An introduction to Bayesian Networks and the Bayes Net Toolbox for Matlab

Kevin Murphy
MIT AI Lab

19 May 2003

Outline

- An introduction to Bayesian networks
- An overview of BNT

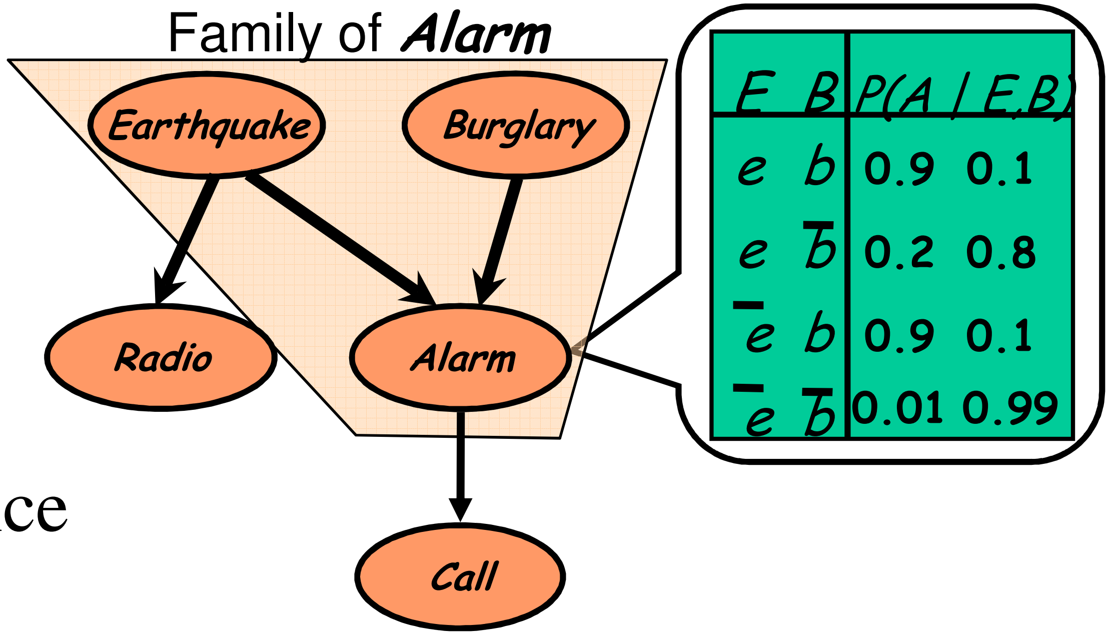
What is a Bayes (belief) net?

Compact representation of joint probability distributions via conditional independence

Qualitative part:

Directed acyclic graph (DAG)

- Nodes - random vars.
- Edges - direct influence



E	B	$P(A \mid E,B)$	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

Together:
Define a unique distribution in a factored form

Quantitative part:
Set of conditional probability distributions

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

Figure from N. Friedman

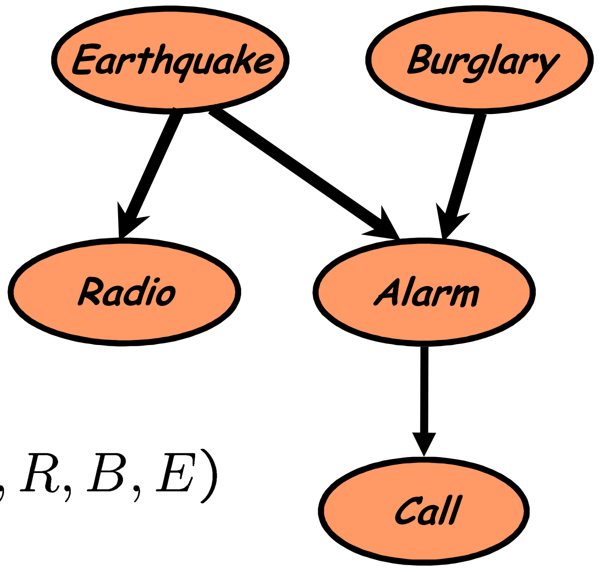
What is a Bayes net?

A node is conditionally independent of its ancestors given its parents, e.g.

C ? R,B,E | A

Hence

$$
\begin{aligned}
P(E,B,R,A,C) \\
&= P(E)P(B|E)P(R|B,E)P(A|R,B,E)P(C|A,R,B,E) \\
&= P(E)P(B)P(R|E)P(A|B,E)P(C|A)
\end{aligned}
$$



From $2^5 - 1 = 31$ parameters to 1+1+2+4+2=10

Why are Bayes nets useful?

- Graph structure supports
  - Modular representation of knowledge
  - Local, distributed algorithms for inference and learning
  - Intuitive (possibly causal) interpretation

- Factored representation may have exponentially fewer parameters than full joint $P(X_1,\ldots,X_n)$ =>
  - lower sample complexity (less data for learning)
  - lower time complexity (less time for inference)

What can Bayes nets be used for?

- **Posterior probabilities**
  - Probability of any event given any evidence
- Most likely explanation
  - Scenario that explains evidence
- Rational decision making
  - Maximize expected utility
  - Value of Information
- Effect of intervention
  - Causal analysis

