A Tutorial on Dynamic Bayesian Networks

Kevin P. Murphy
MIT AI lab

12 November 2002

Modelling sequential data

- Sequential data is everywhere, e.g.,
  - Sequence data (offline): Biosequence analysis, text processing, ...
  - Temporal data (online): Speech recognition, visual tracking, financial forecasting, ...
- Problems: classification, segmentation, state estimation, fault diagnosis, prediction, ...
- Solution: build/learn generative models, then compute $P(\text{quantity of interest}|\text{evidence})$ using Bayes rule.

Outline of talk

- Representation
  - What are DBNs, and what can we use them for?
- Inference
  - How do we compute $P(X_t|y_{1:t})$ and related quantities?
- Learning
  - How do we estimate parameters and model structure?

Representation

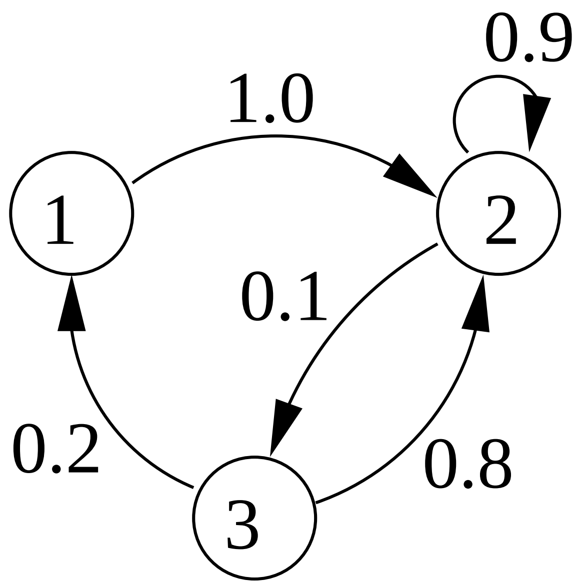
- Hidden Markov Models (HMMs).
- Dynamic Bayesian Networks (DBNs).
- Modelling HMM variants as DBNs.
- State space models (SSMs).
- Modelling SSMs and variants as DBNs.

Hidden Markov Models (HMMs)

- An HMM is a stochastic finite automaton, where each state generates (emits) an observation.
- Let $X_t \in \{1, \ldots, K\}$ represent the hidden state at time t, and Y_t represent the observation.
- e.g., $X =$ phones, $Y =$ acoustic feature vector.
- Transition model: $A(i,j) \stackrel{\triangle}{=} P(X_t = j | X_{t-1} = i)$.
- Observation model: $B(i,k) \stackrel{\triangle}{=} P(Y_t = k | X_t = i)$.
- Initial state distribution: $\pi(i) \stackrel{\triangle}{=} P(X_0 = i)$.

HMM state transition diagram

- Nodes represent states.
- There is an arrow from i to j iff $A(i, j) > 0$.



The 3 main tasks for HMMs

- Computing likelihood: $P(y_{1:t}) = \sum_i P(X_t = i, y_{1:t})$
- Viterbi decoding (most likely explanation): $\arg\max_{x_{1:t}} P(x_{1:t}|y_{1:t})$
- Learning: $\widehat{\theta}_{ML} = \arg\max_\theta P(y_{1:T}|\theta)$, where $\theta = (A, B, \pi)$.
  - Learning can be done with Baum-Welch (EM).
  - Learning uses inference as a subroutine.
  - Inference (forwards-backwards) takes $O(TK^2)$ time, where K is the number of states and T is sequence length.

The problem with HMMs

- Suppose we want to track the state (e.g., the position) of D objects in an image sequence.
- Let each object be in K possible states.
- Then $X_t = (X_t^{(1)}, \ldots, X_t^{(D)})$ can have K^D possible values.

$\Rightarrow$ Inference takes $O\left(T(K^D)^2\right)$ time and $O\left(TK^D\right)$ space.

$\Rightarrow$ $P(X_t|X_{t-1})$ needs $O(K^{2D})$ parameters to specify.

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.

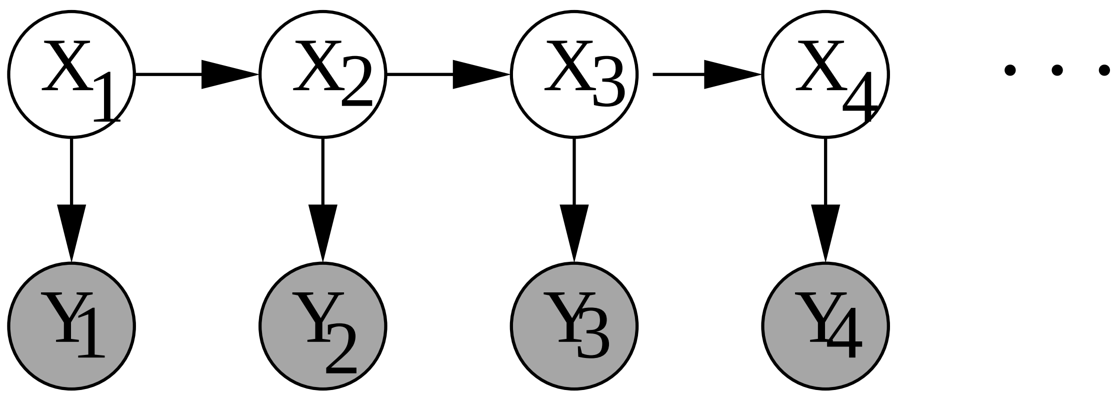
⇒ A DBN may have exponentially fewer parameters than its corresponding HMM.

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

DBNs are a kind of graphical model

- In a graphical model, nodes represent random variables, and (lack of) arcs represents conditional independencies.
- Directed graphical models = Bayes nets = belief nets.
- DBNs are Bayes nets for dynamic processes.
- Informally, an arc from X_i to X_j means X_i "causes" X_j. (Graph must be acyclic!)

