

# Graphical models and their Role in Databases

VLDB 2007 Tutorial

Amol Deshpande<sup>1</sup>   Sunita Sarawagi<sup>2</sup>

<sup>1</sup>University of Maryland

<sup>2</sup>IIT Bombay

# Why a tutorial on graphical models at VLDB?

- VLDB
  - ▶ Many DB tasks use probabilistic modeling
    - ★ Core: Selectivity estimation, Imprecise databases,
    - ★ Application: Information extraction, Duplicate elimination, sensor networks.
    - ★ Data mining: Classification (naive Bayes, logistic), clustering (EM)
  - ▶ Probabilistic modeling is simultaneously
    - ★ intuitive (low barrier to entry)
    - ★ subtle (important to understand well for correctness & efficiency)
- Graphical models
  - ▶ Fundamental tools for intuitively and efficiently modeling probabilities
  - ▶ Distilled body of knowledge from many fields (let us build upon them, instead of reinventing)

VLDB wants to broaden, GM a fun and useful candidate for broadening

# Probabilistic modeling

- Given: several variables:  $x_1, \dots, x_n$ ,  $n$  is large.
- Task: build a joint distribution function  $\Pr(x_1, \dots, x_n)$
- Goal: Answer several kind of projection queries on the distribution
- Basic premise
  - ▶ Explicit joint distribution is dauntingly large
  - ▶ Queries are simple **aggregates** over the joint distribution.

## Example: Selectivity estimation in databases

- Variables are columns of a table

Age	Income	Experience	Degree	Location
10 ranges	7 scales	7 scales	3 scales	30 places

- An explicit joint** distribution over all columns not tractable:  
number of combinations:  $10 \times 7 \times 7 \times 3 \times 30 = 44100$ .
- Queries: Estimate number of people with
  - ▶ Income > 200K and Degree="Bachelors",
  - ▶ Income < 200K, Degree="PhD" and experience > 10 years.
  - ▶ Many, many more.

# Alternatives to an explicit joint distribution

- Assume all columns are independent of each other: **bad assumption**
- Use data to detect pairs of highly correlated column pairs and estimate their pairwise frequencies
  - ▶ Many highly correlated pairs  
income  $\not\perp$  age, income  $\not\perp$  experience, age  $\not\perp$  experience
  - ▶ **Ad hoc methods of combining these into a single estimate**
- Go beyond pairwise correlations: understand finer dependencies
  - ▶ income  $\not\perp$  age, but income  $\perp$  age | experience
  - ▶ experience  $\perp$  degree, but experience  $\not\perp$  degree | income

Graphical models make explicit an efficient joint distribution from these independencies

# Graphical models

Model joint distribution over **several** variables as a product of smaller factors that is

- ① *Intuitive* to represent and visualize
  - ▶ Graph: represent structure of dependencies
  - ▶ Potentials over subsets: quantify the dependencies
- ② *Efficient* to query
  - ▶ given values of any variable subset, reason about probability distribution of others.
  - ▶ many efficient exact and approximate inference algorithms

Graphical models = graph theory + probability theory.

# Graphical models in use

- Roots in statistical physics for modeling interacting atoms in gas and solids [ 1900]
- Early usage in genetics for modeling properties of species [ 1920]
- AI: expert systems ( 1970s-80s)
- Now many new applications:
  - ▶ Error Correcting Codes: Turbo codes, impressive success story (1990s)
  - ▶ Robotics and Vision: image denoising, robot navigation.
  - ▶ Text mining: information extraction, duplicate elimination, hypertext classification, help systems
  - ▶ Bio-informatics: Secondary structure prediction, Gene discovery
  - ▶ Data mining: probabilistic classification and clustering.

