Lecture 18:

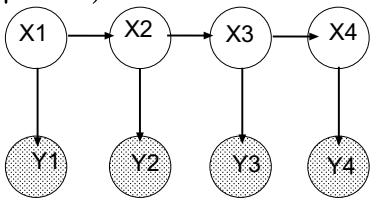
Sequential state estimation in nonlinear, non Gaussian dynamical systems

Kevin Murphy

22 November 2004

Sequential Bayesian updating

- A generic state-space model defines the dynamics $P(X_t|X_{t-1})$ and the observation model $P(y_t|X_t)$.



- Online inference (filtering) means recursively computing the belief state $P(X_t|y_{1:t})$ using Bayes' rule:

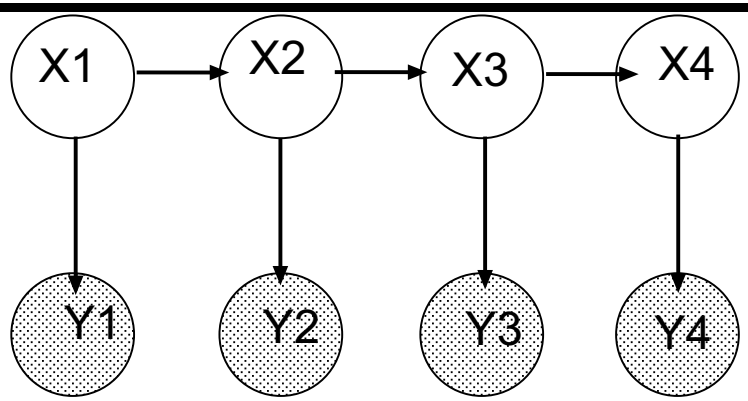
$$P(X_t|y_{1:t}) = \frac{1}{P(y_t|y_{1:t-1})} P(y_t|X_t) \int P(X_t|x_{t-1}) P(x_{t-1}|y_{1:t-1}) dx_{t-1}$$

- If $X_t = \theta$ is a constant, this can be used for recursive parameter estimation.
- There are many different methods, depending on how we represent the dynamical model, the observation model and the belief state.

Overview of filtering methods

Dynamics	Observation	Belief	Method
Trans. mat	Any	Histogram	HMM/ forwards
DBN	DBN	Jtree	Thin Junction tree filter (TJTF)
DBN	DBN	Prod. histo	Boyen-Koller (BK) filter
Lin Gauss	Lin Gauss	Gauss	Kalman filter (KF)
NonLin Gauss	NonLin Gauss	Gauss	Extended Kalman filter (EKF)
NonLin Gauss	NonLin Gauss	Gauss	Unscented Kalman filter (UKF)
Any	Any	Gauss	Assumed Density Filtering (ADF)
Mix Lin Gauss	Mix Lin Gauss	Mix Gauss	Assumed Density Filtering (ADF)
Any	Any	Samples	Particle filtering

HMMs



- Represent belief state as a histogram: $P(X_t = i|y_{1:t}) = \alpha_t(i)$.
- Predict step:

$$P(X_{t+1} = j|y_{1:t}) = \sum_i P(X_{t+1} = j|X_t = i)P(X_t = i|y_{1:t})$$

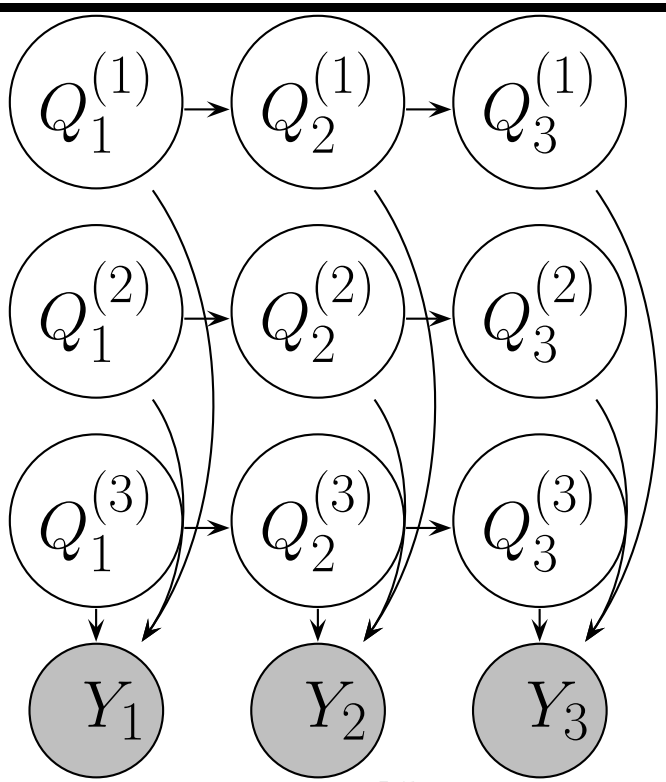
- Update step:

$$P(X_{t+1} = j|y_{1:t+1}) = \frac{p(y_{t+1}|X_{t+1} = j)P(X_{t+1} = j|y_{1:t})}{p(y_{t+1}|y_{1:t})}$$

Dynamic Bayesian Networks (DBN)

- A DBN generalizes a state space model by representing the components of X_t and Y_t and their conditional (in)dependencies using a graph.
- By factorizing the state space in this way, we can substantially reduce the number of free parameters.
- e.g., let X_t be a bit vector of length K. An HMM transition matrix would have $O(2^K \times 2^K)$ parameters. A DBN may have $O(K)$ parameters, depending on the structure.
- For linear Gaussian models, sparse graphs $\equiv$ sparse matrices, so DBNs are not needed as much (but are still helpful).

Factorial HMM

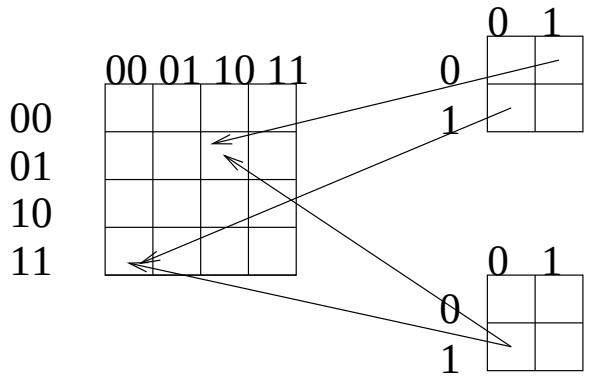


Belief state $P(X_t|y_{1:t})$ has size $O(2^N)$ for N binary chains, because the common observed child Y_t couples all the parents (explaining away).

Sparse graphs $\not\Rightarrow$ sparse discrete transition matrices

- Any discrete DBN can be flattened into an HMM.
- But the resulting transition matrix will not necessarily have 0s in it.

