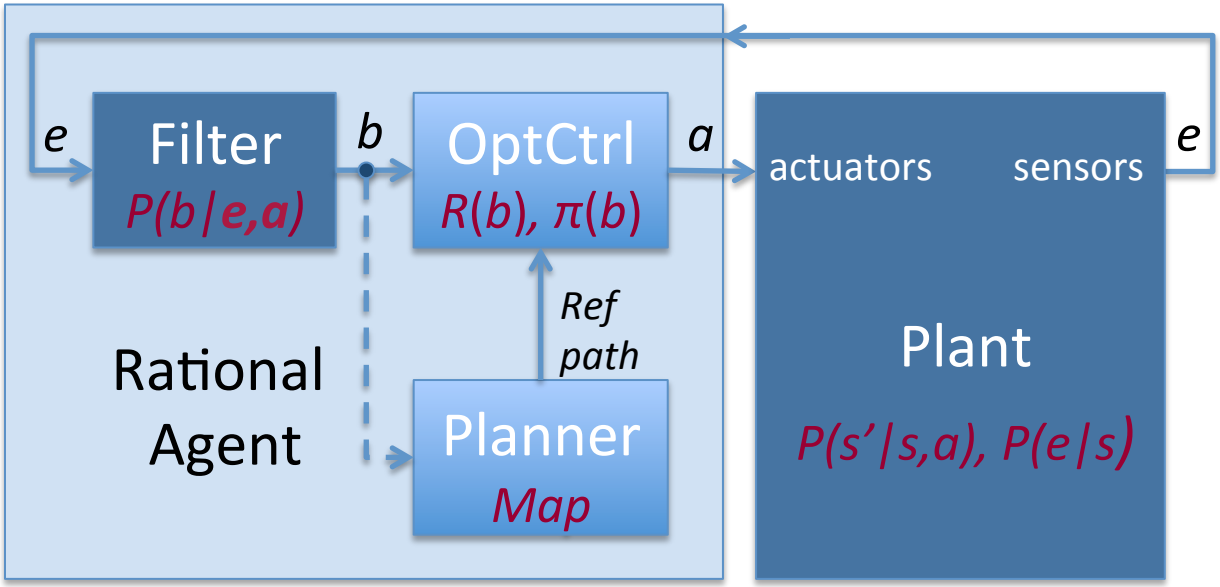
State Estimation in Structured and Continuous Models

Chapters 15.4-5



Temporal-Models Problems

Inference from HMM and observations :

- Filtering $P(X_t \mid \mathbf{e}_{0:t})$: Current-state estimation
- Prediction $P(X_{t+k} \mid \mathbf{e}_{0:t})$: Future-state estimation
- Smoothing $P(X_k \mid \mathbf{e}_{0:t})$: Past-state estimation
- MLE $\text{argmax}_{\mathbf{x}_{0:t}} P(\mathbf{x}_{0:t} \mid \mathbf{e}_{0:t})$: Most-likely explanation

Learning best HMM from observations:

- EM $P(X_0)$, $P(X' \mid X)$, $P(E \mid X)$: Expectation Maximisation

Most likely explanation

Most likely sequence ≠ sequence of most likely states!

Most likely path (MLP) to each x_{t+1}

= MLP to some x_t plus one more step

$$\max_{x_1 \ldots x_t} P(x_1, \ldots, x_t, X_{t+1} \mid \mathbf{e}_{1:t+1})$$

$$= P(e_{t+1} \mid X_{t+1}) \max_{x_t} P(X_{t+1} \mid x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t \mid \mathbf{e}_{1:t})$$

Identical to filtering, except $\mathbf{f}_{1:t}$ replaced by:

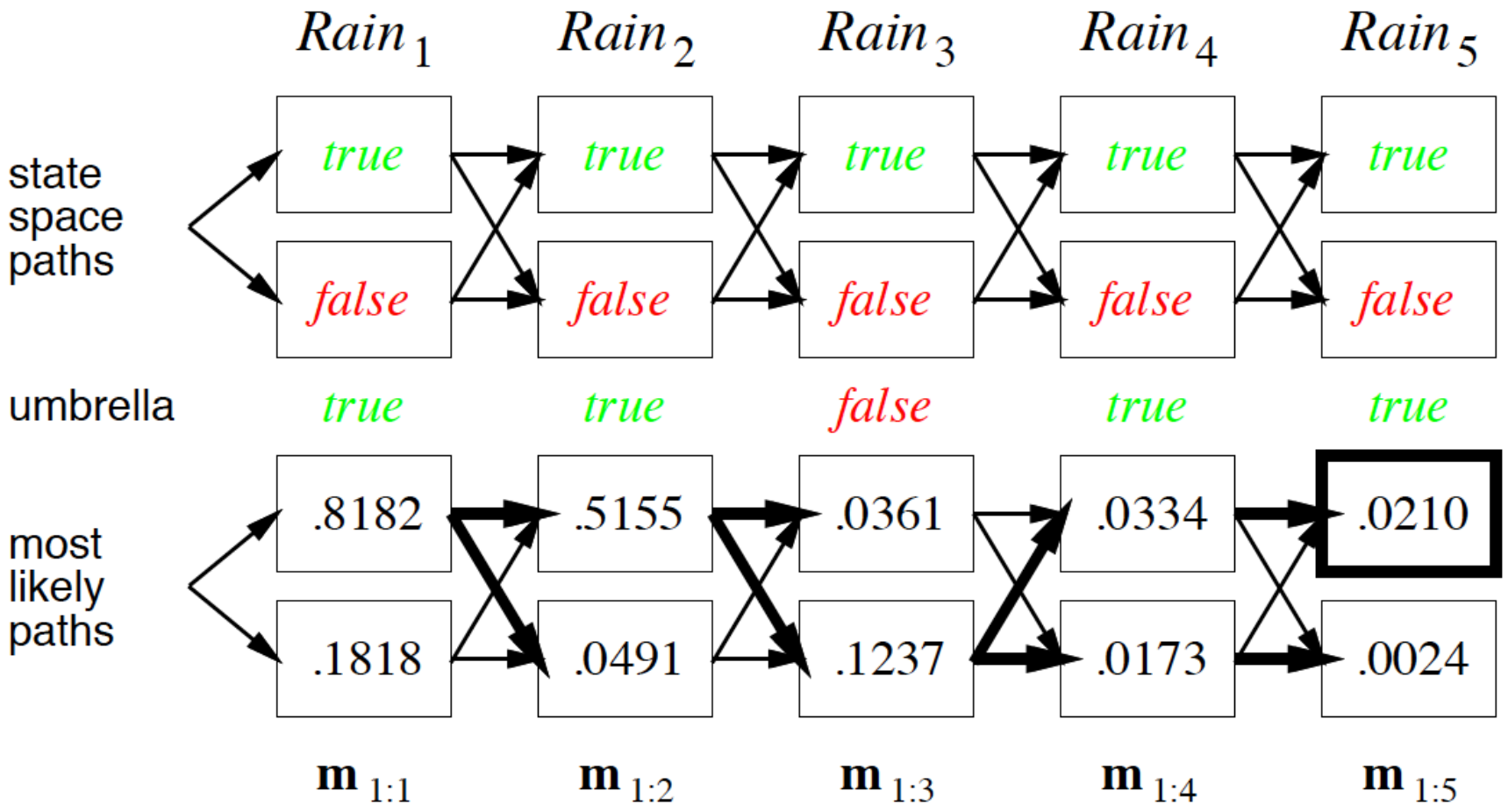
$$\mathbf{m}_{1:t} = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, X_t \mid \mathbf{e}_{1:t})$$