HMM represented as a DBN

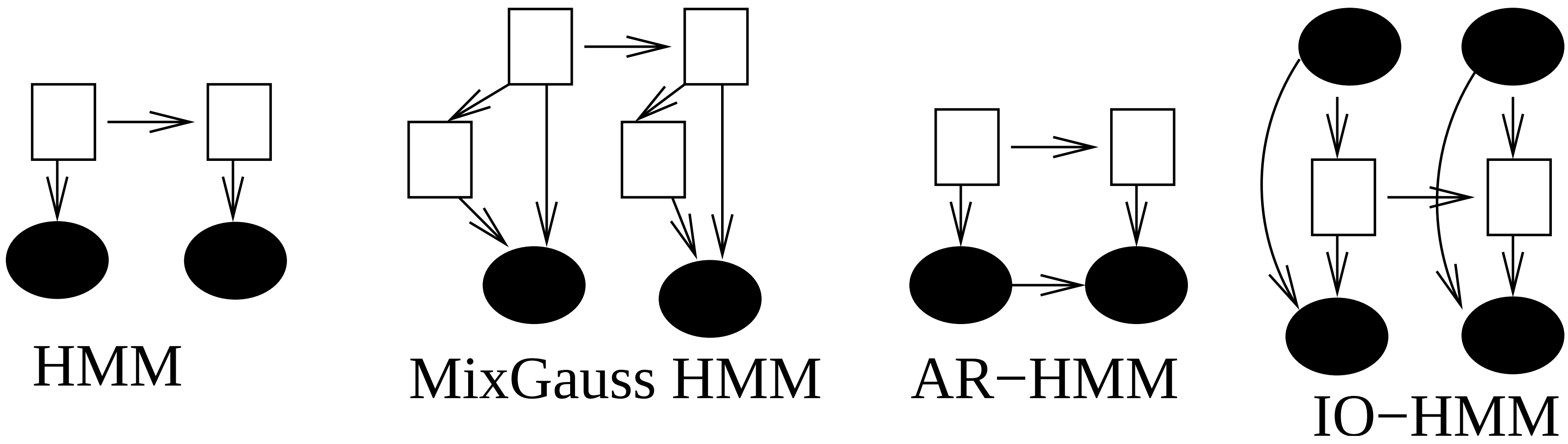


- This graph encodes the assumptions

$$Y_t \perp Y_{t'} | X_t \text{ and } X_{t+1} \perp X_{t-1} | X_t \text{ (Markov)}$$

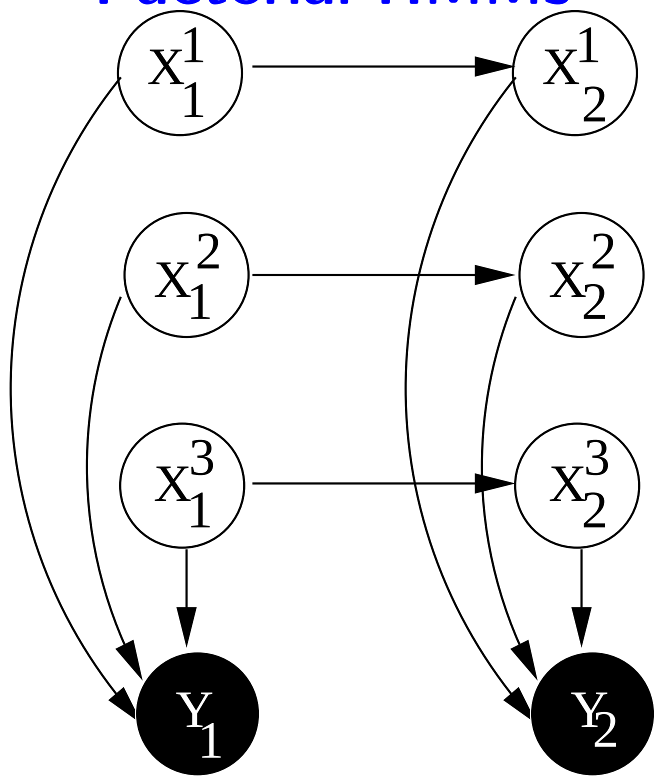
- Shaded nodes are observed, unshaded are hidden.
- Structure and parameters repeat over time.

HMM variants represented as DBNs



⇒ The same code can do inference and learning in all of these models.

