possible combination of values for variables in $Parents(X)$
- A *conditioning case* is a row in the CPT

A Bayesian Network



Bayesian Network: What it Means

A Bayesian Network can be understood as:

1. A representation of the full joint distribution over its random variables; or
2. A collection of conditional independence statements.

(1) is helpful in understanding BN construction
(2) is helpful in understanding BN inference

BN Representation of the Full Joint Distribution

- A generic entry in the full joint distribution is

$$\mathrm{P}(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$$

or $\mathrm{P}(x_1, \ldots, x_n)$ for short

- By definition, in a BN this is given by

$$\mathrm{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathrm{P}(x_i | \mathit{Parents}(X_i))$$

Chain Rule and BNs

- BN

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

- Chain rule

$$\mathrm{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathrm{P}(x_i | x_{i-1}, \ldots, x_1)$$

- For a BN to correctly represent the joint distribution:

$$\mathrm{P}(X_i | Parents(X_i)) = \mathrm{P}(X_i | X_{i-1}, \ldots, X_1)$$

Chain Rule and BNs: contd.

For a BN to correctly represent the joint distribution:

$$\mathrm{P}(X_i|Parents(X_i)) = \mathrm{P}(X_i|X_{i-1}, \ldots, X_1)$$

This follows provided:

- There is an ordering such that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$. (true for DAGs).
- X_i is conditionally independent of its predecessors in the node ordering, given its parents.
- X_i is conditionally independent of its non-descendents, given its parents. (This is called the *Markov condition*)
  - X is a parent of Y if there is an edge from X to Y. Y is a descendent of X if there is a path from X to Y. X is an ancestor (or predecessor) of Y if Y is a descendent of X. Y is a non-descendent of X if there is no path from X to Y.
  - Clearly, $Parents(X_i) \subseteq NonDesc(X_i)$. But, usually, the Markov condition does not include $Parents(X_i)$

  It follows that X_i is conditionally independent of its predecessors in the node ordering, given its parents. That is,

