Fall 2010 Graduate Course on
Dynamic Learning

# Chapter 9: Dynamic Bayesian Networks

November 3, 2010

Byoung-Tak Zhang

School of Computer Science and Engineering &
Cognitive Science and Brain Science Programs
Seoul National University

http://bi.snu.ac.kr/~btzhang/

# Overview

- Motivating Applications
  - Speech Modeling
  - Sequential Data Modeling
- State-Space Models
  - HMM, KFM, DBNs
- DBNs
  - Representation
  - Inference
  - Learning
- References

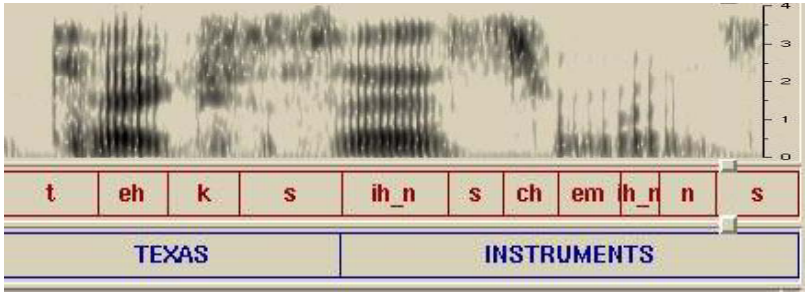
# Feature-Based Pronunciation Modeling

warmth → dictionary →

| | | | | | |
|---|---|---|---|---|---|
| voicing | V | V | V | V | !V |
| velum | Clo | Clo | Clo | Op | Clo |
| lip opening | Nar | Mid | Mid | Clo | Mid |
| ... | ... | ... | ... | ... | ... |

→ lips & velum desynchronize → [ w ao r m p th ]

- wants → [w aa_n t s] -- Phone deletion??
- several → [s eh r v ax l] -- Exchange of two phones???
- *instruments* → [ih_n s ch em ih_n n s]




everybody → [eh r uw ay]


[Karen Livescu, 2004]

# Approach: Main Ideas ([*HLT/NAACL-2004*])

- Begin with usual assumption: Each word has one or more *underlying* pronunciations, given by a dictionary

warmth → dictionary

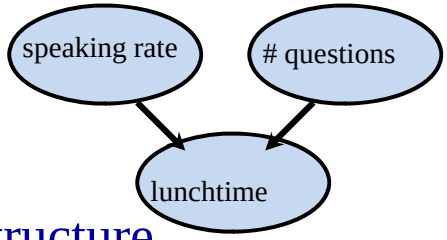
| index | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| voicing | V | V | V | V | !V |
| velum | Off | Off | Off | On | Off |
| lip opening | Nar | Mid | Mid | Clo | Mid |
| ... | ... | ... | ... | ... | … |

- Surface (actual) feature values can stray from underlying values via:
  1) Substitution – modeled by confusion matrices P(s|u)
  2) Asynchrony
     - Assign index (counter) to each feature, and allow index values to differ
     - Apply constraints on the difference between the mean indices of feature subsets
- Natural to implement using graphical models, in particular dynamic Bayesian networks (DBNs)

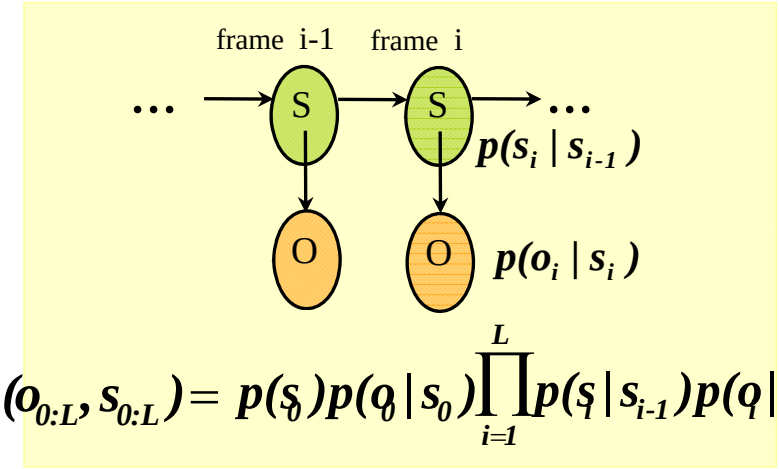
[Karen Livescu, 2004]

# Dynamic Bayesian networks

- **Bayesian network (BN):** Directed-graph representation of a distribution over a set of variables
  - Vertex ⇔ variable + its distribution given the parents
  - Edge ⇔ "dependency"