$m_{1:t}(i)$ gives the probability of the most likely path to state i.

Update has sum replaced by max, giving the Viterbi algorithm:

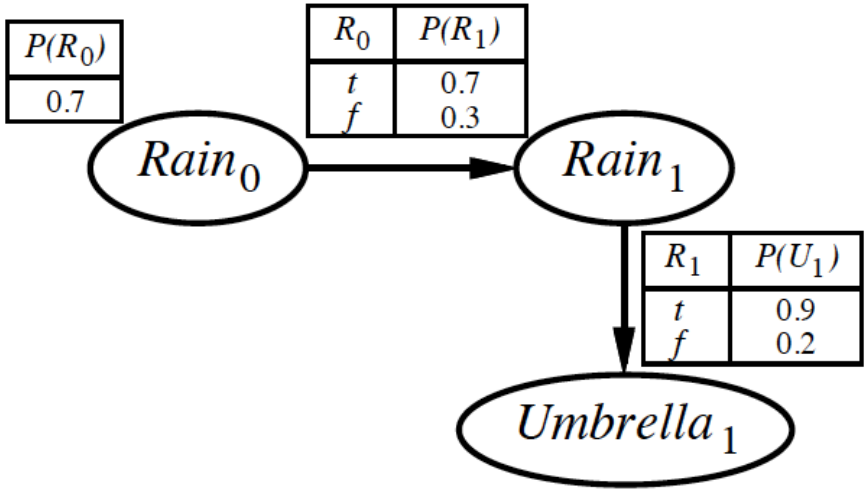
$$\mathbf{m}_{1:t+1} = P(e_{t+1} \mid X_{t+1}) \max_{x_t} (P(x_1, \ldots, x_{t-1}, X_t \mid \mathbf{e}_{1:t}) \mathbf{m}_{1:t})$$