# Overall plan

- Fundamentals
  - ▶ Representation
  - ▶ Exact inference
- Applications
  - ▶ Selectivity estimation
  - ▶ Probabilistic databases
- Applications: Sensor data management
- Fundamentals
  - ① Learning a graphical model
  - ② Conditional Random Fields
- Applications
  - ▶ Information extraction
  - ▶ Duplicate elimination



# Part I: Fundamentals of Graphical Models

# Part I: Outline

## 1 Representation

- Directed graphical models: Bayesian networks
- Undirected graphical models

## 2 Inference Queries

- Exact inference on chains
- Exact inference on general graphs

## 3 Constructing a graphical model

- Graph Structure
- Parameters in Potentials

## 4 Approximate inference

- Generalized belief propagation
- Sampling: Gibbs, Particle filters

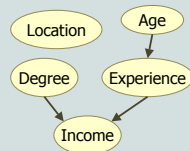
# Representation

Structure of a graphical model: Graph + Potential

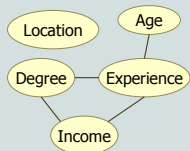
## Graph

- Nodes: variables  $\mathbf{x} = x_1, \dots, x_n$ 
  - ▶ Continuous: Sensor temperatures, income
  - ▶ Discrete: Degree (one of Bachelors, Masters, PhD), Levels of age
- Edges: direct interaction
  - ▶ Directed edges: Bayesian networks
  - ▶ Undirected edges: Markov Random fields

### Directed



### Undirected



# Representation

## Potentials: $\psi_c(\mathbf{x}_c)$

- Scores for assignment of values to subsets  $c$  of directly interacting variables.
- Which subsets? What do the potentials mean?
  - ▶ Different for directed and undirected graphs

## Probability

Factorizes as product of potentials

$$\Pr(\mathbf{x} = x_1, \dots, x_n) \propto \prod \psi_S(\mathbf{x}_S)$$

# Directed graphical models: Bayesian networks

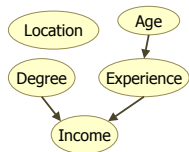
- Graph  $G$ : directed acyclic
  - ▶ Parents of a node:  $\text{Pa}(x_i) =$  set of nodes in  $G$  pointing to  $x_i$
- Potentials: defined at each node in terms of its parents.

$$\psi_i(x_i, \text{Pa}(x_i)) = \Pr(x_i | \text{Pa}(x_i))$$

- Probability distribution

$$\Pr(x_1 \dots x_n) = \prod_{i=1}^n \Pr(x_i | \text{pa}(x_i))$$

# Example of a directed graph



$$\psi_1(L) = \Pr(L)$$

NY	CA	London	Other
0.2	0.3	0.1	0.4

$$\psi_2(A) = \Pr(A)$$

20-30	30-45	> 45
0.3	0.4	0.3

or, a Gaussian distribution  
 $(\mu, \sigma) = (35, 10)$

$$\psi_2(E, A) = \Pr(E|A)$$

	0-10	10-15	> 15
20-30	0.9	0.1	0
30-45	0.4	0.5	0.1
> 45	0.1	0.1	0.8

$$\psi_2(I, E, D) = \Pr(I|D, A)$$

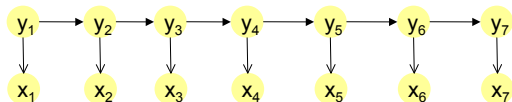
3 dimensional table, or a histogram approximation.

## Probability distribution

$$\text{Pa}(\mathbf{x} = L, D, I, A, E) = \Pr(L) \Pr(D) \Pr(A) \Pr(E|A) \Pr(I|D, E)$$

# Popular Bayesian networks

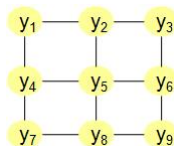
- Hidden Markov Models: **speech recognition, information extraction**



- ▶ State variables: discrete **phoneme, entity tag**
  - ▶ Observation variables: continuous (**speech waveform**), discrete (**Word**)
- Kalman Filters: State variables: continuous
    - ▶ Discussed later
  - PRMs: Probabilistic relational networks:
    - ▶ An important relevant class for relational data
    - ▶ Discussed later
  - QMR (Quick Medical Reference) system

# Undirected graphical models

- Graph  $G$ : arbitrary undirected graph
- Useful when variables interact symmetrically, no natural parent-child relationship
- Example: labeling pixels of an image.
- Potentials defined on arbitrary subcliques  $C$  of  $G$ . Popular choices:
  - ▶ Node potentials
  - ▶ Edge potentials

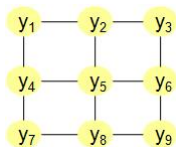


Probability distribution

$$\Pr(\mathbf{x} = y_1 \dots y_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{y}_C)$$

where  $Z = \sum_{\mathbf{y}'} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{y}'_C)$

# Example



$y_i = 1$  (part of foreground), 0 otherwise.