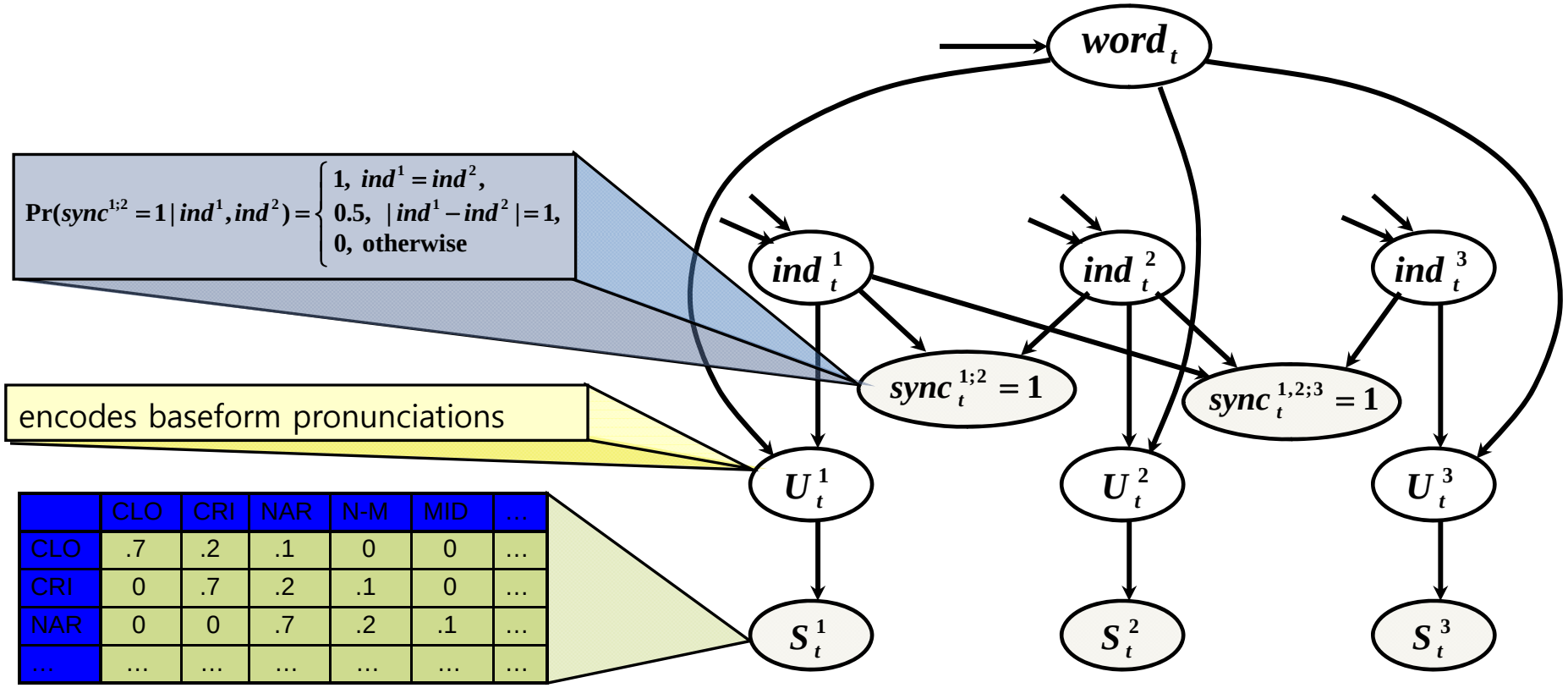

- **Dynamic Bayesian network (DBN):** BN with a repeating structure
- Example: HMM




- Uniform algorithms for (among other things)
  - Finding the most likely values of a subset of the variables, given the rest (analogous to Viterbi algorithm for HMMs)
  - Learning model parameters via EM

# Approach: A DBN-based Model

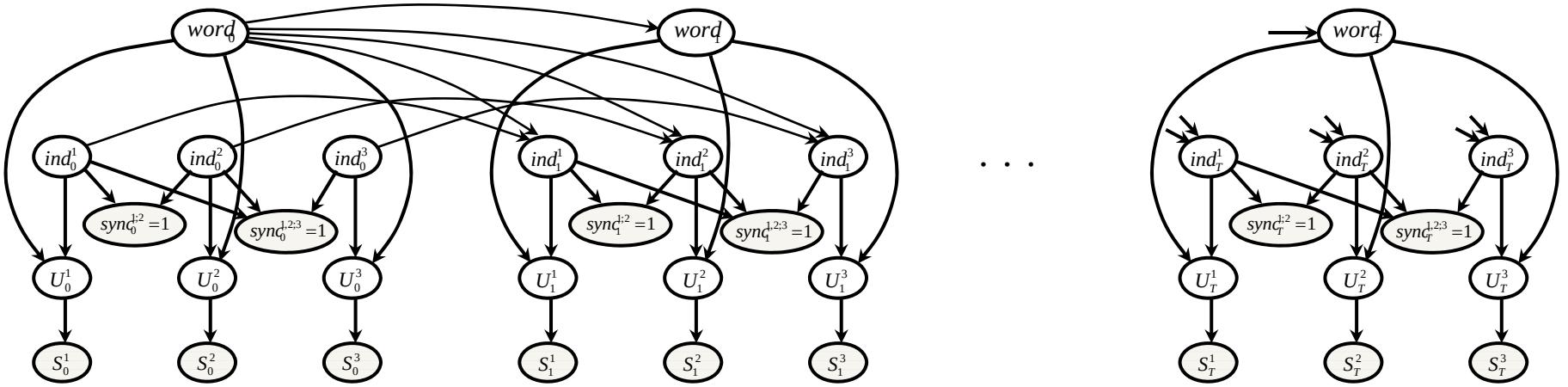
- Example DBN using 3 features:




| | CLO | CRI | NAR | N-M | MID | … |
|---|---|---|---|---|---|---|
| CLO | .7 | .2 | .1 | 0 | 0 | … |
| CRI | 0 | .7 | .2 | .1 | 0 | … |
| NAR | 0 | 0 | .7 | .2 | .1 | … |
| … | … | … | … | … | … | … |

# Approach: A DBN-based Model (2)

- "Unrolled" DBN:




- Parameter learning via Expectation Maximization (EM)
- Training data
  - Articulatory databases
  - Detailed phonetic transcriptions

# Sequential Data Modeling

- Sequential data modeling is important in many areas
- Time series generated by a dynamical system
  - Time series modeling
- A sequence generated by an one-dimensional spatial process
  - Bio-sequences

# The Solutions

- Classic approaches to time-series prediction
  - Linear models: ARIMA (autoregressive integrated moving average), ARMAX (autoregressive moving average exogenous variables model)
  - Nonlinear models: neural networks, decision trees
- Problems with classic approaches
  - Prediction of the future is based on only a finite window
  - It's difficult to incorporate prior knowledge
  - Difficulties with multi-dimensional inputs and/or outputs
- State-space models
  - Assume there is some underlying hidden state of the world (query) that generates the observations (evidence), and that this hidden state evolves in time, possibly as a function of our inputs.
  - The belief state: our belief on the hidden state of the world given the observations up to the current time $y_{1:t}$ and our inputs $u_{1:t}$ to the system, $P( X \mid y_{1:t}, u_{1:t} )$
  - Two most common state-space models: hidden Markov models(HMMs) and Kalman filter models (KFMs)
  - A more general state-space model: dynamic Bayesian networks (DBNs)