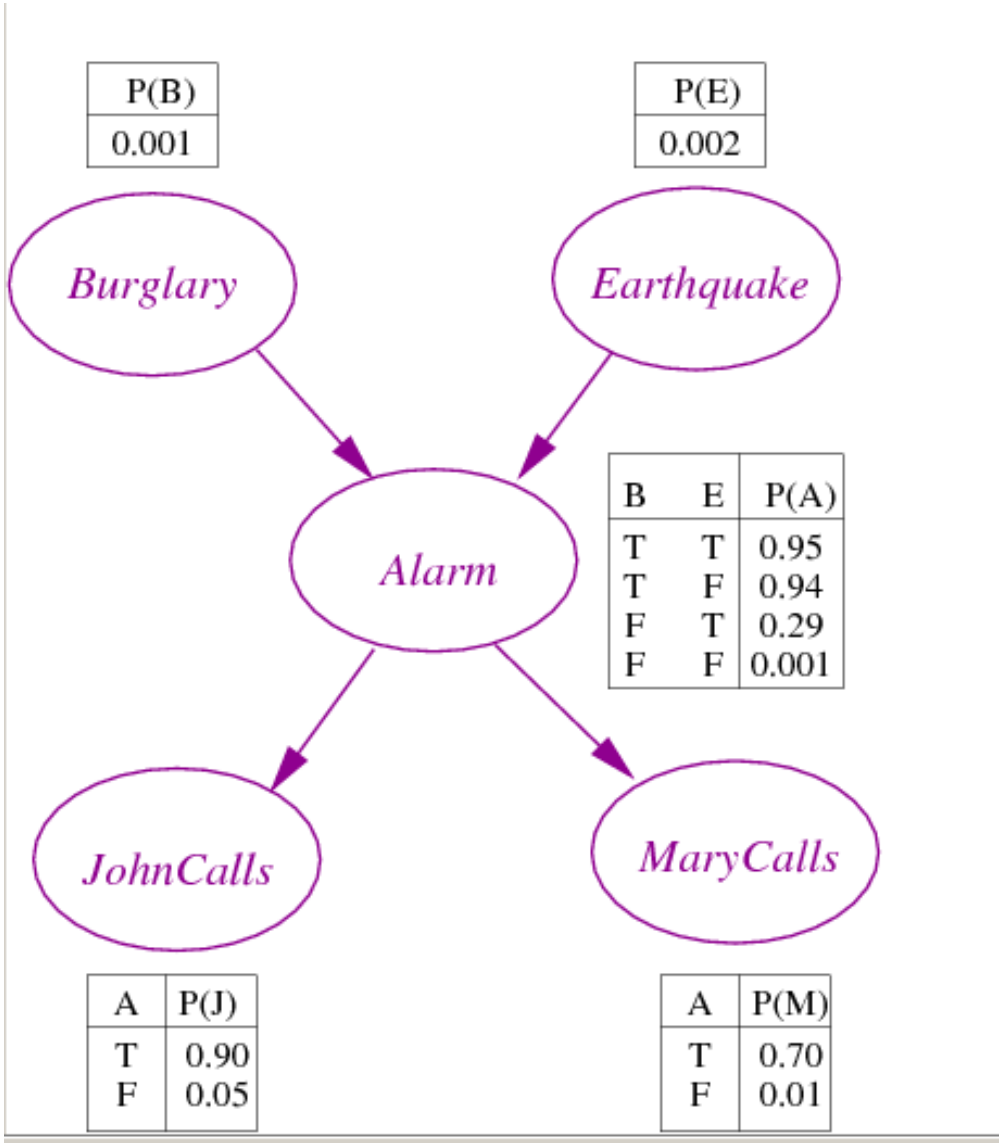
Conditional Independence Again

- Recall that a node X is conditionally independent of its predecessors (in an ancestral ordering) given $Parents(X)$
- *Markov Blanket* of X: the set consisting of the parents of X, children of X, and the children's parents.
- It can be shown that X is conditionally independent of all nodes in the network given its Markov blanket.

BNs: Representation and Inference I

- In the lectures, we will focus on learning Bayesian networks as a way of learning joint probability distributions
- In the tutorials, we will look at some other aspects of BN representation and inference (like notions of d-separation, sampling and so on)
- Here, we will just show a simple example of how the factored joint probability represented by a BN can be used to answer probabilistic queries
- Alarm again:

BNs: Representation and Inference II



BNs: Representation and Inference III

- Then, a BN will allow us to calculate answers like these:

$$\mathrm{P}(\mathit{Burglary}|\mathit{johnCalls}, \mathit{maryCalls}) = \langle 0.284, 0.716 \rangle$$

- How is this done?

$$\begin{aligned} \mathrm{P}(B|j,m) &= \alpha \sum_{\mathrm{e}} \sum_{\mathrm{a}} \mathrm{P}(B, j, m, \mathrm{e}, \mathrm{a}) \\ \mathrm{P}(B|j,m) &= \alpha \sum_{\mathrm{e}} \sum_{\mathrm{a}} \mathrm{P}(B)\mathrm{P}(j|\mathrm{a})\mathrm{P}(m|\mathrm{a})\mathrm{P}(\mathrm{e})\mathrm{P}(\mathrm{a}|B, \mathrm{e}) \end{aligned}$$

Solve separately for $B = \mathit{true}$ and $B = \mathit{false}$:

BNs: Representation and Inference IV

$$\begin{aligned} \mathrm{P}(b|j,m) &= \alpha \sum_{\mathsf{e}} \sum_{\mathsf{a}} \mathrm{P}(b)\mathrm{P}(j|\mathsf{a})\mathrm{P}(m|\mathsf{a})\mathrm{P}(\mathsf{e})\mathrm{P}(\mathsf{a}|b,\mathsf{e}) \\ &= \alpha \mathrm{P}(b) \sum_{\mathsf{e}} \mathrm{P}(\mathsf{e}) \sum_{\mathsf{a}} \mathrm{P}(j|\mathsf{a})\mathrm{P}(m|\mathsf{a})\mathrm{P}(\mathsf{a}|b,\mathsf{e}) \end{aligned}$$

(Similarly for $B = \mathit{false}$)
From the CPTs in the BN:

$$\begin{aligned} \mathrm{P}(b|j,m) &= \alpha 0.000592 \\ \mathrm{P}(\neg b|j,m) &= \alpha 0.001494 \end{aligned}$$