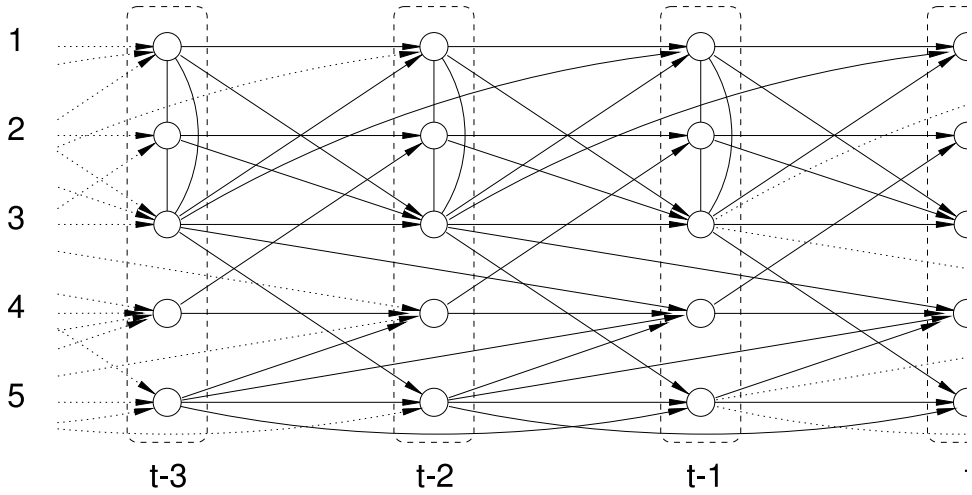
$$P(Q_t^{(1)} = j_1, Q_t^{(2)} = j_2 | Q_{t-1}^{(1)} = i_1, Q_{t-1}^{(2)} = i_2)$$
$$= P(Q_t^{(1)} = j_1 | Q_{t-1}^{(1)} = i_1) \times P(Q_t^{(2)} = j_2 | Q_{t-1}^{(2)} = i_2)$$



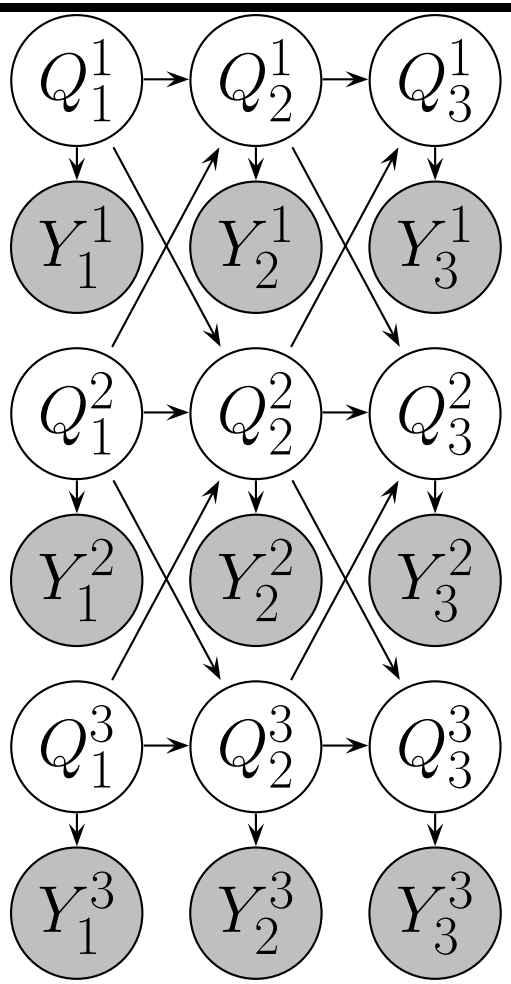
- For discrete state spaces, the graph structure provides a compact representation of the model in a way that cannot be easily captured in the form of the parameter matrices.

Linear Gaussian DBNs

- For linear Gaussian models, sparse graphs $\equiv$ sparse matrices, so DBNs are not needed as much (but are still helpful).
- Consider a Vector Auto Regressive process of order 2: $X_t = A_1 X_{t-1} + A_2 X_{t-2} + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \Sigma)$.
- If $A_k(i,j) = 0$, then the directed arc $X_{t-k}(i) \rightarrow X_t(j)$ is missing (for $k = 1, 2$).
- If $\Sigma^{-1}(i,j) = 0$, then the undirected arc $X_t(i) - X_t(j)$ is missing.

