Advanced Machine Learning Techniques for Temporal, Multimedia, and Relational Data

Vibhav Gogate

The University of Texas at Dallas

Many slides courtesy of Kevin Murphy

Multimedia Data

- Text
  - Ascii documents
  - HTML documents
  - Databases (Structured documents)
  - Annotations
- Images
  - JPG, PNG, BMP, TIFF, etc.
- Audio
  - MP3, WAV files
- Video
  - Sequence of frames

Size and Complexity of processing the data increases as we go from top to bottom

Temporal Data

- Time Series data generated by a dynamic system
  - A user's GPS locations recorded by his Cell-phone
  - Loop Sensors counting cars on a freeway
  - Load monitoring devices capturing power consumed in a household
  - Video as a sequence of frames

Relational Data

- Data resides in multiple tables

Name	Job	Company	Party
adams	researcher	scuf	no
blake	president	jvt	yes
king	manager	pharmadm	no
miller	manager	jvt	yes
scott	researcher	scuf	yes
turner	researcher	pharmadm	no

Name	Course
adams	erm
adams	so2
adams	srw
blake	cso
blake	erm
king	cso
king	erm

Company	Type
jvt	commercial
scuf	university
pharmadm	university

Course	Length	Type
cso	2	introductory
erm	3	introductory
so2	4	introductory
srw	3	advanced

Example Borrowed from Luc De Raedt's textbook, "Logical and Relational Learning"

Machine Learning

- Study of systems that improve their performance over time with experience
- Experience= Data (or examples or observations or evidence)
- Learning = Search for patterns, regularities or rules that provide insights into the data

What we will cover?

- Probabilistic Machine Learning
  - Build a model that describes the distribution that generated the data
  - Representation, Inference and Learning
- Dynamic Probabilistic Networks
  - Temporal Data
- Markov Logic Networks
  - Relational Data

Probabilistic Graphical Models

- "PGMs have revolutionized AI and machine learning over the last two decades" – Eric Horvitz, Director, Microsoft Research
- **Basic Idea**: Compactly represent a joint probability distribution over a large number of variables by taking advantage of conditional independence.
  - Graph describes the conditional independence assumptions

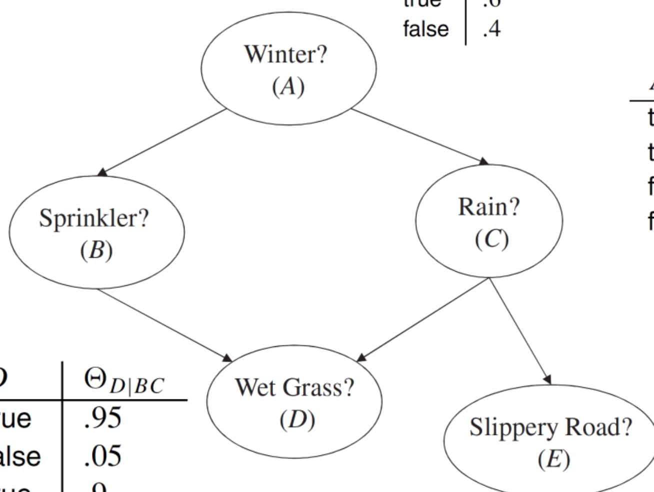
Bayesian networks

- Directed or Causal Networks

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} P(X_i | pa(X_i))$$

A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25



A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

Product of several poly-sized conditional probability tables

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Each table is variable given its parents in the graph

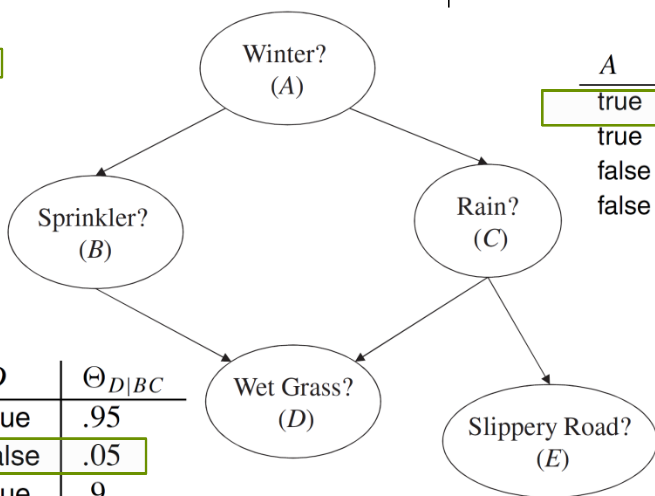
Bayesian networks

31 vs 17 entries

Exponential vs Poly entries

A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25



A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Joint distribution

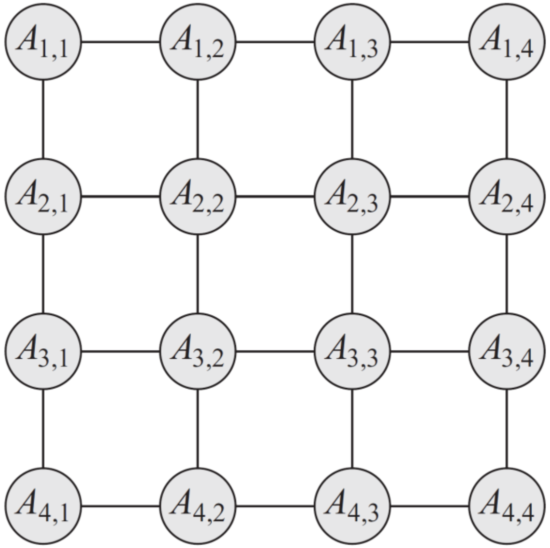
A	B	C	D	E	Pr(.)
true	true	true	true	true	.06384
true	true	true	true	false	.02736
true	true	true	false	true	.00336
true	true	true	false	false	.00144
true	true	false	true	true	0
true	true	false	true	false	.02160
true	true	false	false	true	0
true	true	false	false	false	.00240
true	false	true	true	true	.21504
true	false	true	true	false	.09216
true	false	true	false	true	.05376

....

....

$$0.2 \times 0.05 \times .6 \times .8 \times .3 = .00144$$

Markov networks



$$P(\mathbf{X} = (X_1, \ldots, X_n)) = \frac{1}{Z} \prod_{i=1}^{m} \phi_i(\mathbf{X}_{Vars(\phi(i))})$$

$A_{1,1}$	$A_{1,2}$	Weight
0	0	3
0	1	4.3
1	0	2.2
1	1	33.1

- Functions defined over cliques
  - Don’t have a probabilistic meaning
- Distribution = normalized product of functions

Log-Linear models

- PGM = A set of weighted formulas (features) in propositional logic
- Alternative Representation of a PGM
- Distribution

$$P(\mathbf{X}) = \frac{1}{Z} \exp\left(\sum_i \delta(f_i, \mathbf{X}) w_i\right)$$

where $\delta(f_i, \mathbf{X})$ is a dirac-delta function which is 1 if $\mathbf{X}$ satisfies f_i and 0 otherwise.

Inference Problems

- Probability of Evidence (PR)
  - Find the probability of an assignment to a subset of variables
- Conditional Marginal Estimation (MAR)
  - Find the marginal probability distribution at a variable given evidence
- Maximum a Posteriori (MAP)
  - Find an assignment with the maximum probability given evidence
- All of them are at least NP-hard

Learning problems

- Structure Learning
    - Learn the structure of the graph from data
- Weight Learning
    - Learn the parameters (CPTs, weights of features)
- Structure Learning is often much harder than weight learning
- In practice, we often assume a structure

Inference algorithms

- Exact algorithms
  - Exponential in treewidth (a graph parameter)
- Message-passing algorithms
  - Belief propagation, Expectation propagation, etc.
- Sampling algorithms
  - Importance sampling
  - Markov chain Monte Carlo sampling
    - Gibbs sampling

Dynamic Bayesian networks

- PGMs are static; don't have a concept of time
- Dynamic Bayesian networks are temporal PGMs
- Three assumptions
  - Stationary
  - Time is discrete
  - K-Markov assumption

Dynamic Bayesian networks

- $\mathbf{X}_t$ = Set of variables at time t
- $\mathbf{X}_{a:b}$ = Set of variables from time $t = a$ to $t = b$.