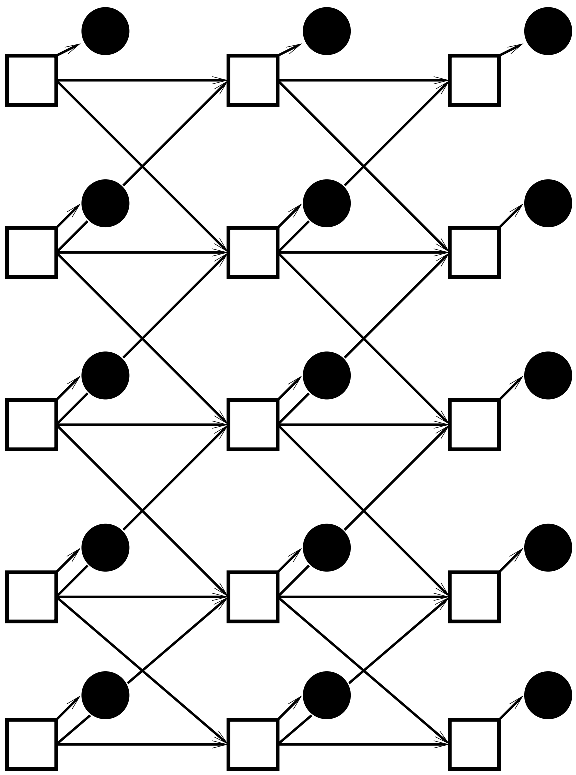
Factorial HMMs



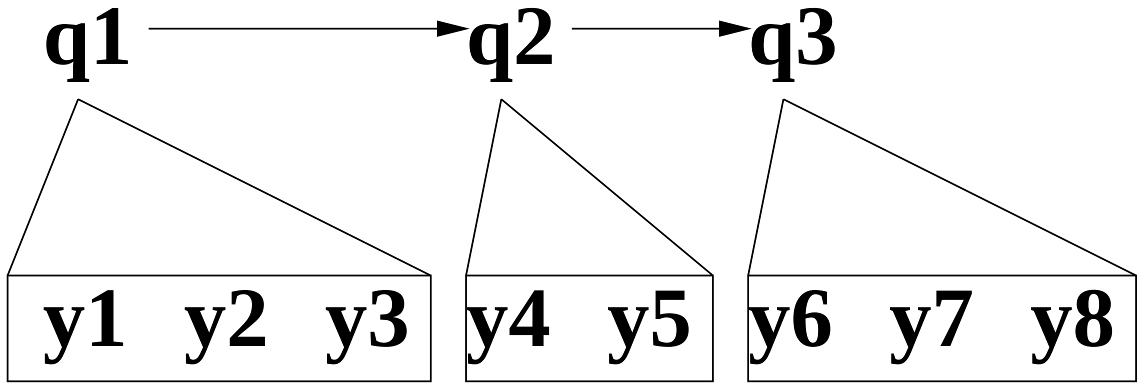
Factorial HMMs vs HMMs

- Let us compare a factorial HMM with D chains, each with K values, to its equivalent HMM.
- 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: $O(TDK^{D+1})$.

Coupled HMMs

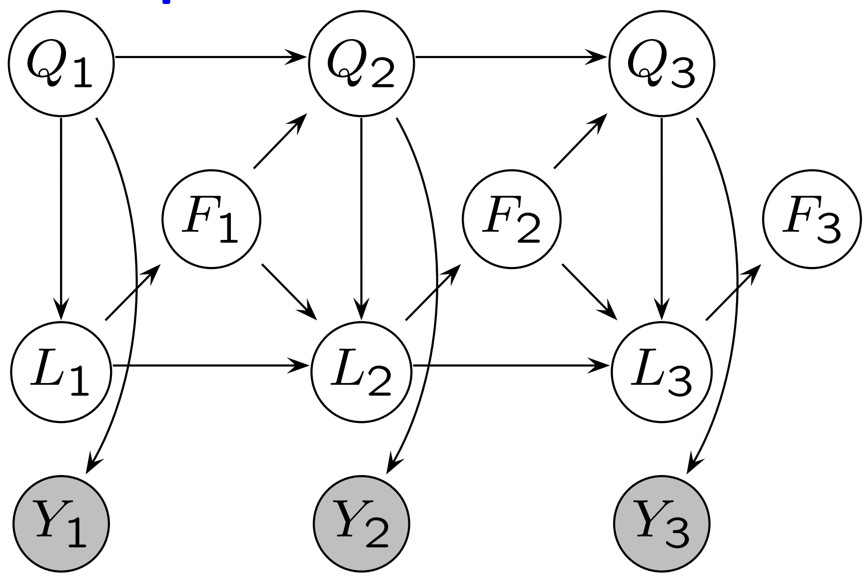


Semi-Markov HMMs

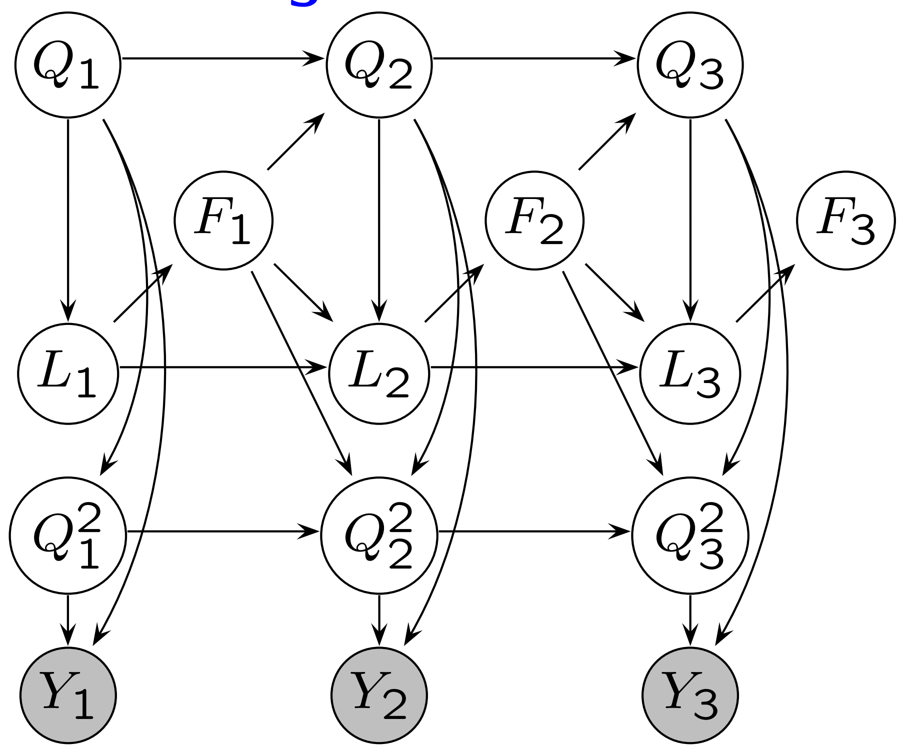


- Each state emits a sequence.
- Explicit-duration HMM:
  $P(Y_{t-l+1:l} | Q_t, L_t = l) = \prod_{i=1}^{l} P(Y_i | Q_t)$
- Segment HMM: $P(Y_{t-l+1:l} | Q_t, L_t = l)$ modelled by an HMM or SSM.
- Multigram: $P(Y_{t-l+1:l} | Q_t, L_t = l)$ is deterministic string, and segments are independent.