Explaining away effect

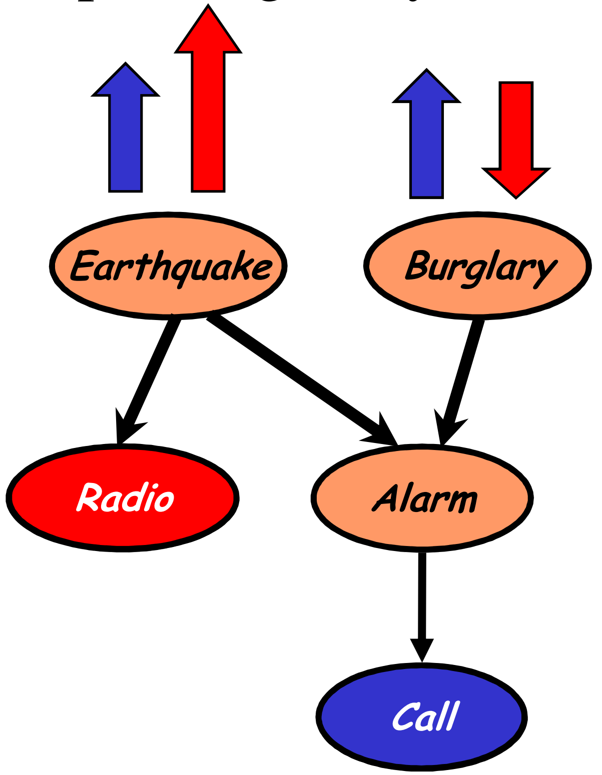


Figure from N. Friedman

A real Bayes net: Alarm

Domain: Monitoring Intensive-Care Patients

- 37 variables
- 509 parameters

 …instead of 2^{37}

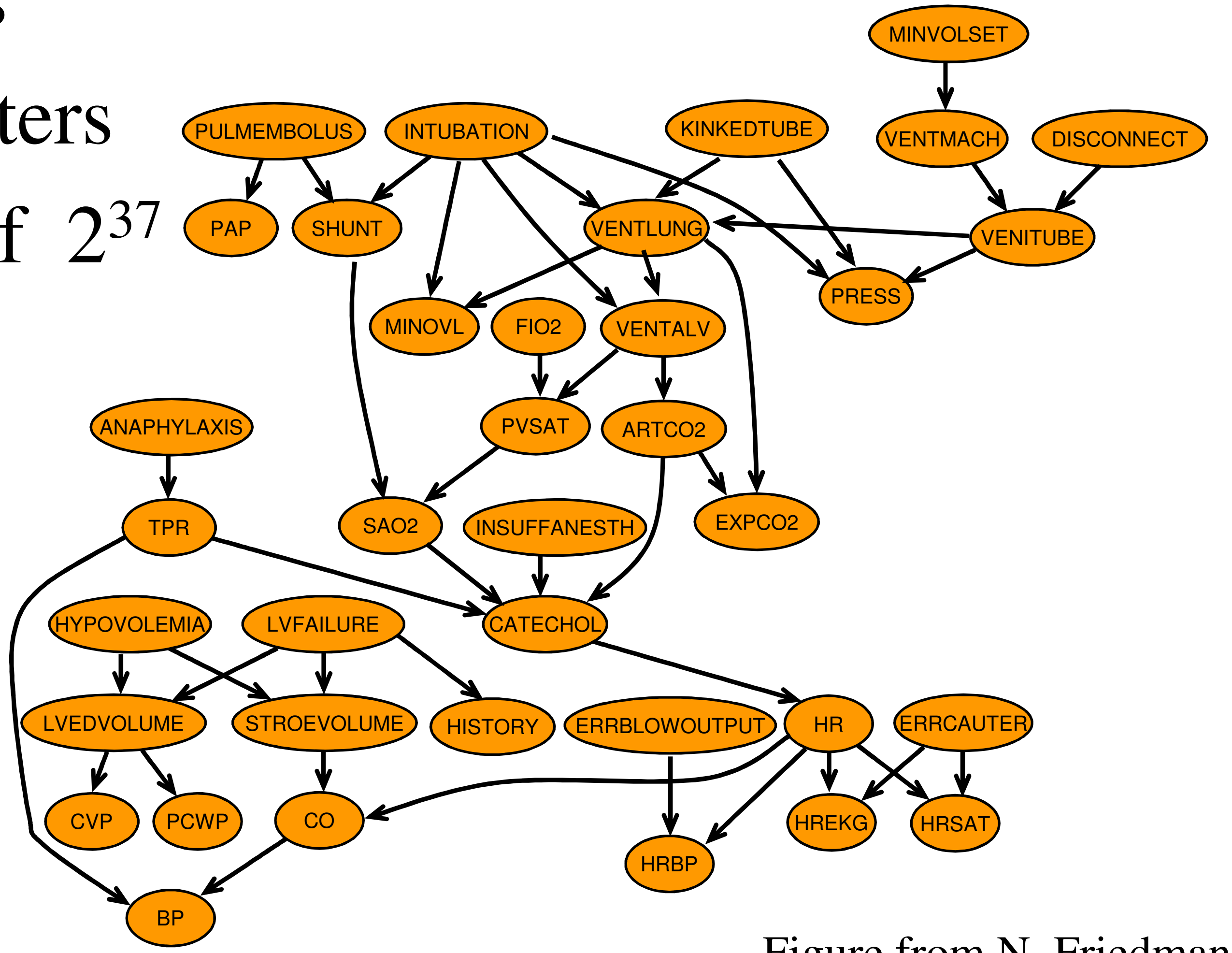


Figure from N. Friedman

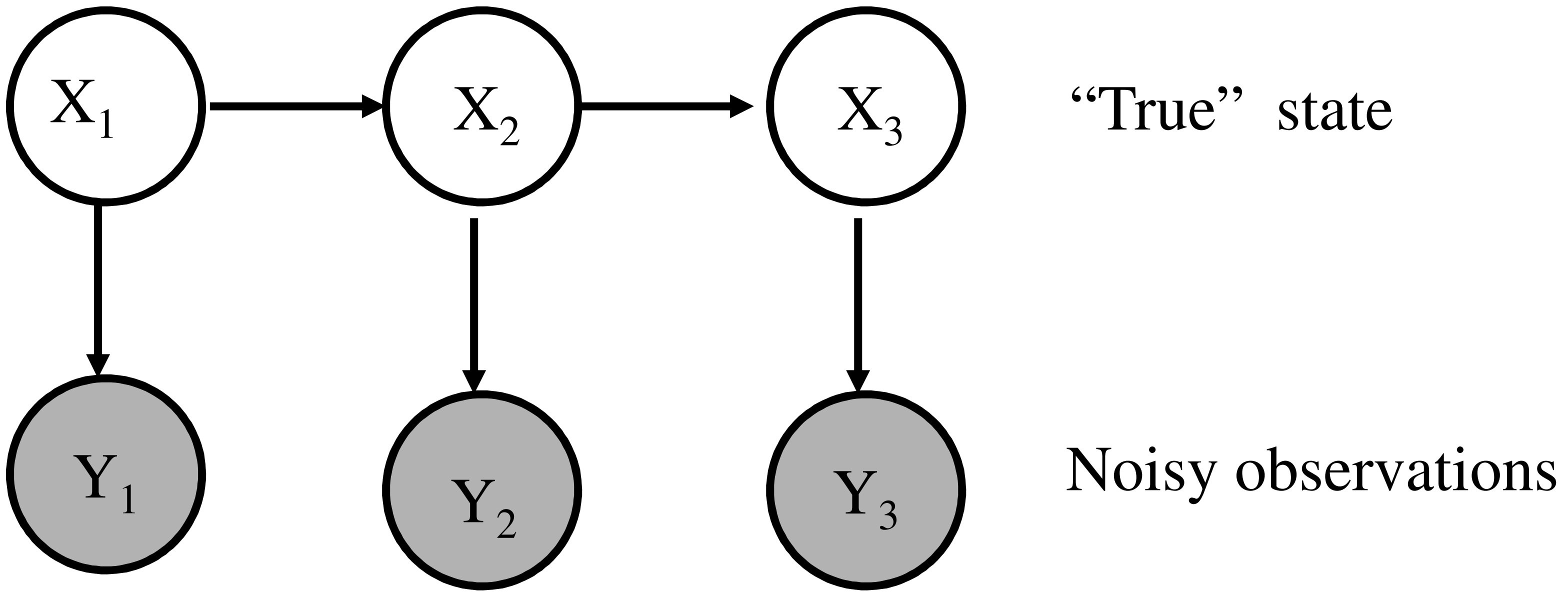
More real-world BN applications

- "Microsoft's competitive advantage lies in its expertise in Bayesian networks"
  -- Bill Gates, quoted in LA Times, 1996
- MS Answer Wizards, (printer) troubleshooters
- Medical diagnosis
- Genetic pedigree analysis
- Speech recognition (HMMs)
- Gene sequence/expression analysis
- Turbocodes (channel coding)

Dealing with time

- In many systems, data arrives sequentially
- Dynamic Bayes nets (DBNs) can be used to model such time-series (sequence) data
- Special cases of DBNs include
  - State-space models
  - Hidden Markov models (HMMs)

State-space model (SSM)/ Linear Dynamical System (LDS)



$$p(X_t|X_{t-1}) = \mathcal{N}(X_t; AX_{t-1}, Q)$$

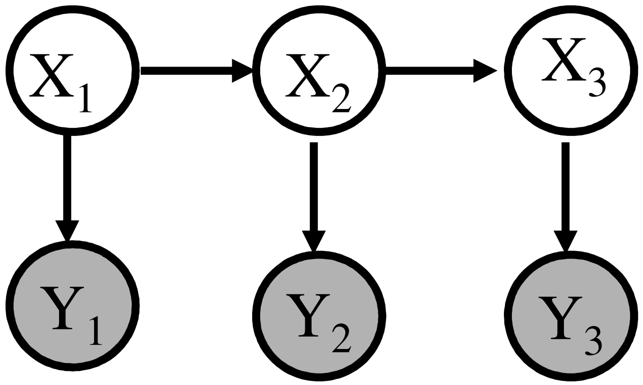
$$p(Y_t|X_t) = \mathcal{N}(Y_t; BX_t, R)$$

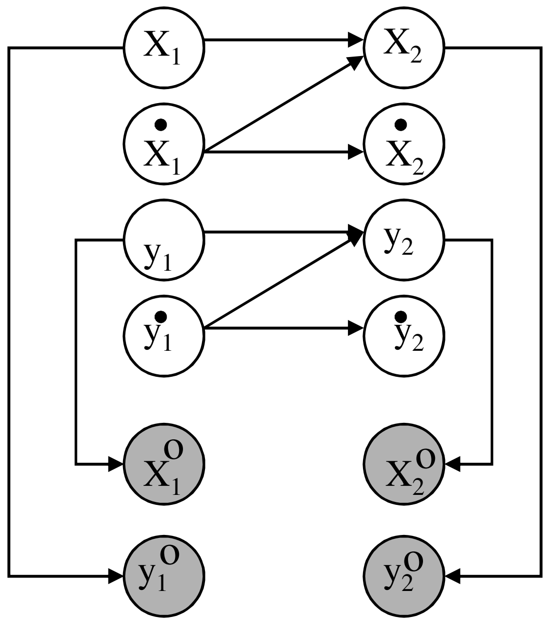
Example: LDS for 2D tracking

$$\begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix} = \begin{pmatrix} 1 & 0 & \triangle & 0 \\ 0 & 1 & 0 & \triangle \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{pmatrix} + v_t$$

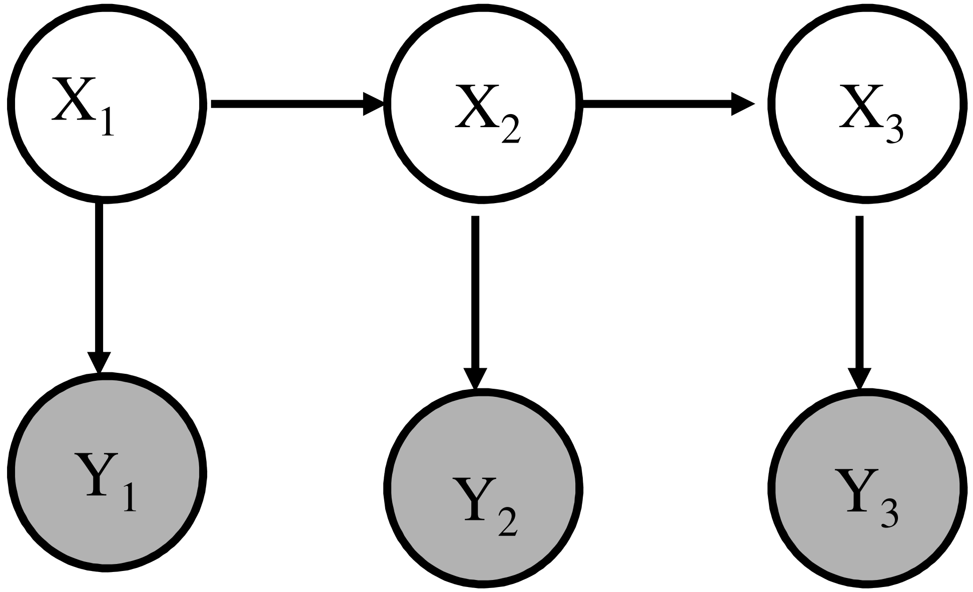
Sparse linear Gaussian systems) sparse graphs

$$\begin{pmatrix} x^o_t \\ y^o_t \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix} + W_t$$





Hidden Markov model (HMM)