Markov Assumption

- $P(\mathbf{X}_t | \mathbf{X}_{0:t-1}) = P(\mathbf{X}_t | \mathbf{X}_{t-1})$

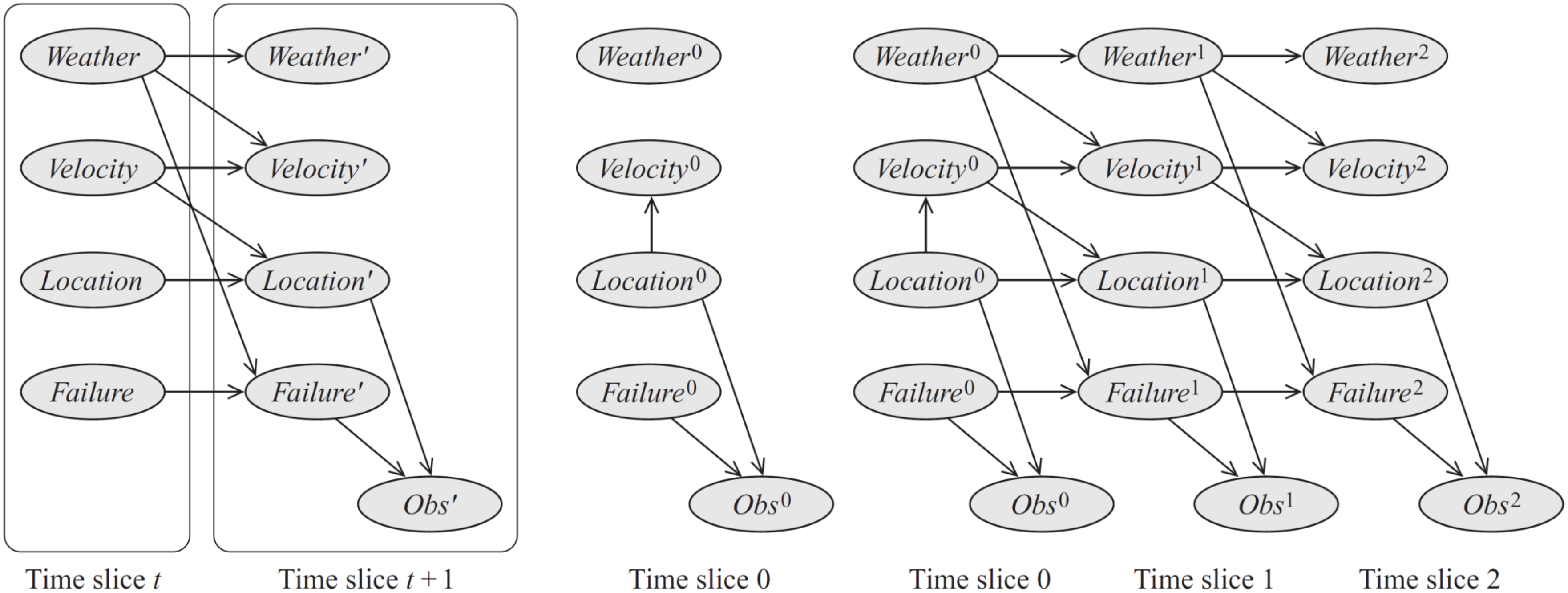
Stationary Process

- $P(\mathbf{X}_t | \mathbf{X}_{t-1})$ is the same for all t

Specification

- $\mathcal{B}_0 \equiv P(\mathbf{X}_0)$ and $\mathcal{B}_{\rightarrow} \equiv P(\mathbf{X}_t | \mathbf{X}_{t-1})$

Example

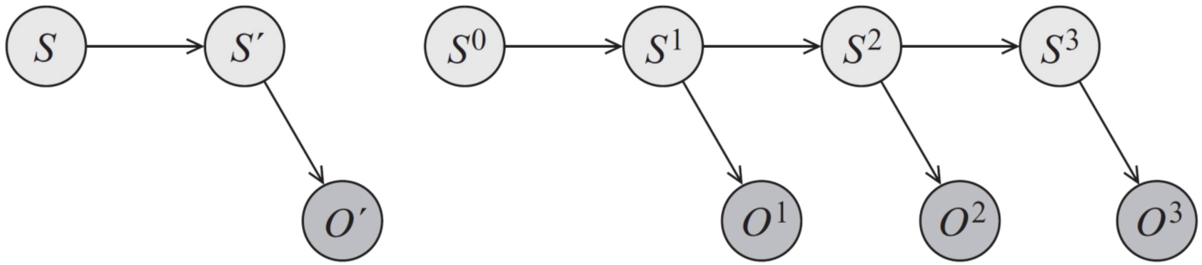


(a) $\mathcal{B}_{\rightarrow}$ (b) $\mathcal{B}_0$ (c) DBN unrolled over 3 steps

- A Dynamic Bayesian network for monitoring a person's car

A DBN as a HMM

- Hidden Markov models (HMMs)
  - Each time slice has one cluster each for observed and unobserved variables



(a) (b)

DBNs vs HMMs

- An HMM represents the state of the world using a single discrete random variable, $X_t \in \{1, \ldots, K\}$.

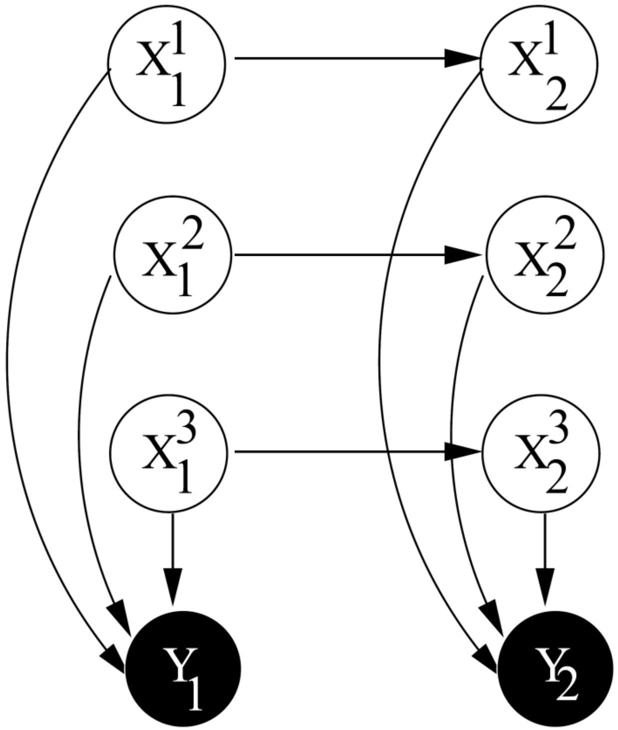
- A DBN represents the state of the world using a set of random variables, $X_t^{(1)}, \ldots, X_t^{(D)}$ (factored/ distributed representation).

- A DBN represents $P(X_t|X_{t-1})$ in a compact way using a parameterized graph.

⇒ A DBN may have exponentially fewer parameters than its corresponding HMM.

⇒ Inference in a DBN may be exponentially faster than in the corresponding HMM.

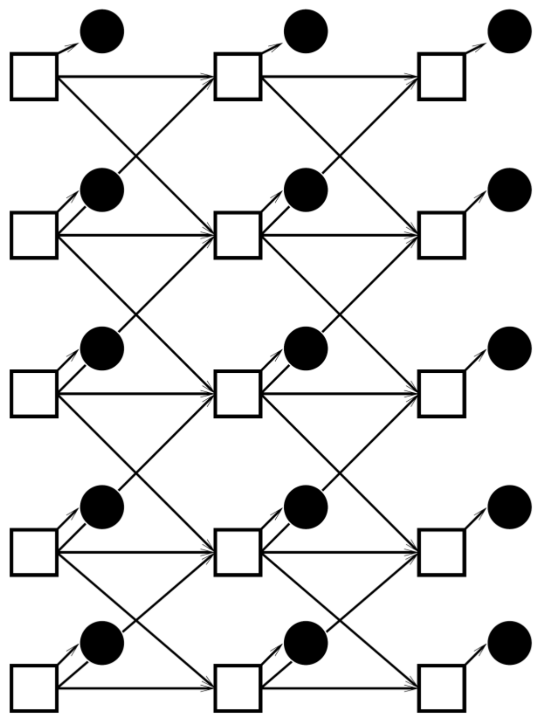
Factorial HMMs as DBNs



- Num. parameters to specify $P(X_t|X_{t-1})$:
  - HMM: $O(K^{2D})$.
  - DBN: $O(DK^2)$.
- Computational complexity of exact inference:
  - HMM: $O(TK^{2D})$.
  - DBN: $O(TDK^{D+1})$.

- Example: Several sources of sound from a single microphone

Coupled HMMs as DBNs



- Example: Temperature in different rooms
  - Adjacent rooms are connected to each other

Inference problems in DBNs

- Tracking or Filtering
  - Find the probability distribution over all variables or the marginal distribution at a variable at time slice "t" given evidence up to slice "t"
- Prediction
  - Find the probability distribution at time slice "k" given evidence up to slice "t" where $k>t$.
- Smoothing
  - Find the probability distribution at time slice "k" given evidence up to slice "t" where $k<t$.
- MAP inference
  - Find the most likely trajectory of the system.

Inference problems in DBNs

- Tracking: $P(\mathbf{X}_t|\mathbf{e}_{1:t})$
- Prediction: $P(\mathbf{X}_k|\mathbf{e}_{1:t})$ where $k > t$
- Interval Smoothing: $P(\mathbf{X}_k|\mathbf{e}_{1:t})$, $k \in [0, T]$; $T \leq t$.
- Fixed-lag Smoothing $P(\mathbf{X}_k|\mathbf{e}_{1:t})$ for a specific $k < t$
- MAP: $\arg\max_{\mathbf{x}_{0:t}} P(\mathbf{X}_{0:t}|\mathbf{e}_{1:t})$

Application Designer

- Select the variables at each time-slice
- Select the edges by following sound probabilistic principles
  - The networks should be a directed acyclic graph (DAG)
  - Each variable should be independent of its non-descendants given its parents
  - Sparesity: Limit the number of parents at each variable

Application Designer

- Remember as you increase the number of variables:
  - The model looks realistic
  - However, the complexity of inference and learning increases and the accuracy goes down because we have to use approximate inference methods
  - Tradeoff between the two is **rarely explored in practice**
- **Don’t think of the technology as a black-box**
  - **We are not there yet!**

Tracking/Filtering Algorithms

- Exact Inference
- Approximate Message passing algorithms
- Sampling Algorithms
  - Particle Filtering
  - Rao-Blackwellised Particle Filtering

Exact Inference on HMMs

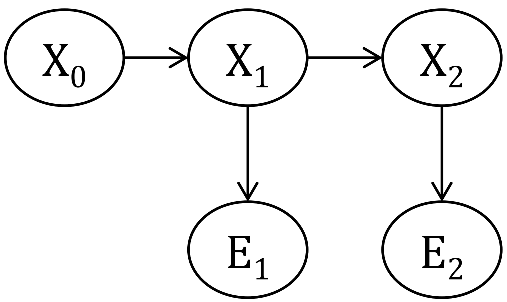
Calculate the belief state $\sigma(\mathbf{X}_{t+1}) = P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1})$ recursively:

$$\begin{aligned}
\sigma_{t+1}(\mathbf{X}_{t+1}) &= P(\mathbf{X}_{t+1}|\mathbf{e}_{t+1}, \mathbf{e}_{1:t}) \\
&= \frac{P(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})}{P(\mathbf{e}_{t+1}|\mathbf{e}_{1:t})} \\
&\propto P(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})
\end{aligned}$$