Explicit duration HMMs

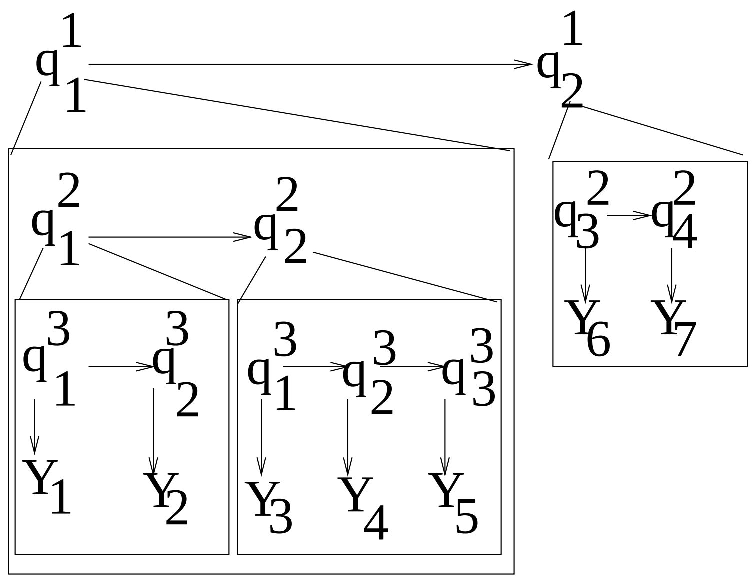


Segment HMMs

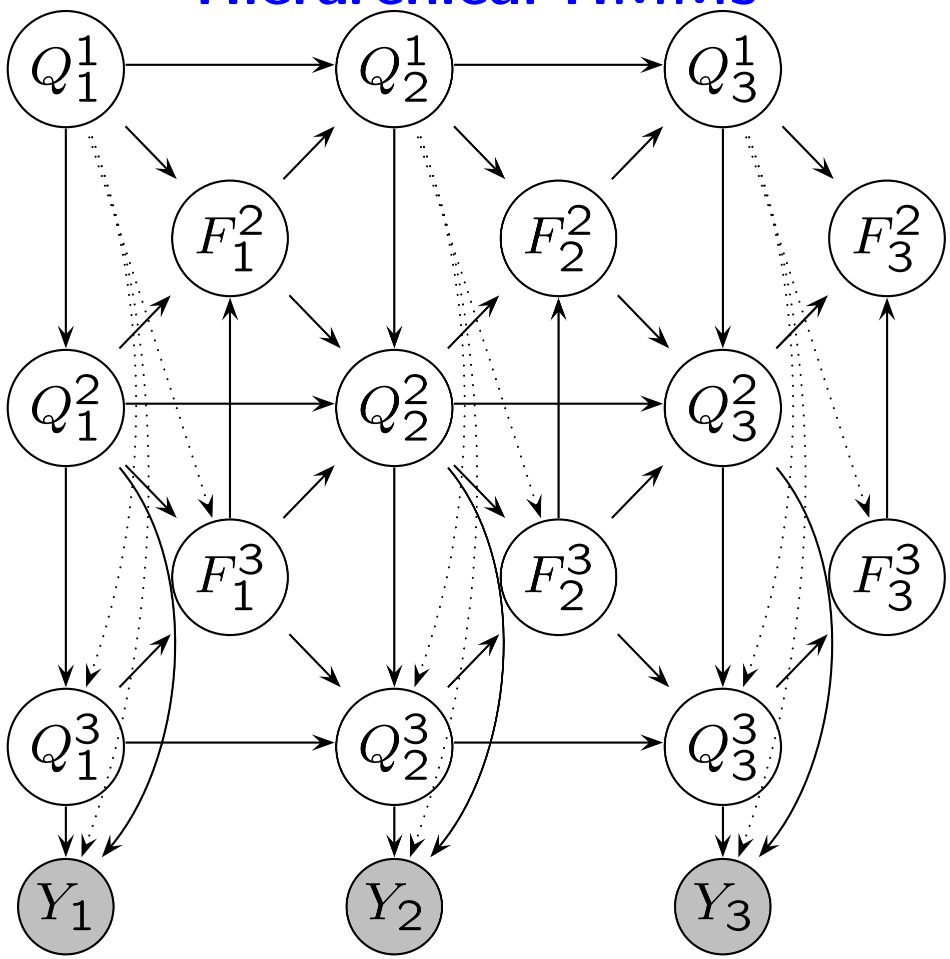


Hierarchical HMMs

- Each state can emit an HMM, which can generate sequences.
- Duration of segments implicitly defined by when sub-HMM enters finish state.



Hierarchical HMMs



State Space Models (SSMs)

- Also known as linear dynamical system, dynamic linear model, Kalman filter model, etc.

- $X_t \in R^D$, $Y_t \in R^M$ and

$$\begin{aligned} P(X_t|X_{t-1}) &= \mathcal{N}(X_t; AX_{t-1}, Q) \\ P(Y_t|X_t) &= \mathcal{N}(Y_t; BX_t, R) \end{aligned}$$

- The Kalman filter can compute $P(X_t|y_{1:t})$ in $O(\min\{M^3, D^2\})$ operations per time step.