# State-Space Models: Representation

- Any state-space model must define a prior $P(X_1)$ and a state-transition function, $P(X_t \mid X_{t-1})$, and an observation function, $P(Y_t \mid X_t)$.
- Assumptions:
  - Models are first-order Markov: $P(X_t \mid X_{1:t-1}) = P(X_t \mid X_{t-1})$
  - Observations are conditional first-order Markov: $P(Y_t \mid X_t , Y_{t-1}) = P(Y_t \mid X_t)$
  - Time-invariant or homogeneous
- Representations:
  - HMMs: $X_t$ is a discrete random variables
  - KFMs: $X_t$ is a vector of continuous random variables
  - DBNs: more general and expressive language for representing state-space models

# State-Space Models: Inference

- A state-space model defines how $X_t$ generates $Y_t$ and $X_{t.}$
- The goal of inference is to infer the hidden states (query) $X_{1:t}$ given the observations (evidence) $Y_{1:t}$.
- Inference tasks:
  - **Filtering (monitoring)**: recursively estimate the belief state using Bayes' rule
    - Predict: computing $P(X_t \mid y_{1:t-1})$
    - Update: computing $P(X_t \mid y_{1:t})$
    - Throw away the old belief state once we have computed the prediction ("rollup")
  - **Smoothing**: estimate the state of the past, given all the evidence up to the current time
    - Fixed-lag smoothing (hindsight): computing $P(X_{t-l} \mid y_{1:t})$ where l > 0 is the lag
  - **Prediction**: predict the future
    - Lookahead: computing $P(X_{t+h} \mid y_{1:t})$ where h > 0 is how far we want to look ahead
  - (Viterbi) **Decoding**: compute the most likely sequence of hidden states given the data
    - MPE(abduction): $x^*_{1:t} = \text{argmax}\ P(x_{1:t} \mid y_{1:t})$

# State-Space Models: Learning

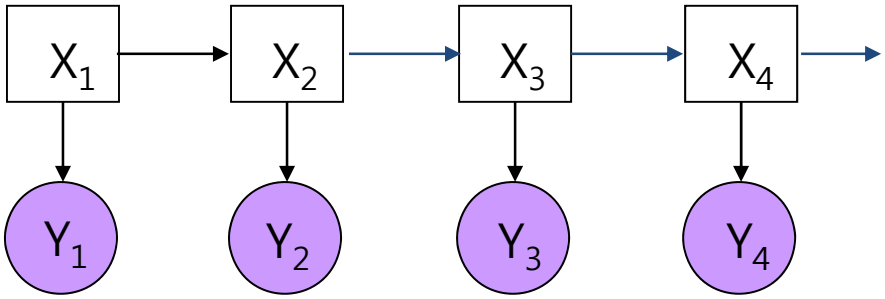
- **Parameter learning** (system identification): estimating from data the parameters that define the transition model $P(X_t \mid X_{t-1})$ and the observation model $P(Y_t \mid X_t)$
  - The usual criterion is maximum-likelihood (ML)
- The goal of parameter learning is to compute

  $\theta^*_{\text{ML}} = \text{argmax}_\theta \; P(Y|\theta) = \text{argmax}_\theta \; \log P(Y|\theta)$ or

  $\theta^*_{\text{MAP}} = \text{argmax}_\theta \; \log P(Y|\theta) + \log P(\theta)$

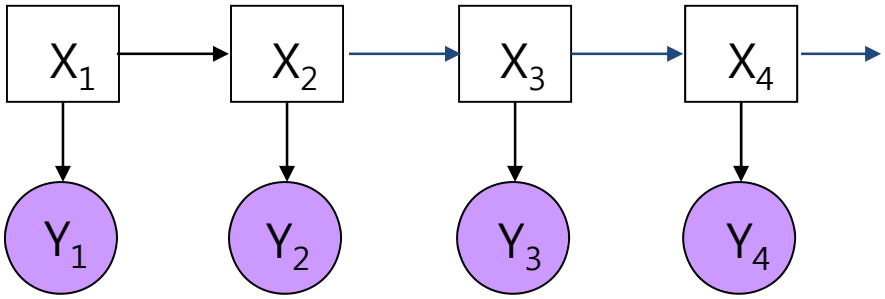
  if we include a prior on the parameters
- Two standard approaches to parameter learning:
  - gradient ascent and EM (expectation maximization)
- **Structure learning**: more ambitious

# HMM: Hidden Markov Model




- One discrete hidden node and one discrete or continuous observed node per time slice.
  - X: hidden variables
  - Y: observations
- Structures and parameters remain the same over time
- Three parameters in an HMM:
  - The initial state distribution $P(X_1)$
  - The transition model $P(X_t \mid X_{t-1})$
  - The observation model $P(Y_t \mid X_t)$
- HMM is the simplest DBN
  - A discrete state variable with arbitrary dynamics and arbitrary measurements

# KFM: Kalman Filter Model




- KFM has the same topology as an HMM
- All the nodes are assumed to have linear-Gaussian distributions