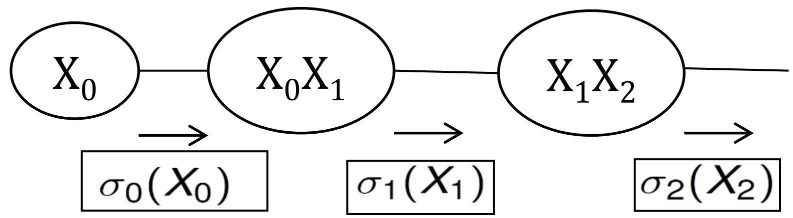
where $P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ can be computed using $\sigma_t(\mathbf{X}_t)$:

$$\begin{aligned}
P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) &= \sum_{\mathbf{X}_t} P(\mathbf{X}_{t+1}|\mathbf{X}_t, \mathbf{e}_{1:t})P(\mathbf{X}_t|\mathbf{e}_{1:t}) \\
&= \sum_{\mathbf{X}_t} P(\mathbf{X}_{t+1}|\mathbf{X}_t)\sigma_t(\mathbf{X}_t)
\end{aligned}$$

Exact Inference in HMMs



Unrolled HMM

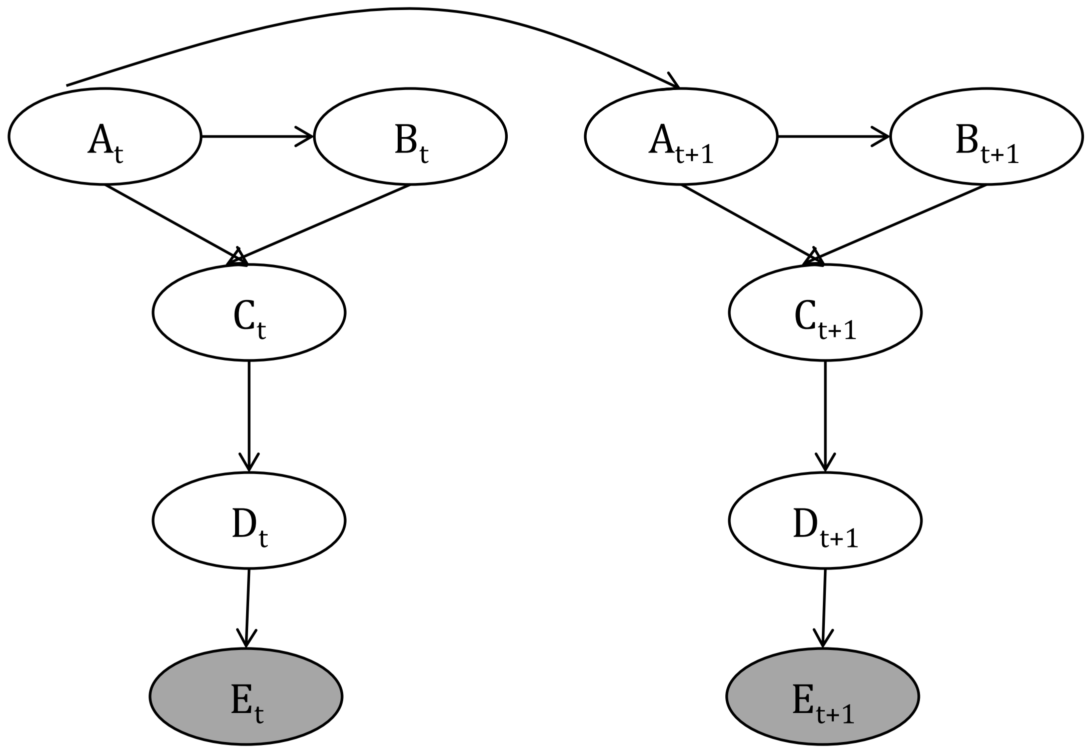


Message-passing algorithm

- Procedural view
  - Cluster (X_0) stores $P(X_0)$
  - Each cluster (X_t,X_{t+1}) stores $P(X_{t+1}|X_t)$ and $P(E_{t+1}|X_{t+1})$
- Multiply incoming message with the functions
- Sum-out X_t and send the resulting function to the next cluster