Normalising:

$$0.000592\alpha + 0.001494\alpha = 1.0$$
$$\alpha \approx 479$$

$$\mathrm{P}(B|j,m) = \langle 0.284, 0.716 \rangle$$

BNs: Representation and Inference V

Special Bayesian Networks I

- If we are concerned with the problem of conditional class probability estimation ($P(Y|X)$), then some special Bayesian networks result by making specific assumptions
- Suppose we have observed n data points, each of which labelled by a random variable Y which takes values from a discrete set (say: $\{+, -\}$, for simplicity) Each data point is a d-dimensional random vector $\mathbf{X} = [X_1, X_2, \ldots, X_d]^T$ where the $X_i \in \Re$.
- Given a particular vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ we wish to obtain an estimate of the conditional probabilities of $Y = +$ (the corresponding probability for $Y = -$ follows automatically). From Bayes rule, the relevant posterior is given as:

$$P(Y = +|\mathbf{x}) = \frac{P(\mathbf{x}|Y = +)P(Y = +)}{P(\mathbf{x})}$$

Special Bayesian Networks II

Two well-known simple cases follow from specific assumptions about the data:

1. The assumption that the class-conditional densities $P(\mathrm{X}|\cdot)$ are from the exponential class (of which the Gaussian is a member), and and equi-probabile priors on the values of Y results in:

$$P(Y = +|\mathrm{X}) \propto \frac{1}{1 + e^{-\mathrm{w}^T \mathrm{x}}}$$

The *logistic regression* procedure uses the sample data and the maximum likelihood principle (correctly, conditional likelilood) to estimate probabilities under this assumption.

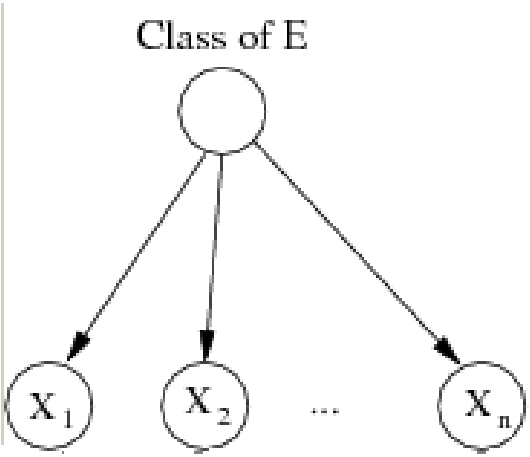
Special Bayesian Networks III

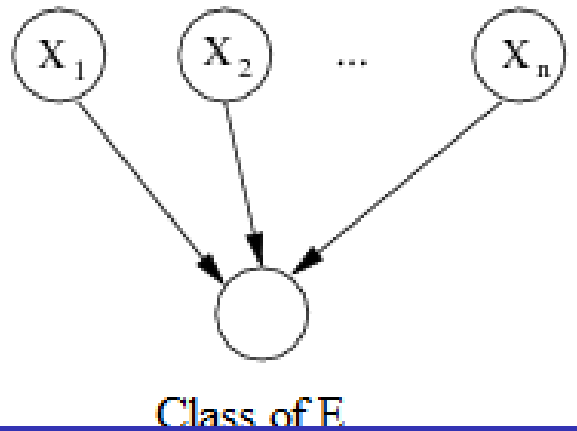
2. The assumption that the X_i are conditionally independent of each other given the value of Y, results in:

$$P(Y = +|\mathrm{X}) \propto \prod_{1}^{d} P(X_i|Y = +)P(Y = +)$$

The *naive Bayes* procedure uses sample data to estimate probabilities under this assumption.

- Both naive Bayes and logistic regression procedures simply encode specific Bayesian network topologies that reflect the underlying assumptions described





Special Bayesian Networks IV

- For both logistic regression and naive Bayes, estimation of parameters (logistic regression) or class-conditional probabilities (naive Bayes) can be done very efficiently
- These represent two of the simplest kinds of Bayesian networks for which tractable computation procedures exist. They have been shown empirically to be able to model a very wide range of observed data quite well

Learning Bayesian Networks

- A Bayesian network is an example of modelling the joint distribution of several observed variables by incorporating prior and conditional dependency information
  - "Fitting the (joint) distribution" requires estimation of two aspects of the model: (a) its structure (the DAG); and (b) the CPTs (its parameters)
  - The structure is dependent on conditional dependencies in the domain. For the present, we will assume that this known beforehand.
- Most of the following is based on material in R.E. Neapolitan's "Learning Bayesian Networks"