- Node potentials

- ▶  $\psi_1(0) = 4, \psi_1(1) = 1$

- ▶  $\psi_2(0) = 2, \psi_2(1) = 3$

- ▶ ....

- ▶  $\psi_9(0) = 1, \psi_9(1) = 1$

- Edge potentials: Same for all edges

- ▶  $\psi(0,0) = 5, \psi(1,1) = 5, \psi(1,0) = 1, \psi(0,1) = 1$

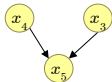
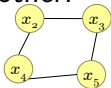
- Probability:  $\Pr(y_1 \dots y_9) \propto \prod_{k=1}^9 \psi_k(y_k) \prod_{(i,j) \in E(G)} \psi(y_i, y_j)$

# Popular undirected graphical models

- Interacting atoms in gas and solids [ 1900]
- Markov Random Fields in vision for image segmentation
- Conditional Random Fields for information extraction

# Comparing directed and undirected graphs

- Some distributions can only be expressed in one and not the other.



- Potentials
  - ▶ Directed: conditional probabilities, more intuitive
  - ▶ Undirected: arbitrary scores, easy to set.
- Dependence structure
  - ▶ Directed: Complicated d-separation test
  - ▶ Undirected: Graph separation:  $A \perp\!\!\!\perp B \mid C$  iff  $C$  separates  $A$  and  $B$  in  $G$ .
- Often application makes the choice clear.
  - ▶ Directed: Causality
  - ▶ Undirected: Symmetric interactions.

# Part I: Outline

## 1 Representation

- Directed graphical models: Bayesian networks
- Undirected graphical models

## 2 Inference Queries

- Exact inference on chains
- Exact inference on general graphs

## 3 Constructing a graphical model

- Graph Structure
- Parameters in Potentials

## 4 Approximate inference

- Generalized belief propagation
- Sampling: Gibbs, Particle filters

# Inference queries

① *Marginal probability queries over a small subset of variables:*

- ▶ Find  $\Pr(\text{Income}=\text{'High'} \ \& \ \text{Degree}=\text{'PhD'})$
- ▶ Find  $\Pr(\text{pixel } y_9 = 1)$

$$\Pr(x_1) = \sum_{x_2 \dots x_n} \Pr(x_1 \dots x_n)$$

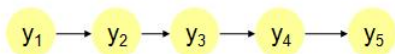
② *Most likely labels of remaining variables: (MAP queries)*

- ▶ Find most likely entity labels of all words in a sentence
- ▶ Find likely temperature at sensors in a room

$$\mathbf{x}^* = \operatorname{argmax}_{x_1 \dots x_n} \Pr(x_1 \dots x_n)$$

# Exact inference on chains

- Given,

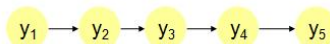


- ▶ Graph
- ▶ Potentials:  $\psi_i(y_i, y_{i+1})$
- ▶  $Pr(y_1, \dots, y_n) = \prod_i \psi_i(y_i, y_{i+1})$
- Find,  $Pr(y_i)$  for any  $i$ , say  $Pr(y_5 = 1)$ 
  - ▶ Exact method:  $Pr(y_5 = 1) = \sum_{y_1, \dots, y_4} Pr(y_1, \dots, y_4, 1)$  requires exponential number of summations.
  - ▶ A more efficient alternative...