  $x(t+1) = F*x(t) + w(t)$,

  $w \sim N(0,Q)$ : process noise, $x(0) \sim N(X(0), V(0))$

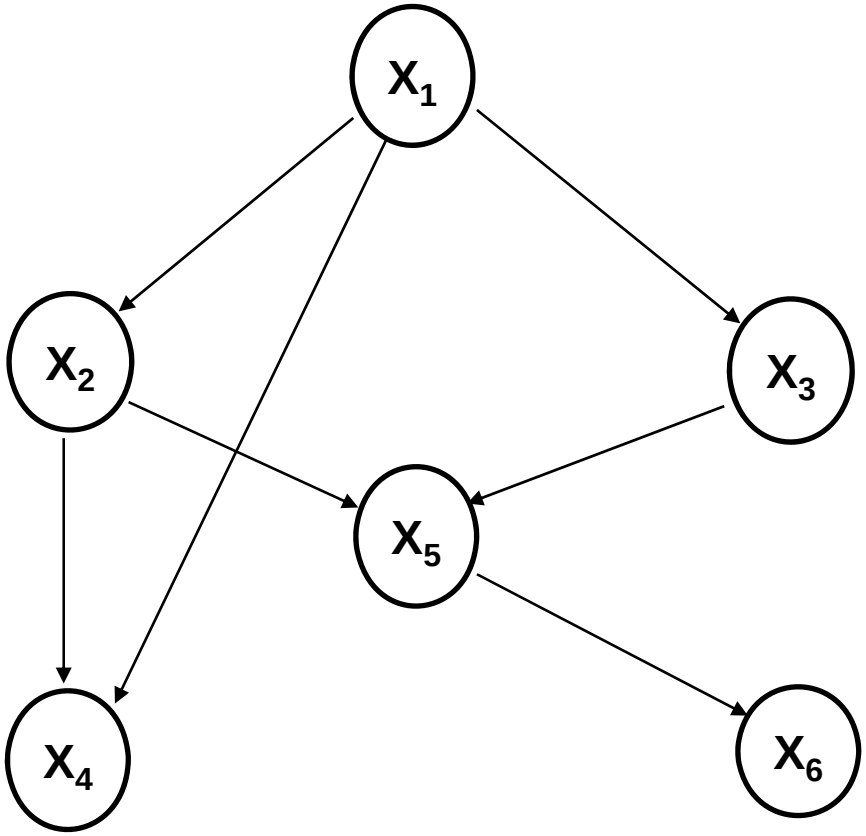
  $y(t) = H*x(t) + v(t)$,

  $v \sim N(0,R)$ : measurement noise
- Also known as Linear Dynamic Systems (LDSs)
  - A partially observed stochastic process
  - with linear dynamics and linear observations: $f(a + b) = f(a) + f(b)$
  - both subject to Gaussian noise
- KFM is the simplest continuous DBN
  - A continuous state variable with linear-Gaussian dynamics and measurements

# Bayesian Network: Definition

- Factored joint probability distribution as a directed graph
  - Structure for representing knowledge about uncertain variables
  - Computational architecture for computing the impact of evidence on beliefs
- Knowledge structure
  - Nodes (vertices) represent the variables
  - Arcs (edges) represent probabilistic dependence between variables
  - Conditional probabilities encode the strength of the dependencies
- Computational architecture
  - Computes posterior probabilities given evidence about selected nodes
  - Exploits probabilistic independence for efficient computation

# Sample Factored Joint Distribution




$$P(x_1, x_2, x_3, x_4, x_5, x_6) = P(x_6 \mid x_5)\, P(x_5 \mid x_3, x_2)\, P(x_4 \mid x_2, x_1)\, P(x_3 \mid x_1)\, P(x_2 \mid x_1)\, P(x_1)$$

# DBN: Dynamic Bayesian Networks

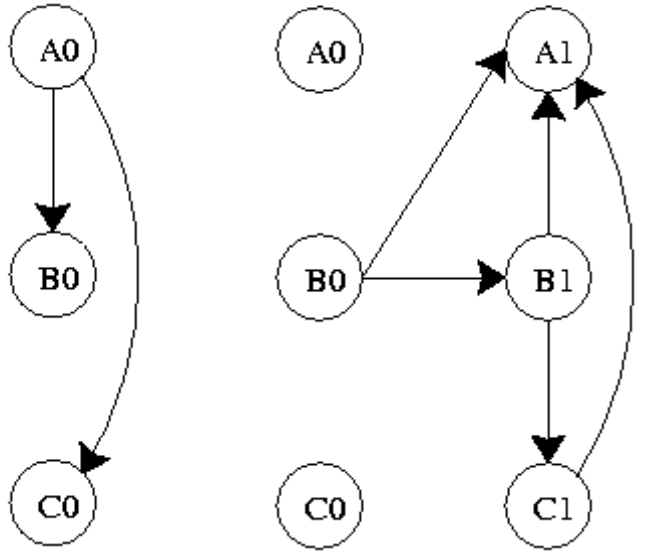
- DBNs are directed graphical models of stochastic process.
- DBNs generalize HMMs and KFMs by representing the hidden and observed states in terms of state variables, which can have complex interdependencies.
- The graphical structure provides an easy way to specify the conditional independencies.
- A compact parameterization of the state-space model.
- An extension of BNs to handle temporal models.
- Time-invariant: the term ‘dynamic’ means that we are modeling a dynamic model, not that the networks change over time

# DBN: A Formal Definition

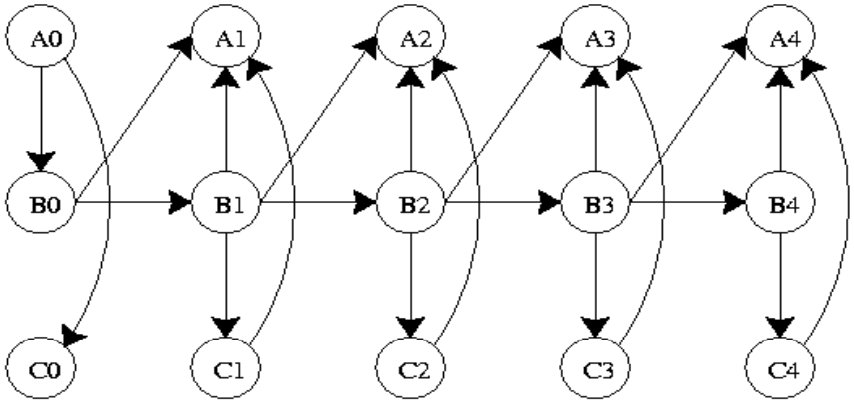
- Definition: A DBN is defined as a pair ($B_0$, $B\rightarrow$), where $B_0$ defines the prior $P(Z_1)$, and is a two-slice temporal Bayes net (2TBN) which defines $P(Z_t \mid Z_{t-1})$ by means of a DAG (directed acyclic graph) as follows:

$$P(Z_t \mid Z_{t-1}) = \prod_{i=1}^{N} P(Z_t^i \mid \pi(Z_t^i))$$

- $Z(i,t)$ is a node at time slice $t$, it can be a hidden node, an observation node, or a control node (optional)
- $Pa(Z_{(i,t)})$ are parent nodes of $Z(i,t)$, they can be at either time slice $t$ or $t-1$
- The nodes in the first slice of a 2TBN do not have parameters associated with them.
- But each node in the second slice has an associated CPD (conditional probability distribution).