Learning Parameters I

- Restricting ourselves to Boolean-valued r.v.'s, each entry in the CPT of a BN is like a biased coin. So, estimating the values in a CPT is like estimating the probabilities of many biased coins p_1, p_2, ...
- Recall that the probability p of biased coin landing *heads* can be obtained from data, by modelling them as a sequence of Bernoulli trials. The maximum likelihood estimate (MLE) of p is then:

$$p_{ML} = \frac{s}{n}$$

  where s is the number of *heads* in n trials.
- Later, we will look at the the EM algorithm is essentially a generalisation of maximum likelihood that enables us to obtained ML estimates even when data are missing

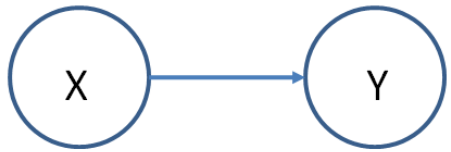
Learning Parameters II

- There are difficulties with the MLE. For example, when estimating p if $s = n = 10$ (either real or weighted counts), the MLE is simply 1
  - Does this seem reasonable? (That is, are we now sure that the coin is a double-headed one?)
  - What about if $s = n = 1000$? And if $s = n = 0$?
  - There is no role for incorporating prior information into the MLE calculations
- One way is to go back and obtain the MAP estimates using the Beta-update formulae

Parameter Estimation in a Bayes Net I

Continuing to assume binary-valued r.v's for the present. Then each CPT entry in the network is simply a specification of coin

	P(X)
X = 1	(a11,b11)



	P(Y=y1|X)
X = 1	(a21,b21)
X =2	(a22,b22)

- For each CPT "coin", we can specify Beta priors

Parameter Estimation in a Bayes Net II

- That is, each probability entry in the CPT has an associated pair of values (a_{ij}, b_{ij}) representing Beta priors. Bayesian networks with such Beta priors are sometimes called *augmented Bayesian networks*
- To be consistent, we would want $a_{11} = a_{21} + b_{21}$; and $b_{11} = a_{22} + b_{22}$
- Estimate the conditional probability from data using the update rule earlier (i.e. a Beta with parameters updated by the corresponding counts)
- For example, suppose you observed data for the network as follows:

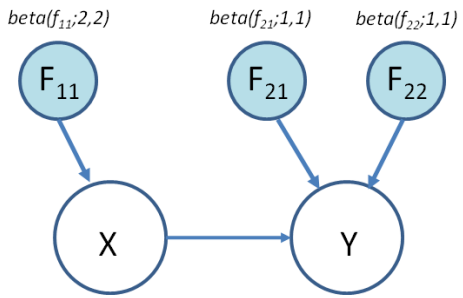
S.No.	X	Y
1	x1	y1
2	x1	y1
3	x1	y1
4	x1	y1
5	x1	y1
6	x1	y2
7	x2	y1
8	x2	y1
9	x2	y2
10	x2	y2

Parameter Estimation in a Bayes Net III

- Then, we update the values for the parameters of the Beta distribution as follows: $a'_{11} = a_{11} + 6$; $b'_{11} = b_{11} + 4$; $a'_{21} = a_{21} + 5$; $b'_{21} = b_{21} + 1$; $a'_{22} = a_{22} + 2$; and $b'_{22} = b_{22} + 2$
  - The probabilities then follow from the expected values of the updated Beta distributions
  - That is $P(X) = \frac{a'_{11}}{a'_{11}+b'_{11}}$; $P(Y = y1|X = x1) = \frac{a'_{21}}{a'_{21}+b'_{21}}$; and so on

MAP Estimates with Missing Data I

- It is quite common in a Bayes Net not to have data for all r.v.'s in the net. What do we do then?
- Here is a Bayesian network augmented with Beta priors:



- Suppose we want to obtain estimates of its parameters based on the following data D_1:

MAP Estimates with Missing Data II

S.No.	X	Y
1	x1	y1
2	x1	y1
3	x1	y1
4	x1	y2
5	x2	y2

- Suppose s_{21} be the instances that have $Y = y1$ and $X = x1$; and t_{21} be the number of instances that have $Y = y2$ and $X = x1$. Then, we know: $f(\theta_{21}|D_1) = Beta(\theta_{21}; a_{21} + s_{21}, b_{21} + t_{21})$
- Now suppose we want instead to update parameters using data D_2:

S.No.	X	Y
1	x1	y1
2	x1	?
3	x1	y1
4	x1	y2
5	x2	?