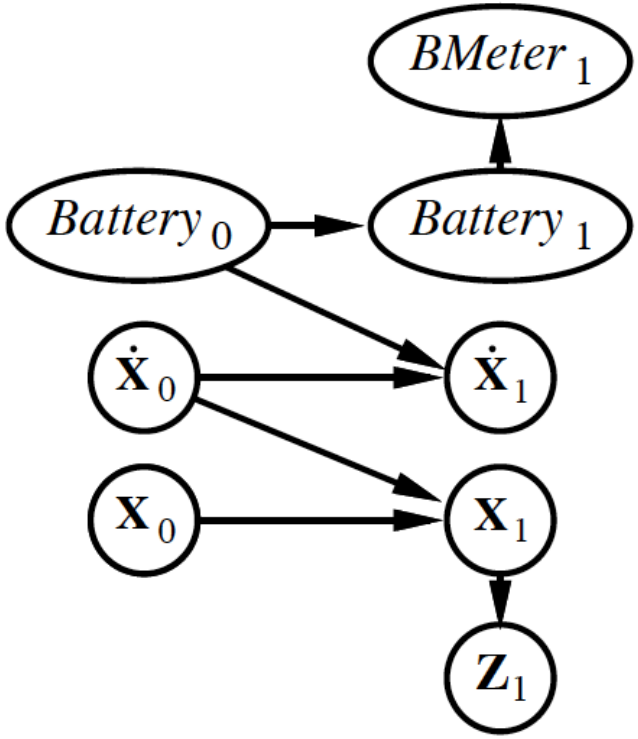
Viterbi example



Dynamic Bayesian Networks

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

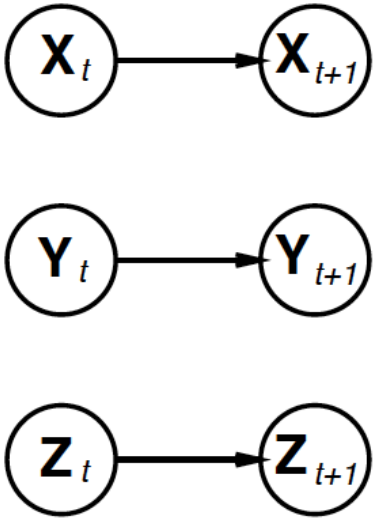


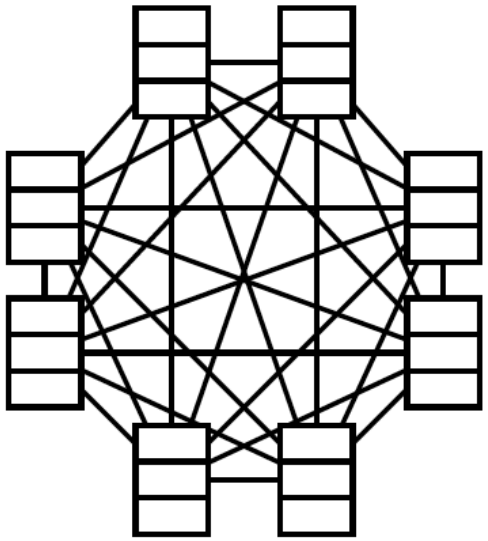


DBNs versus HMMs

Every HMM is a single-variable DBN

Every discrete DBN is an HMM





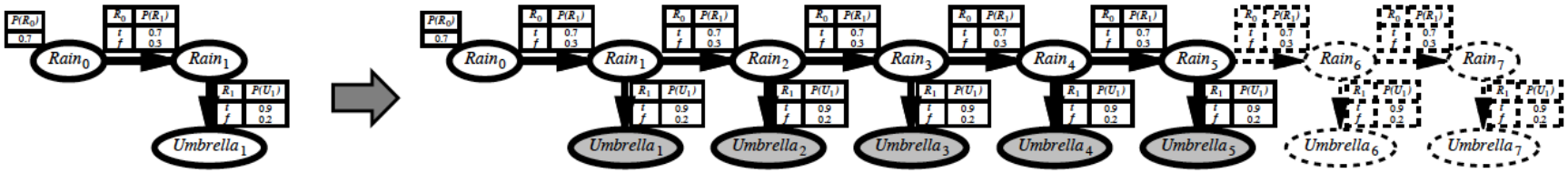
Sparse dependencies imply exponentially fewer parameters

Consider 20 state variables, three parents each:

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

Exact inference in DBNs

Naive method: unroll the network and run any exact algorithm



Problem: inference cost for each update grows with t

Improvement with rollup filtering:

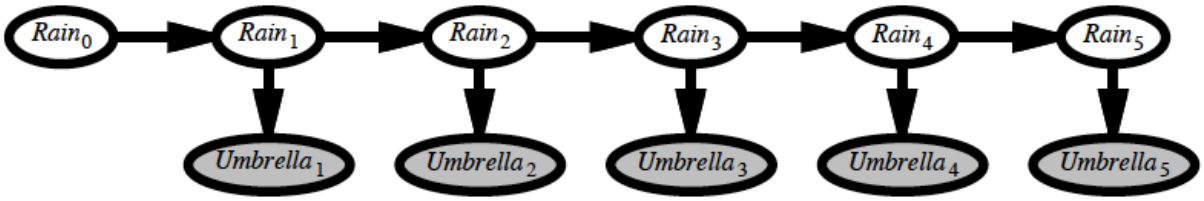
- Add $slice_{t+1}$, sum out $slice_t$, using variable elimination

Largest factor is $O(d^{n+1})$, update cost $O(d^{n+2})$

- The HMM has update cost $O(d^{2n})$

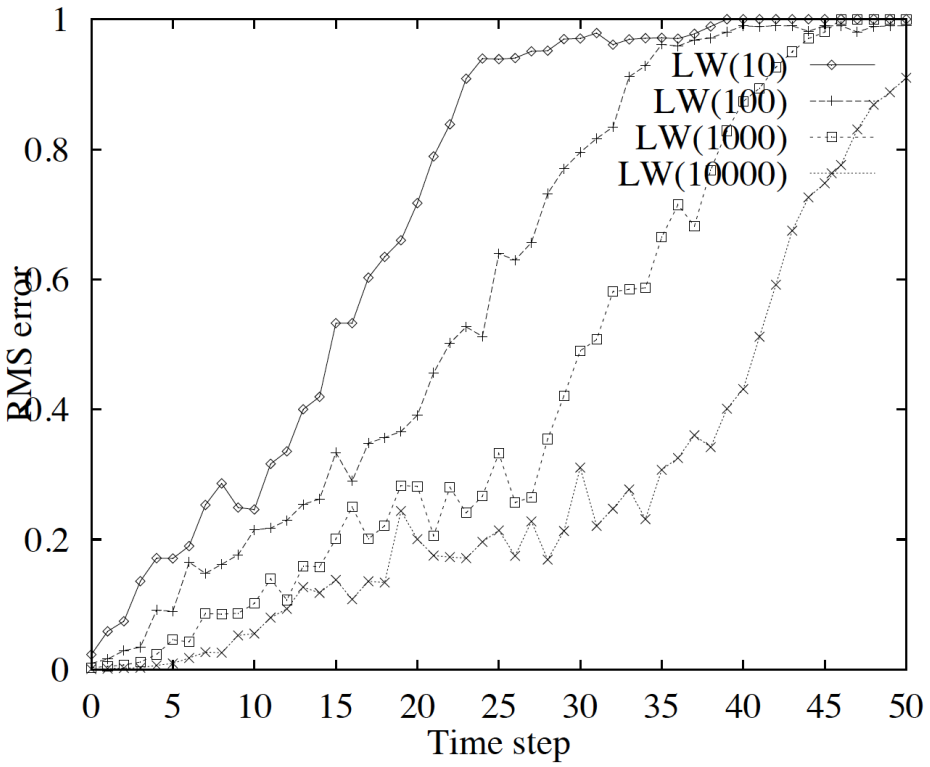
Likelihood weighting for DBNs

Set of weighted samples approximates the belief state



LW samples pay no attention to the evidence!