# The Semantics of a DBN




- First-order Markov assumption: the parents of a node can only be in the same time slice or the previous time slice, i.e., arcs do not across slices
- Inter-slice arcs are all from left to right, reflecting the arrow of time
- Intra-slice arcs can be arbitrary as long as the overall DBN is a DAG
- Time-invariant assumption: the parameters of the CPDs don't change over time
- The semantics of DBN can be defined by "unrolling" the 2TBN to *T* time slices
- The resulting joint probability distribution is then defined by

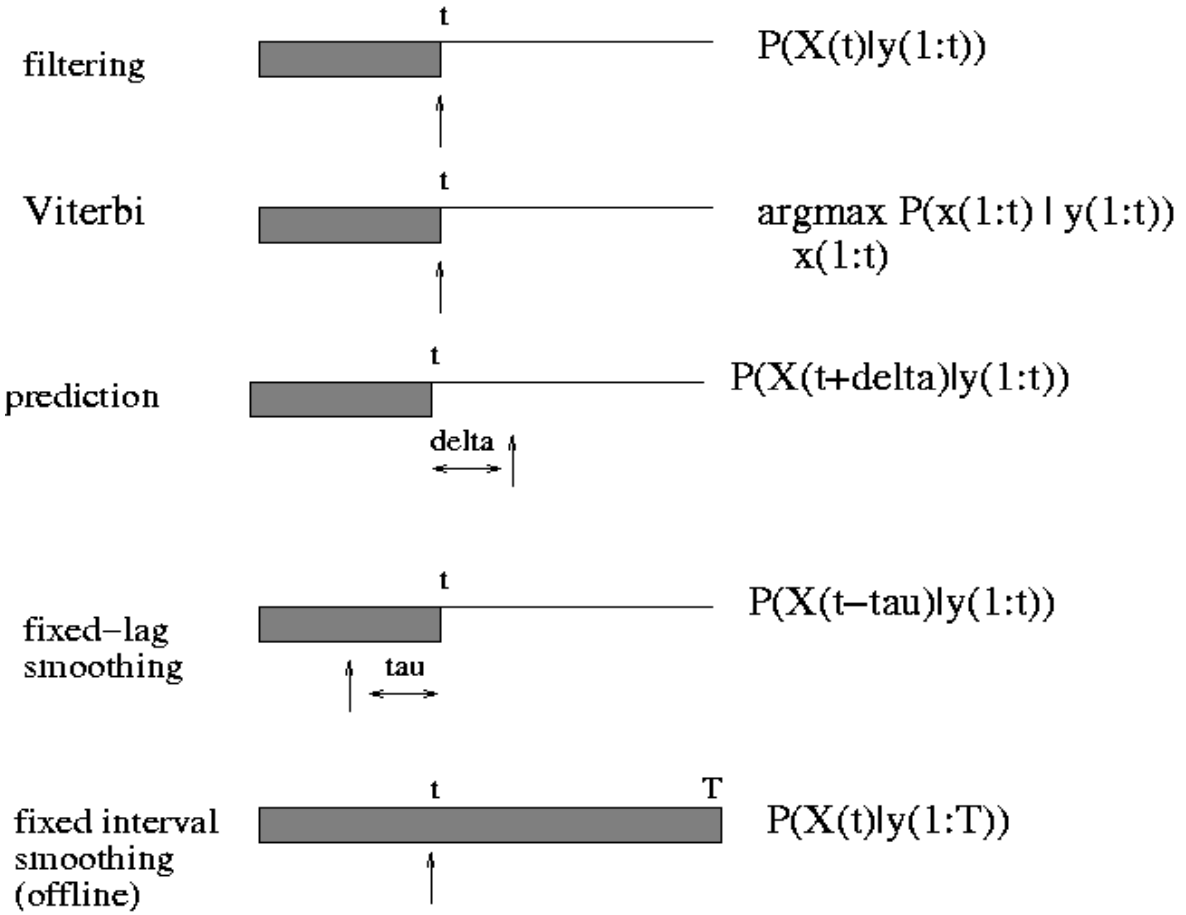
$$P(Z_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} P(Z_t^i \mid \pi(Z_t^i))$$

# DBN, HMM, and KFM

- DBN represents the (hidden) state in terms of a set of random variables.
  - HMM's state space consists of a single random variable.
- DBN allows arbitrary CPDs.
  - KFM requires all the CPDs to be linear-Gaussian.
- DBN allows much more general graph structures.
  - HMMs and KFMs have a restricted topology (chain structure).
- DBN has more expressive power.
  - DBN generalizes HMM and KFM.

# DBN: Inference

- The goal of inference in DBNs is to compute $P(X_t^i \mid y_{1:r})$
  - Filtering: $r = t$
  - Smoothing: $r > t$
  - Prediction: $r < t$
  - Viterbi: MPE



# DBN Inference Algorithms

- DBN inference algorithms extend HMM and KFM's inference algorithms, and call BN inference algorithms as subroutines.
- DBN inference is NP-hard.
- **Exact inference algorithms**
  - Forward-backward smoothing algorithm (on any discrete-state DBN)
  - The frontier algorithm (sweep a Markov blanket, the frontier set *F*, across the DBN, first forwards and then backwards)
  - The interface algorithm (use only the set of nodes with outgoing arcs to the next time slice to d-separate the past from the future)
  - Kalman filtering and smoothing
- **Approximate algorithms**
  - The Boyen-Koller (BK) algorithm (approximate the joint distribution over the interface as a product of marginals)
  - Factored frontier (FF) algorithm
  - Loopy propagation algorithm (LBP)
  - Kalman filtering and smoother
  - Stochastic sampling algorithms
    - Importance sampling or MCMC (offline inference)
    - Particle filtering (PF) (online)