Phones/ words

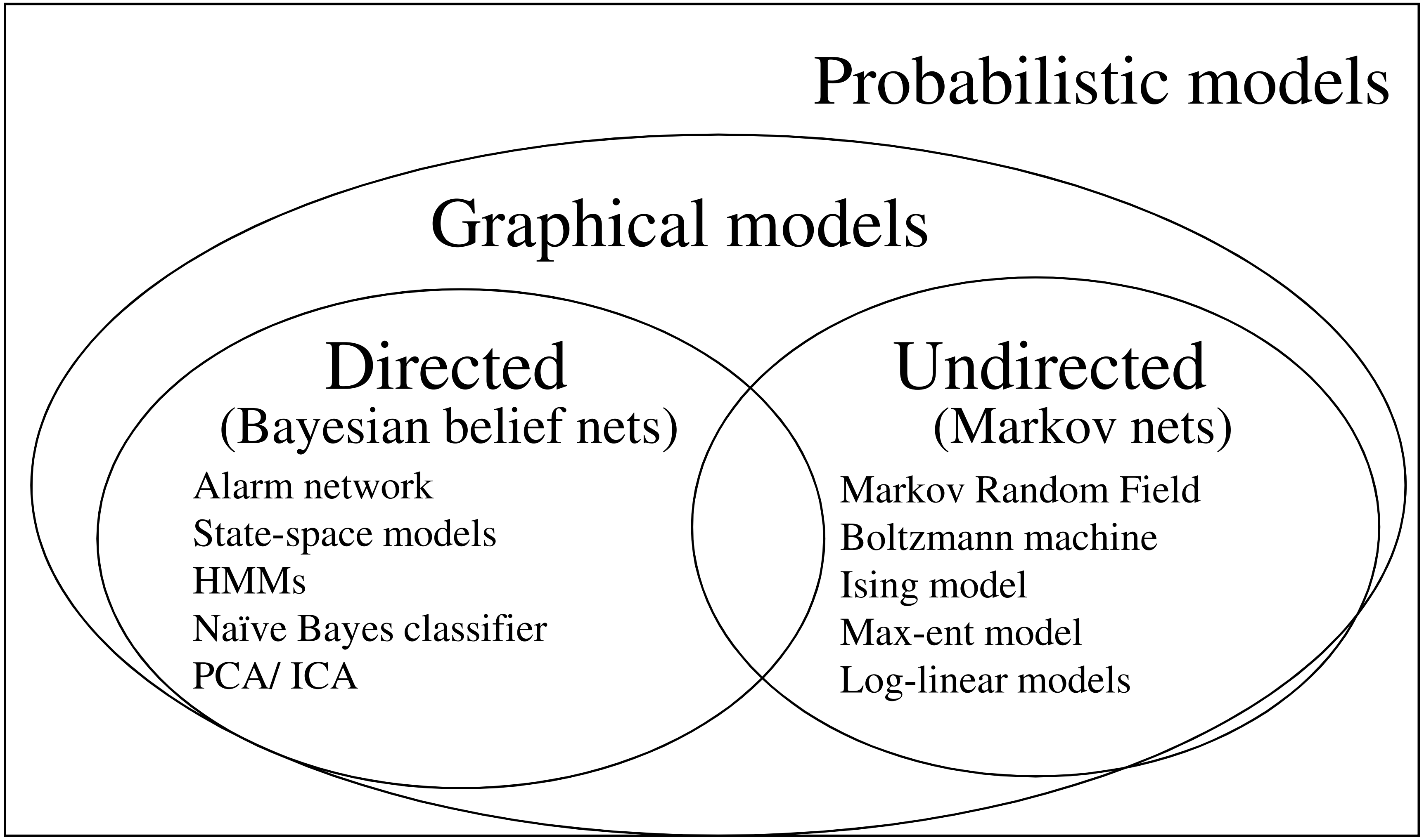
acoustic signal

Sparse transition matrix)/sparse graph

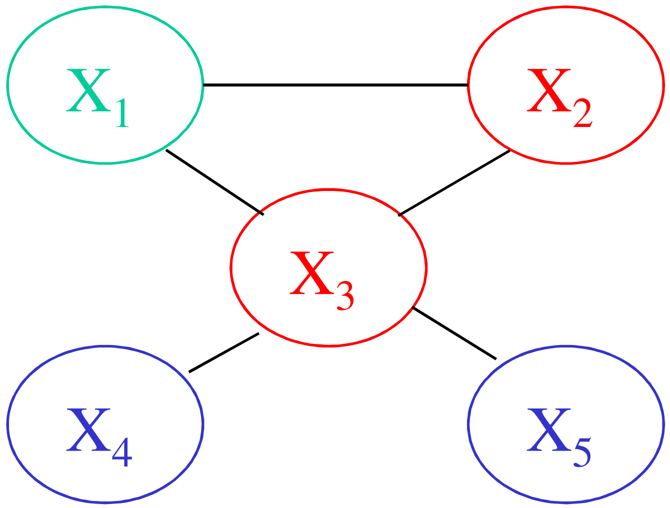
$$P(X_t = j | X_{t-1} = i) = A(i,j)$$ transition matrix

$$p(Y_t = y | X_t = i) = \mathcal{N}(y; \mu_i, \Sigma_i)$$ Gaussian observations

Probabilistic graphical models



Toy example of a Markov net



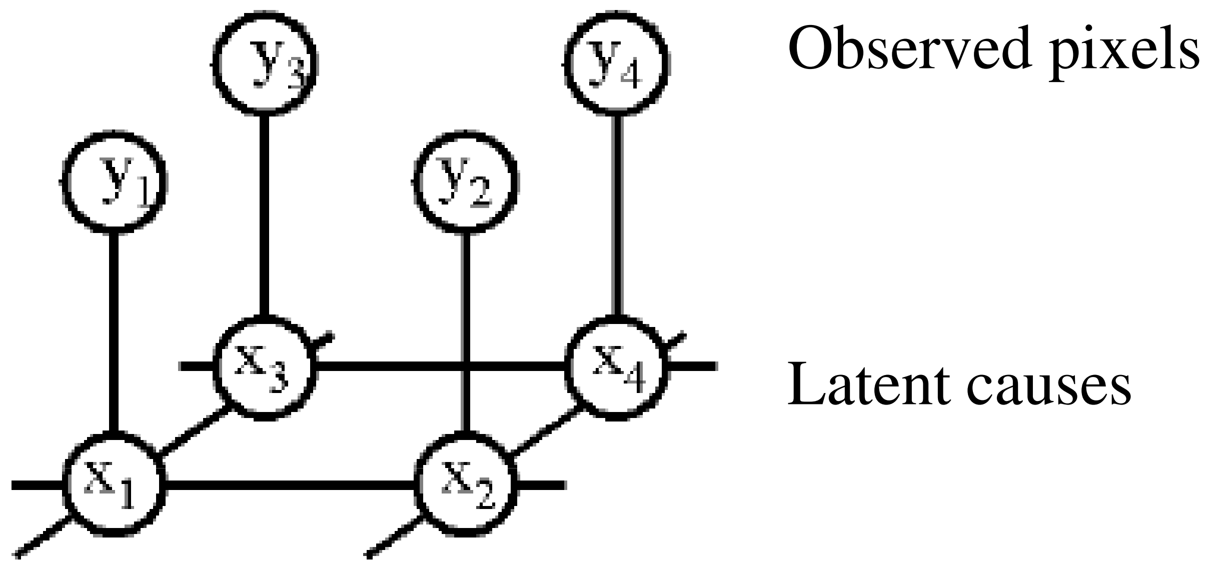
X_i ? X_{rest} | X_{nbrs} e.g, X_1 ? X_4, X_5 | X_2, X_3

Potential functions

$$P(X_{1:5}) = \frac{1}{Z}\psi(X_1, X_2, X_3)\psi(X_3, X_4)\psi(X_4, X_5)$$

Partition function

A real Markov net



•Estimate $P(x_1, \ldots, x_n \mid y_1, \ldots, y_n)$

• $\Psi(x_i, y_i) = P(\text{observe } y_i \mid x_i)$: local evidence

• $\Psi(x_i, x_j) / \exp(-J(x_i, x_j))$: compatibility matrix

c.f., Ising/Potts model

Inference

- **Posterior probabilities**
  - Probability of any event given any evidence
- *Most likely explanation*
  - *Scenario that explains evidence*
- *Rational decision making*
  - *Maximize expected utility*
  - *Value of Information*
- *Effect of intervention*
  - *Causal analysis*

Explaining away effect

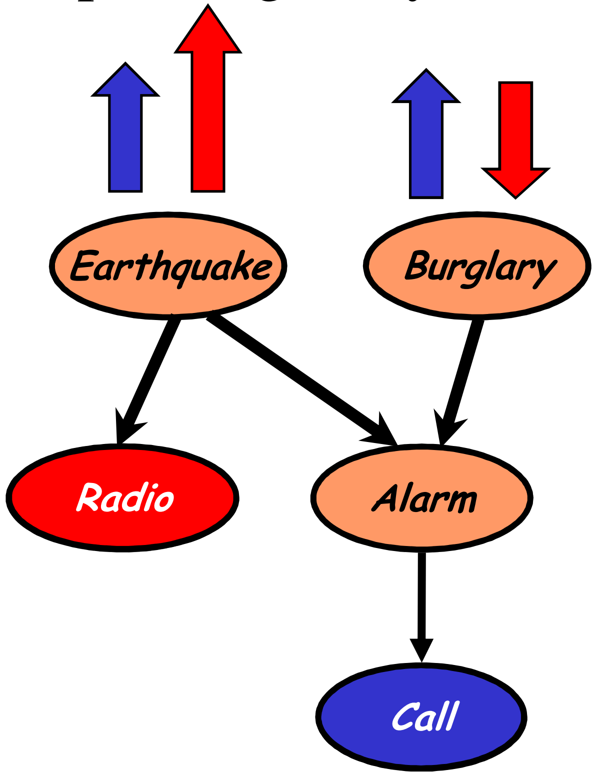
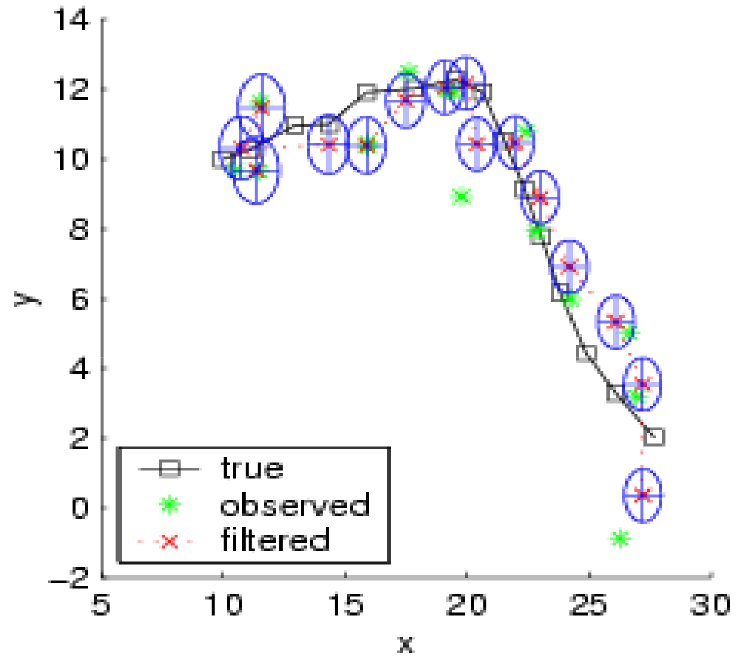
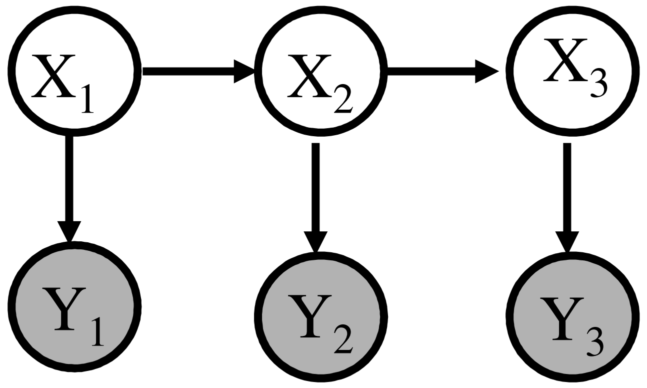