- Fraction agreeing falls exponentially with t
- #samples required grows exponentially with t

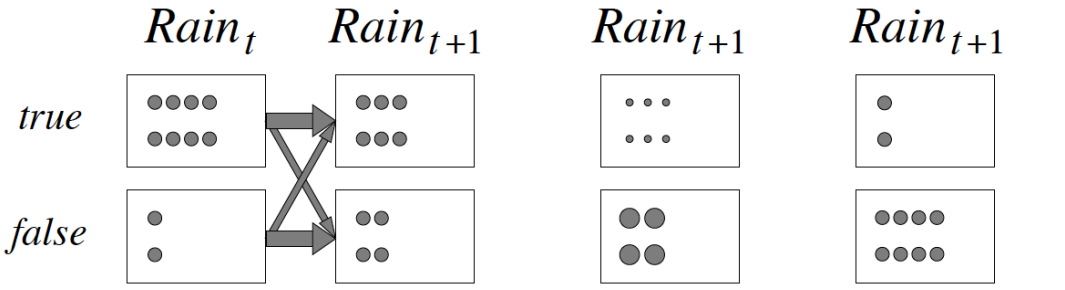


Particle Filtering

Basic idea: ensure that the population of samples (particles)

- Tracks the high-likelihood regions of the state-space

Replicate particles proportional to likelihood for e_t



Widely used for tracking nonlinear systems, e.g. in vision

Also used for SLAM and mapping in mobile robots

- 10^5 – dimensional state space

Particle Filtering

Assume consistent at time t: $N(x_t \mid e_{1:t}) / N = P(x_t \mid e_{1:t})$

Propagate forward: populations of x_{t+1} are

$$N(x_{t+1} \mid e_{1:t}) = \sum_{x_t} P(x_{t+1} \mid x_t)\, N(x_t \mid e_{1:t})$$

Weight samples by their likelihood for e_{t+1}:

$$W(x_{t+1} \mid e_{1:t+1}) = P(e_{t+1} \mid x_{t+1})\, N(x_{t+1} \mid e_{1:t})$$

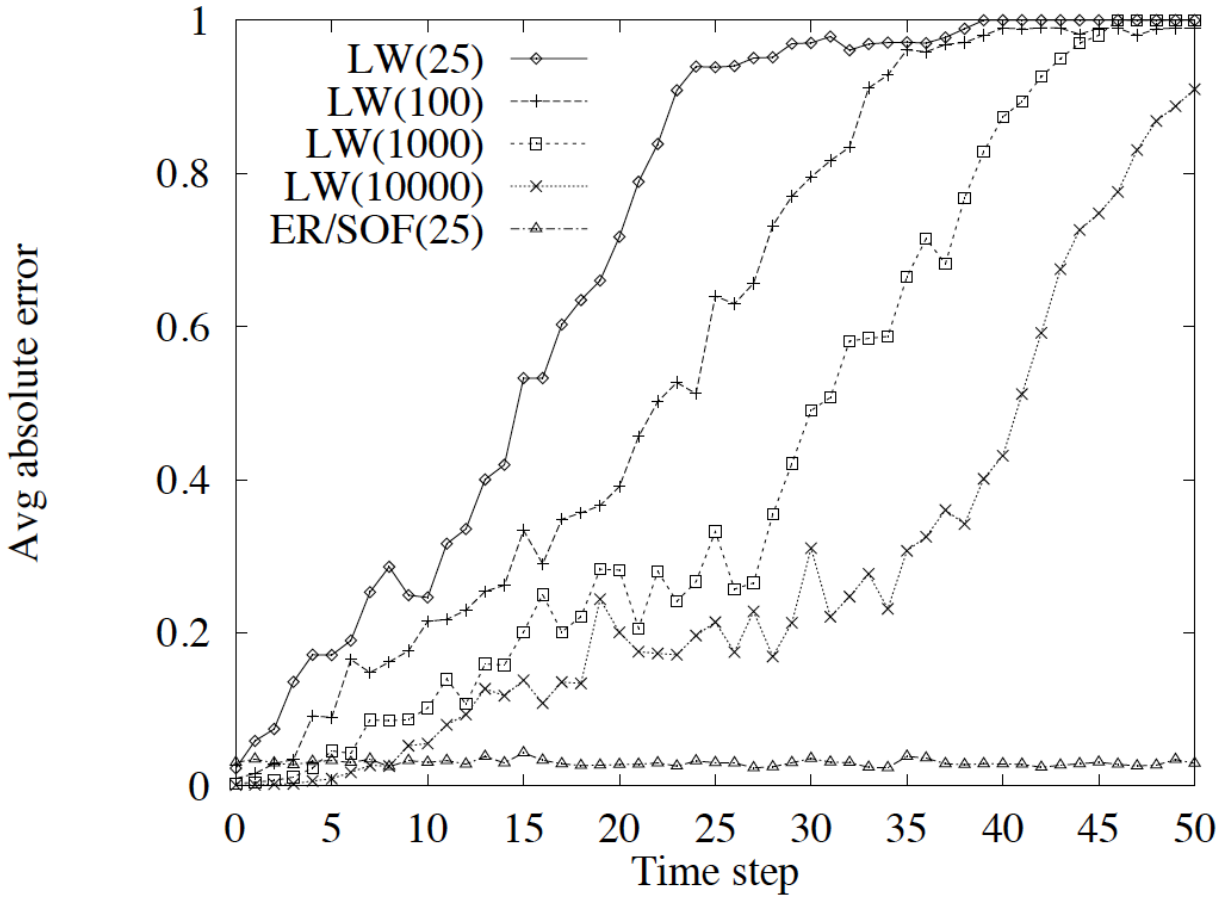
Resample to obtain populations proportional to W:

$$\begin{aligned} N(x_{t+1} \mid e_{1:t+1}) / N &= \alpha W(x_{t+1} \mid e_{1:t+1}) \\ &= \alpha P(e_{t+1} \mid x_{t+1})\, N(x_{t+1} \mid e_{1:t}) \\ &= \alpha P(e_{t+1} \mid x_{t+1}) \sum_{x_t} P(x_{t+1} \mid x_t)\, N(x_t \mid e_{1:t}) \\ &= \alpha' P(e_{t+1} \mid x_{t+1}) \sum_{x_t} P(x_{t+1} \mid x_t)\, P(x_t \mid e_{1:t}) \\ &= P(x_{t+1} \mid e_{1:t+1}) \end{aligned}$$