Factored linear-Gaussian models produce sparse matrices

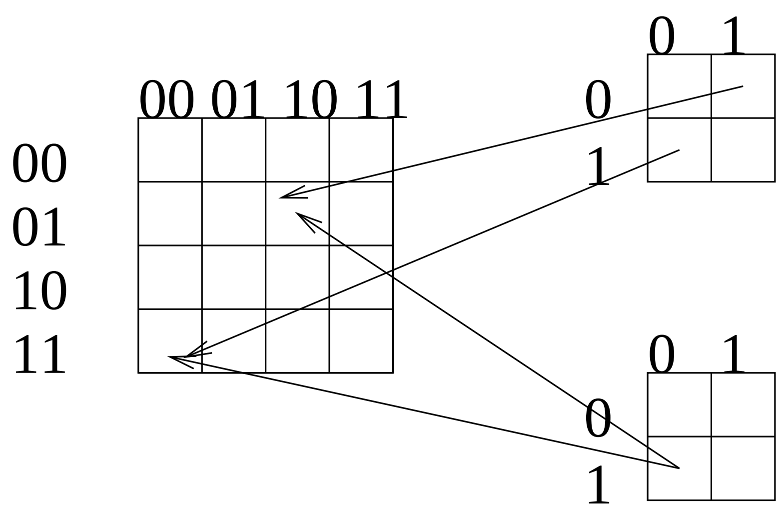
- Directed arc from $X_{t-1}(i)$ to $X_t(j)$ iff $A(i,j) > 0$.
- Undirected between $X_t(i)$ and $X_t(j)$ iff $\Sigma^{-1}(i,j) > 0$.
- e.g., consider a 2-chain factorial SSM
  with $P(X_t^i | X_{t-1}^i) = \mathcal{N}(X_t^i; A^i X_{t-1}, Q_i)$

$$P(X_t^1, X_t^1 | X_{t-1}^1, X_{t-1}^1) = \mathcal{N}\left(\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix}; \begin{pmatrix} A^1 & 0 \\ 0 & A^2 \end{pmatrix} \begin{pmatrix} X_{t-1}^1 \\ X_{t-1}^2 \end{pmatrix}, \begin{pmatrix} Q_1^{-1} & 0 \\ 0 & Q_2^{-1} \end{pmatrix} \right)$$

Factored discrete-state models do NOT produce sparse transition matrices

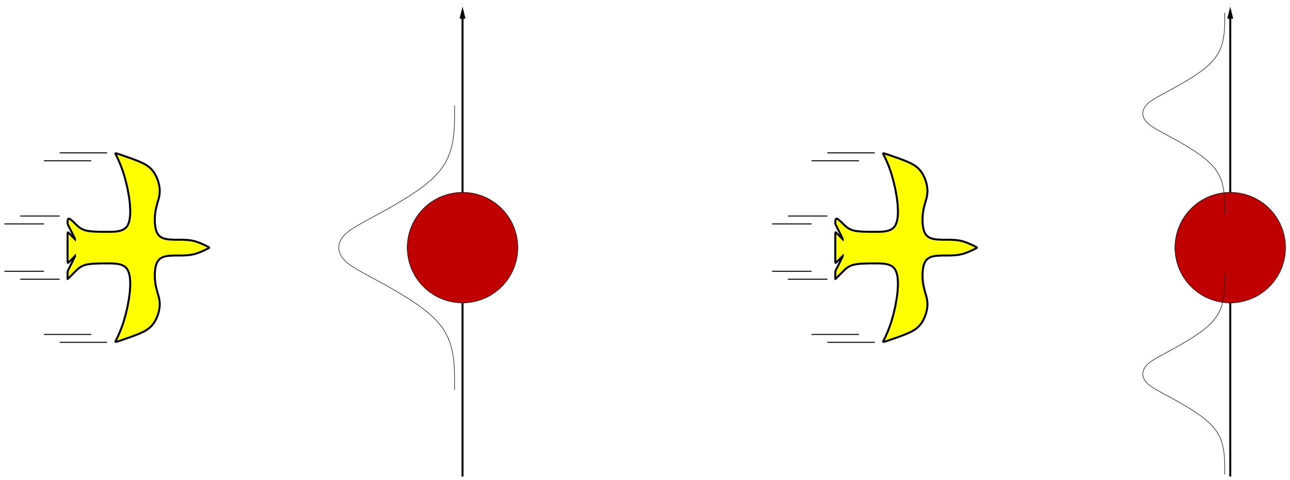
e.g., consider a 2-chain factorial HMM

$$P(X_t^1, X_t^1 | X_{t-1}^1, X_{t-1}^1) = P(X_t^1 | X_{t-1}^1) P(X_t^2 | X_{t-1}^2)$$