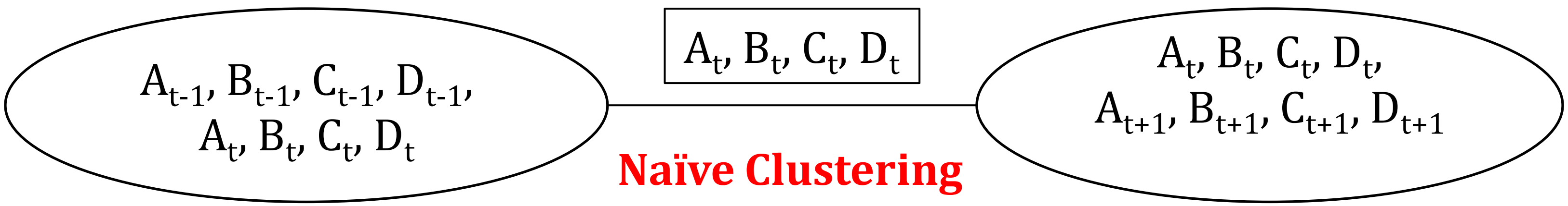
Exact Inference in DBNs

- **Clusters are Factored!**

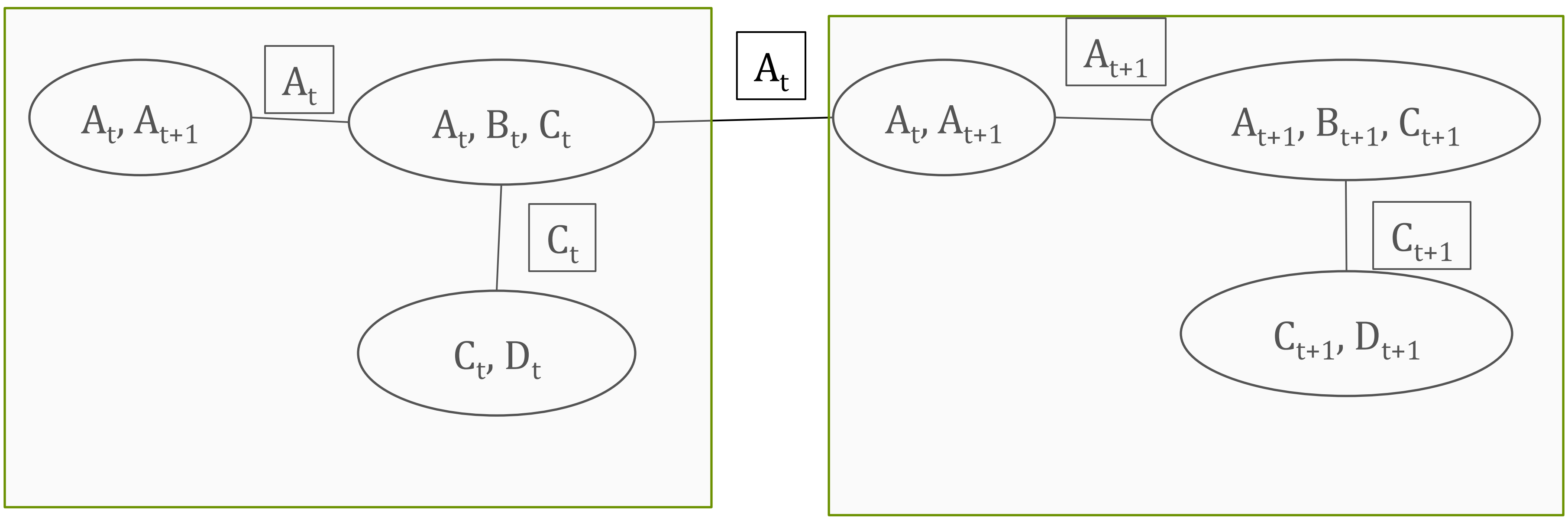


Exact Inference in DBNs

- **Clusters are Factored!**



Naïve Clustering

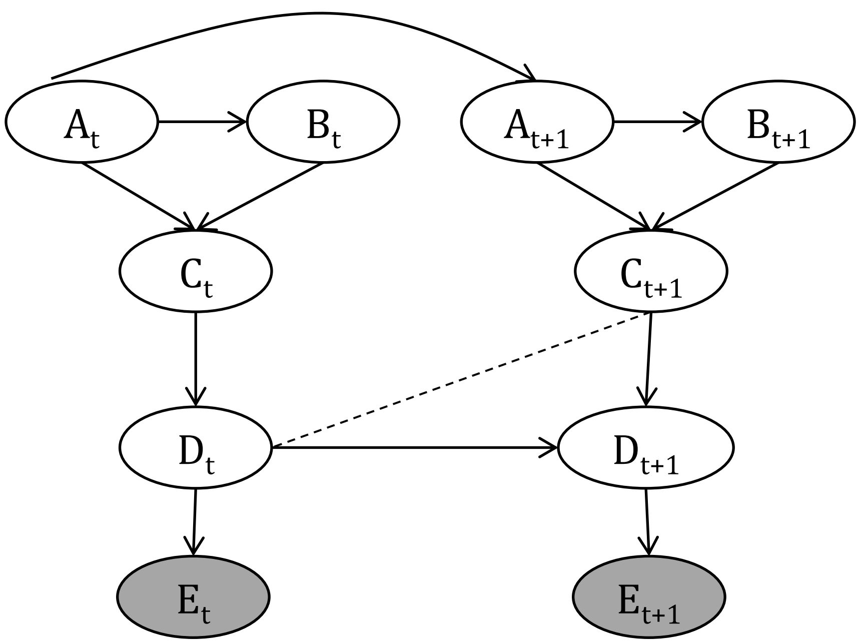


Advanced Clustering

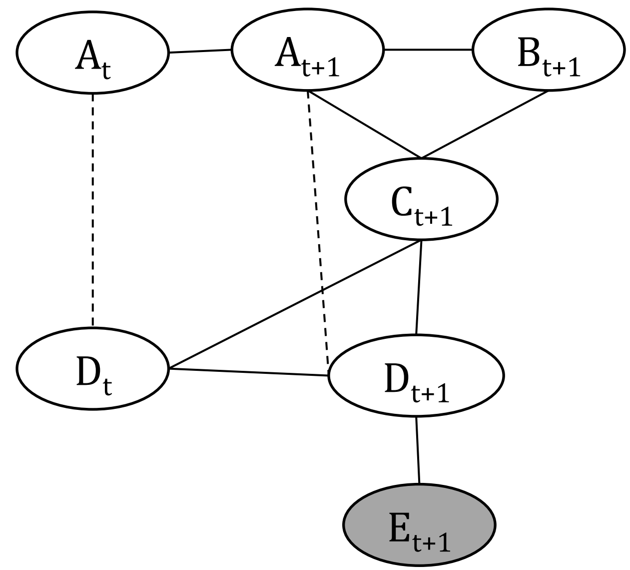
Smaller Clusters

- Complexity: exponential in the number of variables in the cluster
  - Smaller clusters are desirable
- How to construct the clusters?
  - At each time slice, find which nodes are connected to the next time slice and create a clique over them
    - **Interface nodes**
  - At each time slice, create a **junction tree** out of
    - **Moralized graph** over nodes in the time slice
    - Interface cliques over the time slice and previous one
  - Paste the junction trees together.

Building the Clusters



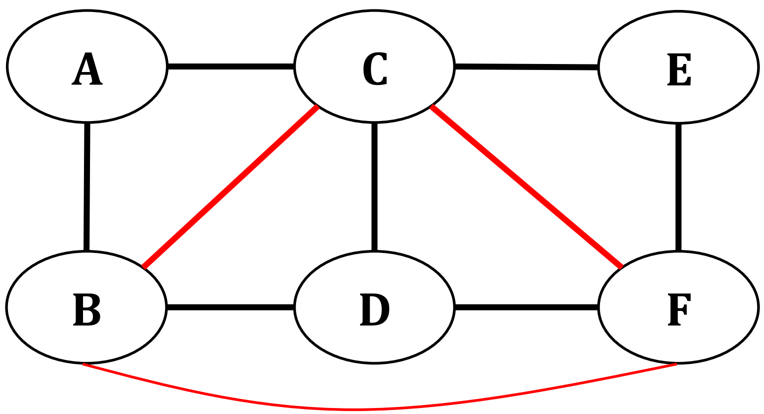
Moral graph:
Connect parents of a node to each other



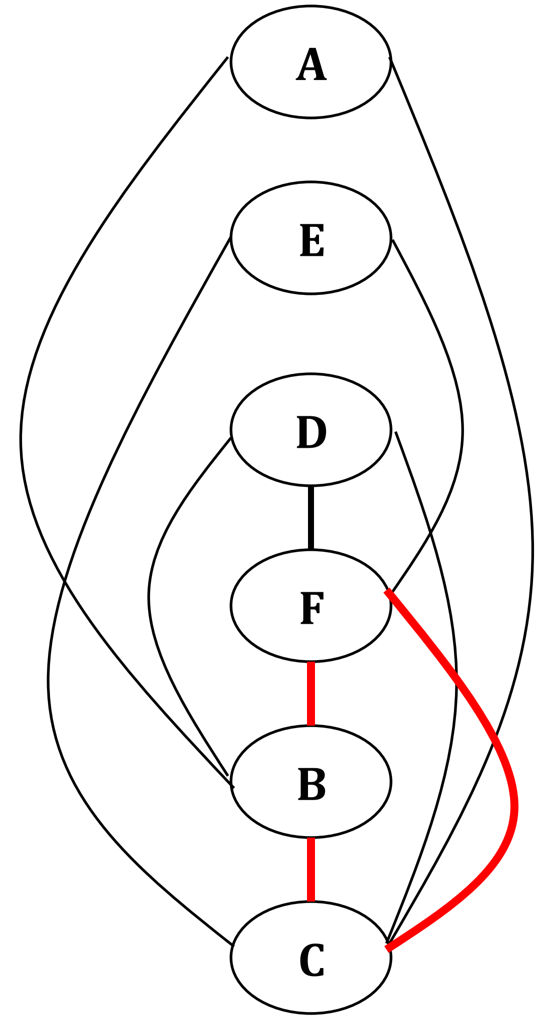
Graph over which a junction tree will be constructed

Building the Clusters

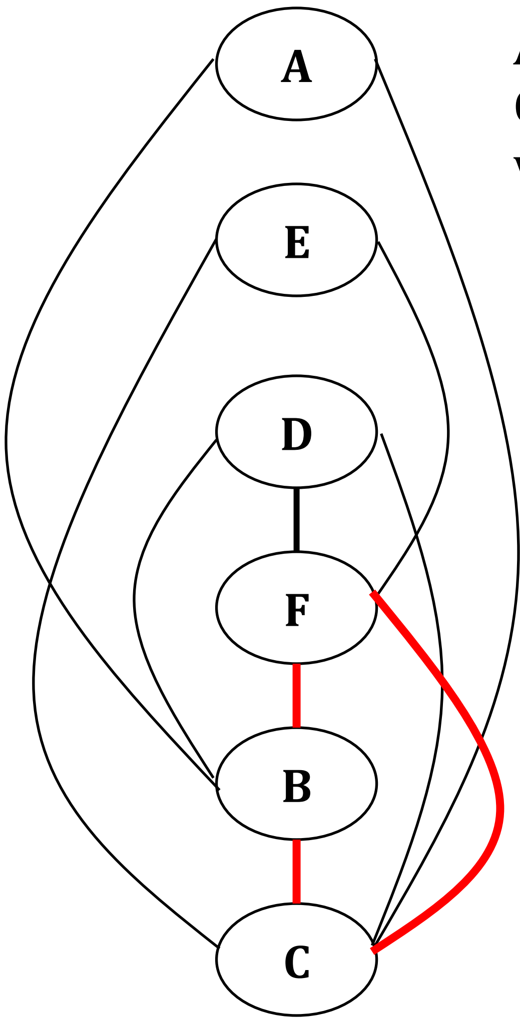
- Make Graph Chordal



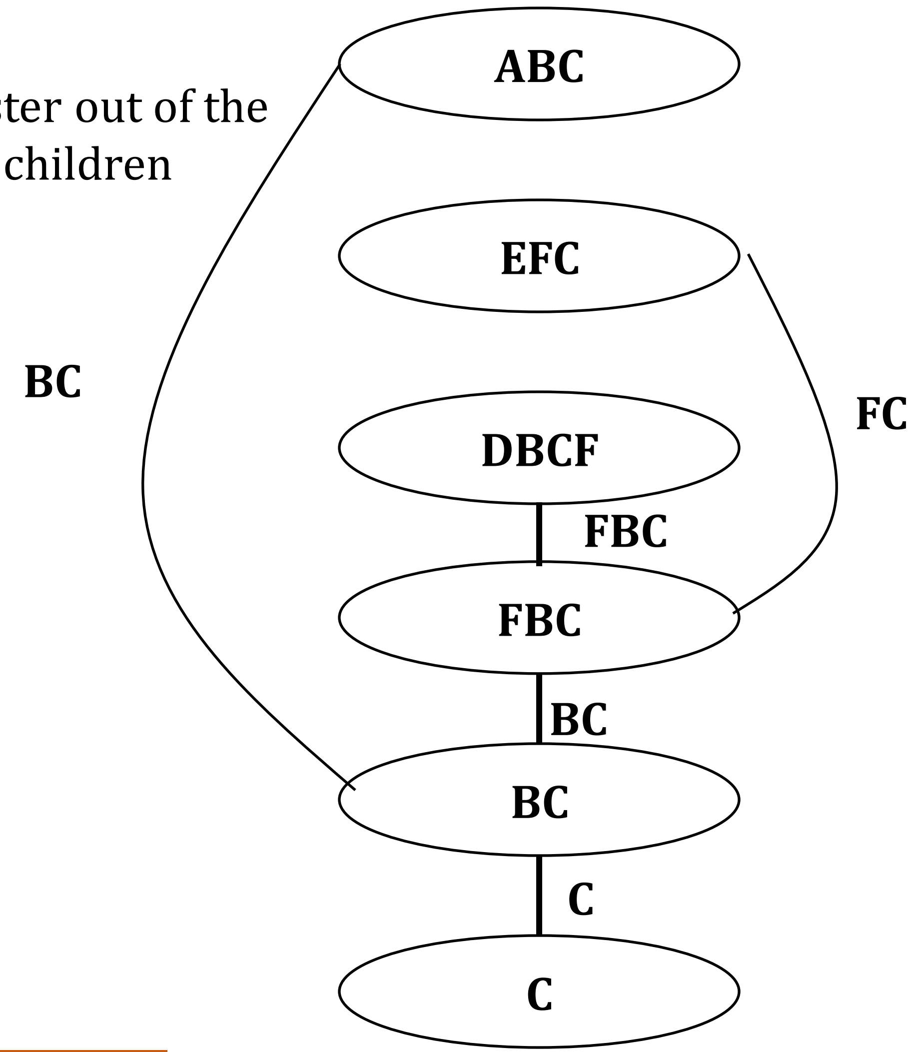
- Process nodes in order
  - Create a clique out of all nodes ordered below the node (its children) and having an edge with the node