Figure from N. Friedman

Kalman filtering (recursive state estimation in an LDS)





Estimate $P(X_t|y_{1:t})$ from $P(X_{t-1}|y_{1:t-1})$ and y_t

- Predict: $P(X_t|y_{1:t-1}) = s_{Xt-1}\ P(X_t|X_{t-1})\ P(X_{t-1}|y_{1:t-1})$
- Update: $P(X_t|y_{1:t})\ /\ P(y_t|X_t)\ P(X_t|y_{1:t-1})$

Forwards algorithm for HMMs

Predict:

$$P(X_t|y_{1:t-1}) = \sum_{x_{t-1}} P(X_t|x_{t-1})P(X_{t-1}|y_{1:t-1})$$

$$\alpha_{t|t-1} = A^T \alpha_{t-1}$$

Update:

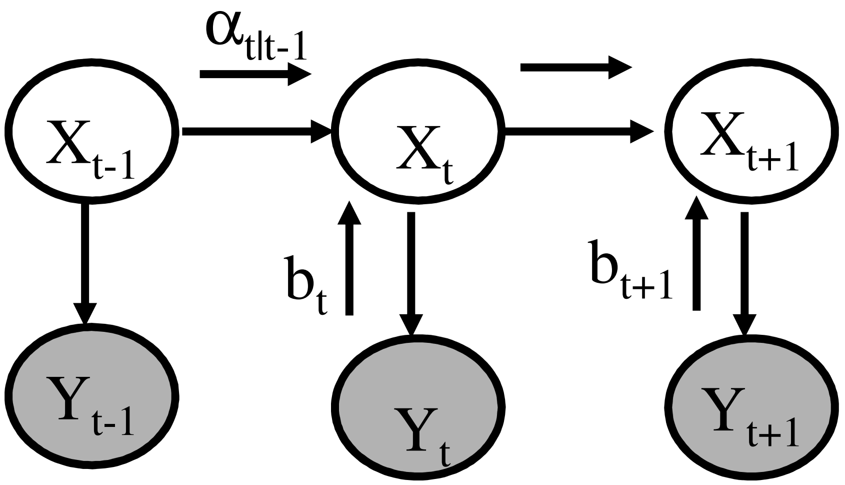
$$P(X_t = i|y_{1:t}) \propto P(X_t = i|y_{1:t-1})p(y_t|X_t = i)$$

$$\alpha_t \propto \alpha_{t|t-1} . * b_t$$

Discrete-state analog of Kalman filter

O(T S^2) time using dynamic programming

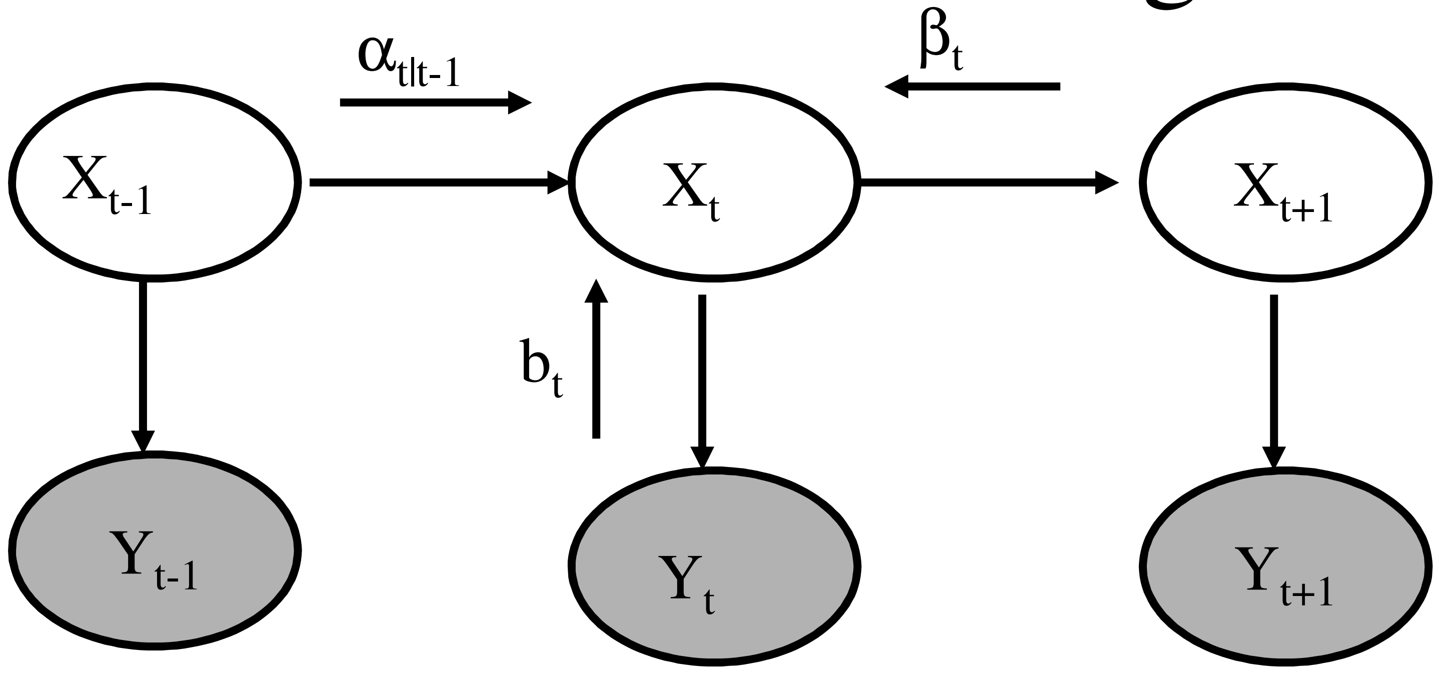
Message passing view of forwards algorithm



$$\alpha_{t|t-1} = A^T \alpha_{t-1}$$

$$\alpha_t \propto \alpha_{t|t-1} .* b_t$$

Forwards-backwards algorithm



Discrete analog of RTS smoother

$$P(X_t|y_{1:T}) \propto P(X_t|y_{1:t-1})P(y_t|X_t)P(y_{t+1:T}|X_t)$$

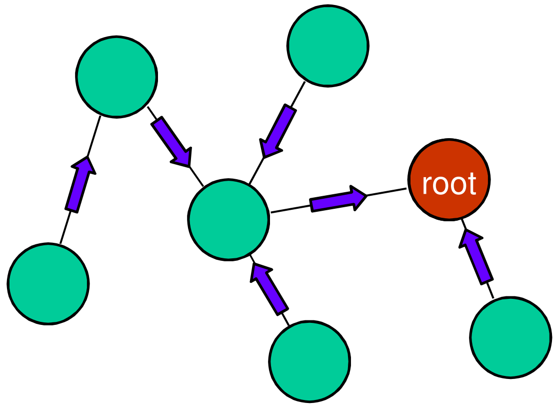
$$\gamma_t(i) \propto \alpha_{t|t-1}(i)b_t(i)\beta_t(i)$$

Belief Propagation

aka Pearl’s algorithm, sum-product algorithm

Generalization of forwards-backwards algo. /RTS smoother from chains to trees - linear time, two-pass algorithm

Collect



Distribute

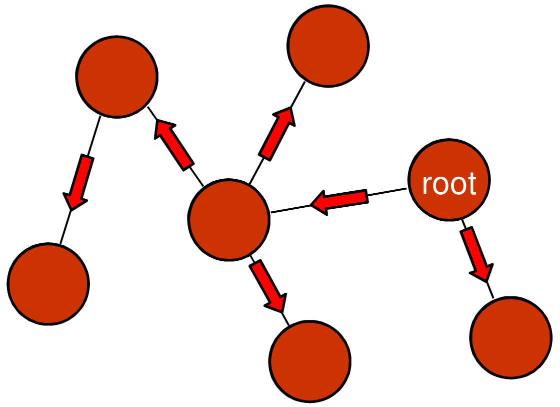
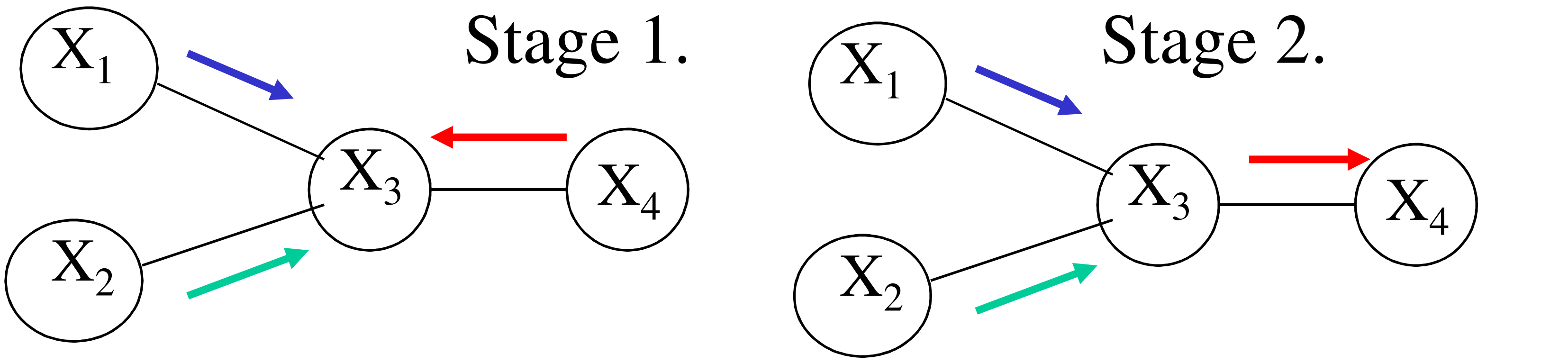