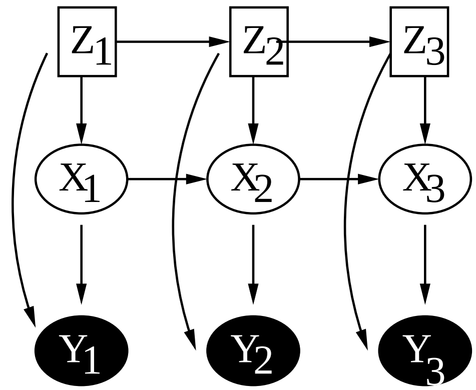


Problems with SSMs

- Non-linearity
- Non-Gaussianity
- Multi-modality



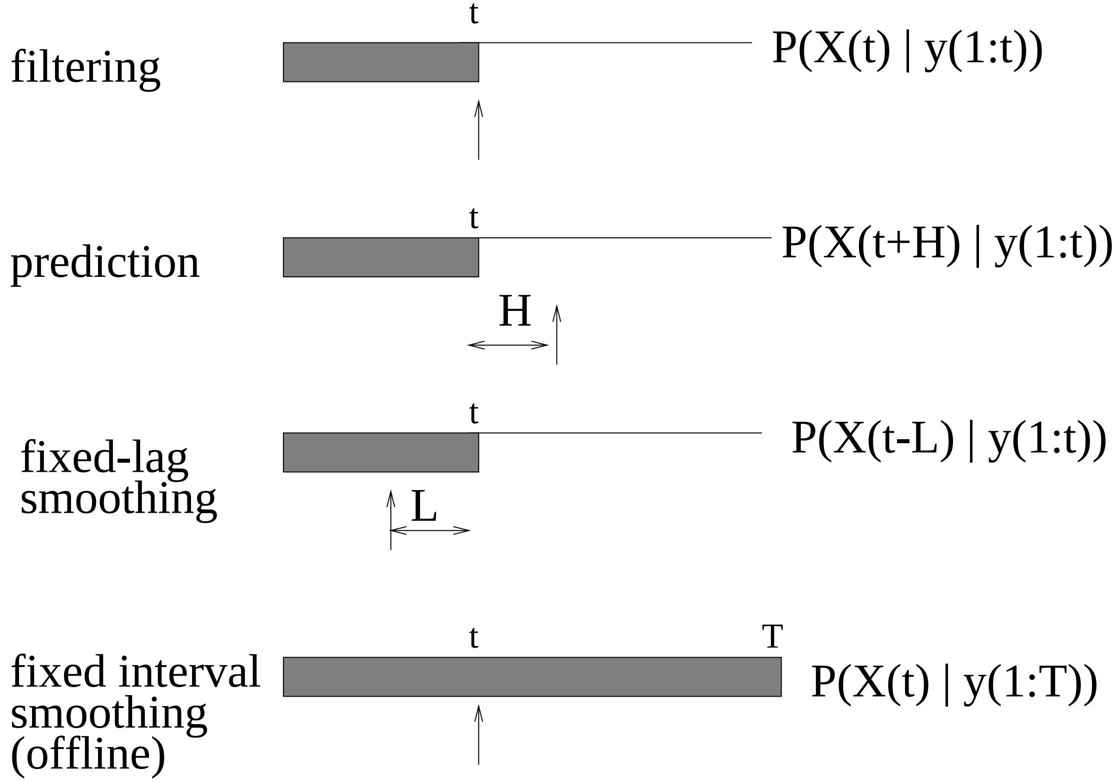
Switching SSMs



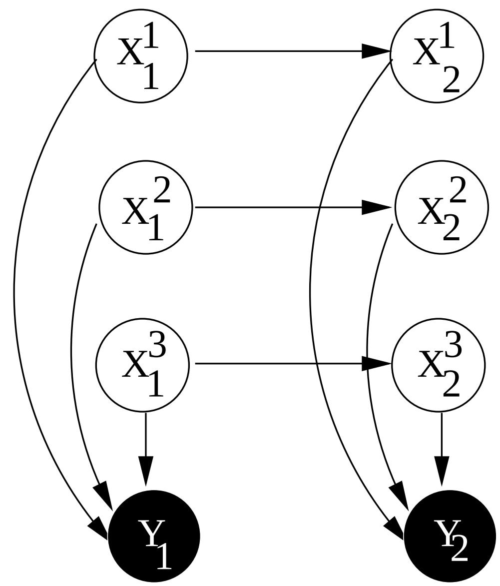
$$
\begin{aligned}
P(X_t|X_{t-1}, Z_t = j) &= \mathcal{N}(X_t; A_j X_{t-1}, Q_j) \\
P(Y_t|X_t, Z_t = j) &= \mathcal{N}(Y_t; B_j X_t, R_j) \\
P(Z_t = j|Z_{t-1} = i) &= M(i, j)
\end{aligned}
$$

- Useful for modelling multiple (linear) regimes/modes, fault diagnosis, data association ambiguity, etc.
- Unfortunately number of modes in posterior grows exponentially, i.e., exact inference takes $O(K^t)$ time.

Kinds of inference for DBNs

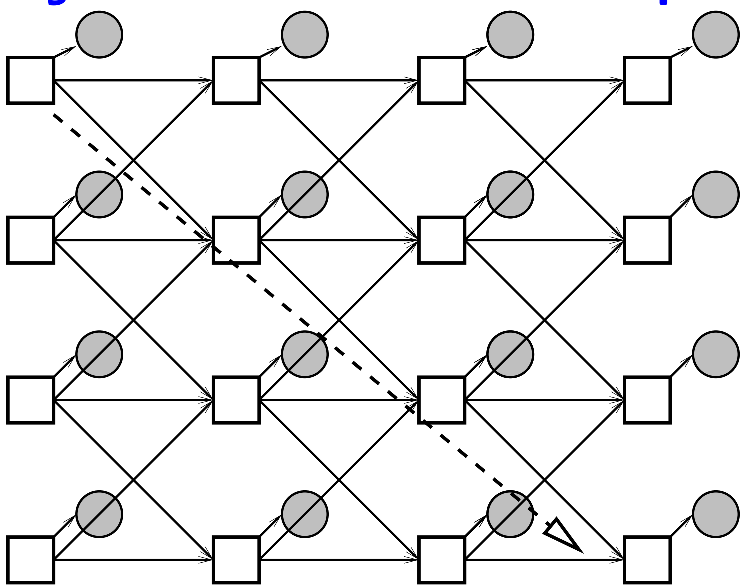


Complexity of inference in factorial HMMs



- $X_t^{(1)}, \ldots, X_t^{(D)}$ become corrrelated due to "explaining away".
- Hence belief state $P(X_t|y_{1:t})$ has size $O(K^D)$.

Complexity of inference in coupled HMMs



- Even with local connectivity, everything becomes correlated due to shared common influences in the past. c.f., MRF.