# Exact inference on chains

$$\begin{aligned}\Pr(y_5 = 1) &= \sum_{y_1, \dots, y_4} \Pr(y_1, \dots, y_4, 1) \\ &= \sum_{y_1} \sum_{y_2} \sum_{y_3} \sum_{y_4} \psi_1(y_1, y_2) \psi_2(y_2, y_3) \psi_3(y_3, y_4) \psi_4(y_4, 1) \\ &= \sum_{y_1} \sum_{y_2} \psi_1(y_1, y_2) \sum_{y_3} \psi_2(y_2, y_3) \sum_{y_4} \psi_3(y_3, y_4) \psi_4(y_4, 1) \\ &= \sum_{y_1} \sum_{y_2} \psi_1(y_1, y_2) \sum_{y_3} \psi_2(y_2, y_3) B_3(y_3) \\ &= \sum_{y_1} \sum_{y_2} \psi_1(y_1, y_2) B_2(y_2) \\ &= \sum_{y_1} B_1(y_1)\end{aligned}$$

An alternative view: flow of beliefs  $B_i(\cdot)$  from node  $i + 1$  to node  $i$



# Adding evidence

Given fixed values of a subset of variables  $\mathbf{x}_e$  (evidence), find the

- 1 *Marginal probability queries over a small subset of variables:*

- ▶ Find  $\Pr(\text{Income}=\text{'High'} \mid \text{Degree}=\text{'PhD'})$

$$\Pr(x_1) = \sum_{x_2 \dots x_m} \Pr(x_1 \dots x_n \mid \mathbf{x}_e)$$

- 2 *Most likely labels of remaining variables: (MAP queries)*

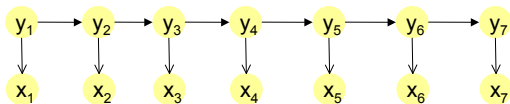
- ▶ Find likely temperature at sensors in a room given readings from a subset of them

$$\mathbf{x}^* = \operatorname{argmax}_{x_1 \dots x_m} \Pr(x_1 \dots x_n \mid \mathbf{x}_e)$$

Easy to add evidence, just change the potential.

# Inference in HMMs

- Given,



- ▶ Graph

- ▶ Potentials:  $\Pr(y_i|y_{i-1}), \Pr(x_i|y_i)$

- ▶ Evidence variables:  $\mathbf{x} = x_1 \dots x_n = o_1 \dots o_n$ .

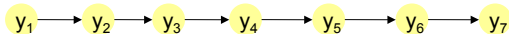
- Find most likely values of the hidden state variables.

$$\mathbf{y} = y_1 \dots y_n$$

$$\operatorname{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x} = \mathbf{o})$$

- Define  $\psi_i(y_{i-1}, y_i) = \Pr(y_i|y_{i-1}) \Pr(x_i = o_i|y_i)$

- Reduced graph only a single chain of  $y$  nodes.



- Algorithm same as earlier, just replace “Sum” with “Max”

This is the well-known Viterbi algorithm

# Exact inference on trees

- Basic steps for marginal and MAP queries.
  - ▶ Perform sum/max over leaf node potential and send resulting “belief” to parent.
  - ▶ Each internal node, on getting beliefs from its children
    - 1 Multiplies incoming beliefs with its own potentials
    - 2 Performs sum/max on the result
    - 3 Sends resulting “belief” factor to parent.
- Root has the answer.

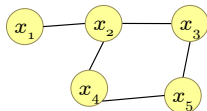
Linear in the number of nodes in the graph

# Junction tree algorithms

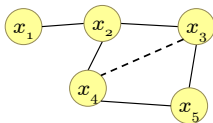
- An **optimal** general-purpose algorithm for **exact** marginal/MAP queries
- Simultaneous computation of many queries
- Efficient data structures
- Complexity:  $O(m^w N)$   $w$  = size of the largest clique in (triangulated) graph,  $m$  = number of values of each discrete variable in the clique. → **linear for trees**.
- Basis for many approximate algorithms.
- Many popular inference algorithms special cases of junction trees
  - ▶ Viterbi algorithm of HMMs
  - ▶ Forward-backward algorithm of Kalman filters

# Creating a junction tree from a graphical model

1. Starting graph



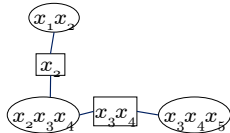
2. Triangulate graph



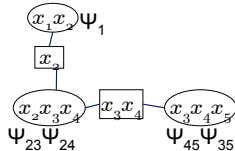
3. Create clique nodes



4. Create tree edges such that variables connected.



5) Assign potentials to exactly one subsumed clique node.



