# Artificial Intelligence

## Time and Uncertainty

CS 444 – Spring 2019
Dr. Kevin Molloy
Department of Computer Science
James Madison University

# Time and Uncertainty

The world changes; we need to track and predict it

Examples: Diabetes management , vehicle diagnosis

**Basic idea**: copy state and evidence variables for each time step

$\mathbf{X}_t$ = set of unobservable state variables at time $t$

e.g. $BloodSugar_t$, $StomachContents_t$, etc.

$\mathbf{E}_t$ = set of observable evidence variables at time $t$

e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$

This assumes discrete time; step size depends on problem.

Notation: $X_{a:b} = X_a, X_{A+1}, \ldots, X_{b-1}, X_b$

# Markov processes (Markov chains)

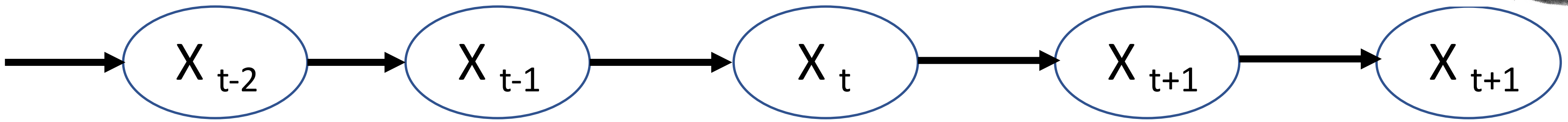
Construct a Bayes net from these variables: parents?

Markov assumption: $X_t$ depends on bounded subset of $X_{0:t-1}$

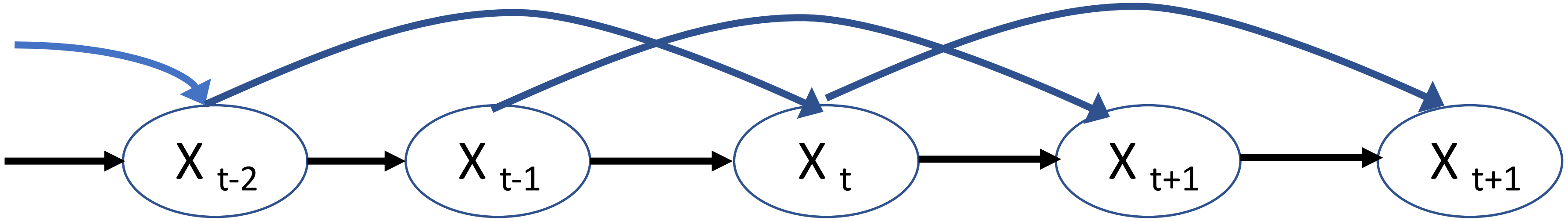
First-order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$

Second-order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1}, X_{t-2})$

First-order




Second-order

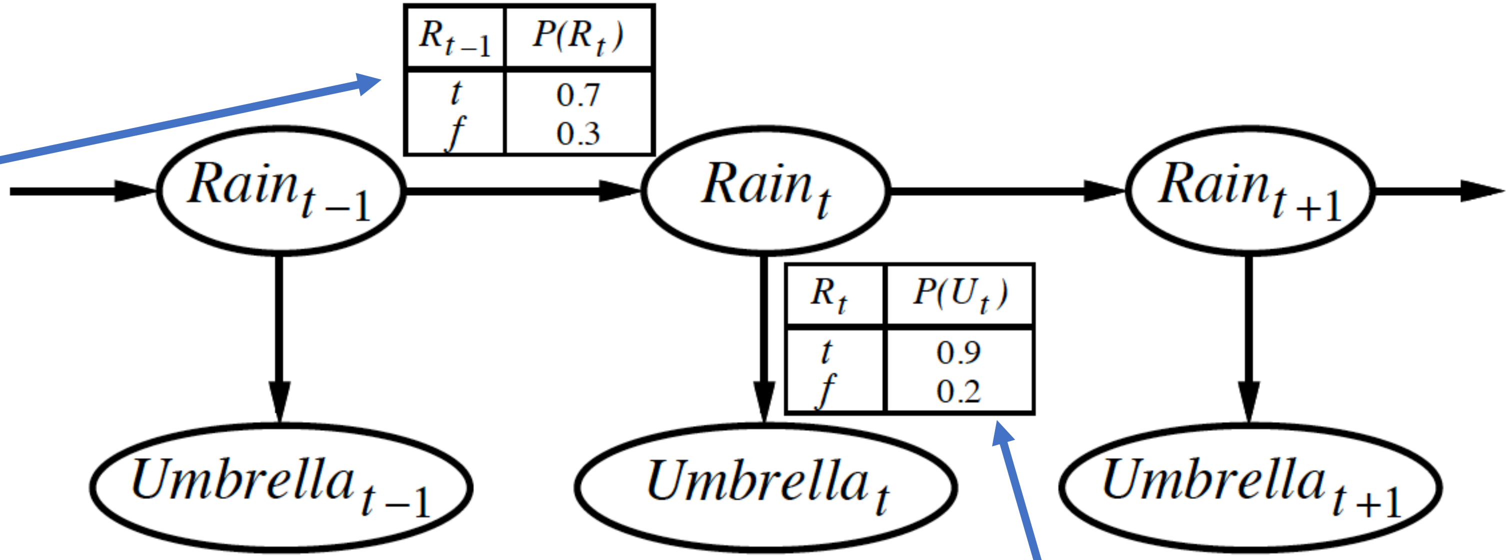



Sensor Markov assumption: $P(E_t \mid X_{0:t}, E_{0:t-1}) = P(E_t \mid X_t)$

Stationary process: transition model $P(X_t \mid X_{t-1})$ and sensor model $P(E_t \mid X_t)$ fixed for all *t*.

# Example

Transition Probabilities
$T_{(i,j)} = P(X_{k+1} = j \mid Z_k = i)$
$(i,j \in m)$. Called the transition or stochastic matrix




First-order Markov assumption not exactly true in the real world.

Possible fixes:

- **Increase order** of Markov process
- **Augment state**, e.g., Add temp, pressure, etc.

Example: robot motion:

Augment position and velocity with Battery

Emission probabilities (called the sensor model in the textbook)

# Inference tasks

Filtering: $P(X_t \mid e_{1:t})$

Belief state – input to the decision process of a rational agent

Prediction: $P(X_{t+k} \mid e_{1:t})$ for $k > 0$

Evaluation of possible action sequences; like filtering without the evidence

Smoothing: $P(X_k \mid e_{1:t})$ for $0 \leq k < t$

Better estimate of past states, essential for learning

Most likely explanation: $\text{ARGMAX}_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$

Speech recognition, decoding with a noisy channel

# Filtering

Goal: compute the belief state – the posterior distribution over the most recent state – given all the evidence seen to date.

Aim: devise a recursive state estimate algorithm:

$P(X_{t+1} \mid e_{1:t+1}) = f(e_{t+1}, P(X_t \mid e_{1:t}))$

$P(X_{t+1} \mid e_{1:t+1}) = P(X_{t+1} \mid e_{1:t}, e_{t+1})$ divide evidence variables

$= \alpha P(e_{t+1} \mid X_{t+1}, e_{1:t})\, P(X_{t+1} \mid e_{1:t})$ using Bayes' rule

$= \alpha P(e_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid e_{1:t})$ Markov assumption

i.e., prediction + estimation. Prediction by summing out and conditioning on $X_t$:

$P(X_{t+1} \mid e_{1:t+1}) = \alpha P(e_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t, e_{1:t}) P(x_t, e_{1:t})$

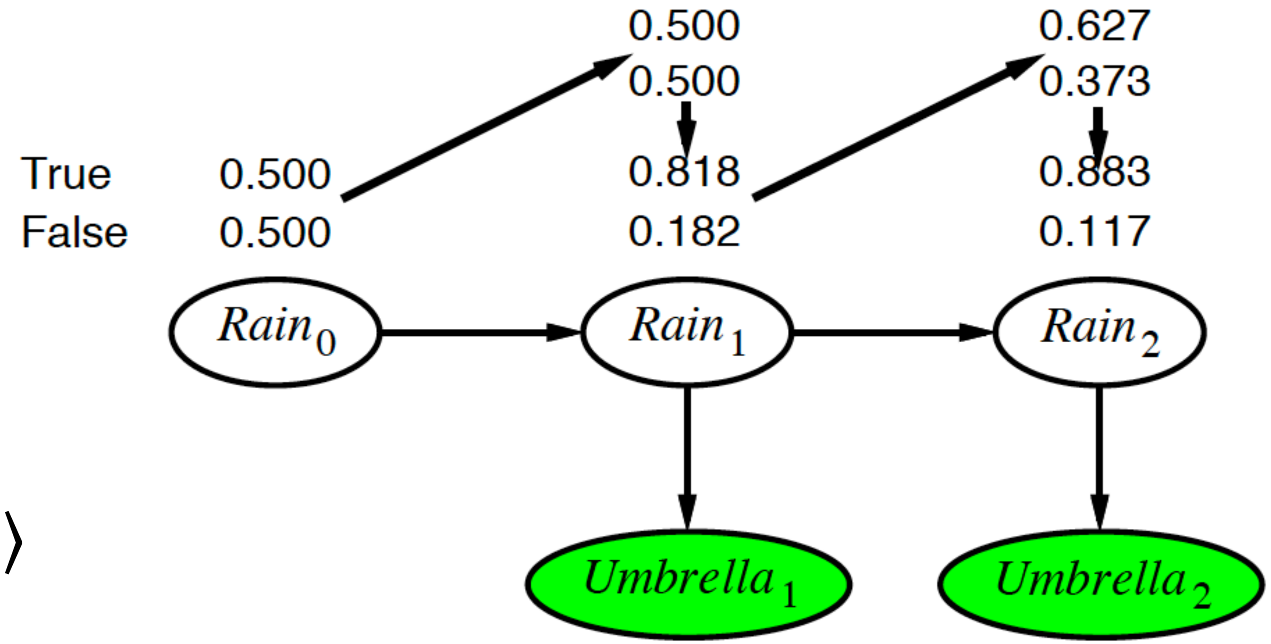
$= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t, e_{1:t})$

$f_{1:t+1} = \text{Forward}(f_{1:t}, e_{t+1})$ where $f_{1:t} = P(X_t \mid e_{1:t})$

Time and space constant (independent of *t*) !!!