# DBN: Learning

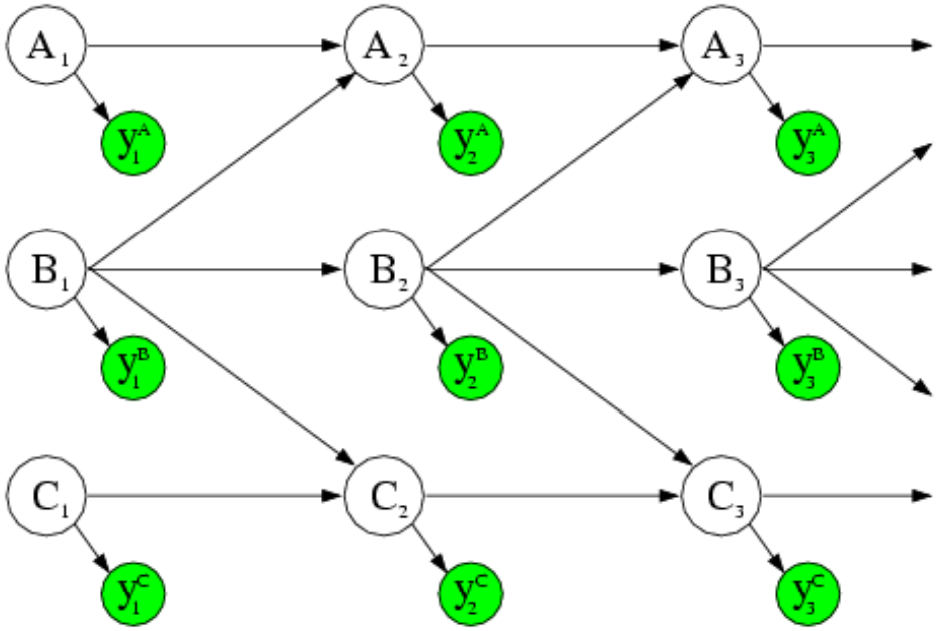
- The techniques for learning DBNs are mostly straightforward extensions of the techniques for learning BNs.
- **Parameter learning**
  - Offline learning
    - Parameters must be tied across time-slices
    - The initial state of the dynamic system can be learned independently of the transition matrix
  - Online learning
    - Add the parameters to the state space and then do online inference (filtering).
- **Structure learning**
  - The intra-slice connectivity must be a DAG.
  - Learning the inter-slice connectivity is equivalent to the variable selection problem, since for each node in slice *t*, we must choose its parents from slice *t-1*.
  - Learning for DBNs reduces to feature selection if we assume the intra-slice connections are fixed
- Learning uses inference algorithms as subroutines.

# DBN Learning Applications

- Learning genetic network topology using structural EM
  - Gene pathway models
- Inferring motifs using HHMMs
  - Motifs are short patterns which occur in DNA and have certain biological significance; $\{A, C, G, T\}^*$
- Inferring people's goals using abstract HMMs
  - Inferring people's intentional states by observing their behavior
- Modeling freeway traffic using coupled HMMs

# Particle Filtering for DBNs

# Example: ABC Network




**Goal**: compute the *joint filtering distribution* $p(A_t, B_t, C_t | \mathbf{y_{1:t}})$

Note the factorisation (using the notation $\mathbf{A_{1:t}} \triangleq \{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_t}\}$):

$$\begin{aligned} p(\mathbf{A_{1:t}}, \mathbf{B_{1:t}}, \mathbf{C_{1:t}} | \mathbf{y_{1:t}}) &= p(\mathbf{A_{1:t}}, \mathbf{C_{1:t}} | \mathbf{y_{1:t}}, \mathbf{B_{1:t}}) \, p(\mathbf{B_{1:t}} | \mathbf{y_{1:t}}) \\ &= p\left(\mathbf{A_{1:t}} | \mathbf{y}^{\mathbf{A}}_{\mathbf{1:t}}, \mathbf{B_{1:t-1}}\right) p\left(\mathbf{C_{1:t}} | \mathbf{y}^{\mathbf{C}}_{\mathbf{1:t}}, \mathbf{B_{1:t-1}}\right) p(\mathbf{B_{1:t}} | \mathbf{y_{1:t}}) \end{aligned}$$

# Inference in DBN

**Exact**:

- $O(q^{2\times 3})$ operations per time step for discrete nodes with $q$ values.
- Impossible for almost all continuous distributions. HZ1

**Particle filtering (PF)**:

- Monte Carlo approximation algorithm that applies to any distribution.
- $O(N)$ operations per time step. $N$ is the number of samples.
- Estimates can have high variance.

**Rao Blackwellised particle filtering (RBPF)**:

- Sample a subset of the variables and compute the distributions of the remaining variables exactly.
- Reduces the variance problem (Casella and Robert, 1996).
- $O(q^2 N')$ operations per time step for the ABC network.

HZ1 for the hidden variables.