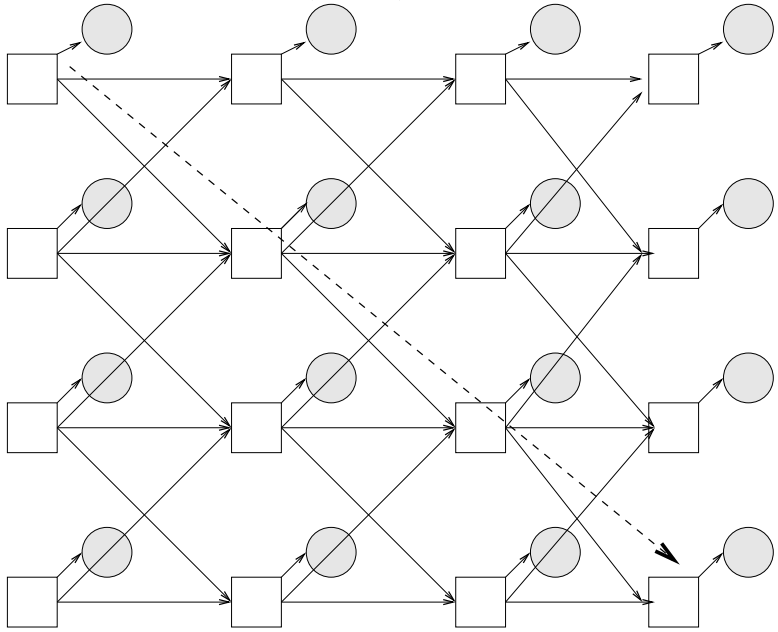


Coupled HMMs



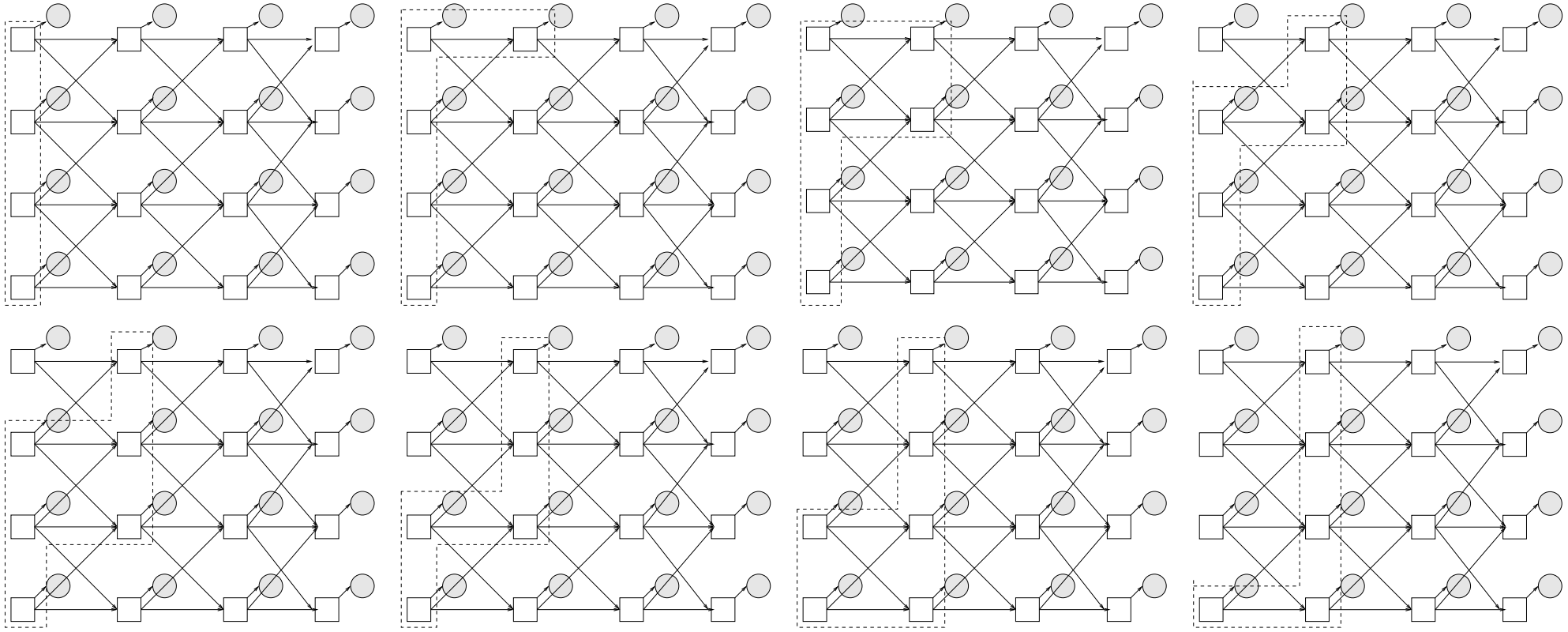
Belief state for coupled HMMs

Even though CHMMs are sparse, all nodes eventually become correlated, so $P(X_t|y_{1:t})$ has size $O(2^N)$.



Junction tree for coupled HMMs

Cliques form a frontier that snakes from X_{t-1} to X_t.

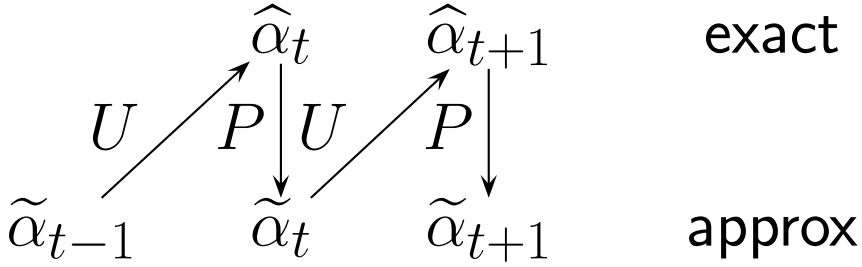


Assumed density filtering (ADF)

- ADF forces the belief state to live in some restricted family $\mathcal{F}$, e.g., product of histograms, Gaussian.
- Given a prior $\tilde{\alpha}_{t-1} \in \mathcal{F}$, do one step of exact Bayesian updating to get $\hat{\alpha}_t \notin \mathcal{F}$. Then do a projection step to find the closest approximation in the family:

$$\tilde{\alpha}_t = \arg\min_{q \in \mathcal{F}} D(\hat{\alpha}_t || q)$$

- If $\mathcal{F}$ is the exponential family, we can solve the KL minimization by moment matching.



Boyen-Koller (BK) algorithm

- The BK algorithm is ADF applied to a DBN.
- The simplest approximation is to let $\mathcal{F}$ be a product of marginals:

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

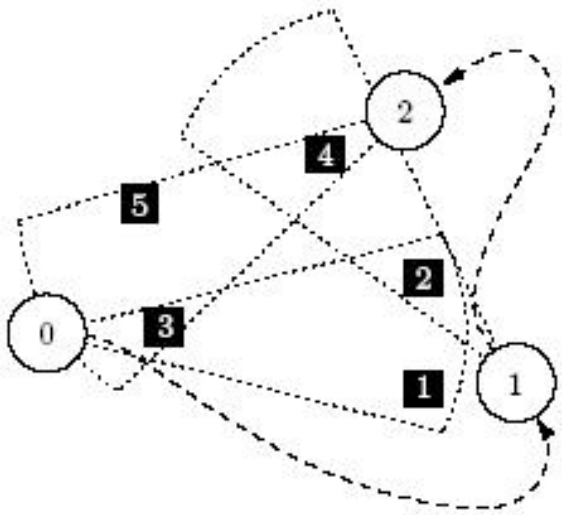
- We start with a prior that is fully factored, $\prod_i P(X_{t-1}^i | y_{1:t-1})$.
- We do one step of exact updating using a 2-slice junction tree. This will couple some nodes, but not as many as in the T-slice (unrolled) jtree.
- Then we compute the marginals $P(X_t^i | y_{1:t})$ (projection step).

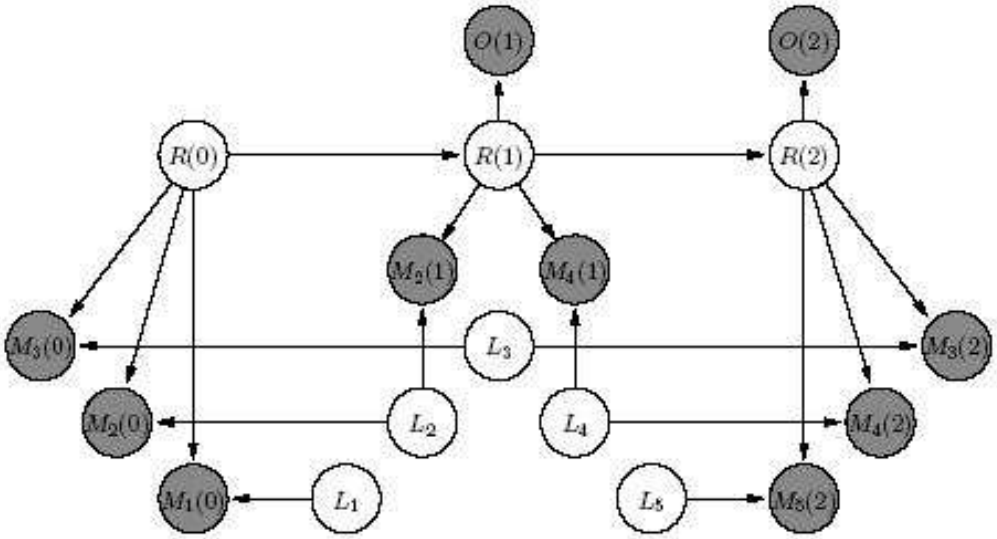
Thin junction tree filter (TJTF)