# Filtering Example

Day 0: All we have are the beliefs (priors)




Day 1: Umbrella appears.

$P(R_1) = \sum_{r_o} P(R_1 | r_o) P(r_0)$

$= \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$

Update based on evidence (Umbrella)

$P(R_1 \mid u_1) = \alpha P(u_1, R_1) P(R_1) = \alpha \langle 0.9, 0.2 \rangle \times \langle 0.5, 0.5 \rangle = \alpha \langle .45, 0.1 \rangle \approx \langle 0.818, 0.182 \rangle$

Day 2: Umbrella appears.

$P(R_2 | u_1) = \sum_{r_1} P(R_2 | r_1) P(r_1 | u_1)$

$= \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 \approx \langle 0.627, 0.373 \rangle$

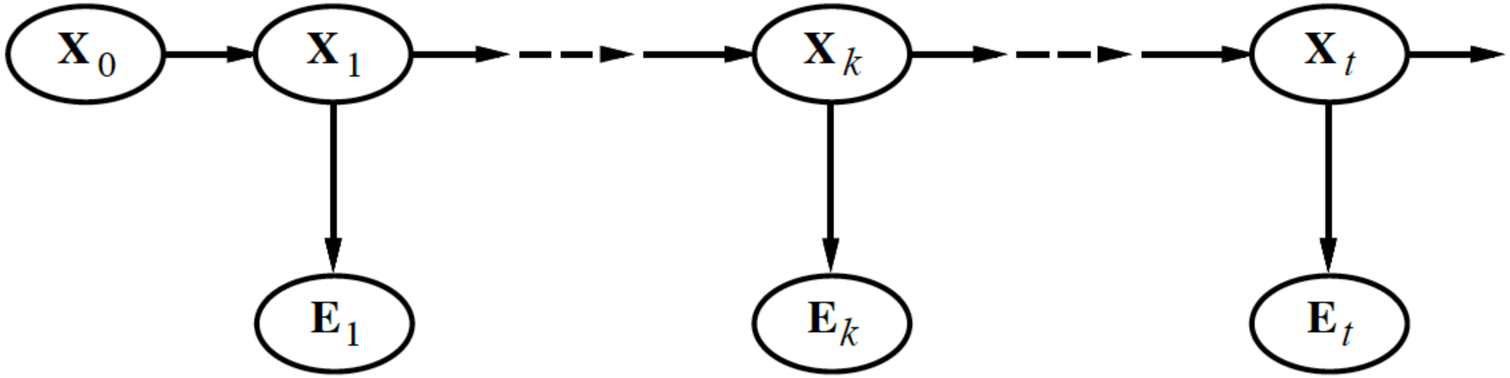
Update: $P(R_2 \mid u_1, u_2) = \alpha P(u_2 \mid R_2) P(R_2 \mid u_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.627, 0.373 \rangle$

$= \alpha \langle 0.565, -.0075 \rangle \approx \langle 0.883, 0.117 \rangle$

| $R_{t-1}$ | $P(R_t)$ |
|---|---|
| t | 0.7 |
| f | 0.3 |

| $R_t$ | $P(U_t)$ |
|---|---|
| t | 0.9 |
| f | 0.2 |

# Smoothing




Divide evidence $e_{1:t}$ into $e_{1:k}, e_{k+1:t}$

$P(X_k \mid e_{1:t}) = P(X_k \mid e_{1:k}, e_{k+1:t})$

$= \alpha P(X_k \mid e_{1:k}) P(e_{k+1:t} \mid X_k, e_{1:k})$

$= \alpha P(X_k \mid e_{1:k})\ P(e_{k+1:t} \mid X_k)$

$= \alpha P(X_k \mid e_{1:k})\ P(e_{k+1:t} \mid X_k)$

$= \alpha f_{1:k} b_{k+1:t}$

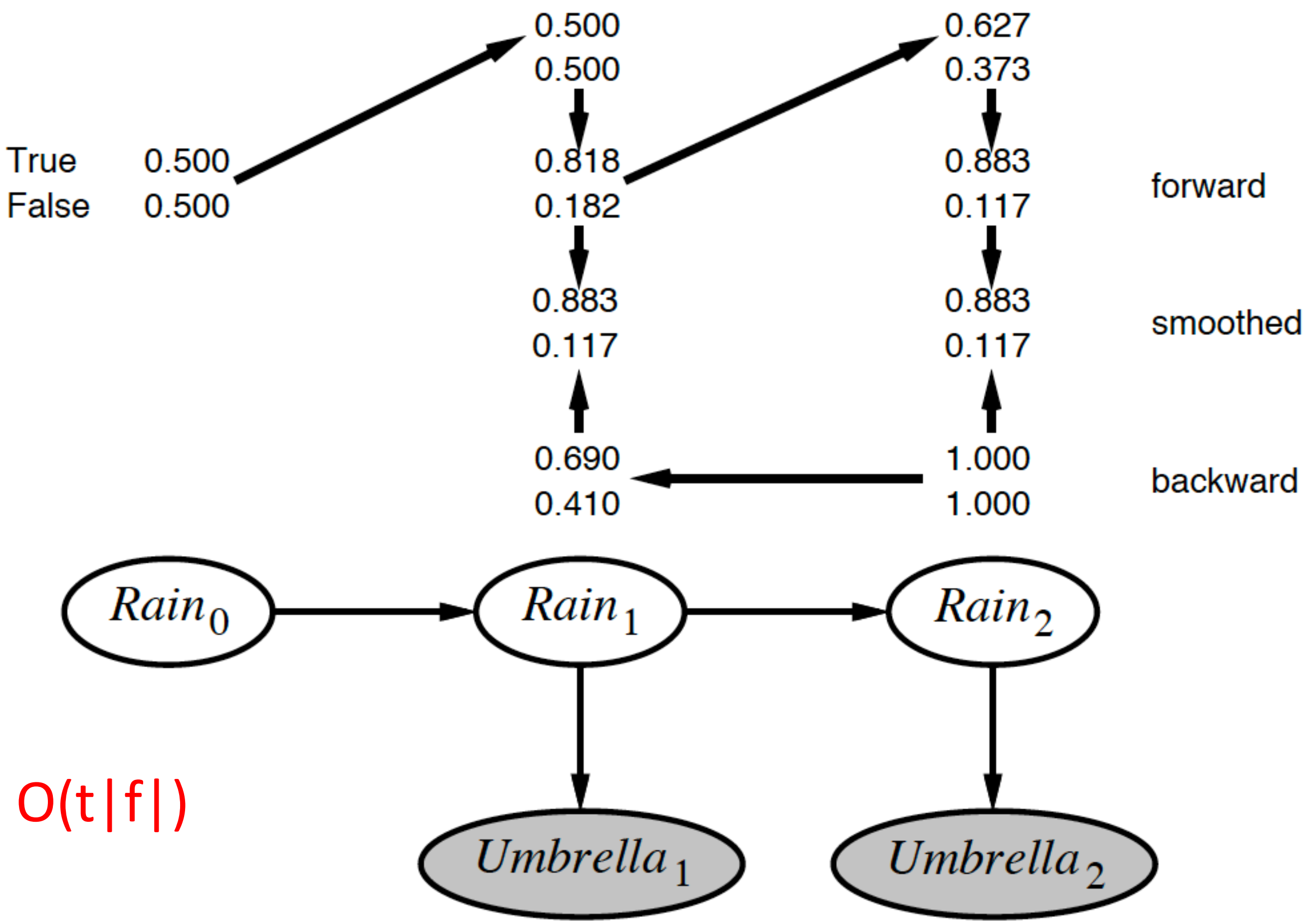
Backward message computed by a backwards recursion:

$P(e_{1+1:t} \mid X_k) = \sum_{x_{k+1}} P(e_{k+1:t} | X_k, X_{k+1})\ P(x_{k+1} | X_k)$

$= \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1})\ P(x_{k+1} | X_k)$

$= \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1})\ P(x_{k+2:t} | X_{k+1})\ P(x_{k+1} | X_k)$

# Smoothing Example




Forward-backward algorithm

Time linear in t (polytree inference) space is O(t|f|)

$P(R_1 \mid u_1, u_2) = \alpha P(R_1|u_1)P(u_2|R_1)$

$P(u_2 \mid R_1) = \sum_{r_2} P(u_2|r_2)\ P(r_2|R_1)$

$= (0.9 \times 1 \times \langle 0.7, 0.3\rangle) + (0.2 \times 1 \times \langle 0.3, 0.7\rangle = \langle 0.69, 0.41\rangle$

$P(R_1 \mid u_1, u_2) = \alpha\langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \approx \langle 0.883, 0.117\rangle$

# Most likely explanation

Most likely sequence ≠ sequence of most likely states!!!