Particle Filtering Performance

Approximation error of PF remains bounded over time

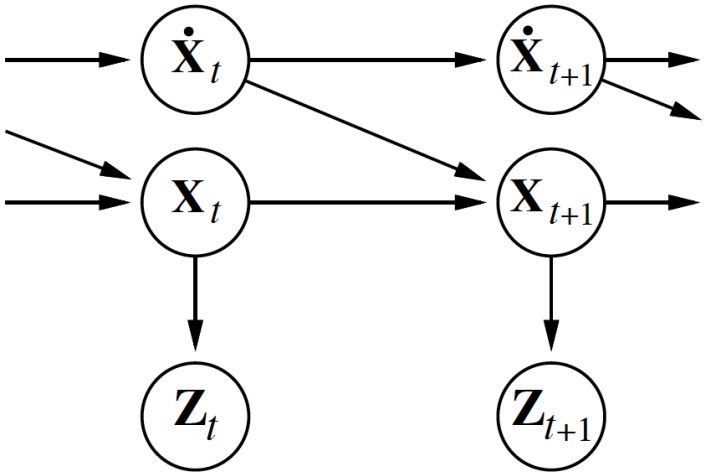
- At least empirically: theoretical analysis is dicult



Kalman Filters

For systems described by a set of continuous variables

- A flying bird's state: $\mathbf{X}_t = (X, Y, Z, \dot{X}, \dot{Y}, \dot{Z})$
- Also: airplanes, robots, ecosystems, economies, chemical plants...



Gaussian prior, linear Gaussian transition model and sensor model

Updating Gaussian Distributions

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

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

Updated distribution is Gaussian if $P(X_{t+1} \mid e_{1:t})$ is Gaussian

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

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

In general: Systems are nonlinear, non-Gaussian

- Deascription of posterior: Grows unboundedly as $t \to \infty$

Simple 1-D example

Gaussian random walk on X-axis: variances σ_x, sensor σ_e

$$P(x_0) = \alpha e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}$$

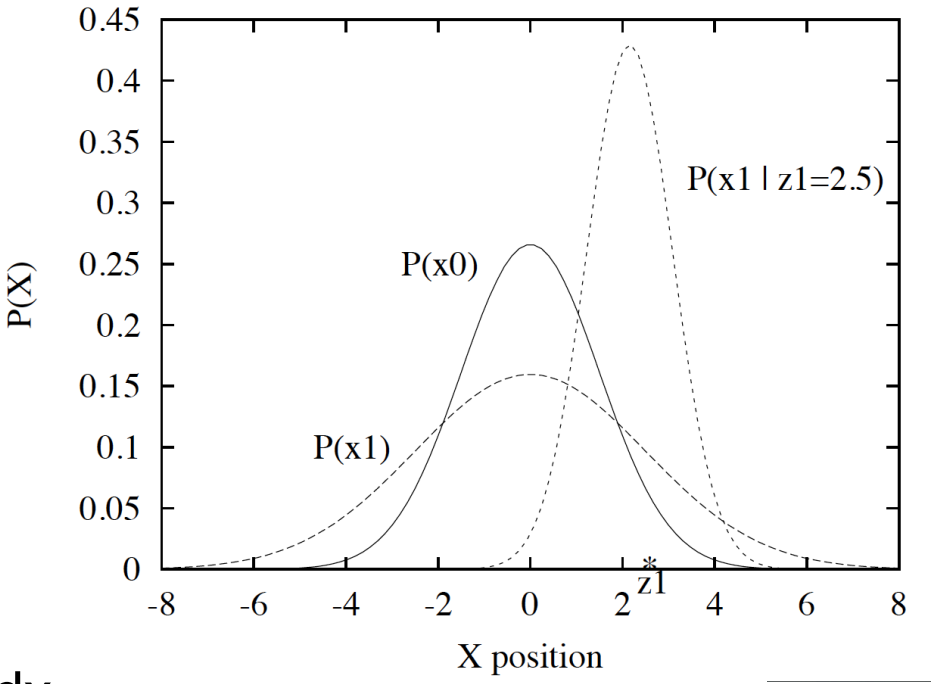
$$P(x_{t+1} \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}$$

$$P(e_t \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(e_t-x_t)^2}{\sigma_e^2}\right)}$$

$$P(x_1 \mid x_0) = \int_{-\infty}^{\infty} P(x_1 \mid x_0) dx_0$$

$$= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(x_1-x_0)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)} dx_0$$

$$= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2(x_1-x_0)^2+\sigma_x^2(x_0-\mu_0)^2}{\sigma_0^2\sigma_x^2}\right)} dx_0$$