- The BK algorithm is ADF applied to a DBN.
- The approximating family $\mathcal{F}$ is chosen a priori.
- Above we considered products of marginals, but BK also considered products of overlapping cliques (i.e., junction tree).
- For some problems, the structure of the approximating family should be chosen dynamically, after seeing the evidence.
- TJTF dynamically thins the junction tree, to keep $\mathcal{F}$ tractable.
- This is useful e.g., for SLAM (simultaneous localization and mapping) problem in mobile robotics.

SLAM (simultaneous localization and mapping)

- State is location of robot and landmarks $X_t = (R_t, L_t^{1:N})$
- Measure location of subset of landmarks at each time step.
- Assume everything is linear Gaussian.
- Use Kalman filter to solve optimally.





Review: Kalman filter

- LDS model: $x_t = Ax_{t-1} + v_t, \;\; y_t = Cx_t + w_t$
- Time update (prediction step):

$$x_{t|t-1} = Ax_{t-1|t-1}, \;\; P_{t|t-1} = AP_{t-1|t-1}A^T + Q, \;\; y_{t|t-1} = Cx_{t|t-1}$$

- Measurement update (correction step):

$$\begin{aligned}
\tilde{y}_t &= y_t - \hat{y}_{t|t-1} \text{ (error/ innovation)} \\
P_{\tilde{y}_t} &= CP_{t|t-1}C^T + R \text{ (covariance of error)} \\
P_{x_t y_t} &= P_{t|t-1}C^T \text{ (cross covariance)} \\
K_t &= P_{x_t y_t} P_{\tilde{y}_t}^{-1} \text{ (Kalman gain matrix)} \\
x_{t|t} &= x_{t|t-1} + K_t(y_t - y_{t|t-1}) \\
P_{t|t} &= P_{t|t-1} - K_t P_{x_t y_t}^T
\end{aligned}$$

Complexity of one KF step

- Let $X_t \in \mathbb{R}^{N_x}$ and $Y_t \in \mathbb{R}^{N_y}$.
- Computing $P_{t|t-1} = AP_{t-1|t-1}A^T + Q$ takes $O(N_x^2)$ time, assuming dense P and dense A.
- Computing $K_t = P_{x_t y_t} P_{\tilde{y}_t}^{-1}$ takes $O(N_y^3)$ time.
- So overall time is, in general, $\max\{N_x^2, N_y^3\}$.

Information filter

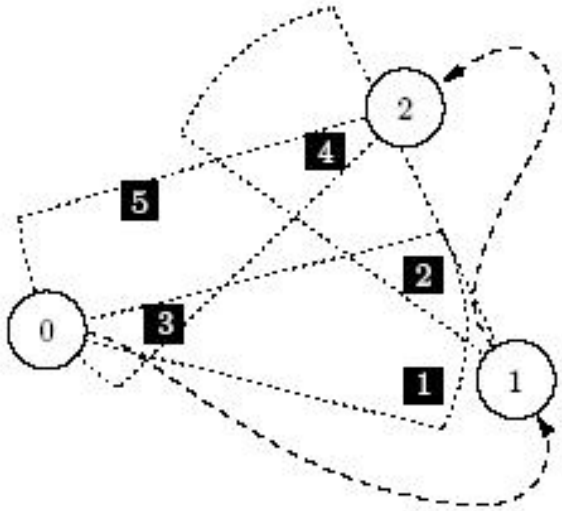
- KF uses moment form, ie. mean μ and covariance Σ.
- Initial uncertainty means covariance can be infinite $\Sigma_{ii} = \infty$.
- It is therefore common to use the *information form*, which works in terms of canonical (natural) parameters $S = \Sigma^{-1}$ and $\xi = \Sigma^{-1}\mu$.
- Substituting into KF equations and using the matrix inversion lemma yields:

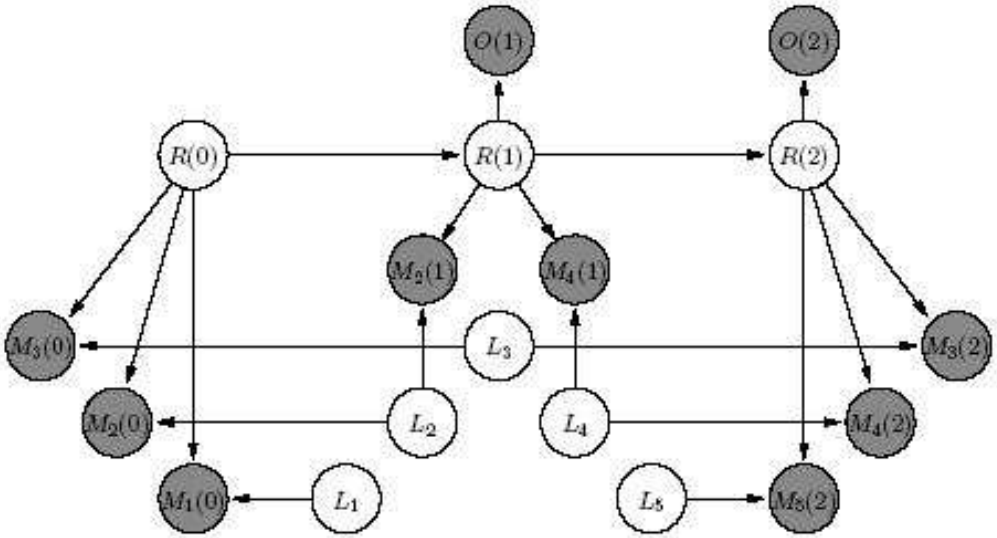
$$\begin{aligned} U_t &= (S_{t|t} + A^T Q A)^{-1} \\ \xi_{t+1|t+1} &= Q^{-1} A U \xi_{t|t} + C^T R^{-1} y_{t+1} \\ S_{t+1|t+1} &= Q^{-1} - Q^{-1} A U A^T Q^{-1} + C^T R^{-1} C \end{aligned}$$

- Now initial uncertainty means the precision is zero $\Sigma_{ii}^{-1} = 0$.
- But now complexity is $O(N_x^3)$.