What should the missing values be?

MAP Estimates with Missing Data III

- Estimate the missing value of Y using $P(Y|X = x1)$. Here: $P(Y = y1|X = x1) = E(Beta(\theta_{21}; 1, 1)) = 1/2$. So, $P(Y = y2|X = x1) = 1 - 1/2 = 1/2$
  - Q: How does this help us work out the value of Y?
  - A: Introduce weighted instances, proportional to the probability of each value:

S.No.	X	Y	Occurs
2	x1	y1	1/2
2'	x1	y2	1/2

  - Similarly, with instance 5:

S.No.	X	Y	Occurs
5	x2	y1	1/2
5'	x2	y2	1/2

  - So, the full table is now D'_2:

S.No.	X	Y	Occurs
1	x1	y1	1
2	x1	y1	1/2
2'	x1	y2	1/2
3	x1	y1	1
4	x1	y2	1
5	x2	y1	1/2
5'	x2	y2	1/2

MAP Estimates with Missing Data IV

We can now calculate the posterior estimates using this table.

- This is one step of the EM algorithm, adapted to MAP estimation in a Bayesian network

- The weighted instances are obtained by calculating the expected value of the number of occurrences for the corresponding values of X and Y (just like we calculated teh expected value for the number of *heads* before)
  - So one step of the estimation process calculates expected values for missing data
- Once we have the expected values, we can calculate the posterior estimates using the usual Beta update rule. These new estimates can then be used to repeat the process, re-calculating the new expected values for the missing data, now using the updated Beta parameters
  - This second step is increases the posterior probability of the parameters given the data

MAP Estimates with Missing Data V

- This can be shown to converge, under some conditions, to a local maximum of the posterior probability of the parameters given the data.

Learning Bayesian Network Structure

- Now we will look at the second part of the Bayesian Network learning problem, of how to learn the structure of the network
- The network we want to identify will be the DAG that has the highest *Bayesian score*. For any DAG $\mathcal{G}$, and observations D, this is just $P(D|\mathcal{G})$ (the likelihood of the data, given G)
- For this, we need to know how to compute $P(D|\mathcal{G})$ for a given D and $\mathcal{G}$

$P(D|\mathcal{G})$ I

- We already know that if we start with a prior distribution $f(\theta) = Beta(a, b)$ on the bias of a coin, then observing data D consisting of s heads and t tails, gives us a posterior distribution

$$f(\theta|D) \;=\; Beta(a+s, b+t)$$

- We have also seen that the probability of the data:

$$P(d|Dir(a,b)) \;=\; \frac{\Gamma(a+s, b+t)}{\Gamma(a,b)}$$

Let us call this $\Gamma(a, b; s, t)$

- Recall also that if the *heads* and *tails* are partitioned across K coins then

$$P(d|Dir(a_1, b1); \cdots; Dir(a_K, b_K)) \;=\; \prod_{k=1}^{K} \Gamma(a_k, b_k; s_k, t_k)$$

$P(D|\mathcal{G})$ II

- Any particular network structure $\mathcal{G}$ simply specifies K coins
- So, to compute likelihood $P(D|\mathcal{G}$ is simply the $P(d)$ value we obtained above. That is, we will obtain one $\Gamma(a_k, b_k; s_k, t_k$ term for each biased in the network
- So, the overall likelihood will be:

$$P(D|\mathcal{G}) = \prod_{i=1}^{n} \prod^{q_i} \Gamma(a_{ij}, b_{ij}; s_{ij}, t_{ij})$$

where:

n is the number of variables
q_i is the number of different instantiations of the parents of X_i (i.e. rows of the CPTs)
a_{ij} is the prior count of the number of times X_i takes the value *heads* when the parents of X_i take their jth instantiation

$P(D|\mathcal{G})$ III

b_{ij} is the prior count of the number of times X_i takes the value *tails* when the parents of X_i take their jth instantiation
s_{ij} is the observed count of the number of times X_i takes the value *heads* when the parents of X_i take their jth instantiation
t_{ij} is the observed count on the number of times X_i takes the value *tails* when the parents of X_i take their jth instantiation