Approximate filtering

- Many possible representations for belief state, $\alpha_t \triangleq P(X_t|y_{1:t})$:
- Discrete distribution (histogram)
- Gaussian
- Mixture of Gaussians
- Set of samples (particles)

Belief state = discrete distribution

- Discrete distribution is non-parametric (flexible), but intractable.
- Only consider k most probable values — Beam search.
- Approximate joint as product of factors
  (ADF/BK approximation)

$$\alpha_t \approx \tilde{\alpha}_t = \prod_{i=1}^{C} P(X_t^i | y_{1:t})$$

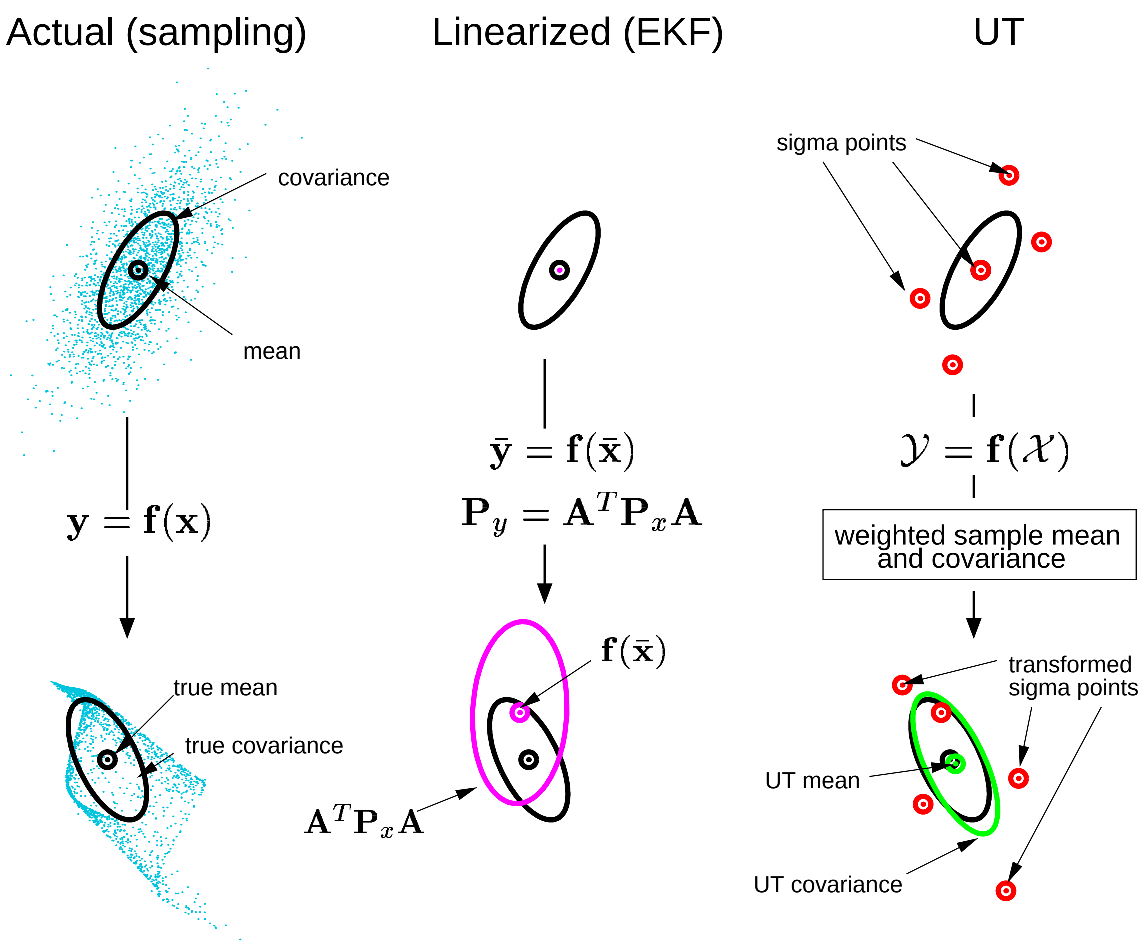
Assumed Density Filtering (ADF)

$$
\begin{tikzcd}
 & \widehat{\alpha}_t \arrow[d, "P"'] & \widehat{\alpha}_{t+1} \arrow[d, "P"'] & \text{exact} \\
\widetilde{\alpha}_{t-1} \arrow[ur, "U"] & \widetilde{\alpha}_t \arrow[ur, "U"] & \widetilde{\alpha}_{t+1} & \text{approx}
\end{tikzcd}
$$

Belief state = Gaussian distribution

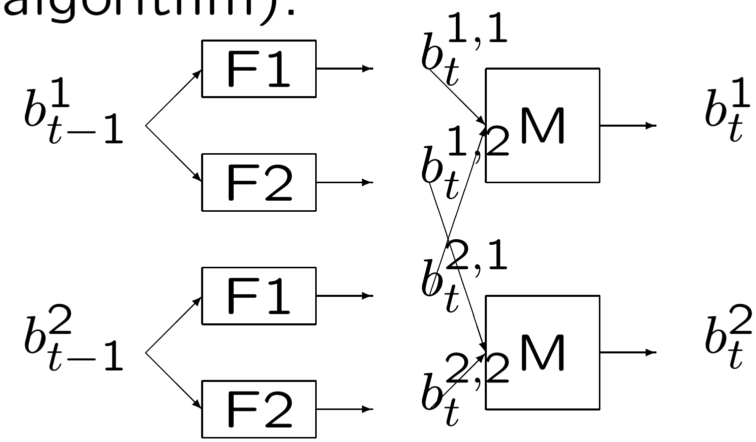
- Kalman filter — exact for SSM.
- Extended Kalman filter — linearize dynamics.
- Unscented Kalman filter — pipe mean $\pm$ sigma points through nonlinearity, and fit Gaussian.