# Belief propagation on junction trees

- Each node  $c$ 
  - ▶ sends *belief*  $B_{c \rightarrow c'}(\cdot)$  to each of its neighbors  $c'$ 
    - ★ once it has beliefs from every other neighbor  $N(c) - \{c'\}$ .
  - ▶  $B_{c \rightarrow c'}(\cdot) =$  belief that clique  $c$  has about the distribution of labels to common variables  $s = c \cap c'$

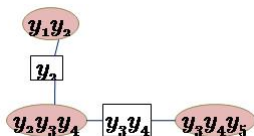
$$B_{c \rightarrow c'}(\mathbf{x}_s) = \sum_{\mathbf{x}_{c-s}} \psi_c(\mathbf{x}_c) \prod_{d \in N(c) - \{c'\}} B_{d \rightarrow c}(\mathbf{x}_{d \cap c})$$

Replace “sum” with “max” for MAP queries.

Compute marginal probability of any variable  $x_i$  as

- 1  $c =$  clique in JT containing  $x_i$
- 2  $\Pr(x_i) \propto \sum_{\mathbf{x}_{c-x_i}} \psi_c(\mathbf{x}_c) \prod_{d \in N(c)} B_{d \rightarrow c}(\mathbf{x}_{d \cap c})$

# Example



$$\psi_{234}(\mathbf{y}_{234}) = \psi_{23}(\mathbf{y}_{23})\psi_{34}(\mathbf{y}_{34})$$

$$\psi_{345}(\mathbf{y}_{345}) = \psi_{35}(\mathbf{y}_{35})\psi_{45}(\mathbf{y}_{45})$$

$$\psi_{234}(\mathbf{y}_{12}) = \psi_{12}(\mathbf{y}_{12})$$

1 Clique "12" sends belief  $B_{12 \rightarrow 234}(y_2) = \sum_{y_1} \psi_{12}(\mathbf{y}_{12})$  to its only neighbor.

2 Clique "345" sends belief  $B_{345 \rightarrow 234}(\mathbf{y}_{34}) = \sum_{y_5} \psi_{234}(\mathbf{y}_{345})$  to "234"

3 Clique "234" sends belief  $B_{234 \rightarrow 345}(\mathbf{y}_{34}) = \sum_{y_2} \psi_{234}(\mathbf{y}_{234})B_{12 \rightarrow 234}(y_2)$  to "345"

4 Clique "234" sends belief  $B_{234 \rightarrow 12}(y_2) = \sum_{y_4} \psi_{234}(\mathbf{y}_{234})B_{345 \rightarrow 234}(\mathbf{y}_{34})$  to "12"

$$\Pr(y_1) \propto \sum_{y_2} \psi_{12}(\mathbf{y}_{12})B_{234 \rightarrow 12}(y_2)$$

# Part I: Outline

## 1 Representation

- Directed graphical models: Bayesian networks
- Undirected graphical models

## 2 Inference Queries

- Exact inference on chains
- Exact inference on general graphs

## 3 Constructing a graphical model

- Graph Structure
- Parameters in Potentials

## 4 Approximate inference

- Generalized belief propagation
- Sampling: Gibbs, Particle filters

# Graph Structure

- 1 Manual: Designed by domain expert
  - ▶ Used in applications where dependency structure is well-understood
  - ▶ Example: QMR systems, Kalman filters, Vision (Grids), HMM for speech recognition and IE.
- 2 Learnt: from examples
  - ▶ NP hard to find the optimal structure.
  - ▶ Widely researched, mostly posed as a branch and bound search problem.
  - ▶ Useful in dynamic situations
  - ▶ Example: Selectivity estimation over attributes of arbitrary tables.