Figure from P. Green

BP: parallel, distributed version



$$bel(x_3) \propto$$

$$\underline{\mu_{1\to 3}(x_3)}\underline{\mu_{2\to 3}(x_3)}\underline{\mu_{4\to 3}(x_3)}$$

$$\underline{\mu_{3\to 4}(x_4)} =$$

$$\sum_{x_1,x_2,x_3} \underline{\mu_{1\to 3}(x_3)}\underline{\mu_{2\to 3}(x_3)}\psi(x_1,x_2,x_3,x_4)$$

Representing potentials

- For discrete variables, potentials can be represented as multi-dimensional arrays (vectors for single node potentials)
- For jointly Gaussian variables, we can use $\psi(X) = (\mu, \Sigma)$ or $\psi(X) = (\Sigma^{-1} \mu , \Sigma^{-1})$
- In general, we can use mixtures of Gaussians or non-parametric forms

Manipulating discrete potentials

Marginalization

$$\mu_3(x_3, x_4) = \sum_{x_1, x_2} \psi(x_1, x_2, x_3, x_4)$$

Multiplication

$$\phi(x_1, x_3, x_4) = \mu_3(x_3, x_4) \times \mu_1(x_1, x_3)$$

80% of time is spent manipulating such multi-dimensional arrays!

Manipulating Gaussian potentials

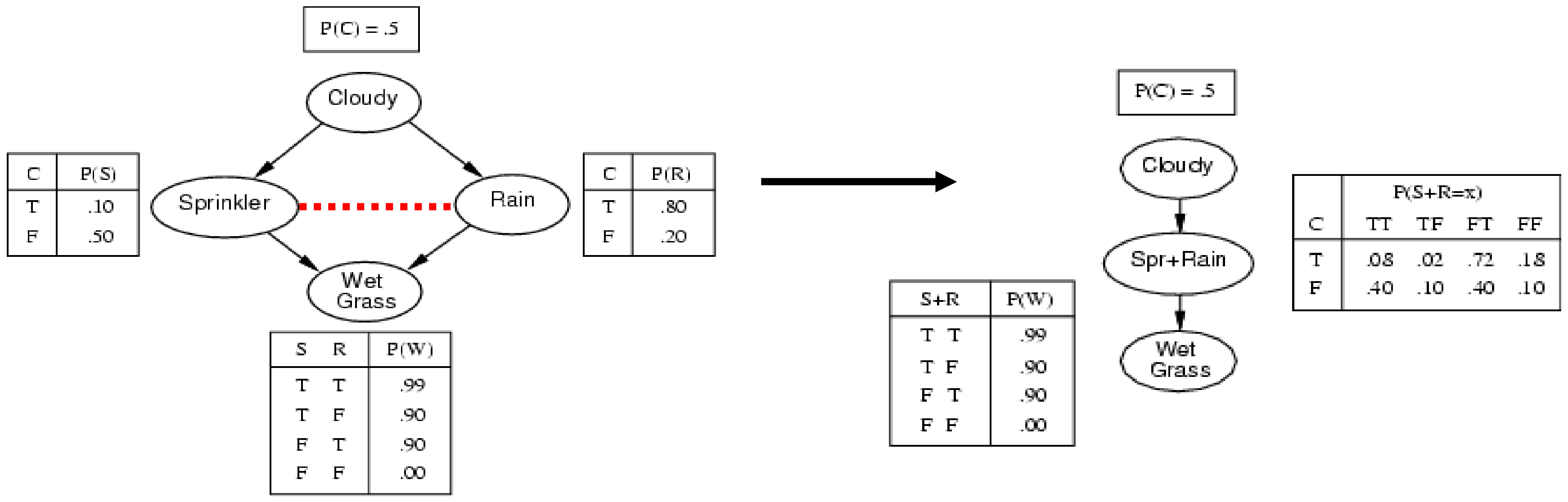
- Closed-form formulae for marginalization and multiplication
- O(1)/O(n^3) complexity per operation
- Mixtures of Gaussian potentials are not closed under marginalization, so need approximations (moment matching)

Semi-rings

- By redefining * and +, same code implements Kalman filter and forwards algorithm
- By replacing + with max, can convert from forwards (sum-product) to Viterbi algorithm (max-product)
- BP works on any commutative semi-ring!

Inference in general graphs

- BP is only guaranteed to be correct for trees
- A general graph should be converted to a junction tree, by clustering nodes
- Computationally complexity is exponential in size of the resulting clusters (NP-hard)



C	P(S)
T	.10
F	.50

C	P(R)
T	.80
F	.20

S	R	P(W)
T	T	.99
T	F	.90
F	T	.90
F	F	.00

S+R	P(W)
T T	.99
T F	.90
F T	.90
F F	.00

	P(S+R=x)			
C	TT	TF	FT	FF
T	.08	.02	.72	.18
F	.40	.10	.40	.10

Approximate inference

- Why?
  - to avoid exponential complexity of exact inference in discrete loopy graphs
  - Because cannot compute messages in closed form (even for trees) in the non-linear/non-Gaussian case
- How?
  - Deterministic approximations: loopy BP, mean field, structured variational, etc
  - Stochastic approximations: MCMC (Gibbs sampling), likelihood weighting, particle filtering, etc

- Algorithms make different speed/accuracy tradeoffs

- Should provide the user with a choice of algorithms

Learning

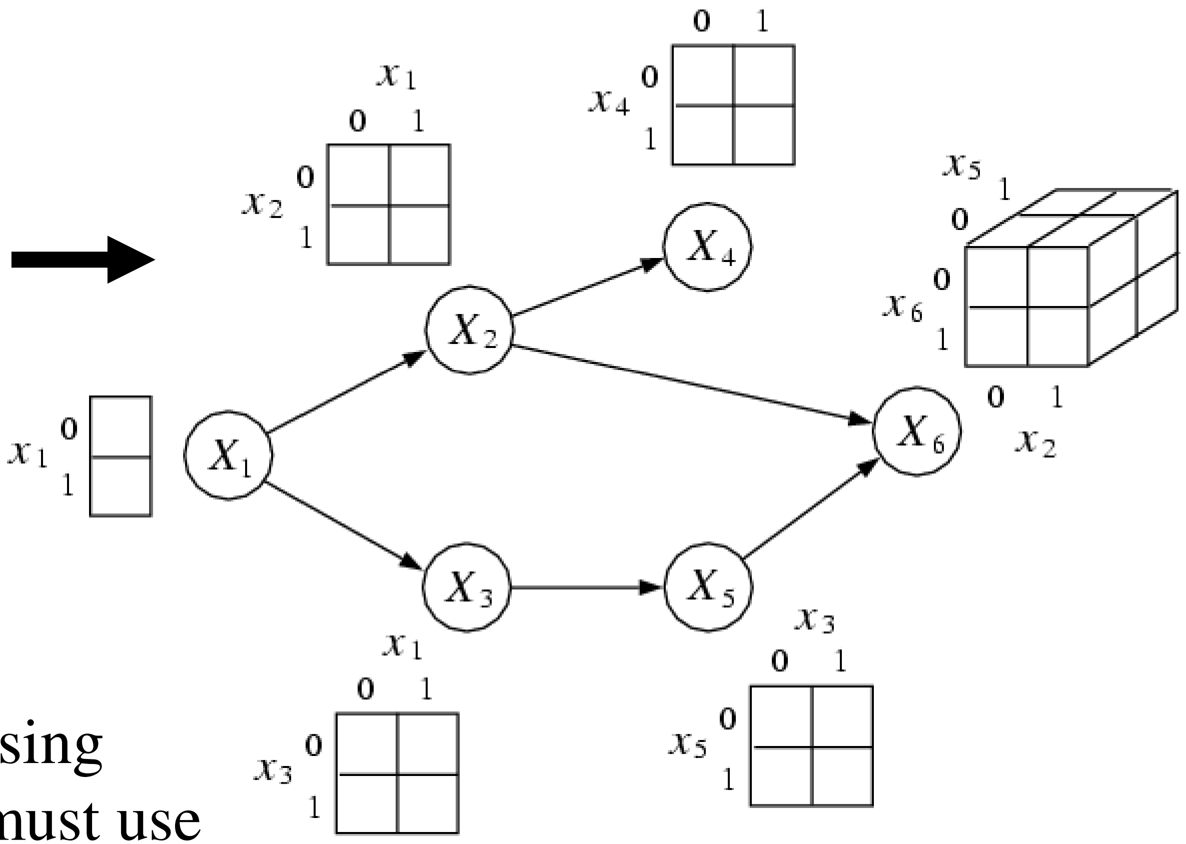
- Parameter estimation
- Model selection (structure learning)

Parameter learning

iid data

X_1	X_2	X_3	X_4	X_5	X_6
0	1	0	0	0	0
1	?	1	1	?	1
		…			
1	1	1	0	1	1

Conditional Probability Tables (CPTs)

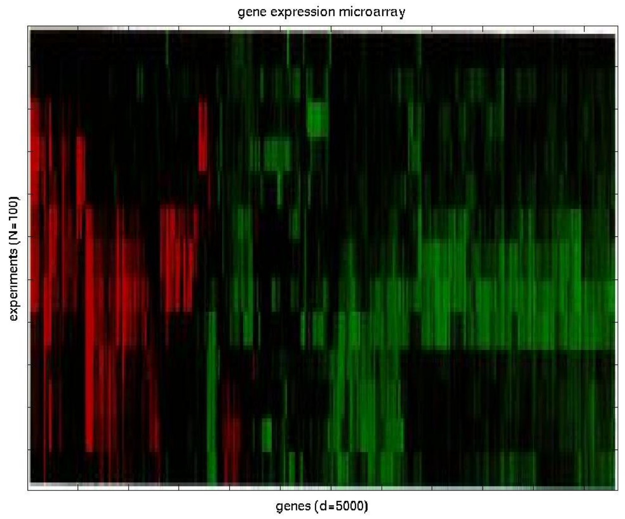


If some values are missing (latent variables), we must use gradient descent or EM to compute the (locally) maximum likelihood estimates