Unscented transform



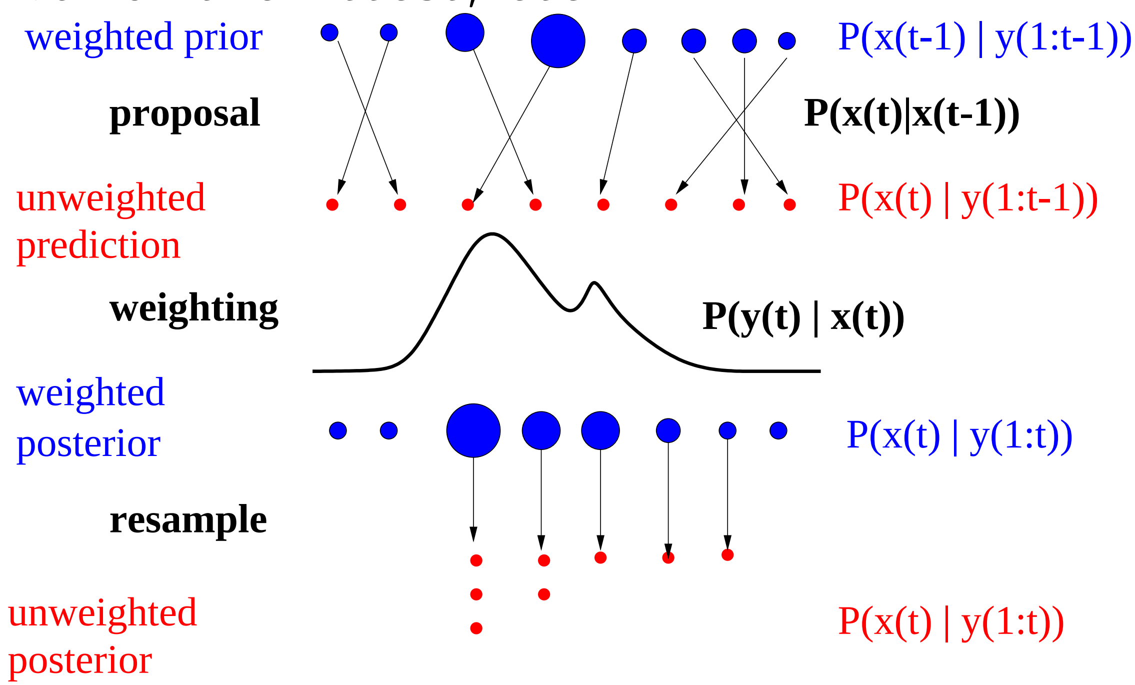
Belief state = mixture of Gaussians

- Hard in general.
- For switching SSMs, can apply ADF: collapse mixture of K Gaussians to best single Gaussian by moment matching (GPB/IMM algorithm).



Belief state = set of samples

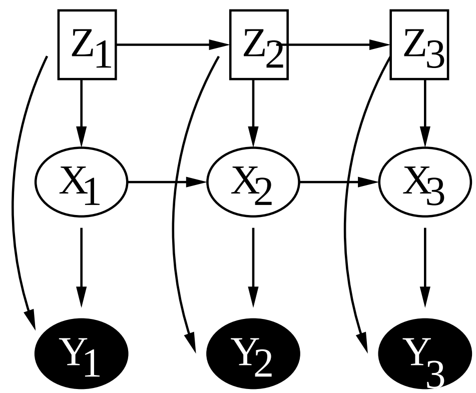
Particle filtering, sequential Monte Carlo, condensation, SISR, survival of the fittest, etc.



Rao-Blackwellised particle filtering (RBPF)

- Particle filtering in high dimensional spaces has high variance.
- Suppose we partition $X_t = (U_t, V_t)$.
- If $V_{1:t}$ can be integrated out analytically, conditional on $U_{1:t}$ and $Y_{1:t}$, we only need to sample $U_{1:t}$.
- Integrating out $V_{1:t}$ reduces the size of the state space, and provably reduces the number of particles needed to achieve a given variance.

RBPF for switching SSMs



- Given $Z_{1:t}$, we can use a Kalman filter to compute $P(X_t|y_{1:t}, z_{1:t})$.
- Each particle represents $(w, z_{1:t}, E[X_t|y_{1:t}, z_{1:t}], \text{Var}[X_t|y_{1:t}, z_{1:t}])$.
- c.f., stochastic bank of Kalman filters.

Approximate smoothing (offline)

- Two-filter smoothing
- Loopy belief propagation
- Variational methods
- Gibbs sampling
- Can combine exact and approximate methods
- Used as a subroutine for learning

Learning (frequentist)

- Parameter learning

$$\widehat{\theta}_{MAP} = \arg\max_{\theta} \log P(\theta|D,M) = \arg\max_{\theta} \log(D|\theta,M) + \log P(\theta|M)$$

where

$$\log P(D|\theta,M) = \sum_{d} \log P(X_d|\theta,M)$$

- Structure learning

$$\widehat{M}_{MAP} = \arg\max_{M} \log P(M|D) = \arg\max_{M} \log P(D|M) + \log P(M)$$

where

$$\log P(D|M) \;=\; \log \int P(D|\theta,M)P(\theta|M)P(M)d\theta$$

Parameter learning: full observability

- 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: partial observability

- 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.

Structure learning (model selection)

- How many nodes?
- Which arcs?
- How many values (states) per node?
- How many levels in the hierarchical HMM?
- Which parameter tying pattern?
- Structural zeros:
  - In a (generalized) linear model, zeros correspond to absent directed arcs (feature selection).
  - In an HMM, zeros correspond to impossible transitions.

Structure learning (model selection)

- Basic approach: search and score.
- Scoring function is marginal likelihood, or an approximation such as penalized likelihood or cross-validated likelihood

$$
\begin{aligned}
\log P(D|M) &= \log \int P(D|\theta, M) P(\theta|M) P(M) d\theta \\
&\overset{BIC}{\approx} \log P(D|\hat{\theta}_{ML}, M) - \frac{dim(M)}{2} \log |D|
\end{aligned}
$$

- Search algorithms: bottom up, top down, middle out.
- Initialization very important.
- Avoiding local minima very important.

Summary

- Representation
  - What are DBNs, and what can we use them for?
- Inference
  - How do we compute $P(X_t|y_{1:t})$ and related quantities?
- Learning
  - How do we estimate parameters and model structure?

Open problems

- Representing richer models, e.g., relational models, SCFGs.
- Efficient inference in large discrete models.
- Inference in models with non-linear, non-Gaussian CPDs.
- Online inference in models with variable-sized state-spaces, e.g., tracking objects and their relations.
- Parameter learning for undirected and chain graph models.
- Structure learning. Discriminative learning.
  Bayesian learning. Online learning. Active learning. etc.

The end