# Parameters in Potentials

- 1 Manual: Provided by domain expert
  - ▶ Used in infrequently constructed graphs, example QMR systems
  - ▶ Also where potentials are an easy function of the attributes of connected graphs, example: vision networks.
- 2 Learnt: from examples
  - ▶ More popular since difficult for humans to assign numeric values
  - ▶ Many variants of parameterizing potentials.
    - 1 Each potential entry a parameter, example, HMMs
    - 2 Potentials: combination of shared parameters and data attributes: example, CRFs. (Discussed in later with extraction)

# Part I: Outline

## 1 Representation

- Directed graphical models: Bayesian networks
- Undirected graphical models

## 2 Inference Queries

- Exact inference on chains
- Exact inference on general graphs

## 3 Constructing a graphical model

- Graph Structure
- Parameters in Potentials

## 4 Approximate inference

- Generalized belief propagation
- Sampling: Gibbs, Particle filters

# Why approximate inference

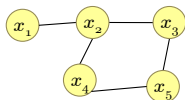
- Exact inference is NP hard. Complexity:  $O(w^m)$ 
  - ▶  $w$  = tree width = size of the largest clique in (triangulated) graph-1,
  - ▶  $m$  = number of values of each discrete variable in the clique.
- Many real-life graphs produce large cliques on triangulation
  - ▶ A  $n \times n$  grid has a tree width of  $n$
  - ▶ A Kalman filter on  $K$  parallel state variables influencing a common observation variable, has a tree width of size  $K + 1$

# Generalized belief propagation

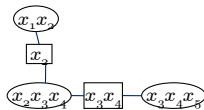
- Approximate junction tree with a cluster graph where
  - 1 Nodes = arbitrary clusters, not cliques in triangulated graph.  
Only ensure all potentials subsumed.
  - 2 Separator nodes on edges = *subset* of intersecting variables.

## Example cluster graph

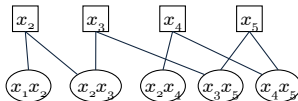
Starting graph



Junction tree.



Cluster graph



# Belief propagation in cluster graphs

- Graph can have loops, tree-based two-phase method not applicable.
- Many variants on scheduling order of propagating beliefs.
  - ▶ Simple loopy belief propagation [Pea88]
  - ▶ Tree-reweighted message passing [Kol04]
  - ▶ Residual belief propagation [EMK06]
- Most have no guarantees of convergence
- Works well in practice, default method of choice.
  - ▶ Success story: Error correction using Turbo code

# MCMC (Gibbs) sampling

- Useful when all else fails, guaranteed to converge to the optimal over infinite number of samples.
- Basic premise: easy to compute conditional probability  $\Pr(x_i | \text{fixed values of remaining variables})$

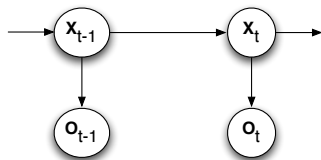
## Algorithm

- Start with some initial assignment, say  $\mathbf{x}^1 = [x_1, \dots, x_n] = [0, \dots, 0]$
- For several iterations
  - ▶ For each variable  $x_i$   
Get a new sample  $\mathbf{x}^{t+1}$  by replacing value of  $x_i$  with a new value sampled according to probability  $\Pr(x_i | x_1^t, \dots, x_{i-1}^t, x_{i+1}^t, \dots, x_n^t)$

# Others

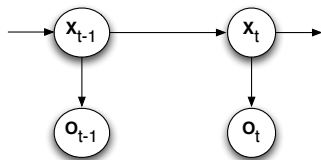
- Combinatorial algorithms for MAP [BVZ01, DTEK07, GDS07]
- Greedy algorithms: relaxation labeling
- Variational methods
- LP and QP based approaches

# Inference Task in DBNs



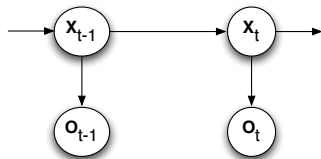
- Simplified representation of a dynamic Bayesian network
  - ▶ Hidden *state* variables:  $\mathbf{x}$ ; Observed variables:  $\mathbf{o}$
  - ▶ Assumed to be vector valued
- Given:
  - ▶ Prior on the initial state:  $p(\mathbf{x}_0)$
  - ▶ How state evolves:  $p(\mathbf{x}_t | \mathbf{x}_{t-1})$
  - ▶ How observations depend on state:  $p(\mathbf{o}_t | \mathbf{x}_t)$
- Estimate the *state at time  $t$*  given *observations till time  $t$* 
  - ▶ The posterior distribution:  $p(\mathbf{x}_t | \mathbf{o}_{1:t})$