Most likely path to each $x_{t+1}$ = most likely path to some $x_t$ plus one more step

$$\max_{x_1 \ldots x_t} P(x_1, \ldots, x_t, X_{t+1} | e_{1:t+1})$$
$$= P(e_{t+1}|X_{t+1}) \max_{X_t}(P(X_{t+1}|x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t | e_{1:t}))$$
$$= P(e_{t+1}|X_{t+1}) \left(\max_{X_t}(P(X_{t+1}|x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t | e_{1:t})\right)$$

Identical to filtering, except $f_{1:t}$ replaced by

$$m_{1:t} = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, X_t | e_{1:t})$$

i.e., $m_{1:t}(i)$ gives the probability of the most likely path to state i. Update has sum replaced by max, giving the Viterbi algorithm.

$$m_{1:t+1} = \mathrm{P}(e_{t+1}|X_{t+1}) \max_{x_t}(P(X_{t+1}|x_t) m_{1:t})$$

# Hidden Markov Model

$X_t$ is a single, discrete variable (usually $E_t$ is too). Domain of $X_t$ is {1, …., S}

Transition matrix $T_{ij} = P(X_t = j \mid X_{t-1} = i)$, e.g. $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

Sensor matrix $O_t$ for each time step, diagonal elements $P(e_t \mid X_t = i)$

e.g. With $U_1$ = true, $O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages as column vectors

$$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$
$$b_{k+1:t} = T O_{k+1} b_{k+2:t}$$

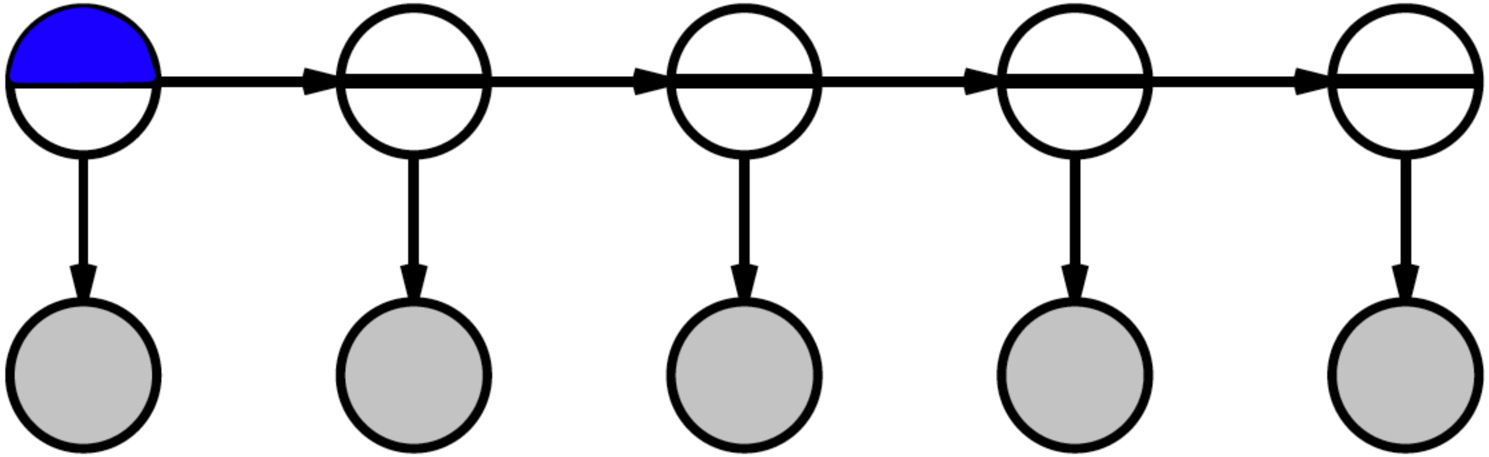
Forward-backward algorithm needs time $O(S^2t)$ and space $O(St)$

# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

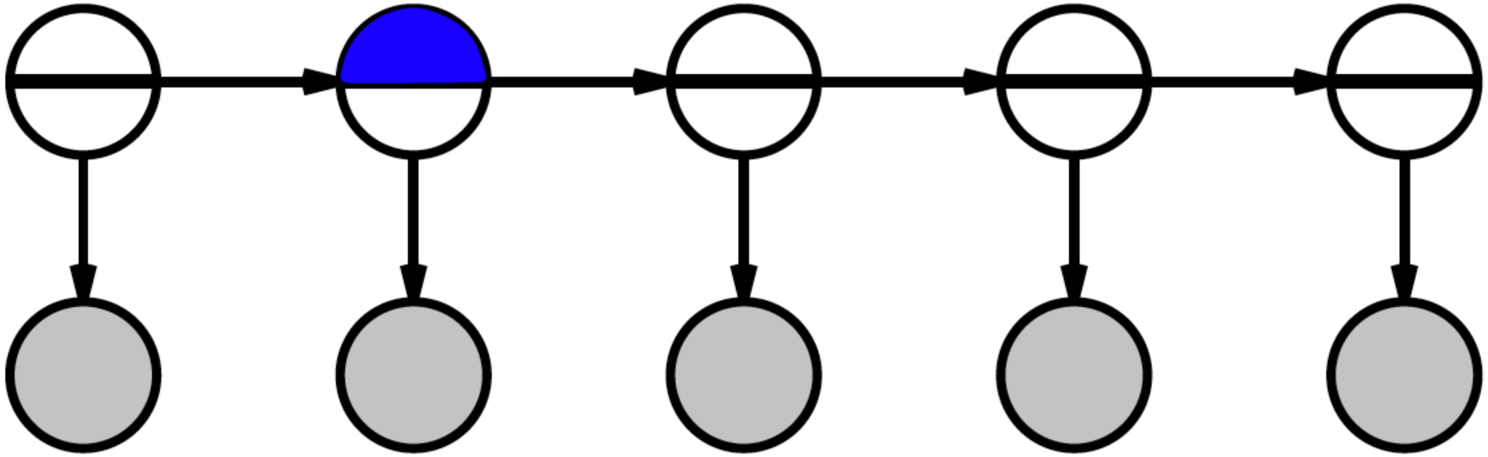


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$



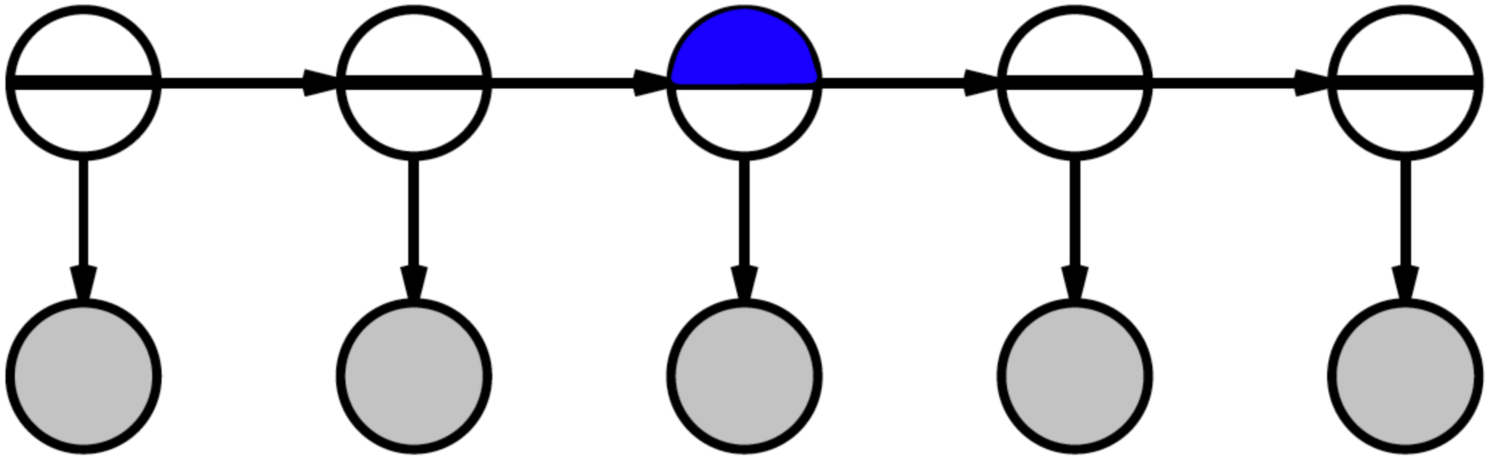
JAMES MADISON UNIVERSITY

# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

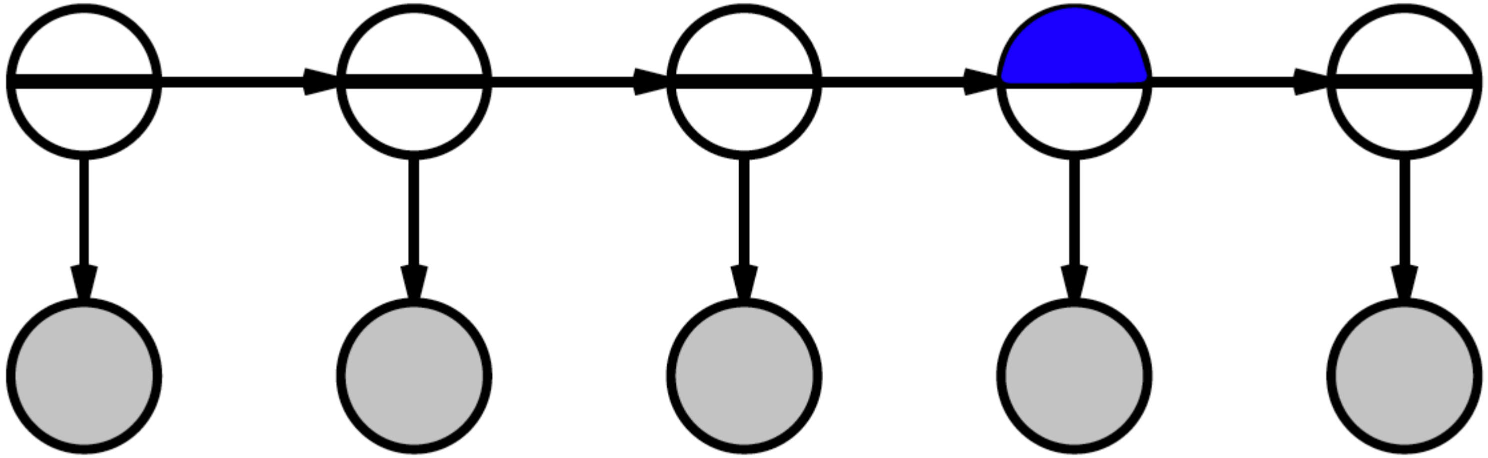


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

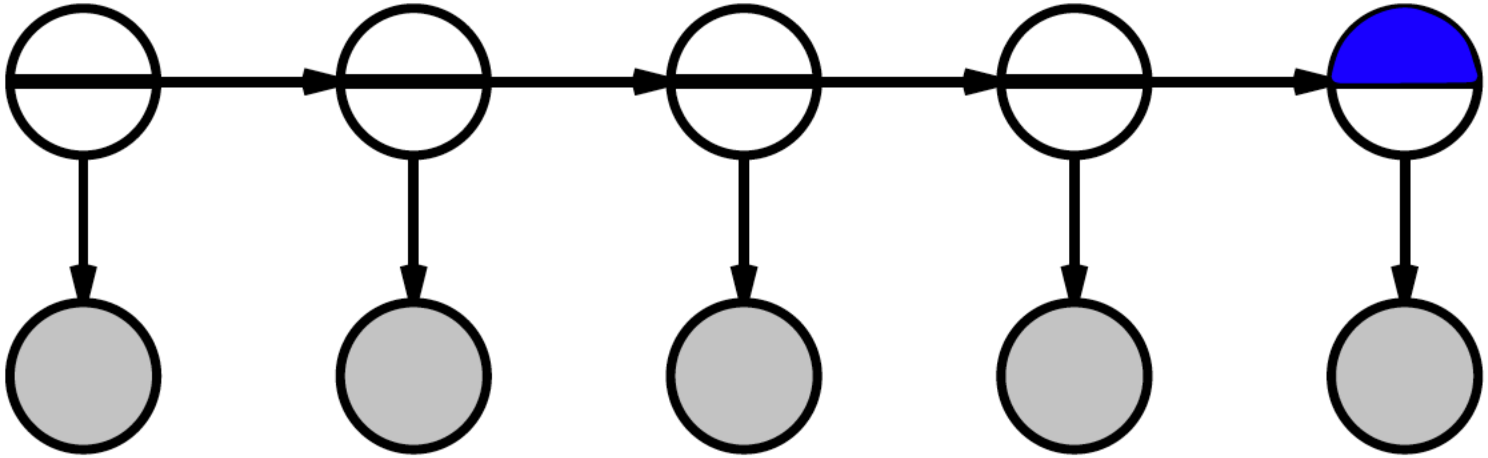


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

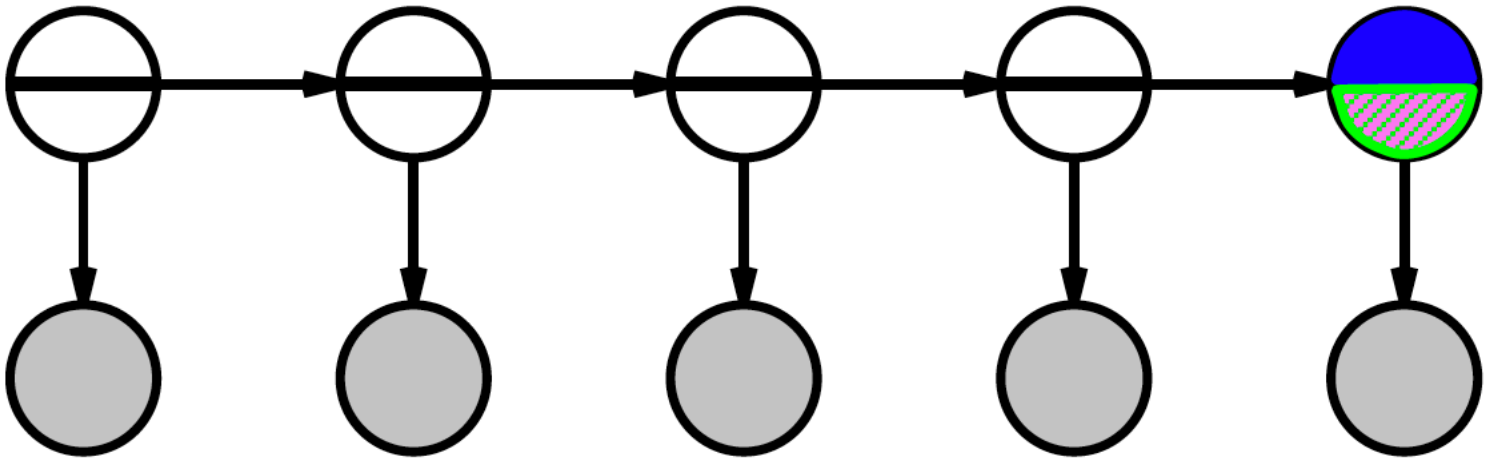


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

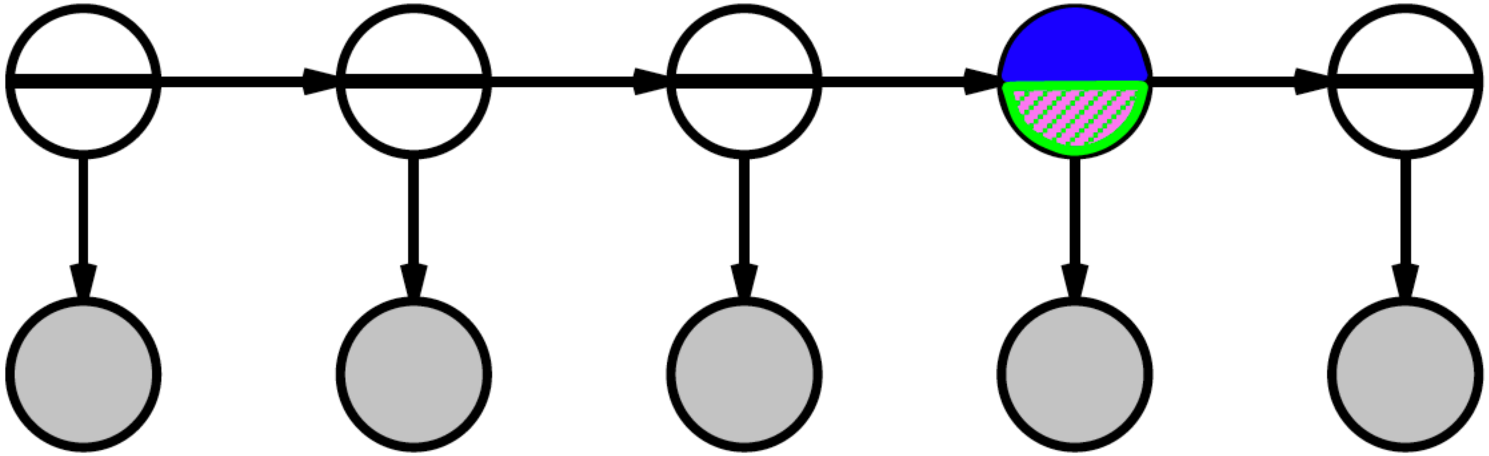


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

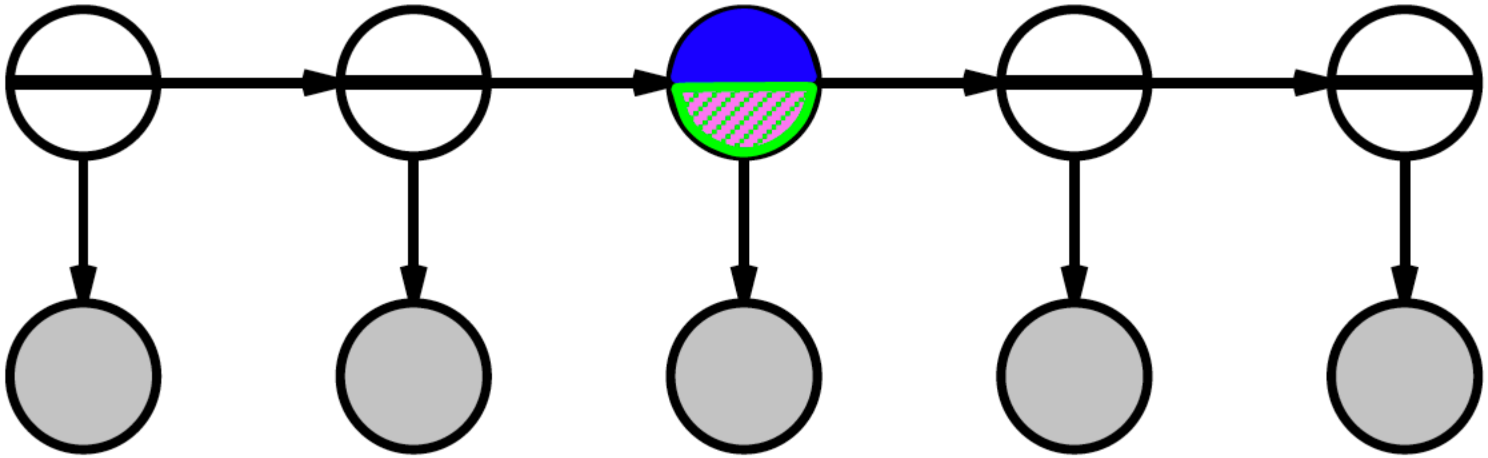


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$

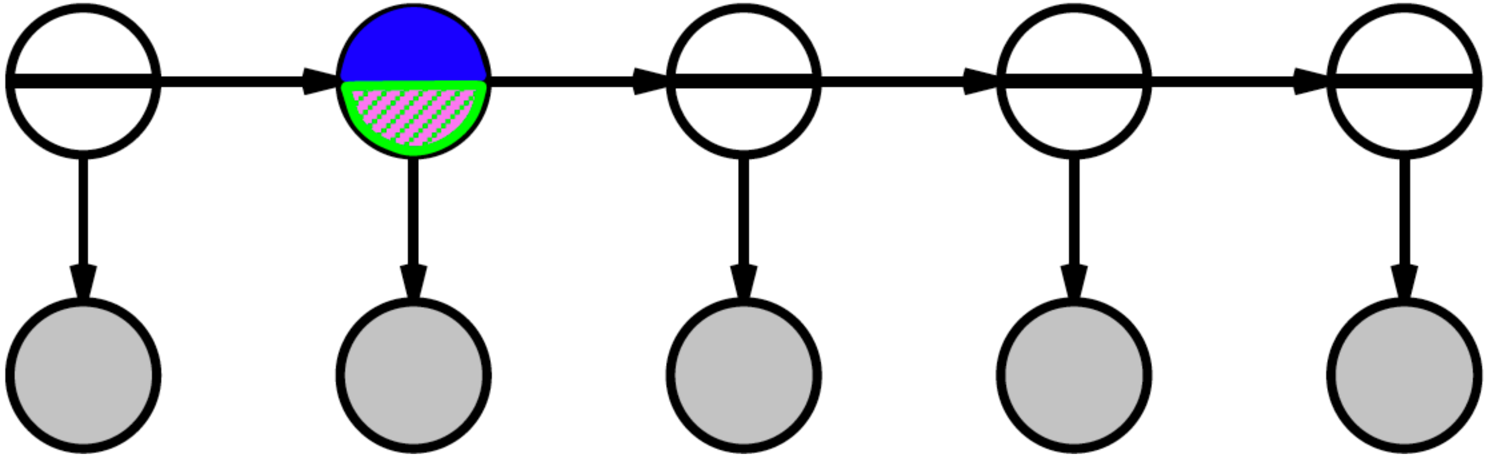


# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha'(T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$



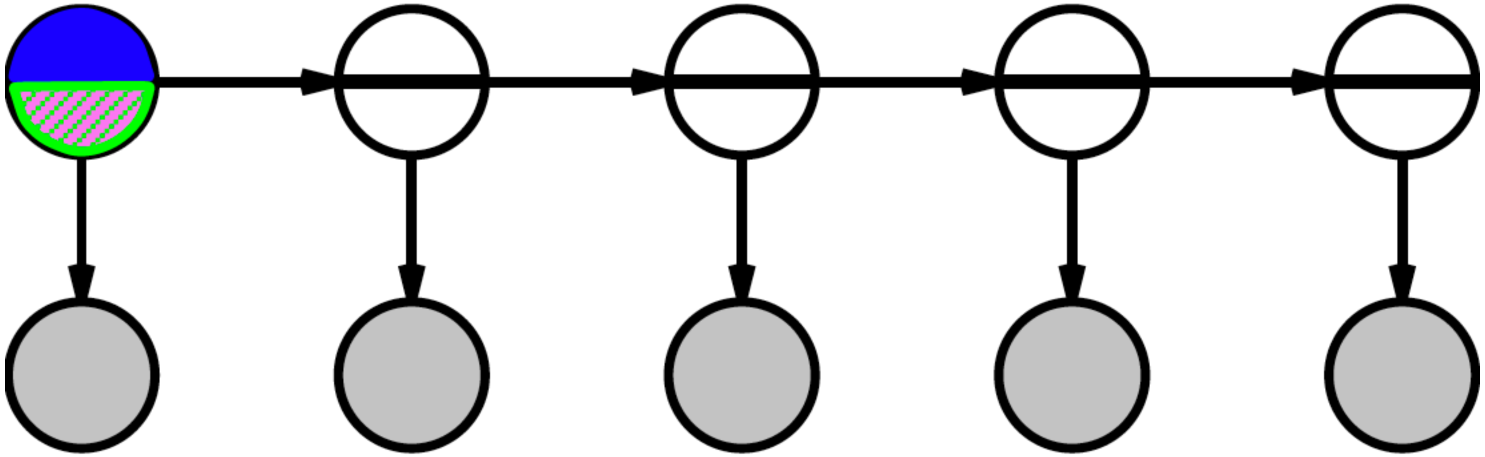
JAMES MADISON UNIVERSITY

# Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$f_{i:t=1} = \alpha O_{t+1} T^t f_{1:t}$$
$$O_{t+1}^{-1} f_{1:t+1} = \alpha T^t f_{1:t}$$
$$\alpha' (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

Algorithm: forward pass computes $f_t$, backward pass foes $f_i$, $b_i$



# Updating Gaussian distributions

Prediction step: if P($X_t$ | $e_{1:t}$) is Gaussian, then prediction

$$P(X_{t+1}|e_{1:t}) = \int_{X_t} P(X_{t+1}|\, x_t)P(x_t|\, e_{1:t})dx_t$$

is Gaussian. If P(Xt+1 | e1:t) is Gaussian, then the update distribution

$$P(X_{t+1}|\, e_{1:t+1}) = \alpha P(e_{t+1}|\, X_{t+1})P(X_{t+1}|\, e_{1:t})$$

is also Gaussian.