- The extension to multinomial variables is as before. Priors are given by the Dirichlet distribution, and we get a formula for $P(D|\mathcal{G})$ that generalises the one above
- There are many other scoring functions (for example, the "Bayesian Information Criterion" or BIC). These are called *asymptotically correct* if with large datasets, the DAG with the maximum value is the same as the DAG with the maximum $P(D|\mathcal{G})$

Structure Estimation as Model Selection I

- If we have several potential DAG structures $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$ we can use Bayes' theorem to calculate:

$$P(\mathcal{G}_i|D) \;=\; \alpha P(D|\mathcal{G}_i)P(\mathcal{G}_i)$$

  (recall: Discrete Hypotheses, Discrete Data in BML) provided we know the $P(\mathcal{G}_i)$ (prior over possible DAGs). We want to find the DAG that maximises $P(\mathcal{G}_i|D)$

- In general, it will be easier to use a logarithmic form of this

Structure Estimation as Model Selection II

- Taking logs on both sides, and ignoring the constant, we want to maximise: get:

$$\log P(\mathcal{G}_i|D) \;=\; \log P(D|\mathcal{G}_i) \;+\; \log P(\mathcal{G}_i)$$

Alternatively, we want to minimise:

$$L(\mathcal{G}_i, D) \;=\; -\log P(D|\mathcal{G}_i) \;-\; \log P(\mathcal{G}_i))$$

The quantity on the r.h.s. is called the *description length* of the network. It is composed of two terms: the length (in bits, using an optimal code) to encode the data, given a network; and the length (in bits, using an optimal code) to encode the network itself. That is:

$$L(\mathcal{G}_i, D) \;=\; L(D|\mathcal{G}_i) + L(\mathcal{G}_i)$$

Structure Estimation as Model Selection III

Maximising using Bayes Rule is therefore equivalent to finding the network with minimum description length. This is called the MDL principle.

- If we use a prior in which larger networks have lower prior probability, then the second term in the description length is larger. So, this would bias the search towards the smallest DAGs that explain the data adequately
  - The BIC scoring function which we will see shortly is an example of a score that employs such a prior
- Provided the space of DAGs is small, we can search all of it, either using $L(\cdot, \cdot)$ or an asymptotically correct scoring function
- Otherwise, we will have to resort to heuristic search methods, that search the space of DAGs. This search may be incomplete and we may not get the optimal answer

Structure Estimation as Model Selection IV

- If our priors are consistently assigned, we will also not be able to distinguish between (equivalence) classes of DAGs given the data. These DAGs are "Markov equivalent"

Search for Models I

- It is possible to devise heuristic search procedures that traverse the space of all possible DAGs. We will not have time to look into those here
- These procedures are usually not exhaustive, in the sense that they are not going to examine every possible DAG (actually, we need only examine DAGs in each Markov equivalence class anyway)
- An example of a such a search procedure is a standard hill-climbing procedure that starts with all disconnected nodes, and then makes small local "refinements", like adding or deleting edges, reversing edges *etc.* and recompute the score. Each new DAG will require estimation of CPT entries (as before). The DAG with the best score (lowest description length) is selected and the procedure repeated.

Search for Models II

- It would also be more efficient if the search performed by the procedure was such that a move from a DAG $\mathcal{G}_1$ to another $\mathcal{G}_2$ only required a small amount of recomputation to obtain the value of $P(D|\mathcal{G}_2)$ from $P(D|\mathcal{G}_1)$ (or the corresponding L values) This is called a *local scoring update*
- For each DAG $\mathcal{G}$ examined by the search, the score is computed. The DAG returned is the one with the highest score.

Summary

- A special kind of numeric model is one that predicts probability of events co-occurring or occurring conditionally
- Probability models are often described as *discriminative* (*i.e.* models for $P(Y|X)$ or *generative* (*i.e.* models for $P(Y, X)$). Using the rules of the usual probability calculus, we can obtain the former from the latter, but not *vice-versa*.
  - So, why would you build discriminative models at all?
- The simplest kind of discriminative model we looked at was *Logistic Regression*. The simplest kind of generative model we looked at was *Naive Bayes*. Both are Bayesian networks with a fixed structure, and learning only deals with estimating parameters
- Bayesian networks are a flexible way of modelling almost any kind of joint distribution. We looked at learning parameters and learning the structure of Bayesian networks, using Beta (Dirichlet) priors for parameters, and graph-based learning for structure