Figure from M. Jordan

Structure learning (data mining)

Gene expression data



Genetic pathway

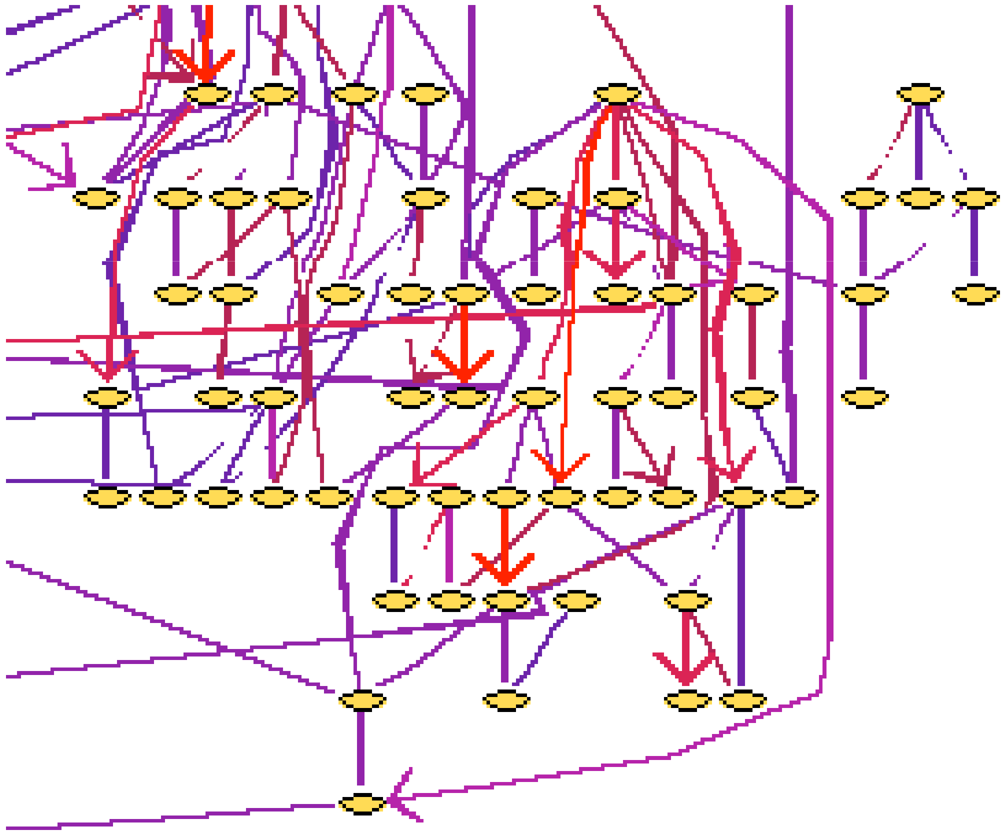


Figure from N. Friedman

Structure learning

•Learning the optimal structure is NP-hard (except for trees)
•Hence use heuristic search through space of DAGs or PDAGs or node orderings
•Search algorithms: hill climbing, simulated annealing, GAs
•Scoring function is often marginal likelihood, or an approximation like BIC/MDL or AIC

$$\begin{aligned} G^* &= \arg\max_G \log P(D|G)P(G) \\ &= \log \int_\theta P(D|G,\theta)P(\theta|G) \\ &\overset{BIC}{\approx} \log P(D|G,\theta^{ML}) \underset{\nwarrow}{-} \lambda \text{dim}(G) \end{aligned}$$

Structural complexity penalty

Summary: why are graphical models useful?

- Factored representation may have exponentially fewer parameters than full joint $P(X_1,\ldots,X_n)$ =>
  - lower time complexity (less time for inference)
  - lower sample complexity (less data for learning)

- Graph structure supports
  - Modular representation of knowledge
  - Local, distributed algorithms for inference and learning
  - Intuitive (possibly causal) interpretation

The Bayes Net Toolbox for Matlab

- What is BNT?
- Why yet another BN toolbox?
- Why Matlab?
- An overview of BNT's design
- How to use BNT
- Other GM projects

What is BNT?

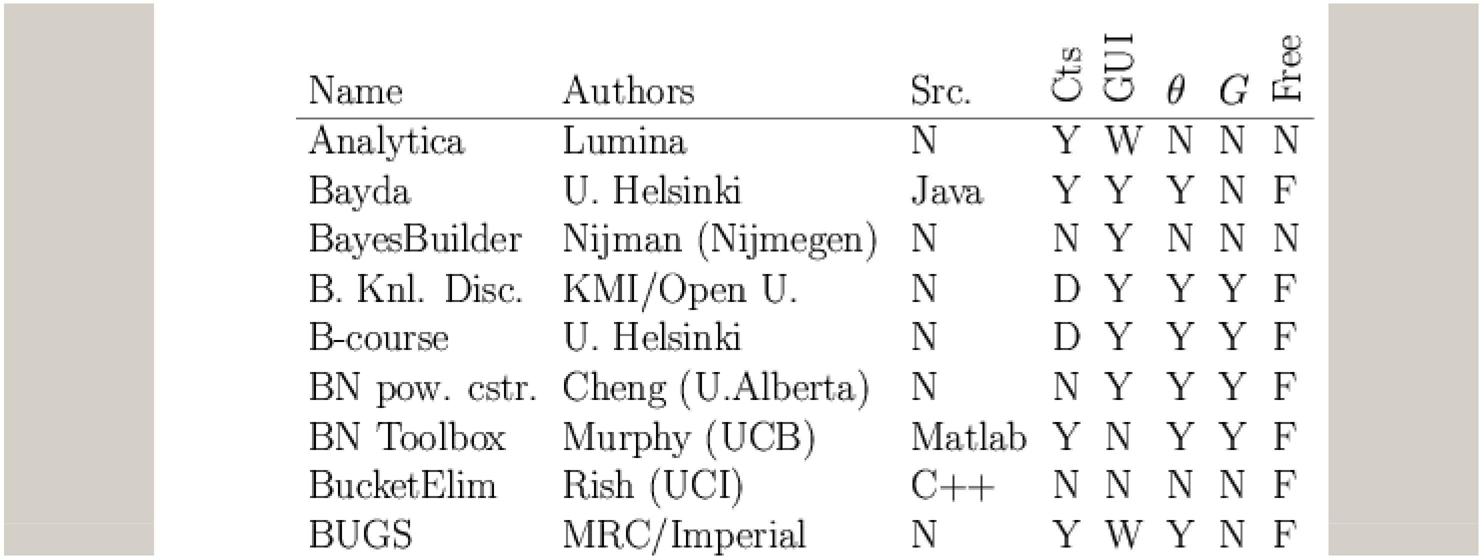
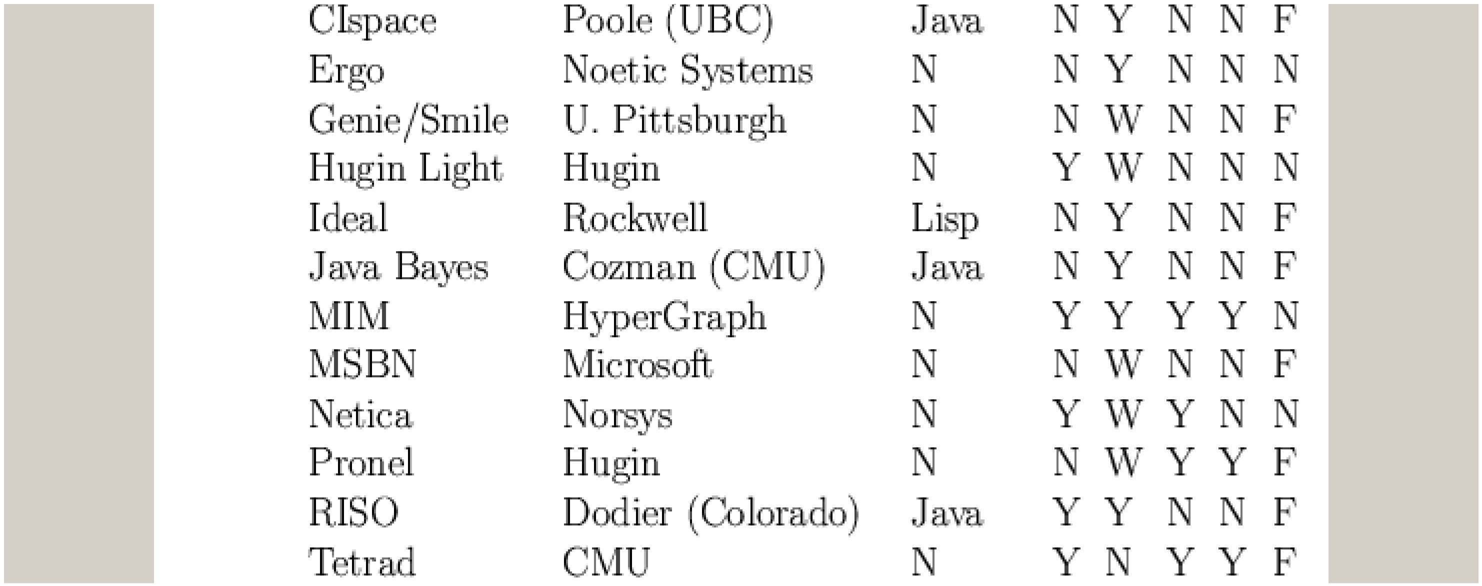
- BNT is an open-source collection of matlab functions for inference and learning of (directed) graphical models
- Started in Summer 1997 (DEC CRL), development continued while at UCB
- Over 100,000 hits and about 30,000 downloads since May 2000
- About 43,000 lines of code (of which 8,000 are comments)

Why yet another BN toolbox?