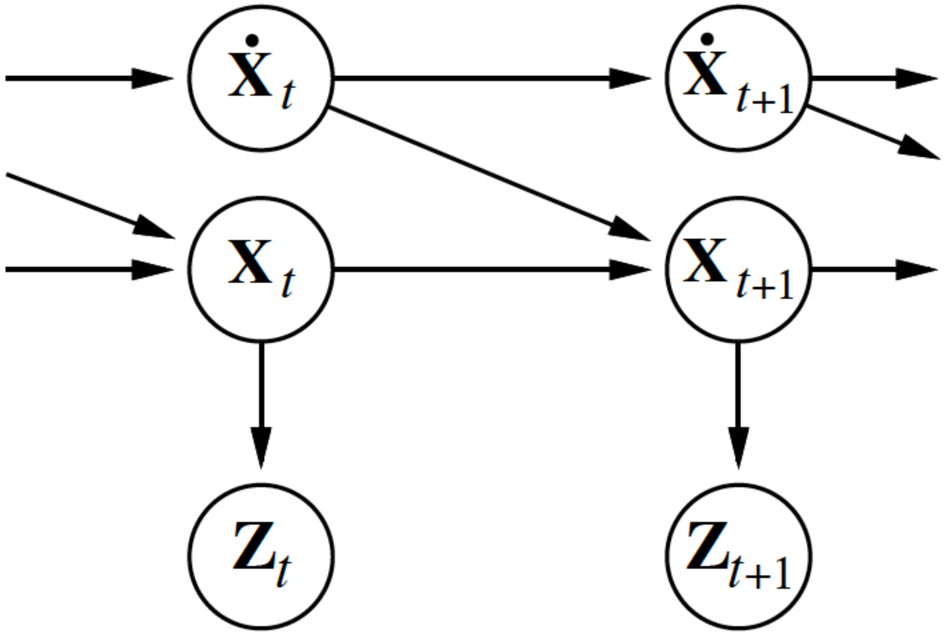
Hence, P($X_t$| $e_{1:t}$) is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all *t*

General (nonlinear, non-Gaussian) process: description of posterior grows unbounded as t $\rightarrow\infty$

# Kalman Filters

Modelling systems described by a set of continuous variables, e.g., tracking a bird flying – Xy = X, Y, Z, X, Y, Z.

Airplanes, robots, ecosystems, economies, chemical plants, planets




Gaussian prior, linear Gaussian transition model and sensor model

# Updating Gaussian distributions

Prediction step: if $P(X_t \mid e_{1:t})$ is Gaussian, then prediction

$$P(X_{t+1}|e_{1:t}) = \int_{x_t} P(X_{t+1}|\, x_t)P(x_t|\, e_{1:t})dx_t$$

is Gaussian. If P(Xt+1 | e1:t) is Gaussian, then the update distribution

$$P(X_{t+1}|\, e_{1:t+1}) = \alpha P(e_{t+1}|\, X_{t+1})P(X_{t+1}|\, e_{1:t})$$

is also Gaussian.

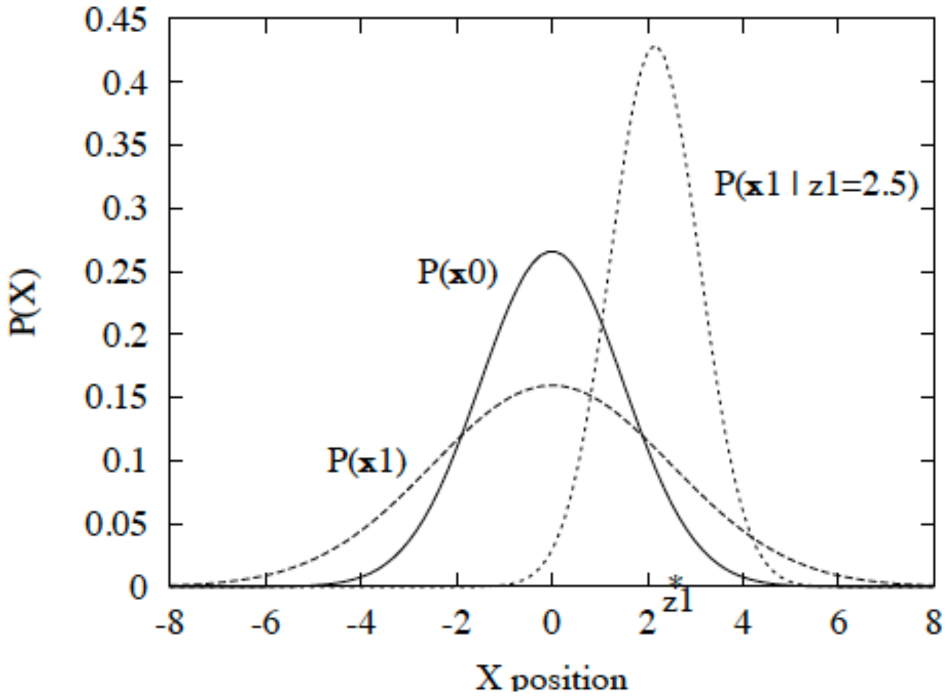
Hence, $P(X_t|\, e_{1:t})$ is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all $t$

General (nonlinear, non-Gaussian) process: description of posterior grows unbounded as $t \rightarrow \infty$

# Simple 1-D example

Gaussian random walk on X-axis, s.d., $\sigma_x$, sensor s.d. $\sigma_x$

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2) z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\,\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$



# General Kalmann Update

$Transition\ and\ sensor\ models$:

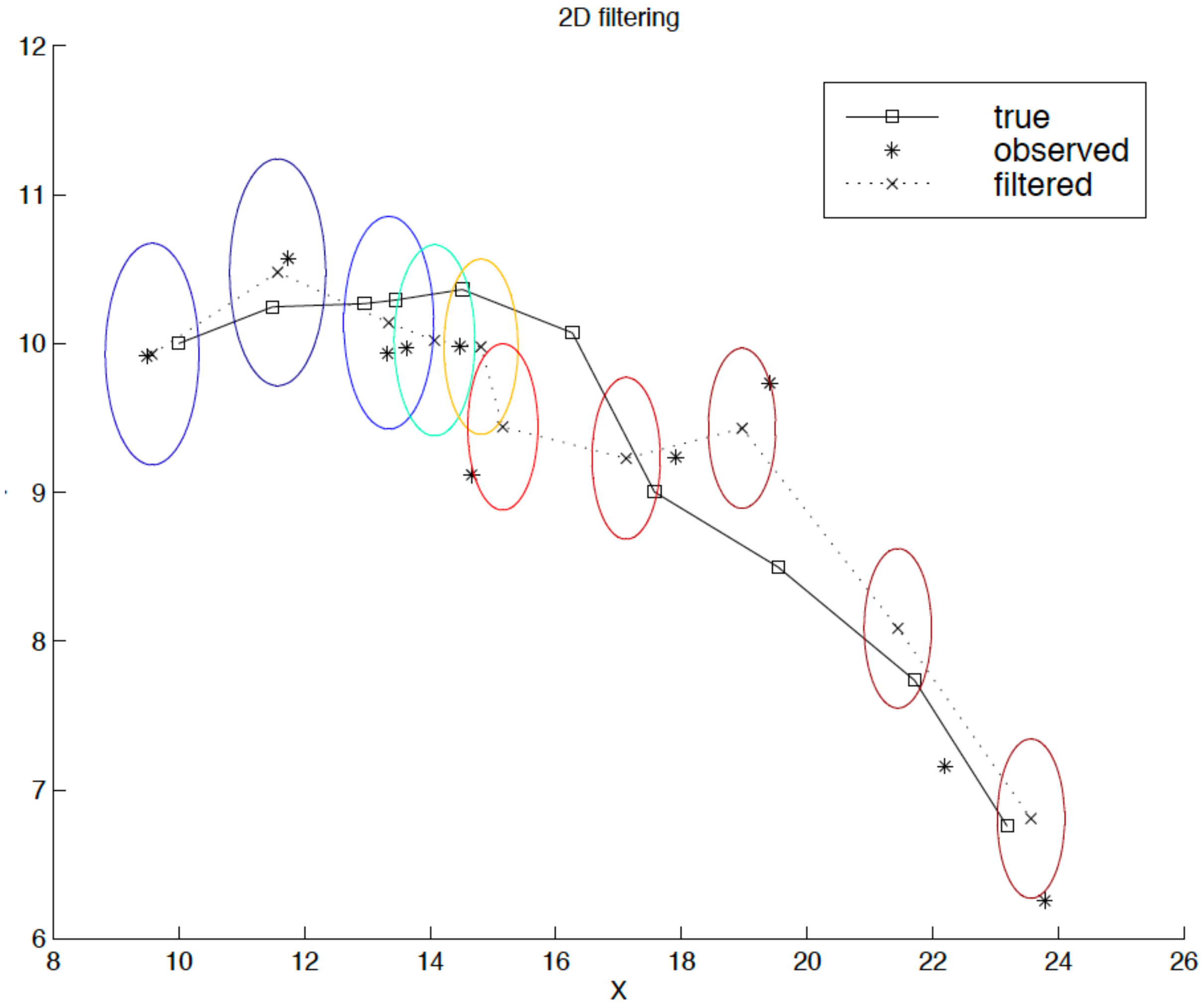
$$P(x_{t+1}|\, x_t) = N(Fx_t, \Sigma_x)(x_{t+1})$$
$$P(z_t|\, x_t) = N(Hx_t, \Sigma_z)(z_t)$$

$F\ is\ the\ matrix\ for\ the\ transition;\ \Sigma x$the transitiobn noise covariance