Simple 1-D example

Gaussian random walk on X-axis: variances σ_x, sensor σ_z

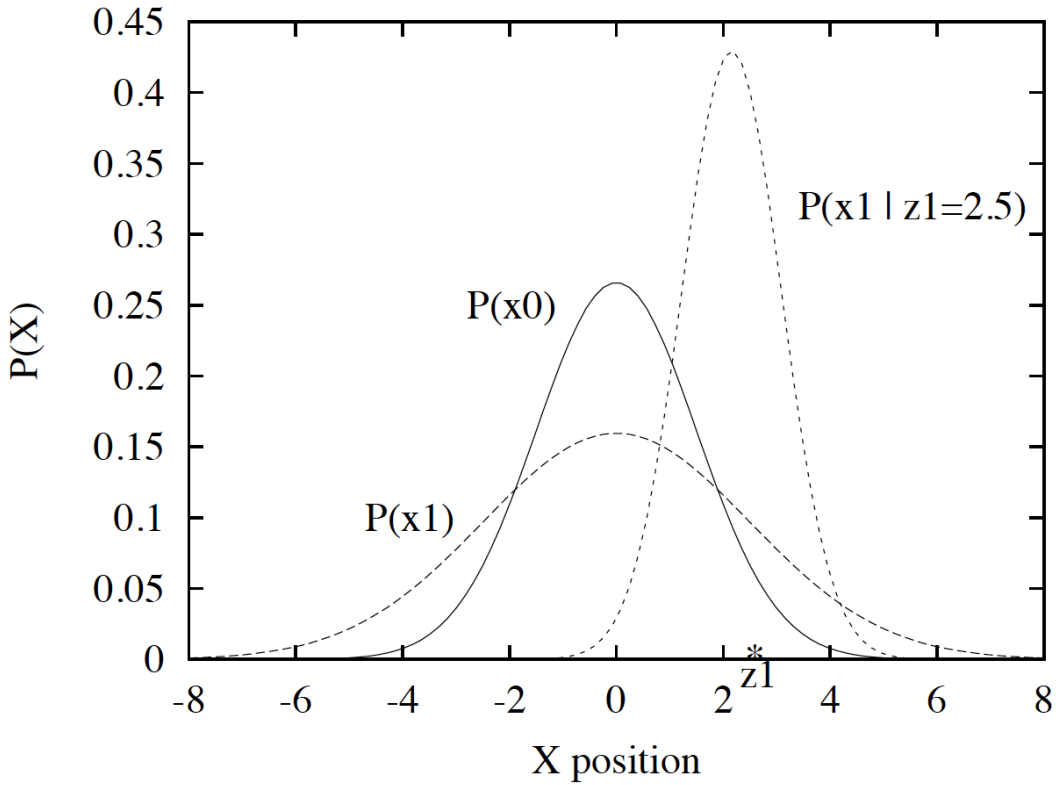
$$P(x_0) = \alpha e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}$$

$$P(x_{t+1} \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}$$

$$P(e_t \mid x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(e_t-x_t)^2}{\sigma_e^2}\right)}$$

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)e_{t+1} + \sigma_e^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}$$

$$\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_e^2}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}$$



General Kalman update

Transition and sensor models:

$$\mathbf{P}(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = \mathbf{N}(\mathbf{A}\mathbf{x}_t \mid \Sigma_x)(\mathbf{x}_{t+1})$$

$$\mathbf{P}(\mathbf{e}_t \mid \mathbf{x}_t) = \mathbf{N}(\mathbf{C}\mathbf{x}_t \mid \Sigma_e)(\mathbf{e}_t)$$

Transition matrix **A**, Transition noise covariance Σ_x

Output matrix **C**, Sensor noise covariance Σ_e

Filter computes the following update:

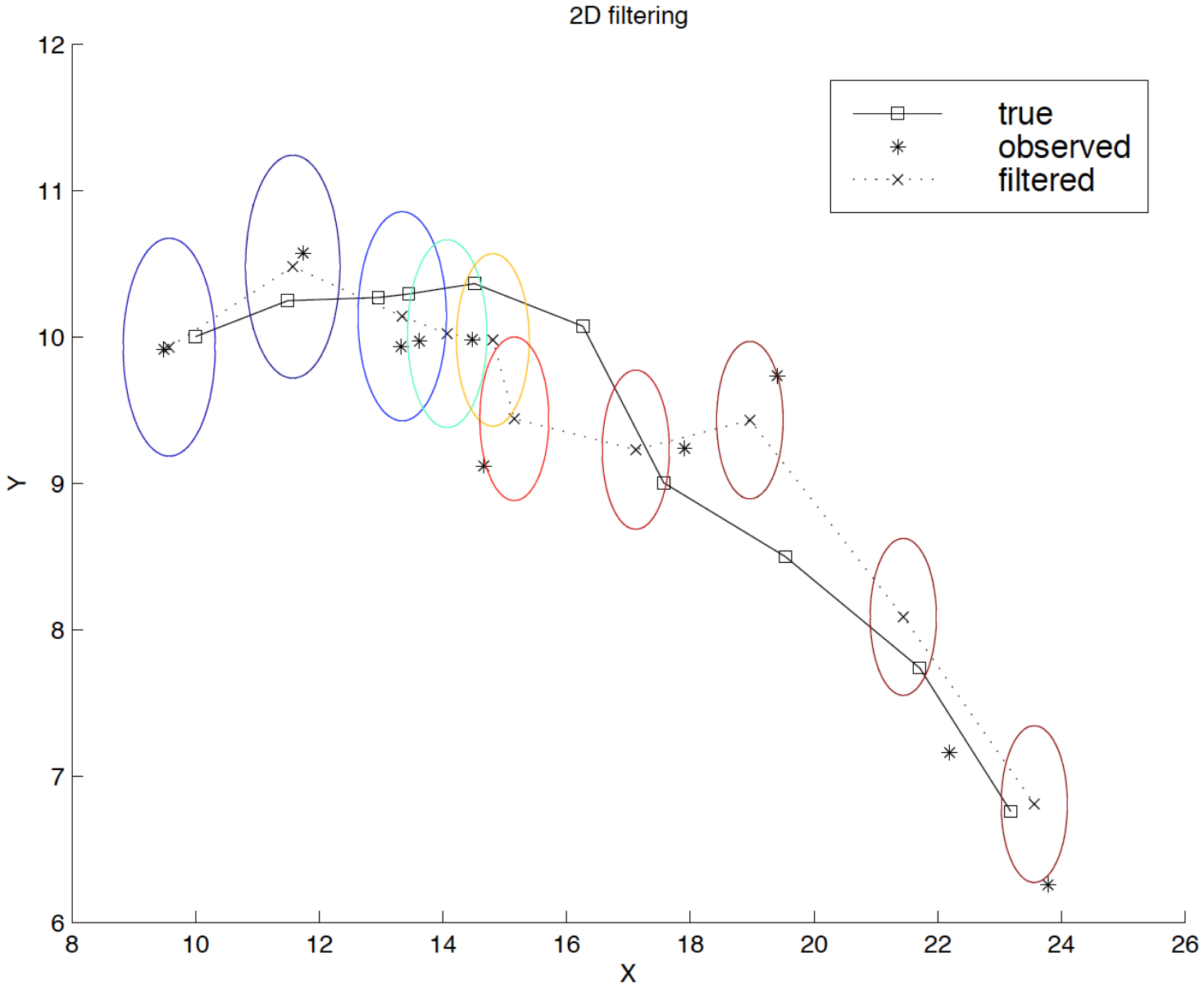
Mean: $\mu_{t+1} = \mathbf{A}\mu_t + \mathbf{K}_{t+1}(\mathbf{e}_{t+1} - \mathbf{C}\mathbf{A}\mu_t)$