- In 1997, there were very few BN programs, and all failed to satisfy the following desiderata:
  - Must support real-valued (vector) data
  - Must support learning (params and struct)
  - Must support time series
  - Must support exact and approximate inference
  - Must separate API from UI
  - Must support MRFs as well as BNs
  - Must be possible to add new models and algorithms
  - Preferably free
  - Preferably open-source
  - Preferably easy to read/ modify
  - Preferably fast

BNT meets all these criteria except for the last

A comparison of GM software

Name	Authors	Src.	Cts	GUI	θ	G	Free
Analytica	Lumina	N	Y	W	N	N	N
Bayda	U. Helsinki	Java	Y	Y	Y	N	F
BayesBuilder	Nijman (Nijmegen)	N	N	Y	N	N	N
B. Knl. Disc.	KMI/Open U.	N	D	Y	Y	Y	F
B-course	U. Helsinki	N	D	Y	Y	Y	F
BN pow. cstr.	Cheng (U.Alberta)	N	N	Y	Y	Y	F
BN Toolbox	Murphy (UCB)	Matlab	Y	N	Y	Y	F
BucketElim	Rish (UCI)	C++	N	N	N	N	F
BUGS	MRC/Imperial	N	Y	W	Y	N	F
CIspace	Poole (UBC)	Java	N	Y	N	N	F
Ergo	Noetic Systems	N	N	Y	N	N	N
Genie/Smile	U. Pittsburgh	N	N	W	N	N	F
Hugin Light	Hugin	N	Y	W	N	N	N
Ideal	Rockwell	Lisp	N	Y	N	N	F
Java Bayes	Cozman (CMU)	Java	N	Y	N	N	F
MIM	HyperGraph	N	Y	Y	Y	Y	N
MSBN	Microsoft	N	N	W	N	N	F
Netica	Norsys	N	Y	W	Y	N	N
Pronel	Hugin	N	N	W	Y	Y	F
RISO	Dodier (Colorado)	Java	Y	Y	N	N	F
Tetrad	CMU	N	Y	N	Y	Y	F

www.ai.mit.edu/~murphyk/Software/Bayes/bnsoft.html

Summary of existing GM software

- ~8 commercial products (Analytica, BayesiaLab, Bayesware, Business Navigator, Ergo, Hugin, MIM, Netica), focused on data mining and decision support; most have free "student" versions
- ~30 academic programs, of which ~20 have source code (mostly Java, some C++/ Lisp)
- Most focus on exact inference in discrete, static, directed graphs (notable exceptions: BUGS and VIBES)
- Many have nice GUIs and database support

BNT contains more features than most of these packages combined!

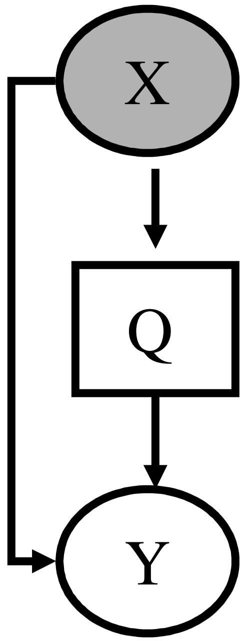
Why Matlab?

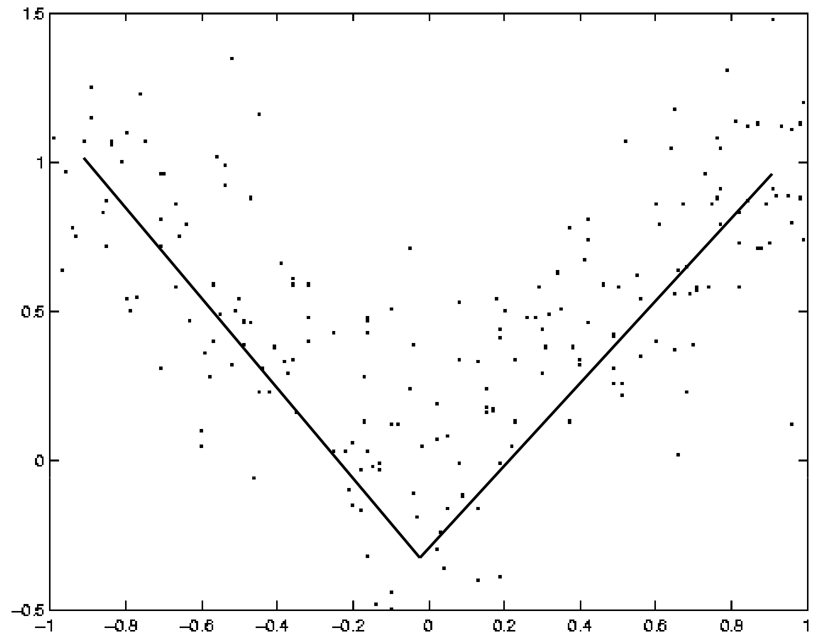
- Pros
  - Excellent interactive development environment
  - Excellent numerical algorithms (e.g., SVD)
  - Excellent data visualization
  - Many other toolboxes, e.g., netlab
  - Code is high-level and easy to read (e.g., Kalman filter in 5 lines of code)
  - Matlab is the lingua franca of engineers and NIPS
- Cons:
  - Slow
  - Commercial license is expensive
  - Poor support for complex data structures
- Other languages I would consider in hindsight:
  - Lush, R, Ocaml, Numpy, Lisp, Java

BNT's class structure

- Models – bnet, mnet, DBN, factor graph, influence (decision) diagram
- CPDs – Gaussian, tabular, softmax, etc
- Potentials – discrete, Gaussian, mixed
- Inference engines
  - Exact - junction tree, variable elimination
  - Approximate - (loopy) belief propagation, sampling
- Learning engines
  - Parameters – EM, (conjugate gradient)
  - Structure - MCMC over graphs, K2

Example: mixture of experts



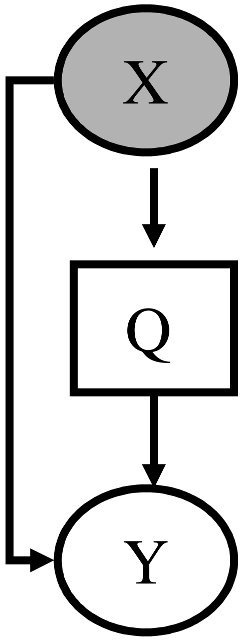


$$P(Q=i|x) = \frac{e^{w_i^T x}}{\sum_j e^{w_j^T x}}$$ softmax/logistic function

$$p(y|Q=i,x) = \mathcal{N}(y; \mu_i + \beta_i^T x, \sigma_i)$$

1. Making the graph

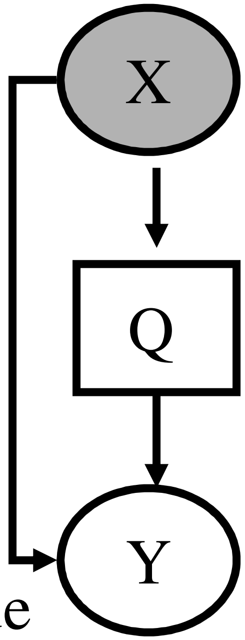
```
X = 1; Q = 2; Y = 3;
dag = zeros(3,3);
dag(X, [Q Y]) = 1;
dag(Q, Y) = 1;
```



- Graphs are (sparse) adjacency matrices
- GUI would be useful for creating complex graphs
- Repetitive graph structure (e.g., chains, grids) is best created using a script (as above)

2. Making the model

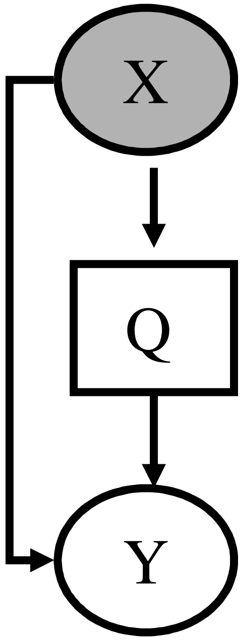
```
node_sizes = [1 2 1];
dnodes = [2];
bnet = mk_bnet(dag, node_sizes, …
    ‘discrete’, dnodes);
```



•X is always observed input, hence only one effective value
•Q is a hidden binary node
•Y is a hidden scalar node
•bnet is a struct, but should be an object
•mk_bnet has many optional arguments, passed as string/value pairs

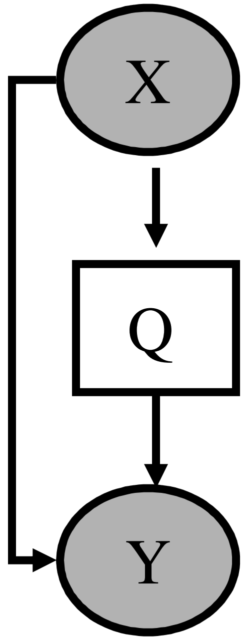
3. Specifying the parameters

```
bnet.CPD{X} = root_CPD(bnet, X);
bnet.CPD{Q} = softmax_CPD(bnet, Q);
bnet.CPD{Y} = gaussian_CPD(bnet, Y);
```



- CPDs are objects which support various methods such as
    - Convert_from_CPD_to_potential
    - Maximize_params_given_expected_suff_stats
- Each CPD is created with random parameters
- Each CPD constructor has many optional arguments

4. Training the model



```
load data -ascii;
ncases = size(data, 1);
cases = cell(3, ncases);
observed = [X Y];
cases(observed, :) = num2cell(data');
```

•Training data is stored in cell arrays (slow!), to allow for variable-sized nodes and missing values
•cases{i,t} = value of node i in case t