Chordal Graph → Junction Tree



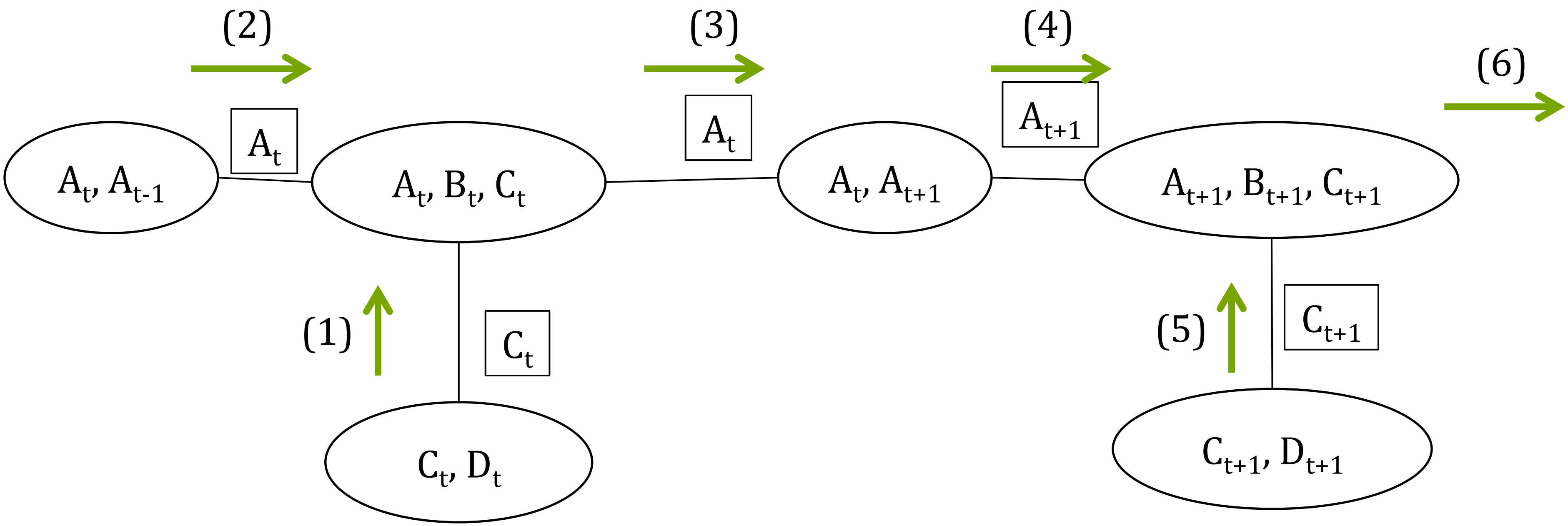
At each node:
Construct a cluster out of the variable and its children



Message-passing

- At each slice
  - Put each function in a cluster that contains all variables mentioned in the function
  - Order messages such that the outgoing interface message is the last one computed
  - Perform message passing
    - Multiply all incoming messages with the functions in the cluster
    - Sum-out all variables that are mentioned in the cluster but not mentioned in the receiving cluster.

Message-passing



Advanced Clustering

A possible message ordering (other orderings are also possible)

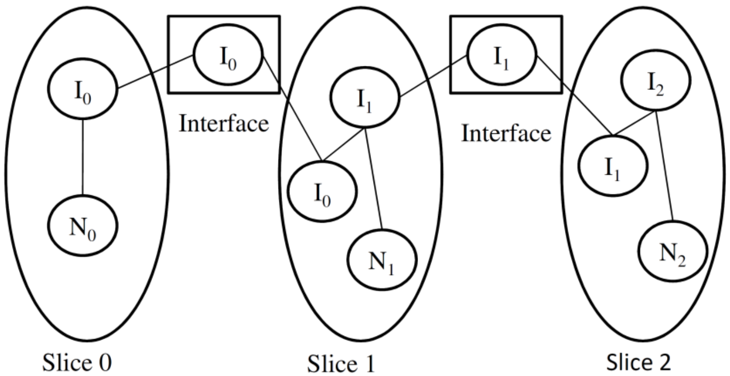
Why it works?

- Laws of probability theory
- It turns out that
  - $P(\mathbf{X}_t|\mathbf{X}_{1:t-1},\mathbf{e}_{1:t-1},\mathbf{e}_t)=P(\mathbf{X}_t|\mathbf{I}_{t-1},\mathbf{e}_t)$
  - Namely, $\mathbf{X}_t$ is conditionally independent of the past given the interface nodes $\mathbf{I}_{t-1}$ at time slice t-1.
- The message passing algorithm is an instance of the variable elimination algorithm
  - Eliminate all variables except the ones required by the next time slice

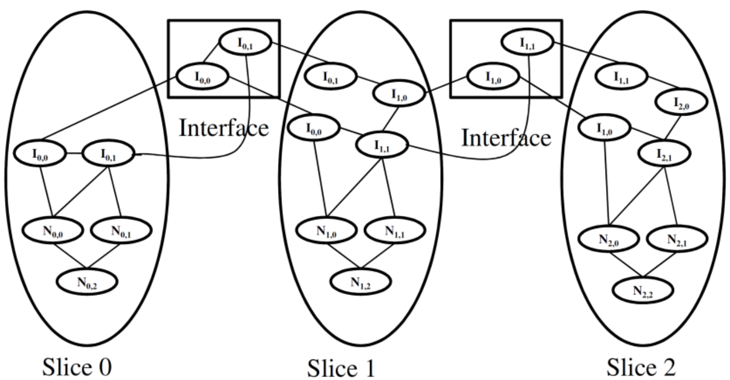
Approximate Inference

- Sometimes the cluster size is just too large to allow exact inference
- Resort to approximate inference
  - Message Passing
  - Sampling-based

Approximate Message Passing



(a) Join tree construction



(b) Join graph construction

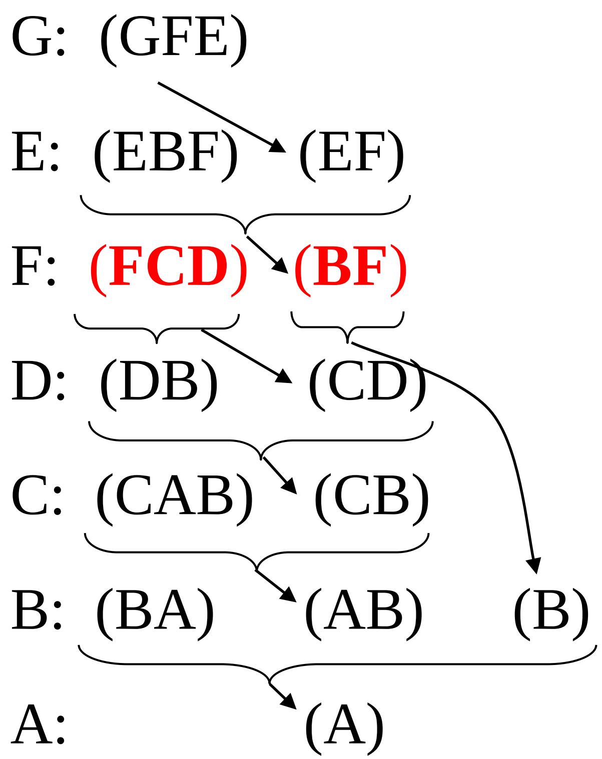
Junction Tree

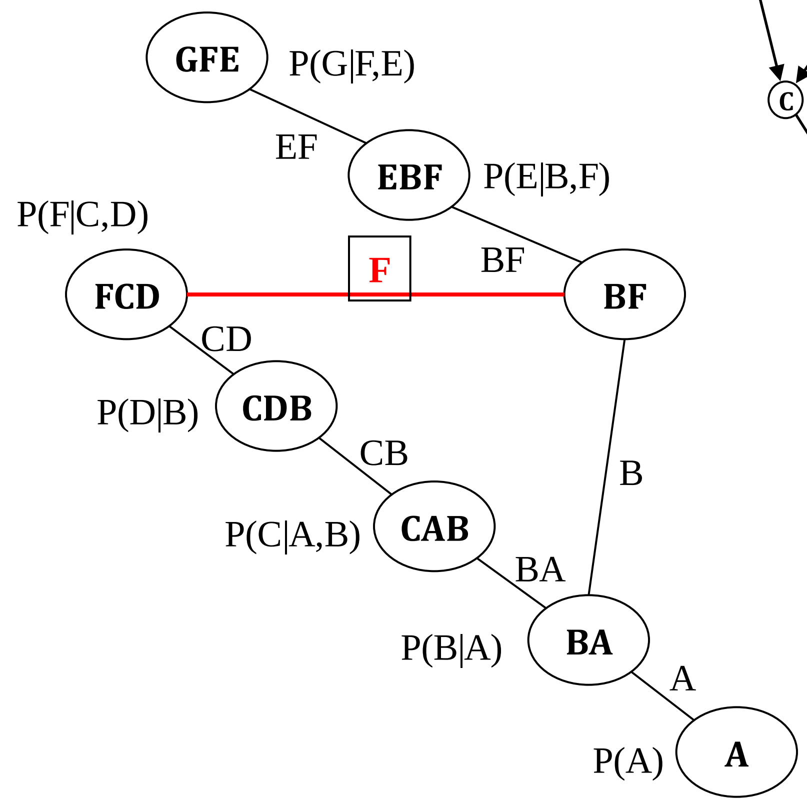
Gogate et.al, 2005, 2009
Kevin Murphy, 2002

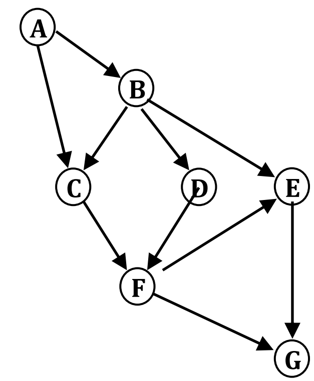
Split the clusters by relaxing the tree requirement