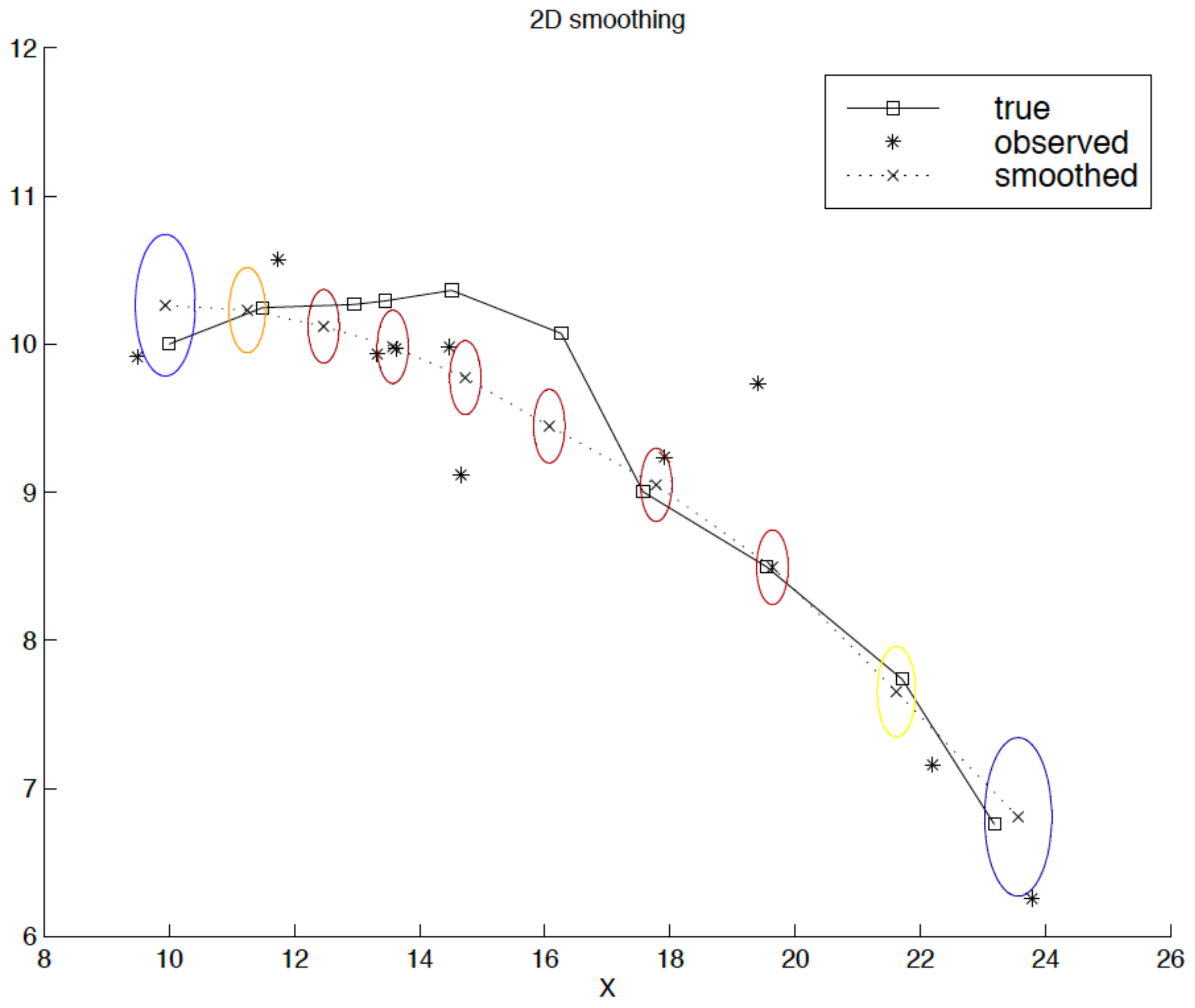
H is the matrix for the sensors, $\Sigma x$ the sensor noise covariance

$Filter$ computes the following update:

# 2-D tracking example: filtering



# 2-D tracking example: smoothing



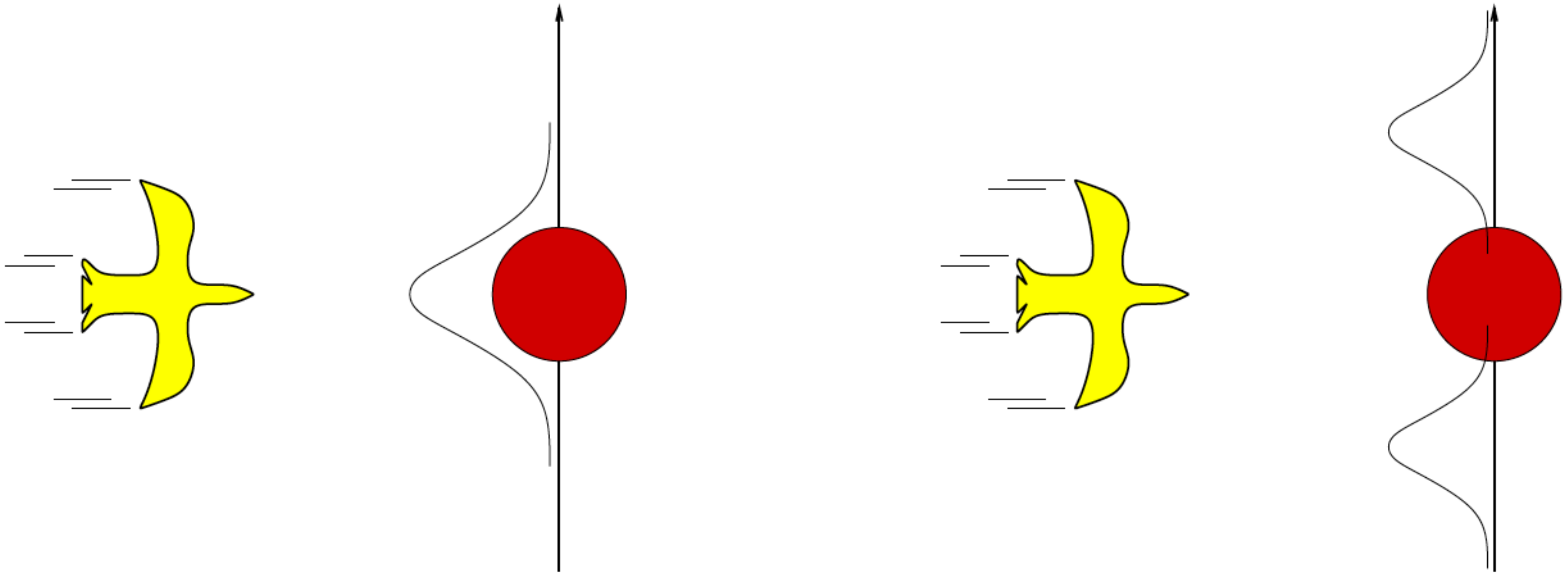
# Where is breaks

*Cannot be applied if the transition model is nonlinear*

Main idea:

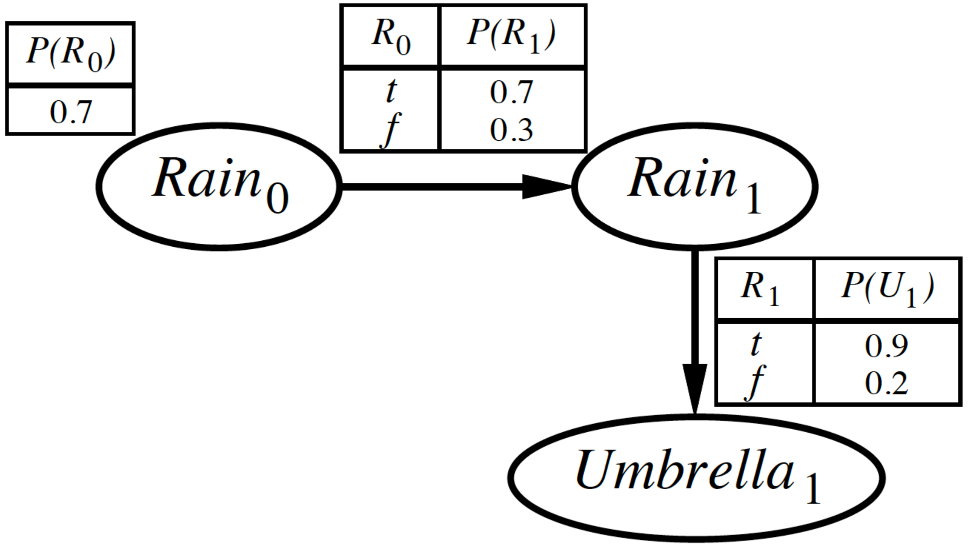
Extended Kalman Filter models transition as locally linear around xt = ut.