Covariance: $\Sigma_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{A}\Sigma_t\mathbf{C}^t + \Sigma_x)$

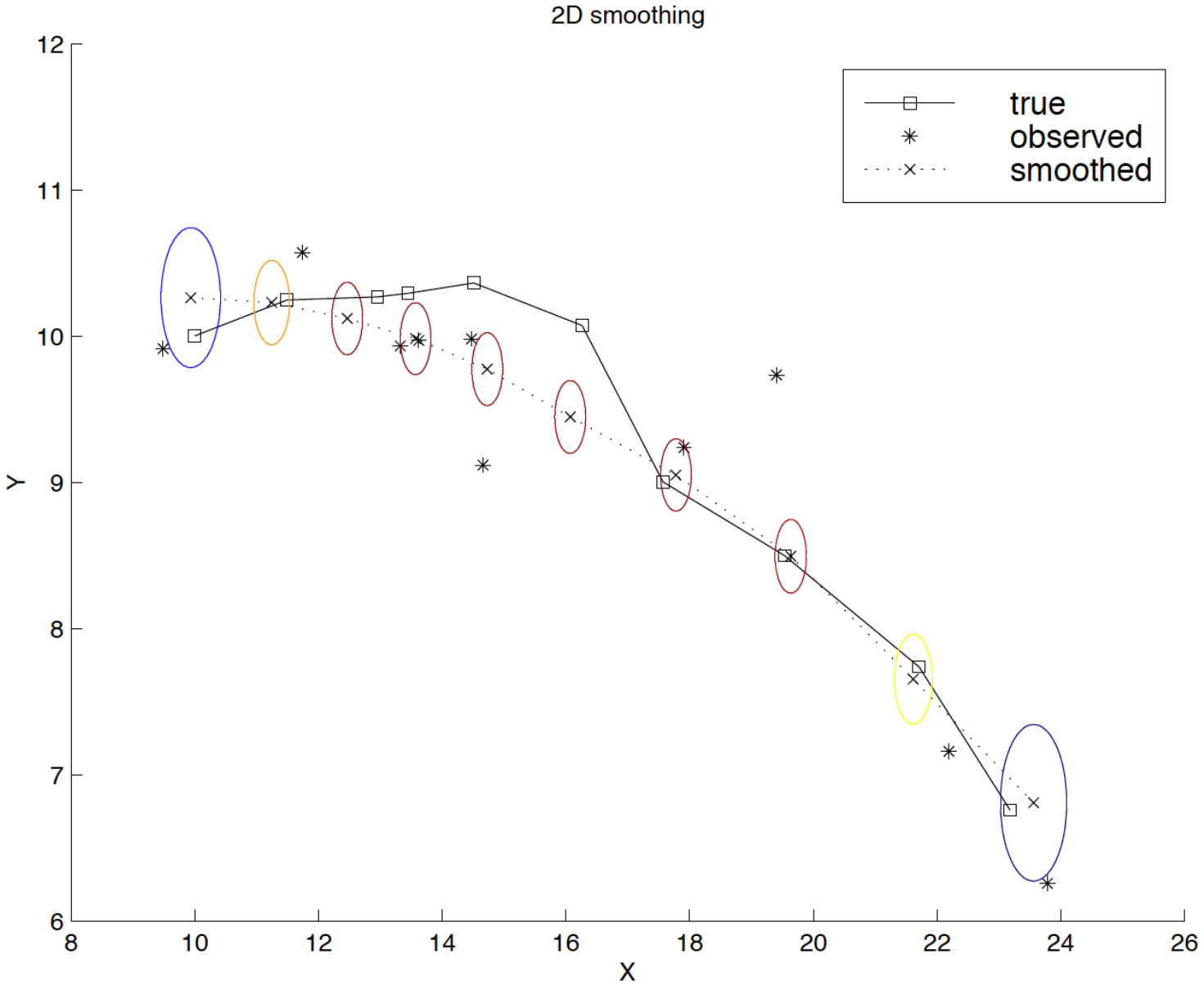
Kalman gain: $\mathbf{K}_{t+1} = (\mathbf{A}\Sigma_t\mathbf{A}^t + \Sigma_x)\mathbf{C}^t(\mathbf{C}(\mathbf{A}\Sigma_t\mathbf{A}^t + \Sigma_x)\mathbf{C}^t + \Sigma_e)^{-1}$