Perform loopy propagation in each slice

Iterative Join Graph Propagation



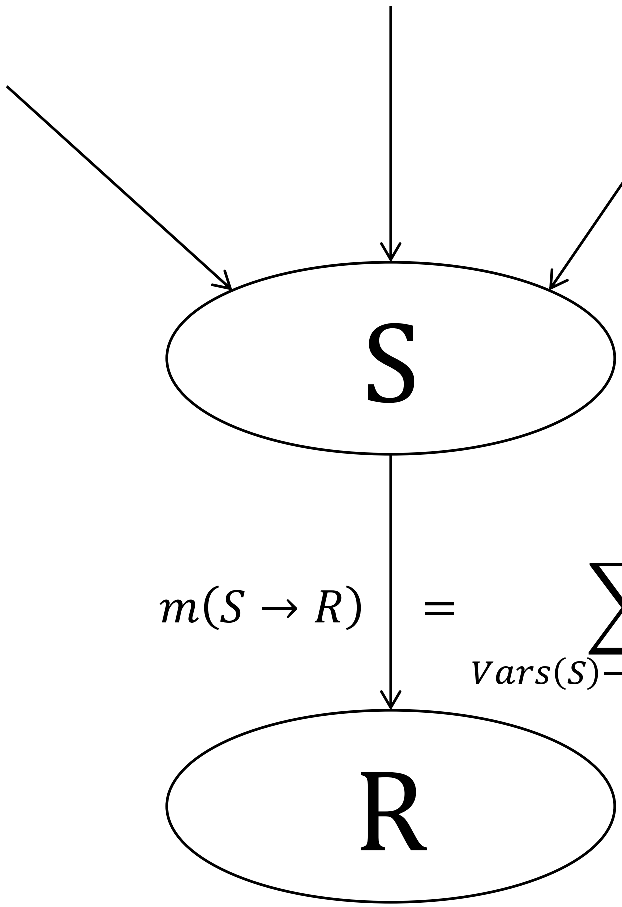




a) schematic mini-bucket(i), i=3

b) arc-labeled join-graph decomposition

Message passing Equations



- Multiply all received messages except from R
- Multiply all functions
- Sum-out all variables except the separator

$$m(S \rightarrow R) = \sum_{Vars(S)-Sep(S,R)} \prod_{f \in functions(S)} f \prod_{G \in Neighbors(S)-R} m(\mathrm{G} \rightarrow \mathrm{R})$$

Particle Filtering

- At each time slice
  - Generate N particles from a distribution Q
    - It is difficult to sample from P
  - Compute the weight of each particle
    - Correct for the fact that you are sampling from Q and not the target distribution P
  - Represent the belief state using the weighted particles

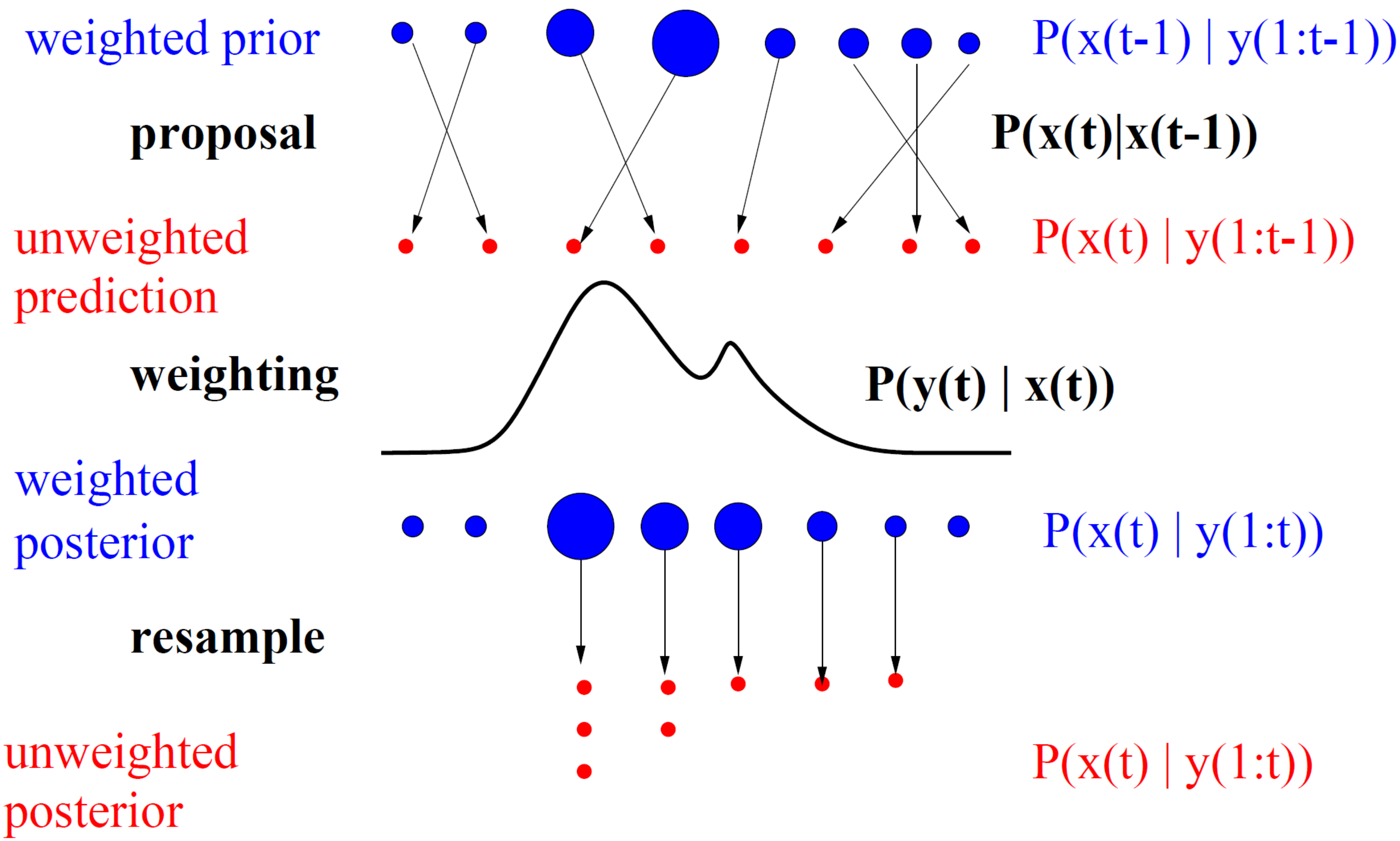
Particle Filtering

- **Given:** Samples: $(\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)})_{i=1}^{N})$ and Evidence $\mathbf{e}_t$
- For i = 1 to N do
  - Sample $\mathbf{x}_t^{(i)}$ from Q_t
  - Compute $w_t^{(i)}$ using the following equation

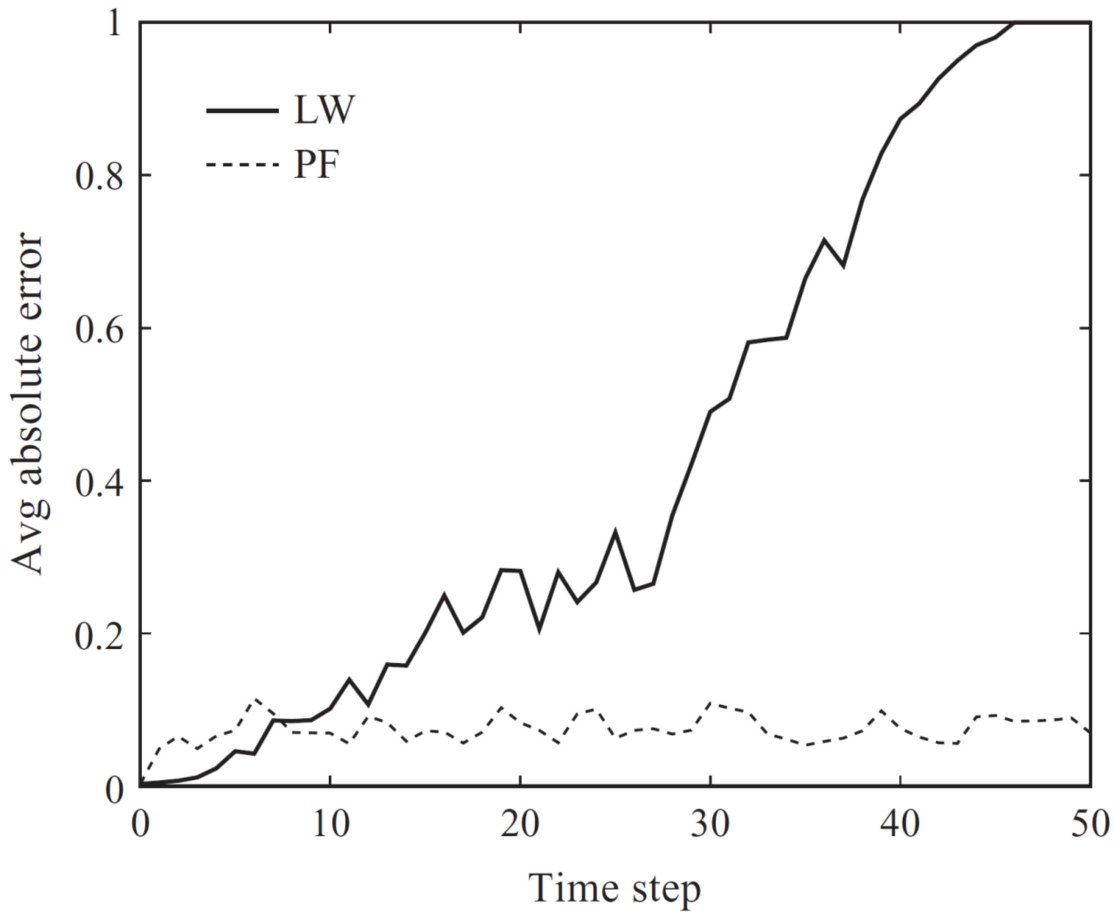
$$
\begin{aligned}
w_t^{(i)} &= \frac{P(\mathbf{e}_t|\mathbf{x}_{t-1}^{(i)})P(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})P(\mathbf{x}_{1:t-1}^{(i)}|\mathbf{e}_{1:t-1})}{Q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)})Q(\mathbf{x}_{1:t-1}^{(i)})} \\
&= \frac{P(\mathbf{e}_t|\mathbf{x}_{t-1}^{(i)})P(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{Q(\mathbf{x}_t^{(i)}|\mathbf{x}_{1:t-1}^{(i)})} w_{t-1}^{(i)}
\end{aligned}
$$