Obs variables can have Gaussian distribution
hzhou, 2004-03-15

# Exact inference in ABC network

**Prediction:** By standard marginalisation

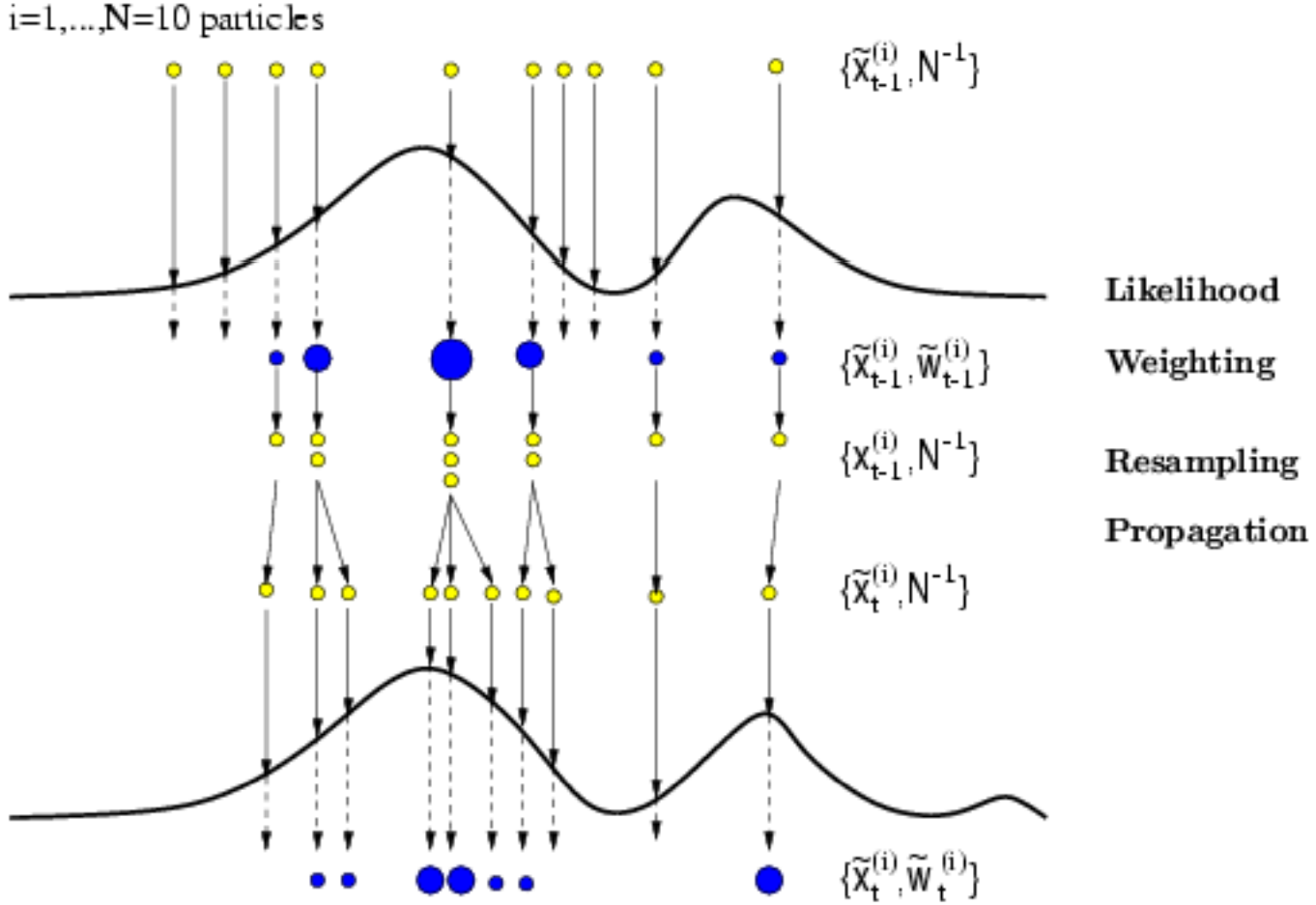
$$
\begin{aligned}
p(A_t, B_t, C_t|\mathbf{y}_{1:t-1}) &= \sum_{A_{t-1}} \sum_{B_{t-1}} \sum_{C_{t-1}} p(A_t, B_t, C_t|A_{t-1}, B_{t-1}, C_{t-1}) \\
&\quad \times p(A_{t-1}, B_{t-1}, C_{t-1}|\mathbf{y}_{1:t-1}) \\
&= \sum_{A_{t-1}} \sum_{B_{t-1}} \sum_{C_{t-1}} p(A_t|A_{t-1}, B_{t-1}) p(B_t|B_{t-1}) \\
&\quad \times p(C_t|B_{t-1}, C_{t-1}) p(A_{t-1}, B_{t-1}, C_{t-1}|\mathbf{y}_{1:t-1}) \\
&= \sum_{B_{t-1}} p(B_t|B_{t-1}) \sum_{A_{t-1}} p(A_t|A_{t-1}, B_{t-1}) \\
&\quad \times \sum_{C_{t-1}} p(C_t|B_{t-1}, C_{t-1}) p(A_{t-1}, B_{t-1}, C_{t-1}|\mathbf{y}_{1:t-1})
\end{aligned}
$$

**Update:** Using Bayes' rule

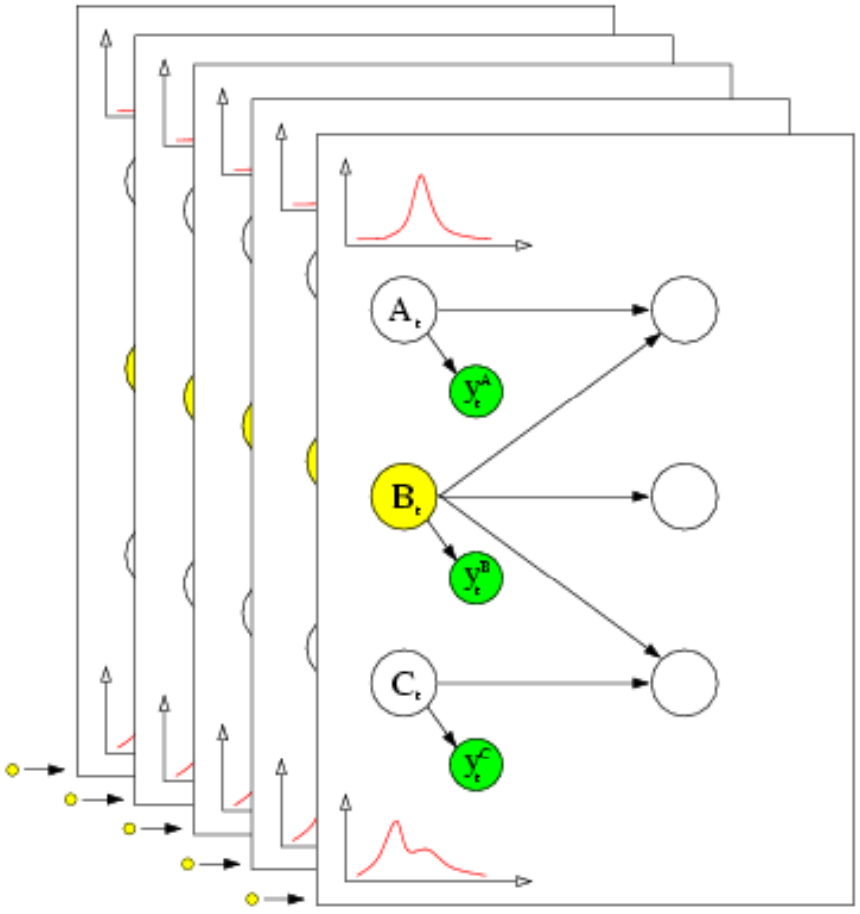
$$
\begin{aligned}
p(A_t, B_t, C_t|\mathbf{y}_{1:t}) &= \frac{p(y_t|A_t, B_t, C_t) p(A_t, B_t, C_t|\mathbf{y}_{1:t-1})}{\sum_{A_t} \sum_{B_t} \sum_{C_t} p(y_t|A_t, B_t, C_t) p(A_t, B_t, C_t|\mathbf{y}_{1:t-1})} \\
&= \frac{p(y_t^A|A_t) p(y_t^B|B_t) p(y_t^C|C_t) p(A_t, B_t, C_t|\mathbf{y}_{1:t-1})}{\sum_{A_t} \sum_{B_t} \sum_{C_t} p(y_t|A_t, B_t, C_t) p(A_t, B_t, C_t|\mathbf{y}_{1:t-1})}
\end{aligned}
$$

# Particle filtering

**Idea**: approximate the filtering distribution with a finite set of samples (particles). Then compute estimates by Monte Carlo averaging.