# Alternative Inference Tasks in DBNs



- Estimate the most likely sequence of states (for discrete  $\mathbf{x}$ )
  - ▶  $\operatorname{argmax}_{\mathbf{x}_{1:t}} p(\mathbf{o}_{1:t} | \mathbf{x}_{1:t})$  (Cf. *Viterbi Algorithm*)
- Estimate the distribution of all states till time  $t$ 
  - ▶  $p(\mathbf{x}_{1:t} | \mathbf{o}_{1:t})$
- Estimate the state at time  $t$  given measurements till time  $t + l$  (*fixed-lag smoothing*)
  - ▶  $p(\mathbf{x}_t | \mathbf{o}_{1:t+l})$
  - ▶ Why ? Belief about the state at time  $t$  may change drastically given future observations

# Exact Inference in DBNs

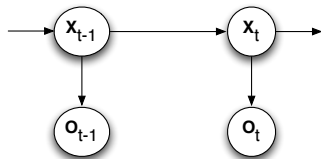


- Easy to write down
  - ▶ Using Bayes rule and Chain rule, we get:

$$p(\mathbf{x}_t | \mathbf{o}_{1:t}) = \frac{p(\mathbf{o}_t | \mathbf{x}_t) \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{o}_{1:t-1}) d\mathbf{x}_{t-1}}{p(\mathbf{o}_t | \mathbf{o}_{1:t-1})}$$

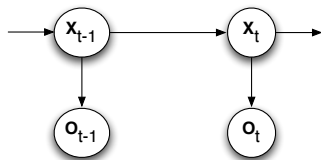
- ▶ Where:
  - ★  $p(\mathbf{o}_t | \mathbf{x}_t)$  and  $p(\mathbf{x}_t | \mathbf{x}_{t-1})$  are known model parameters
  - ★  $p(\mathbf{x}_{t-1} | \mathbf{o}_{1:t-1})$  is available from the previous time
  - ★  $p(\mathbf{o}_t | \mathbf{o}_{1:t-1}) = \int p(\mathbf{o}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{o}_{1:t-1}) d\mathbf{x}_t$  is a normalization constant (so may not need to be evaluated)

# Exact Inference in DBNs



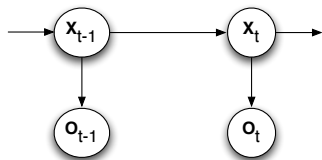
- However, can solve exactly in very few cases:
  - ▶ Kalman filters: if the system is linear Gaussian
    - ★ If  $p(\mathbf{x}_{t-1}|\mathbf{o}_{1:t-1})$  is Gaussian and the system is linear Gaussian,  $p(\mathbf{x}_t|\mathbf{o}_{1:t})$  is Gaussian
    - ★ Very efficient
    - ★ Backward smoothing also easily doable
  - ▶ Grid-based method: if the state space is discrete and finite
    - ★ Can compute the integral as a sum exactly

# Approximate Inference in DBNs



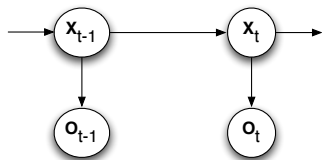
- Extended Kalman Filter
  - ▶ Approximate the process as a linear Gaussian system
  - ▶ Will fail if the posterior density not close to a Gaussian (e.g. if it is bimodal or heavily skewed)
- Approximate Grid-based methods
  - ▶ Discretize the continuous state space using a grid
  - ▶ Need sufficiently dense grid for good approximation
  - ▶ Suffers from “curse of dimensionality”