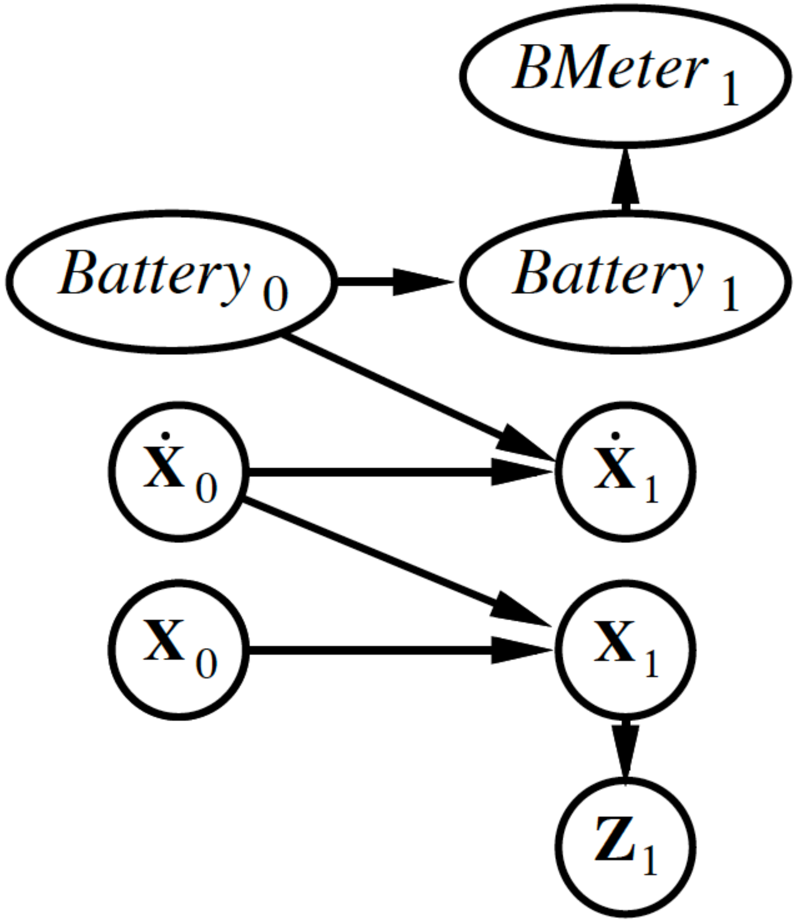
Fails if systems is locally unsmooth



# Dynamic Bayesian networks

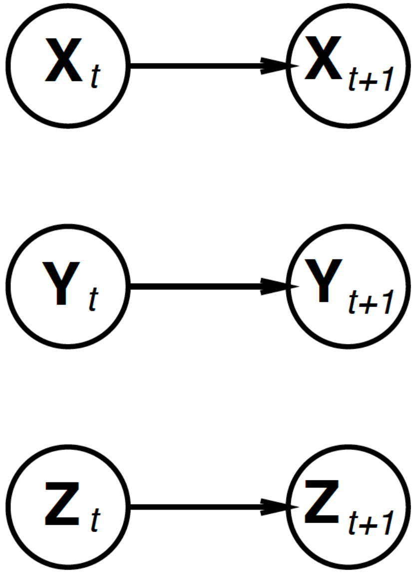
$X_t$, $E_t$ contain arbitrarily many variables in a replicated Bayes net

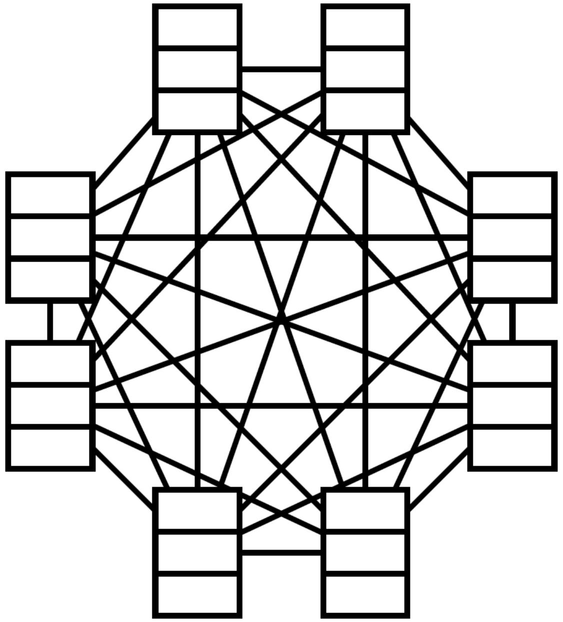


# Dynamic BN vs. Hidden Markov Models

Every HMM is a single variable DBN; every discrete DBN is an HMM



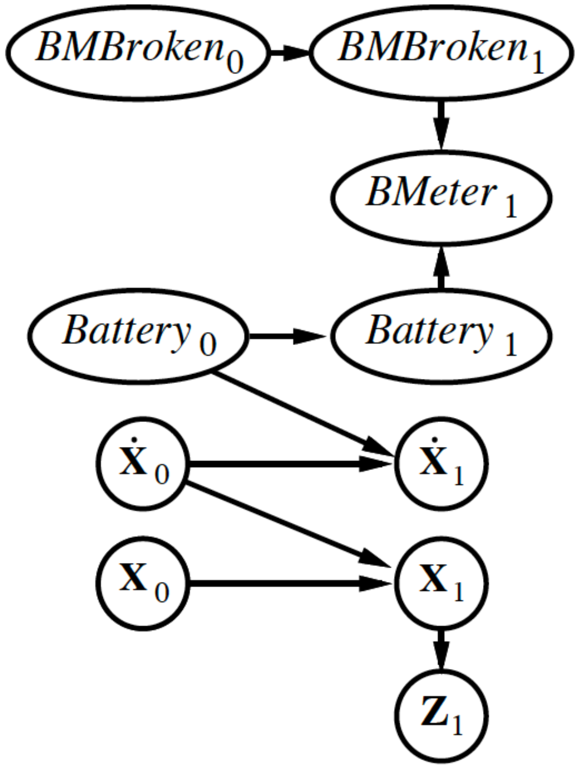
Sparse dependencies ⟹ exponentially fewer parameters

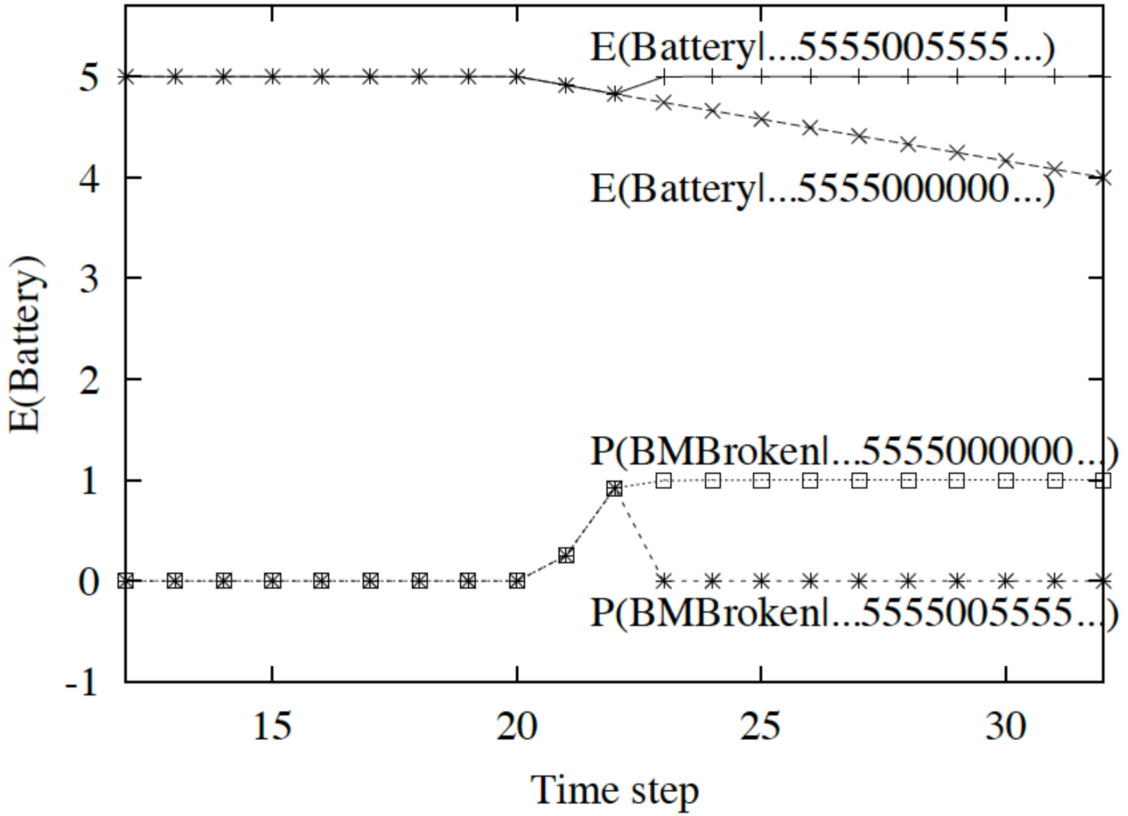
e.g. 20 state variables, three parents each

DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

# Dynamic BN verus Kalman Filter

Every Kalman filter model is a DBM, but few DBM are KFs; real world requires non-Gaussian posteriors. E.g., where are bin Laden and my keys? What's the battery charge?



# Viterbi Example