# Rao-Blackwellized PF



# Rao-Blackwellized PF (2)

**Prediction:**

$$p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) = \sum_{A_{t-1}} p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}, A_{t-1}) p(A_{t-1}|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t})$$
$$= \sum_{A_{t-1}} p(A_t|B_{t-1}, A_{t-1}) p(A_{t-1}|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1})$$

and similarly

$$p(C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) = \sum_{C_{t-1}} p(C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}, C_{t-1}) p(C_{t-1}|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t})$$
$$= \sum_{C_{t-1}} p(C_t|B_{t-1}, C_{t-1}) p(C_{t-1}|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1})$$

# Rao-Blackwellized PF (3)

**Update:**

$$\begin{aligned} p(A_t|\mathbf{y}_{1:t}, \mathbf{B}_{1:t}) &\propto p(y_t|\mathbf{y}_{1:t-1}, A_t, \mathbf{B}_{1:t}) p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) \\ &\propto p(y_t^A|A_t) p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) \end{aligned}$$

and similarly,

$$p(C_t|\mathbf{y}_{1:t}, \mathbf{B}_{1:t}) \propto p(y_t^C|C_t) p(C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t})$$

Once again, we use the transition prior as proposal distribution

$$q(\mathbf{B}_{1:t}|\mathbf{y}_{1:t}) = p(B_t|B_{t-1}) p(\mathbf{B}_{1:t-1}|\mathbf{y}_{1:t-1})$$

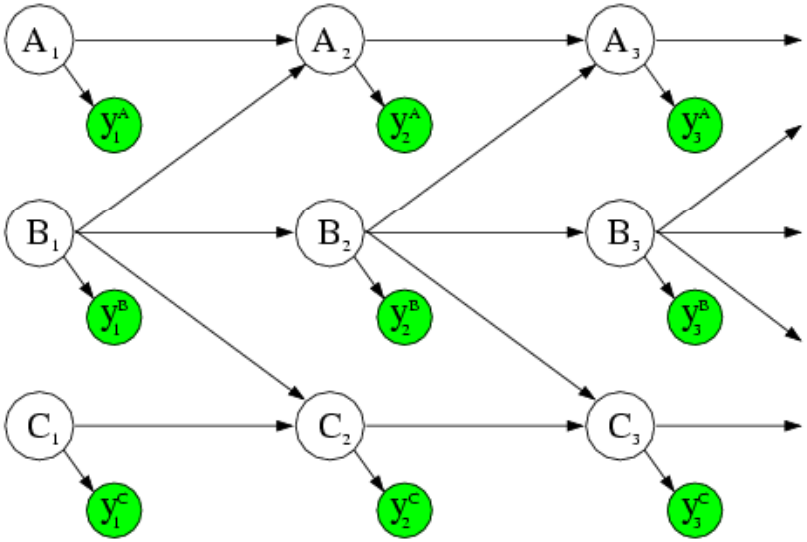
$$w_t = \frac{p(\mathbf{B}_{1:t}|\mathbf{y}_{1:t})}{q(\mathbf{B}_{1:t}|\mathbf{y}_{1:t})} \propto p(y_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t})$$

# Rao-Blackwellised PF (4)

$$
\begin{aligned}
p(y_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) &= \sum_{A_t}\sum_{C_t} p(y_t, A_t, C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) \\
&= \sum_{A_t}\sum_{C_t} p(y_t|\mathbf{y}_{1:t-1}, A_t, C_t, \mathbf{B}_{1:t}) p(A_t, C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t}) \\
&= \sum_{A_t}\sum_{C_t} p(y_t^A|A_t) p(y_t^B|B_t) p(y_t^C|C_t) p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1}) \\
&\qquad \times p(C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1}) \\
&= p(y_t^B|B_t) \sum_{A_t} p(y_t^A|A_t) p(A_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1}) \\
&\qquad \times \sum_{C_t} p(y_t^C|C_t) p(C_t|\mathbf{y}_{1:t-1}, \mathbf{B}_{1:t-1})
\end{aligned}
$$

# Discussions

- Structure of the network:
  - A, C dependent on B
  - $y_t$ can be also separated into 3 independent parts



# Summary

- DBN is a general state-space model to describe a stochastic dynamic system.
- HMMs and KFMs are special cases of DBNs.
- DBNs have more “expressive power”.
- DBN inference includes filtering, smoothing, prediction; uses BN inference as subroutines
- DBN structure learning includes the learning of intra-slice connections and inter-slice connections.
- DBN has a broad range of real world applications.

# References


- Boyen, X. and Koller R, D., Tractable Inference for Complex Stochastic Processes, In UAI-98, 1998.
- Friedman, N., Murphy, K., and Russel, S., Learning the structure of dynamic probabilistic networks, In 12th UAI, 1998.
- Guo, H., Dynamic Bayesian Networks, Slide File, Kensas State University, 2002.
- Kjrulff, U., A computational scheme for reasoning in dynamic probabilistic networks , Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, 121-129, Morgan Kaufmann, San Francisco, 1992.
- Murphy, K. P., Dynamic Bayesian Networks: Representation, Inference and Learning, PhD thesis, UC Berkeley, Computer Science Division, July 2002.
- Murphy, K., and Mian, S., Modelling Gene Expression Data using Dynamic Bayesian Networks, Technical Report, UC Berkeley, 1999.
- Stephenson, T. A., An Introduction to Bayesian Network Theory and Usage , 2000.
- Zweig, G. Speech Recognition with Dynamic Bayesian Networks. Ph.D. Thesis, University of California, Berkeley, 1997.