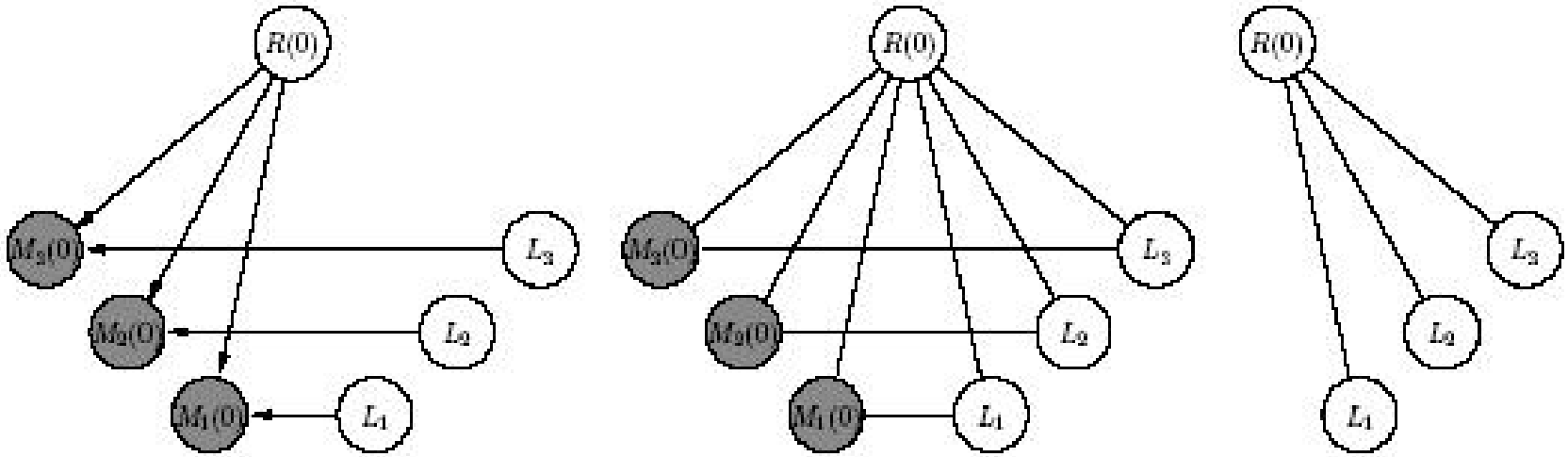
Review: KF for SLAM

- State is location of robot and landmarks $X_t = (R_t, L_t^{1:N})$
- Measure location of subset of landmarks at each time step.
- Assume everything is linear Gaussian.
- Use Kalman filter to solve optimally.





SLAM: Measurement (update) step



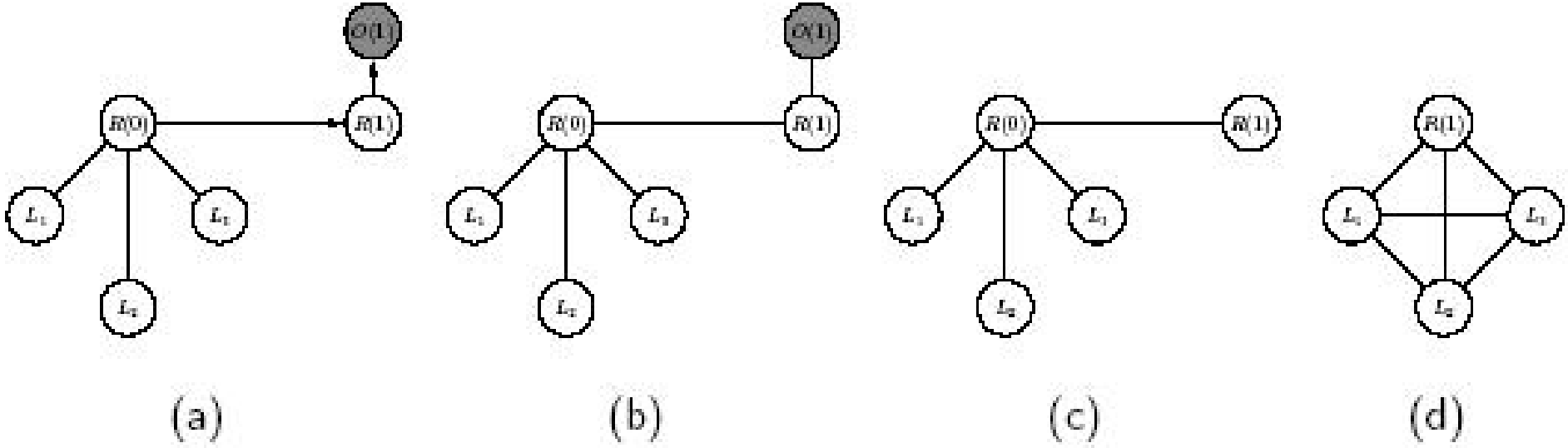
$P(X_t|y_{0:t}) = \mathcal{N}(\mu_t, \Sigma_t)$:

$$P(X_0) \qquad\qquad P(X_0|y_0)$$

$$\begin{pmatrix} & R & L^1 & L^2 & L^3 & L^4 & L^5 \\ R & \cdot & & & & & \\ L^1 & & \cdot & & & & \\ L^2 & & & \cdot & & & \\ L^3 & & & & \cdot & & \\ L^4 & & & & & \cdot & \\ L^5 & & & & & & \cdot \end{pmatrix}, \quad \begin{pmatrix} & R & L^1 & L^2 & L^3 & L^4 & L^5 \\ R & \cdot & \cdot & \cdot & \cdot & & \\ L^1 & \cdot & \cdot & & & & \\ L^2 & \cdot & & \cdot & & & \\ L^3 & \cdot & & & \cdot & & \\ L^4 & & & & & \cdot & \\ L^5 & & & & & & \cdot \end{pmatrix}$$

Observe, moralize, absorb evidence. Correlates R with active L.

SLAM: Motion (predict) step



Marginalizing out old robot position correlates all active landmarks with each other.

$$P(X_1|y_0)$$

$$\begin{array}{c} \\ R \\ L^1 \\ L^2 \\ L^3 \\ L^4 \\ L^5 \end{array} \begin{pmatrix} R & L^1 & L^2 & L^3 & L^4 & L^5 \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & & & & \cdot & \\ & & & & & \cdot \end{pmatrix}$$

Complexity of KF for SLAM

- In general, we have the following time complexities:
  - Computing $P_{t|t-1} = AP_{t-1|t-1}A^T + Q$ takes $O(N_x^2)$ time.
  - Computing $K_t = P_{x_t y_t} P_{\tilde{y}_t}^{-1}$ takes $O(N_y^3)$ time.
  - Computing $P_{t|t} = P_{t|t-1} - K_t P_{xy}^T$ takes $O(N_x^2)$ time.
- For SLAM, the landmarks are stationary, so only the RR and RL components of $P_{t|t-1}$ need be updated in $O(N_x)$ time.
- Unfortunately, computing $P_{t|t} = P_{t|t-1} - K_t P_{xy}^T$ takes $O(N_x^2)$ time, since $P_{t|t}$ becomes dense.
- Doesn't scale to many landmarks.