# Approximate Inference in DBNs: Particle Filters



- Approximate the state using a set of weighted samples, called *particles*
- At time  $t - 1$ , approximate  $p(\mathbf{x}_{t-1} | \mathbf{o}_{1:t-1})$  using  $n$  particles:
  - ▶  $\{\mathbf{x}_{t-1}^1, w_{t-1}^1\}, \{\mathbf{x}_{t-1}^2, w_{t-1}^2\}, \dots, \{\mathbf{x}_{t-1}^n, w_{t-1}^n\}$
- Can estimate any statistic using these particles
  - ▶ e.g.  $E(\mathbf{x}_{t-1} | \mathbf{o}_{1:t-1}) \approx \sum_{i=1}^n w_{t-1}^i \mathbf{x}_{t-1}^i$
- Inference Task: Generate a set of particles corresponding to  $p(\mathbf{x}_t | \mathbf{o}_{1:t})$  given  $\mathbf{o}_t$

# Approximate Inference in DBNs: Particle Filters

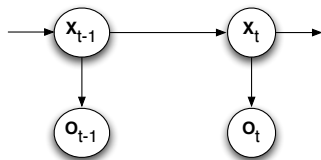


- Generate one sample each from:  $p(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{o}_t)$
- Assign weights as:

$$w_t^i \propto w_{t-1}^i p(\mathbf{o}_t | \mathbf{x}_{t-1}^i) = w_{t-1}^i \int p(\mathbf{o}_t | \mathbf{x}'_t) p(\mathbf{x}'_t | \mathbf{x}_{t-1}^i) d\mathbf{x}'_t$$

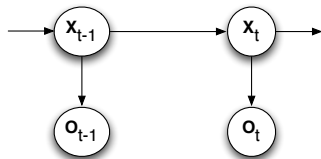
- Problems:
  - ▶ Requires sampling from  $p(\mathbf{x}_t | \dots)$  and computing  $p(\mathbf{o}_t | \dots)$
  - ▶ Requires evaluating complex integrals
- Can solve in very few cases:
  - ▶  $\mathbf{x}_t$  is discrete, or
  - ▶  $p(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{o}_t)$  is Gaussian (evolution can still be non-linear)

# Approximate Inference in DBNs: Particle Filters



- Must use *importance sampling*
  - ▶ Use an *importance density*  $q()$  to generate samples from
  - ▶ ...that closely approximates the true density  $p()$
  - ▶ No magic bullet for choosing  $q()$
- Degeneracy issues
  - ▶ After a while, a single particle has all the weight
  - ▶ Need to resample periodically

# Approximate Inference in DBNs: Particle Filters



- Many other extensions/variations have been considered
  - ▶ A lot more art than science at this point
- For an approachable introduction, see “A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking”; Arulampalam et al.; IEEE Trans. Signal Processing; 2002
  - ▶ Our discussion heavily borrows from it

## More on graphical models

- Koller and Friedman book (Structured Probabilistic Models) not published yet but you could request authors for a draft.
- Kevin Murphy's brief online introduction (<http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html>)
- Graphical models. M. I. Jordan. Statistical Science (Special Issue on Bayesian Statistics), 19, 140-155, 2004. (<http://www.cs.berkeley.edu/~jordan/papers/statsci.ps.gz>)
- Other text books:
  - ▶ R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter. "Probabilistic Networks and Expert Systems". Springer-Verlag. 1999.
  - ▶ J. Pearl. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference." Morgan Kaufmann. 1988.
  - ▶ Graphical models by Lauritzen, Oxford science publications F. V. Jensen. "Bayesian Networks and Decision Graphs". Springer. 2001.



Yuri Boykov, Olga Veksler, and Ramin Zabih.

Fast approximate energy minimization via graph cuts.

*IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.



J. Duchi, D. Tarlow, G. Elidan, and D. Koller.

Using combinatorial optimization within max-product belief propagation.

In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.



G. Elidan, I. McGraw, and D. Koller.

Residual belief propagation: Informed scheduling for asynchronous message passing.

In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, Boston, Massachusetts, July 2006.



Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi.

Efficient inference with cardinality-based clique potentials.

In *Proceedings of the 24<sup>th</sup> International Conference on Machine Learning (ICML), USA*, 2007.



Vladimir Kolmogorov.

Convergent tree-reweighted message passing for energy minimization.

Technical Report MSR-TR-2004-90, Microsoft Research (MSR), September 2004.



Judea Pearl.

*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*.

Morgan Kaufmann, 1988.



## Part II: Applications













































































































































## Part III: Graphical models for Information extraction and data integration



























