- Normalize the weights, namely set $w_t^{(i)} = w_t^{(i)} / \sum_{i=1}^{N} w_t^{(i)}$
- if $\frac{1}{\sum_{i=1}^{N}(w_t^{(i)})^2}$ is below some pre-defined threshold
  - Resample N particles from the distribution defined by the weights
  - Set all weights to $1/N$
- **Send** the particles and their weights to the next time slice

Particle Filtering: Picture



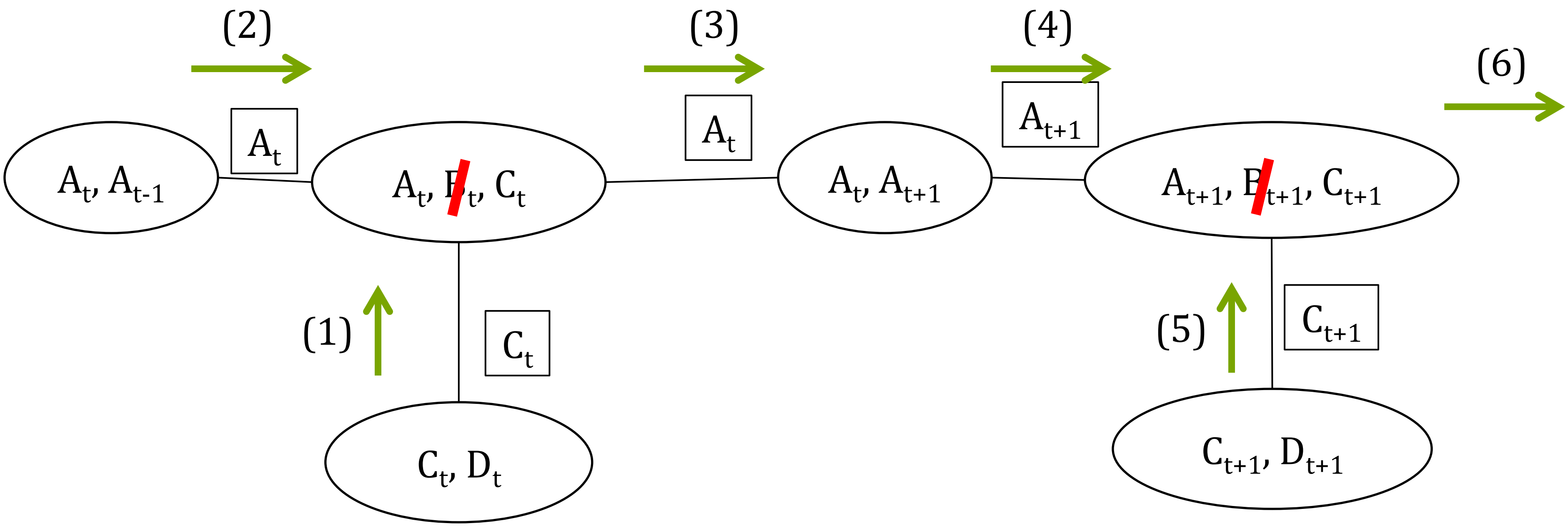
Impact of the Resampling Step



Rao-Blackwellised Particle Filtering

- Combine particle filtering with exact inference. Sample enough variables so that the resulting DBN is tractable
- Suppose we partition $\mathbf{X}_t = (\mathbf{U}_t, \mathbf{V}_t)$
- If the conditional distribution $P(\mathbf{V}_t | \mathbf{u}_t)$ can be computed exactly using junction tree inference or some other method, we only need to sample $\mathbf{U}_t$
- Reduces the variance and thus improves the accuracy

Rao-Blackwellised Particle Filtering



- Suppose we have enough computational resources to do inference with two variables in each cluster!
- Sample B_t at each time slice, Exactly infer others!

Interval Smoothing

- Interval Smoothing is a much harder problem because we have to traverse back in time
- Naïve Algorithm
  - Store all the clusters+messages at each time slice and traverse backwards
    - Large space complexity
  - Clever idea (Murphy, 2002)
    - Stores only O(logT) time slices
    - Factor of O(logT) more expensive time-wise

MAP estimation

- Instead of using sum-out operation we use a max-out operation.

Parameter Learning: FOD

- FOD: fully observable data
  - If every node is observed in every case, the likelihood decomposes into a sum of terms, one per node:

$$
\begin{aligned}
\log P(D|\theta, M) &= \sum_d \log P(X_d|\theta, M) \\
&= \sum_d \log \prod_i P(X_{d,i}|\pi_{d,i}, \theta_i, M) \\
&= \sum_i \sum_d \log P(X_{d,i}|\pi_{d,i}, \theta_i, M)
\end{aligned}
$$

    where $\pi_{d,i}$ are the values of the parents of node i in case d, and θ_i are the parameters associated with CPD i.

Parameter Learning: POD

- POD: Partially observable data

  - If some nodes are sometimes hidden, the likelihood does not decompose.

$$\log P(D|\theta, M) = \sum_d \log \sum_h P(H = h, V = v_d|\theta, M)$$

  - In this case, can use gradient descent or EM to find local maximum.
  - EM iteratively maximizes the expected complete-data log-likelihood, which does decompose into a sum of local terms.

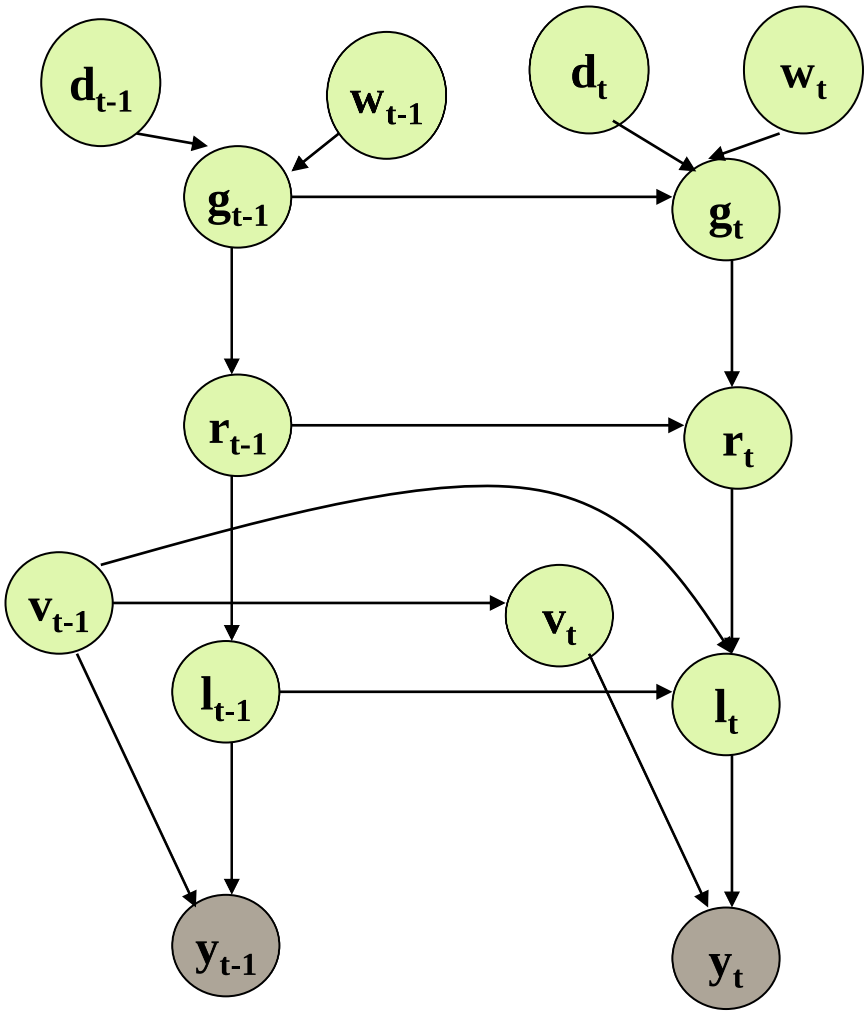
Handling Continuous variables

- Dependencies are often modeled as linear Gaussian
- Example: Current position (X) and Velocity (V)
  - $P(X_t | X_{t-1}, V_t) = X_{t-1} + V_t \Delta + N(0; \sigma^2_X)$; Δ: length of slice
  - $P(V_t | V_{t-1}) = V_{t-1} + N(0; \sigma^2_V)$
- Conditional linear Gaussian
  - Continuous variables have discrete parents
- Hybrid Particle Filtering and GBP algorithms

Applications

- Recognizing activities and transportation routines
- Robotics
- Object tracking
- Bio-informatics
- Speech recognition
- Event detection in Videos

Recognizing travel routines



D: Time-of-day (discrete)

W: Day of week (discrete)

Goal: collection of locations where the person spends significant amount of time. (discrete)

Route: A hidden variable that just predicts what path the person takes (discrete)

Location: A pair (e,d) e is the edge on which the person is and d is the distance of the person from one of the end-points of the edge (continuous)

Velocity: Continuous

GPS reading: (lat,lon,spd,utc).