Thin junction tree filters for SLAM

- The precision matrix Σ^{-1} will eventually become non-zero everywhere (although edge strengths may be small).
- Hence the covariance matrix Σ becomes dense.
- Hence Kalman filter will take $O(N^2)$ time.
- Junction tree filtering is exact, and hence has the same complexity as KF.
- However, we can adaptively reduce the size of "fat" cliques; this is similar to pruning weak edges in the graphical model (i.e., set elements of Σ^{-1} to 0).
- This yields an $O(N)$ algorithm.
- Further approximations yield an $O(1)$ algorithm.

Nonlinear systems

- In robotics and other problems, the motion model and the observation model are often *nonlinear*:

$$x_t = f(x_{t-1}) + v_t, \;\; y_t = g(x_t) + w_t$$

- An optimal closed form solution to the filtering problem is no longer possible.
- The nonlinear functions f and g are sometimes represented by *neural networks* (multi-layer perceptrons or radial basis function networks).
- The parameters of f,g may be learned offline using EM, where we do gradient descent (back propagation) in the M step, c.f. learning a MRF/CRF with hidden nodes.
- Or we may learn the parameters online by adding them to the state space: $\tilde{x}_t = (x_t, \theta)$. This makes the problem even more nonlinear.

Extended Kalman Filter (EKF)

- The basic idea of the EKF is to linearize f and g using a second order Taylor expansion, and then apply the standard KF.
- i.e., we approximate a stationary nonlinear system with a non-stationary linear system.

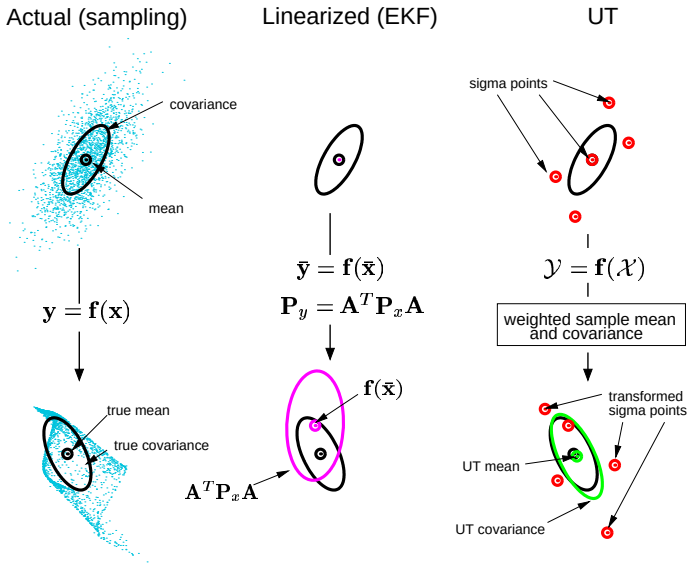
$$
\begin{aligned}
x_t &= f(\hat{x}_{t-1|t-1}) + A_{\hat{x}_{t|t-1}}(x_{t-1} - \hat{x}_{t-1|t-1}) + v_t \\
y_t &= g(\hat{x}_{t|t-1}) + C_{\hat{x}_{t|t-1}}(x_t - \hat{x}_{t|t-1}) + w_t
\end{aligned}
$$

  where $\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1})$ and $A_{\hat{x}} \stackrel{\text{def}}{=} \left.\frac{\partial f}{\partial x}\right|_{\hat{x}}$ and $C_{\hat{x}} \stackrel{\text{def}}{=} \left.\frac{\partial g}{\partial x}\right|_{\hat{x}}$.

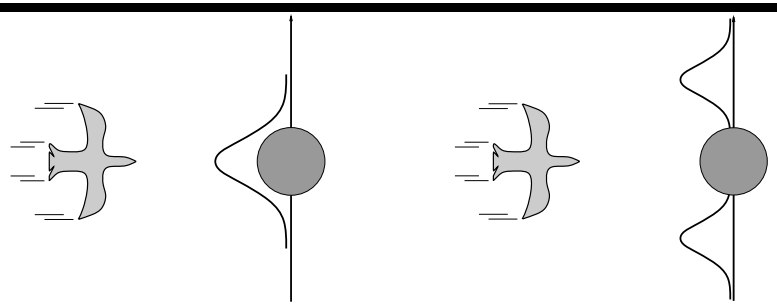
- The noise covariance (Q and R) is not changed, i.e., the additional error due to linearization is not modeled.

Unscented Kalman Filter (UKF)

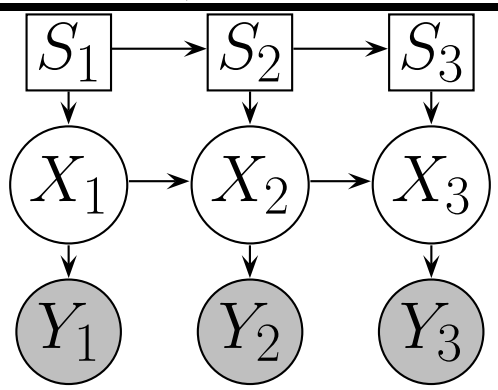
- UKF does not require computing derivatives of f or g, and is accurate to second order.
- The UKF applies the unscented transform twice, once to compute $P(X_t|y_{1:t-1})$ and once to compute $P(X_t|y_{1:t})$.
- The unscented transform passes the mean $\pm$ σ in each dimension, and fits an ellipse to the resulting points.



The need for multimodal belief states



Switching LDS (jump linear system)



Combination of HMM and LDS.