Σ_t and $\mathbf{K}_t$: Independent of sequence, computed offline

2-D tracking example: Filtering



2-D tracking example: Smoothing

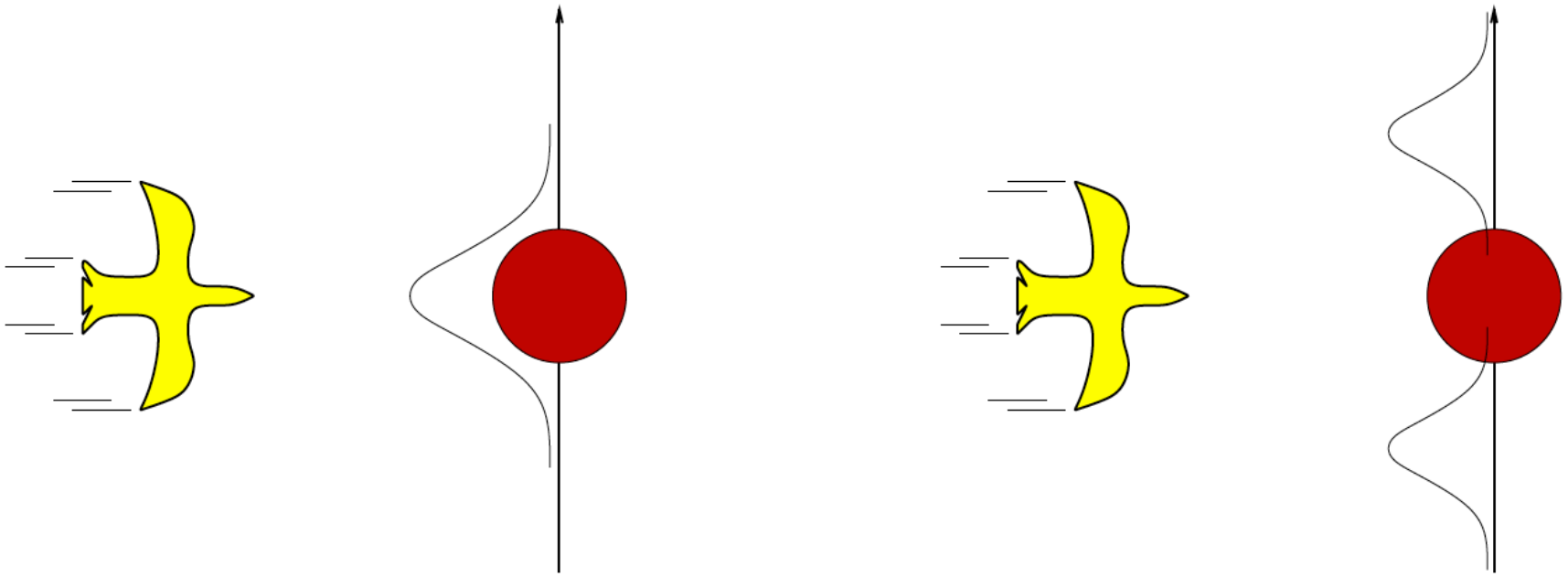


Where it Breaks

Cannot be applied if the transition model is nonlinear

Extended Kalman Filter:

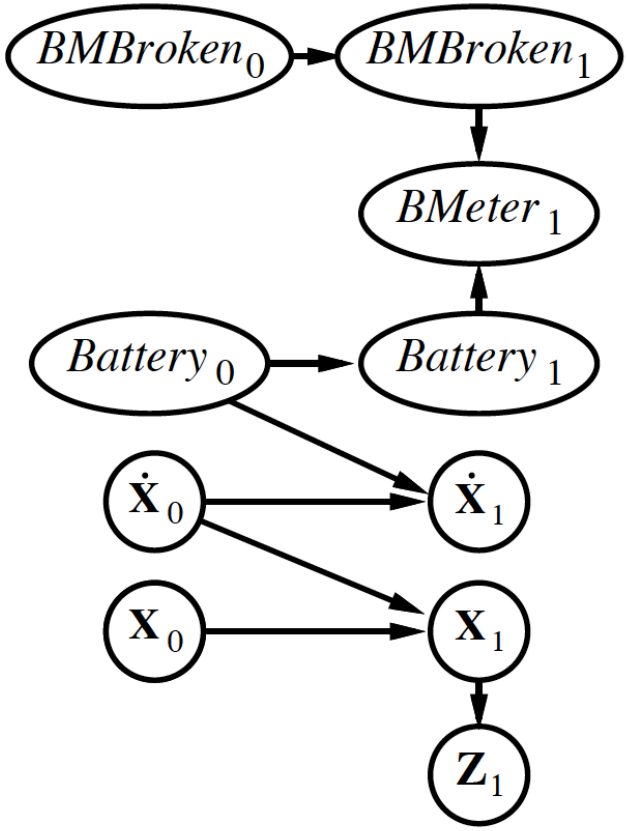
- Models transition as locally linear around $x_t = \mu_t$
- Fails if the system is locally unsmooth

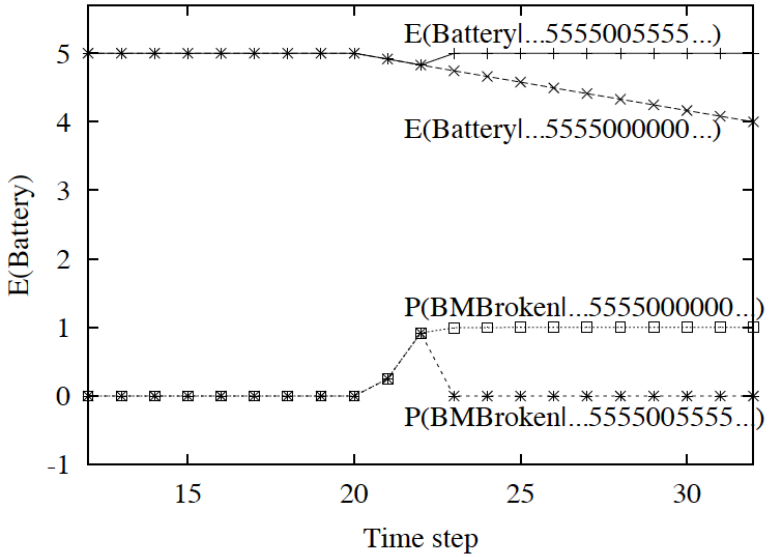


DBNs versus Kalman Filters

Every Kalman filter model is a DBN, but few DBNs are KFs;

- Real world requires non-Gaussian posteriors
- Example: What's the battery charge?





Summary

Temporal models use state & sensor variables replicated over time

Markov assumptions and stationarity assumption, so we need

- Transition Model: $P(X_{t+1} \mid X_t)$ Sensor model: $P(E_t \mid X_t)$

Tasks are filtering, prediction, smoothing, most likely sequence

- All done recursively with constant cost per time step

Hidden Markov models have a single discrete state variable

- Used e.g. for speech recognition

Kalman filters allow n state variables, linear Gaussian

- Complexity of update is $O(n^3)$

Dynamic Bayes nets subsume HMMs & KFs

- Exact update intractable

Particle filtering is a good approximate filtering alg for DBNs