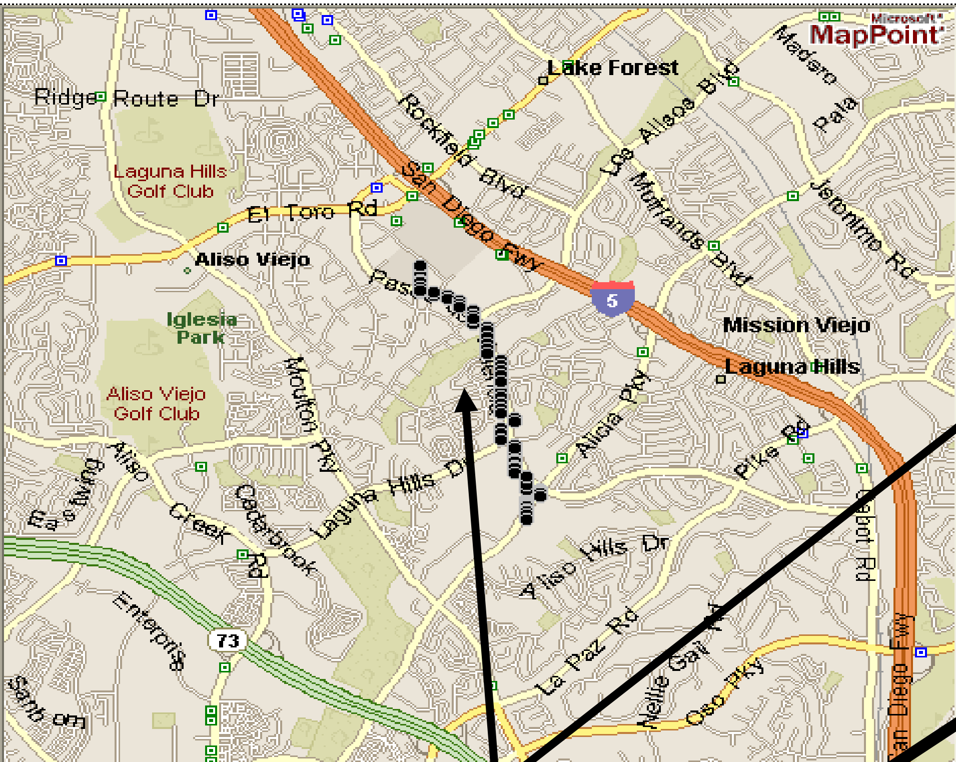
Example Queries

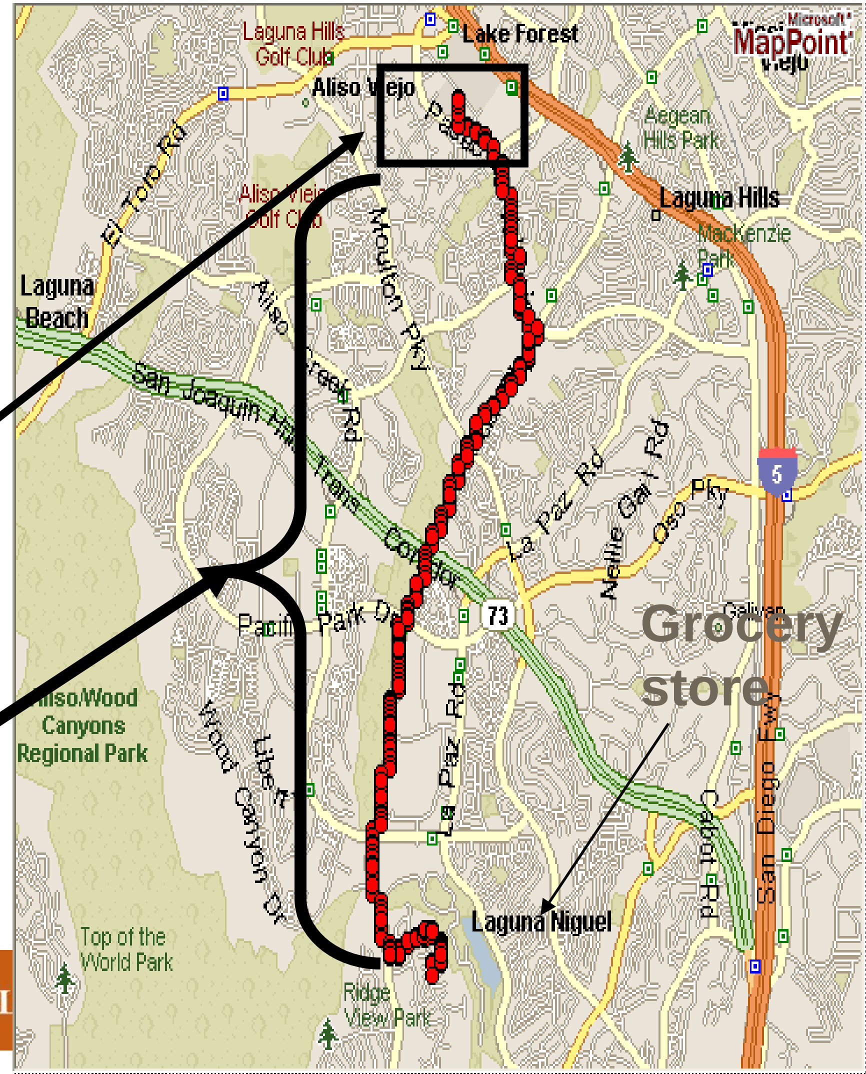
- Where the person will be 10 minutes from now?
  - $P(l_T | d_{1:t}, w_{1:t}, y_{1:t})$ where T=t+10 minutes
- What is the person's next goal?
  - $P(g_T | d_{1:t}, w_{1:t}, y_{1:t})$

UT DALLAS

Example of Goals

Example of Route





Route Seen

- **Route Predicted**

Experimental Results: Data Collection

- GPS data was collected by one of the authors for a period of 6 months.
    - Latitude and longitude pairs
- 3 months data was used for training and 3 months for testing.
- Data divided into segments
    - A segment is a series of GPS readings such that two consecutive readings are less than 15 minutes apart.

Experimental Results: Models and algorithms

- Test if adding new variables improves prediction accuracy.
  - Model-1: Model as described before
  - Model-2: Remove variables d_t and w_t
  - Model-3: Remove variables d_t, w_t,f_t,r_t,g_t from each time slice.
- Algorithms:
  - IJGP-RBPF(1,2), IJGP-RBPF(2,1), IJGP-S(1) and IJGP-S(2)

Learning the models from data

- EM algorithm used for learning the models
- Takes about 3 to 5 days to learn data that is distributed over 3 months.
- Since EM uses inference as a sub-step, we have 4 EM algorithms corresponding to the 4 algorithms used for inference
  - IJGP-RBPF(1,2), IJGP-RBPF(2,1), IJGP-S(1) and IJGP-S(2)

Predicting Goals (MODEL-1)

			Model-1 (20% of the trip seen)			
N	Inference\Learning		IJGP-RBPF(1,1)	IJGP-RBPF(1,2)	IJGP(1)	IJGP(2)
		Time	Accuracy	Accuracy	Accuracy	Accuracy
100	IJGP-RBPF(1,1)	12.3	78	80	79	80
100	IJGP-RBPF(1,2)	15.8	81	84	78	81
200	IJGP-RBPF(1,1)	33.2	80	84	77	82
200	IJGP-RBPF(1,2)	60.3	80	84	76	82
500	IJGP-RBPF(1,1)	123.4	81	84	80	82
500	IJGP-RBPF(1,2)	200.12	84	84	81	82
	IJGP(1)	9	79	79	77	79
	IJGP(2)	34.3	74	84	78	82

- Compute $P(g_t | e_{1:t})$ and compare it with the actual goal.
- Accuracy = percentage of goals predicted correctly.
- N = number of particles
- Column: learning algorithm
- Row: inference algorithm

Predicting Goals (Model-2)

			IJGP-RBPF(1,1)	IJGP-RBPF(1,2)	IJGP(1)	IJGP(2)
N	Inference\Learning	Time	Accuracy	Accuracy	Accuracy	Accuracy
100	IJGP-RBPF(1,1)	8.3	73	73	71	73
100	IJGP-RBPF(1,2)	14.5	76	76	71	75
200	IJGP-RBPF(1,1)	23.4	76	77	71	75
200	IJGP-RBPF(1,2)	31.4	76	77	71	76
500	IJGP-RBPF(1,1)	40.08	76	77	71	76
500	IJGP-RBPF(1,2)	51.87	76	77	71	76
	IJGP(1)	6.34	71	73	71	74
	IJGP(2)	10.78	76	76	72	76

- Compute P(gt|e1:t) and compare it with the actual goal.
- Accuracy = percentage of goals predicted correctly.
- N = number of particles
- Column: learning algorithm
- Row: inference algorithm

Predicting Goals (Model-3)

N	Inference/Learning		IJGP-RBPF(1,1)	IJGP(1)
		Time	Accuracy	
100	IJGP-RBPF(1,1)	2.2	68	61
200	IJGP-RBPF(1,1)	4.7	67	64
500	IJGP-RBPF(1,1)	12.45	68	63
	IJGP(1)	1.23	66	62

- Compute P(gt|e1:t) and compare it with the actual goal.
- Accuracy = percentage of goals predicted correctly.
- N = number of particles
- Column: learning algorithm
- Row: inference algorithm

Predicting Routes

- Compare the path of the person predicted by the model with the actual path.
- False positives (FP)
  - count the number of roads that were not taken by the person but were in the predicted path.
- False Negatives (FN)
  - count the number of roads that were taken by the person but were not in the predicted path.

False Positives and False Negatives for Route prediction

		Model-1	Model-2	Model-3
N	INFERENCE	FP/FN	FP/FN	FP/FN
	IJGP(1)	33/23	39/34	60/55
	IJGP(2)	31/17	39/33	
100	IJGP-RBPF(1,1)	33/21	39/33	60/54
200	IJGP-RBPF(1,1)	33/21	39/33	58/43
100	IJGP-RBPF(1,2)	32/22	42/33	
200	IJGP-RBPF(1,2)	31/22	38/33	

Model-1 shows the highest route prediction accuracy, given by low false positives and false negatives.

Software

- BNT toolkit by Kevin Murphy
- GMTK toolkit by Jeff Bilmes