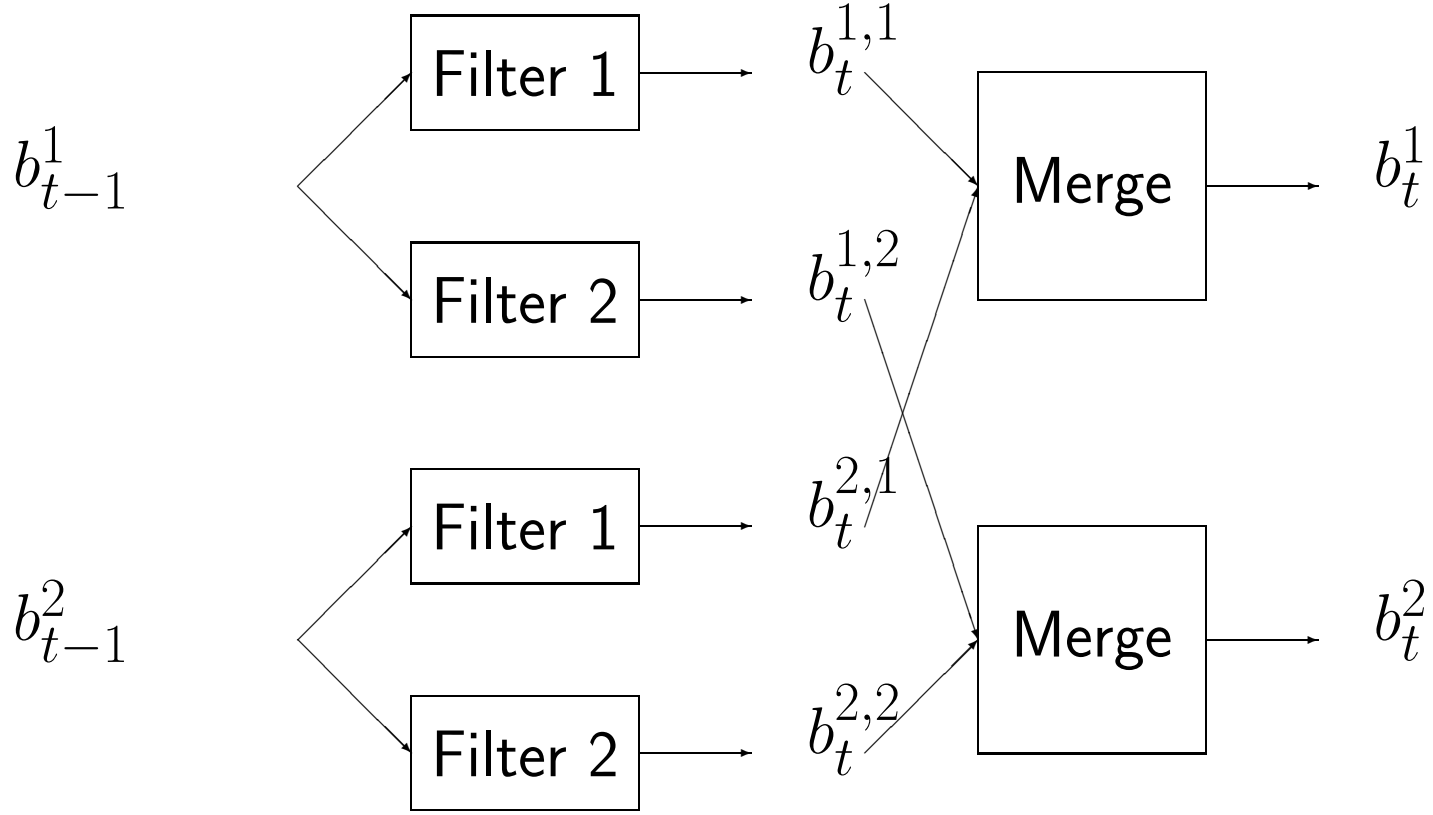
$$\begin{aligned} P(X_t = x_t | X_{t-1} = x_{t-1}, S_t = i) &= \mathcal{N}(x_t; A_i x_{t-1}, Q_i) \\ P(Y_t = y | X_t = x) &= \mathcal{N}(y; Cx, R) \\ P(S_t = j | S_{t-1} = i) &= M(i, j) \end{aligned}$$

Belief state has $O(2^t)$ Gaussian modes:

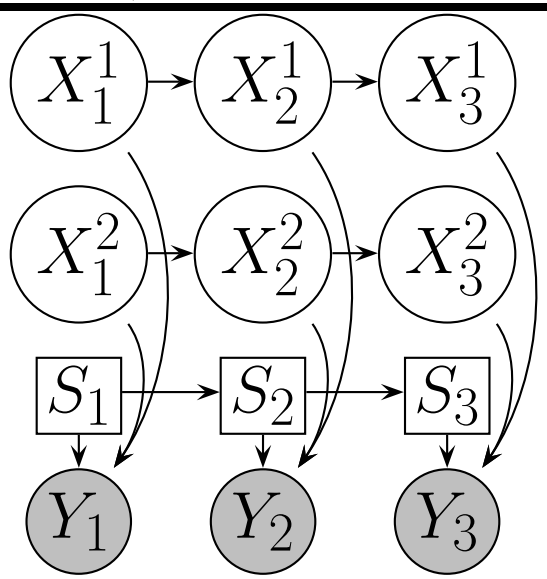
$$P(X_t, S_t | y_{1:t}) = \sum_{s_{1:t-1}} \int dx_{1:t-1} P(X_{1:t}, S_{1:t} | y_{1:t})$$

GPB2 for switching LDS

Generalized pseudo Bayesian = ADF per mode of S_t.



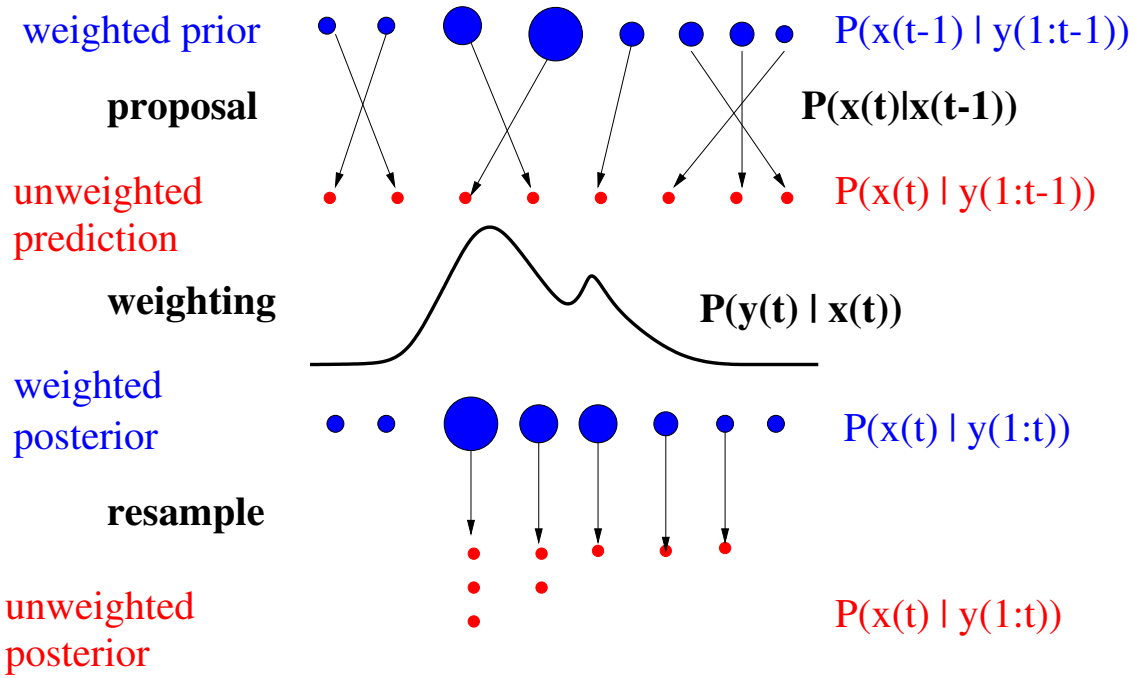
Data association (correspondence problem)



- Optimal belief state has $O(2^t)$ modes.
- Common to use nearest neighbor approximation.
- For each time slice, can enforce that at most one source causes each observation (maximal matching problem, solvable in $O(N^3)$ time using Hungarian algorithm).
- Correspondence problem also arises in shape matching and stereo vision.