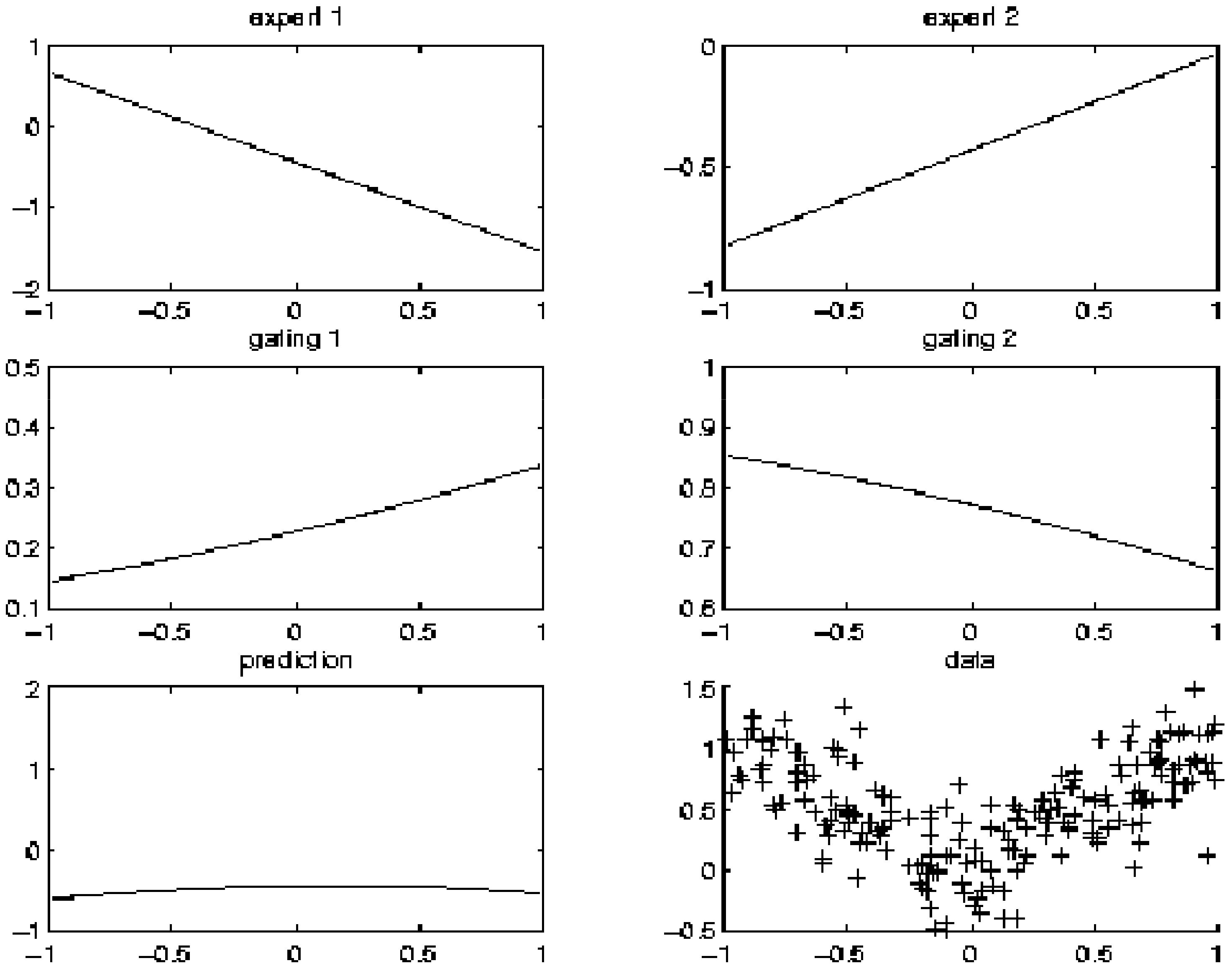
```
engine = jtree_inf_engine(bnet, observed);
```

•Any inference engine could be used for this trivial model

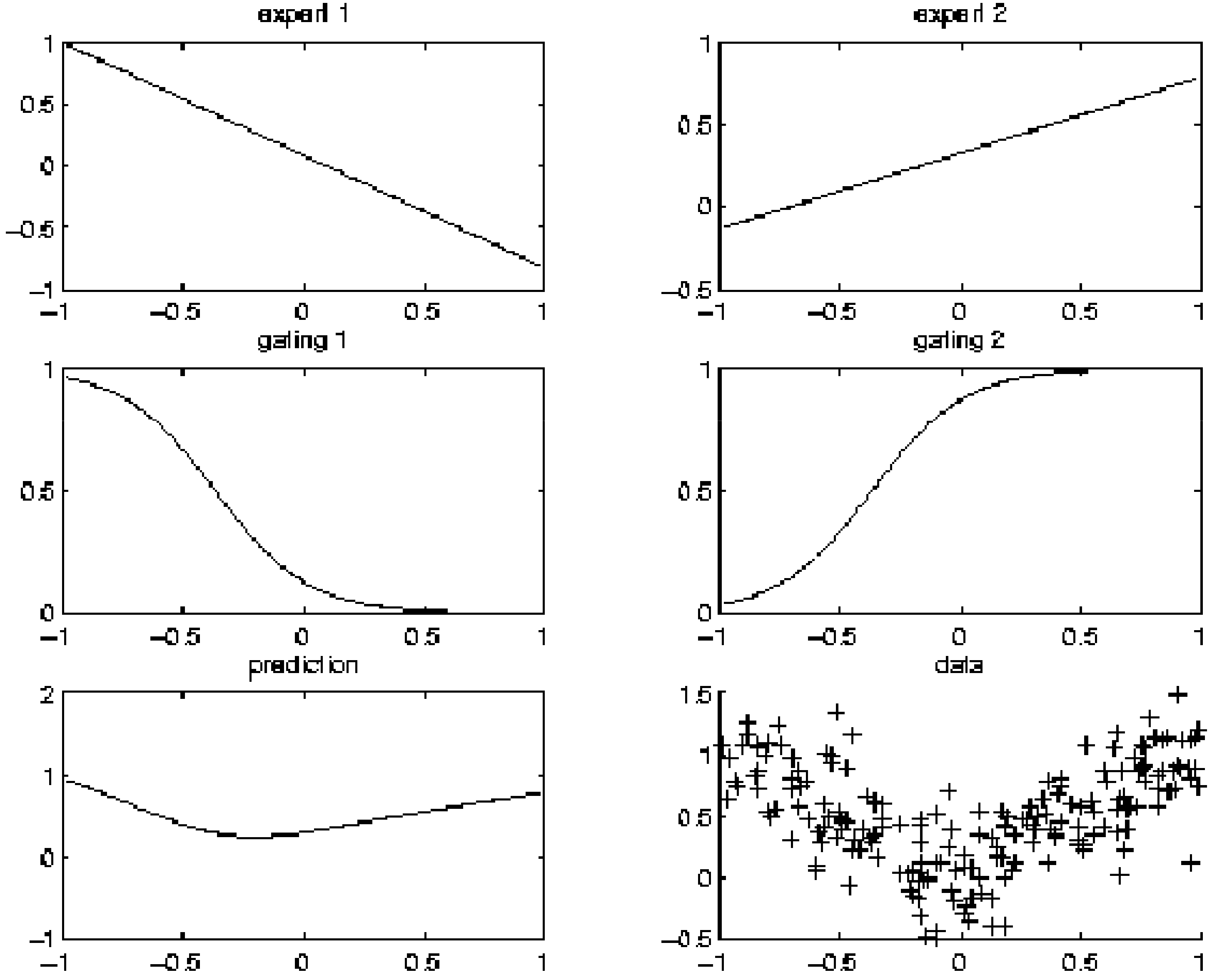
```
bnet2 = learn_params_em(engine, cases);
```

•We use EM since the Q nodes are hidden during training
•learn_params_em is a function, but should be an object

Before training

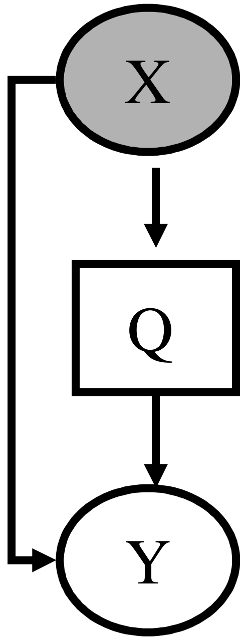


After training



5. Inference/ prediction

```
engine = jtree_inf_engine(bnet2);
evidence = cell(1,3);
evidence{X} = 0.68; % Q and Y are hidden
engine = enter_evidence(engine, evidence);
m = marginal_nodes(engine, Y);
m.mu % E[Y|X]
m.Sigma % Cov[Y|X]
```



Other kinds of models that BNT supports

- Classification/ regression: linear regression, logistic regression, cluster weighted regression, hierarchical mixtures of experts, naïve Bayes
- Dimensionality reduction: probabilistic PCA, factor analysis, probabilistic ICA
- Density estimation: mixtures of Gaussians
- State-space models: LDS, switching LDS, tree-structured AR models
- HMM variants: input-output HMM, factorial HMM, coupled HMM, DBNs
- Probabilistic expert systems: QMR, Alarm, etc.
- Limited-memory influence diagrams (LIMID)
- Undirected graphical models (MRFs)

Summary of BNT

- Provides many different kinds of models/ CPDs – lego brick philosophy
- Provides many inference algorithms, with different speed/ accuracy/ generality tradeoffs (to be chosen by user)
- Provides several learning algorithms (parameters and structure)
- Source code is easy to read and extend

What is wrong with BNT?

- It is slow
- It has little support for undirected models
- Models are not bona fide objects
- Learning engines are not objects
- It does not support online inference/learning
- It does not support Bayesian estimation
- It has no GUI
- It has no file parser
- It is more complex than necessary

Some alternatives to BNT?

- HUGIN: commercial
  - Junction tree inference only, no support for DBNs
- PNL: Probabilistic Networks Library (Intel)
  - Open-source C++, based on BNT, work in progress (due 12/03)
- GMTk: Graphical Models toolkit (Bilmes, Zweig/ UW)
  - Open source C++, designed for ASR (HTK), binary avail now
- AutoBayes: code generator (Fischer, Buntine/NASA Ames)
  - Prolog generates matlab/C, not avail. to public
- VIBES: variational inference (Winn / Bishop, U. Cambridge)
  - conjugate exponential models, work in progress
- BUGS: (Spiegelhalter et al., MRC UK)
  - Gibbs sampling for Bayesian DAGs, binary avail. since ’96

Why yet another GM toolbox?

- In 2003, there are still very few GM programs that satisfy the following desiderata:
  - Must support real-valued (vector) data
  - Must support learning (params and struct)
  - Must support time series
  - Must support exact and approximate inference
  - Must separate API from UI
  - Must support MRFs as well as BNs
  - Must be possible to add new models and algorithms
  - ~~Preferably free~~
  - ~~Preferably open-source~~
  - Must be easy to read/ modify
  - Must be fast (smarter algorithms, not better coding!)
  - Must be integrated with data analysis environment