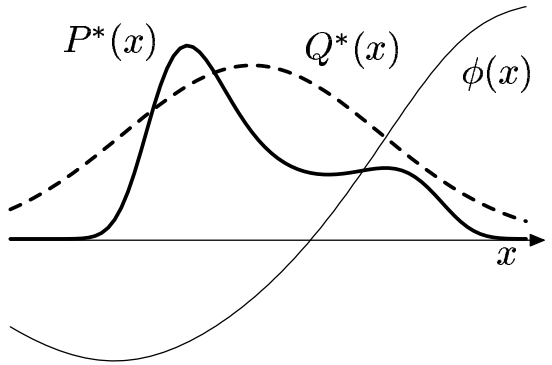
Particle filtering (sequential Monte Carlo)

- Represent belief state as weighted set of samples (non-parametric).
- Can handle nonlinear f/g and multi-modality.
- Easy to implement.
- Only works well in small dimensions.



Importance sampling

- Assume we want to compute $\Phi = E\phi(x) = \int d^N x P(x)\phi(x)$.
- Drawing samples from $P(x)$ may be too hard, but we can evaluate $P^*(x)$ where $P(x) = P^*(x)/Z_P$.
- Assume we can sample from a proposal density $Q(x)$ and can evaluate $Q^*(x)$, where $Q(x) = Q^*(x)/Z_Q$.
- We assign each sample an importance weight $w_r = P^*(x^r)/Q^*(x^r)$ and then approximate $\Phi = \frac{\sum_r w_r \phi(x^r)}{\sum_r w_r}$.
- Does not work well if $Q(x)$ is small where $|\phi(x)P^*(x)|$ is large; $Q(x)$ should have *heavy tails*.



Sequential Importance sampling

- Suppose the target density is $P(x_{1:t}|y_{1:t})$ and the proposal is $q(x_{1:t}|y_{1:t})$, so $w_t^i \propto P(x_{1:t}^i|y_{1:t})/Q(x_{1:t}^i|y_{1:t})$.
- The probability of a sample path can be computed recursively using Bayes' rule:

$$\begin{aligned} w_t^i &\propto \frac{P(y_t|x_t^i)P(x_t^i|x_{t-1}^i)P(x_{1:t-1}^i|y_{1:t-1})}{Q(x_t^i|x_{1:t-1}^i, y_{1:t})Q(x_{1:t-1}^i|y_{1:t-1})} \\ &= \frac{P(y_t|x_t^i)P(x_t^i|x_{t-1}^i)}{Q(x_t^i|x_{1:t-1}^i, y_{1:t})} w_{t-1}^i \\ &= \hat{w}_t^i w_{t-1}^i \end{aligned}$$

- For online problems, we typically use $Q(x_t|x_{1:t-1}^i, y_{1:t}) = Q(x_t|x_{t-1}^i, y_{1:t})$ so we don't have to store the entire history. Hence

$$\hat{w}_t^i = \frac{P(y_t|x_t^i)P(x_t^i|x_{t-1}^i)}{Q(x_t^i|x_{t-1}^i, y_{1:t})}$$

Sequential Importance sampling with resamling (SISR)

- As time increases, one sample path will turn out to be exponentially more likely than any other, so all the weights except one go to 0.
- This is called sample impoverishment.
- Whenever the effective number of samples $N_{eff} = 1/\sum_i (w_t^i)^2$ drops below a threshold, we resample with replacement.
- The resampled weights are set to $1/N$, since the past weights are reflected in the empirical frequency.
- There are various ways to do the resampling in $O(N)$ time.
- PF is the same as SISR.

Proposal distribution for PF

- The simplest proposal is to sample from the prior $Q(x_t|x_{t-1}^i, y_{1:t}) = P(X_t|x_{t-1}^i)$. In vision, this is called the condensation algorithm.
- Recall that the incremental weight is

$$\hat{w}_t^i = \frac{P(y_t|x_t^i)P(x_t^i|x_{t-1}^i)}{Q(x_t^i|x_{t-1}^i, y_{1:t})}$$

- So for condensation, $\hat{w}_t^i = P(y_t|x_t^i)$.
- It is better to look at the evidence before proposing:

$$q(x_t|x_{t-1}^i, y_t) = P(x_t|x_{t-1}^i, y_t) = \frac{P(y_t|x_t)P(x_t|x_{t-1}^i)}{\int dx_t P(y_t|x_t)P(x_t|x_{t-1}^i)}$$

- In this case, the incremental weight is the denominator $\hat{w}_t^i = P(y_t|x_{t-1}^i)$.

Unscented particle filtering

- Often it is too hard to compute the optimal proposal $P(X_t|x^i_{t-1}, y_{1:t})$ exactly.
- But sometimes we can approximate this.
- Consider a nonlinear system with Gaussian process noise and linear-Gaussian observations:

$$\begin{aligned} P(X_t|x^i_{t-1}) &= \mathcal{N}(X_t; f_t(x^i_{t-1}), Q_t) \\ P(Y_t|X_t) &= \mathcal{N}(y_t; C_t X_t, R_t) \end{aligned}$$

- Then we can compute $Q(X_t|x^i_{t-1}, y_{1:t})$ using an EKF/UKF (with a delta function prior on x^i_{t-1}), and sample from this.

Rao-Blackwellised PF (RBPF)

- Sampling in high dimensional spaces causes high variance in the estimate.
- RBPF idea: sample some variables R, and conditional on that, compute expected value of rest X analytically.
- So-called because of RB theorem, which is based on this identity:

$$\mathsf{Var}[\tau(X,R)] = \mathsf{Var}[E(\tau(X,R)|R)] + E[\mathsf{Var}(\tau(X,R)|R)]$$

- Hence $\mathsf{Var}[E(\tau(X,R)|R)] \leq \mathsf{Var}[\tau(X,R)]$, so $\tau'(X,R) = E(\tau(X,R)|R)$ is a lower variance estimator.

RBPF for SLAM ("FastSLAM")

- Key idea: if you always know the robot's location, the posterior over landmarks factorizes, so KF takes $O(N)$ time.
- So sample $R_{1:t}$, and for each particle/ trajectory, run a Kalman filter.