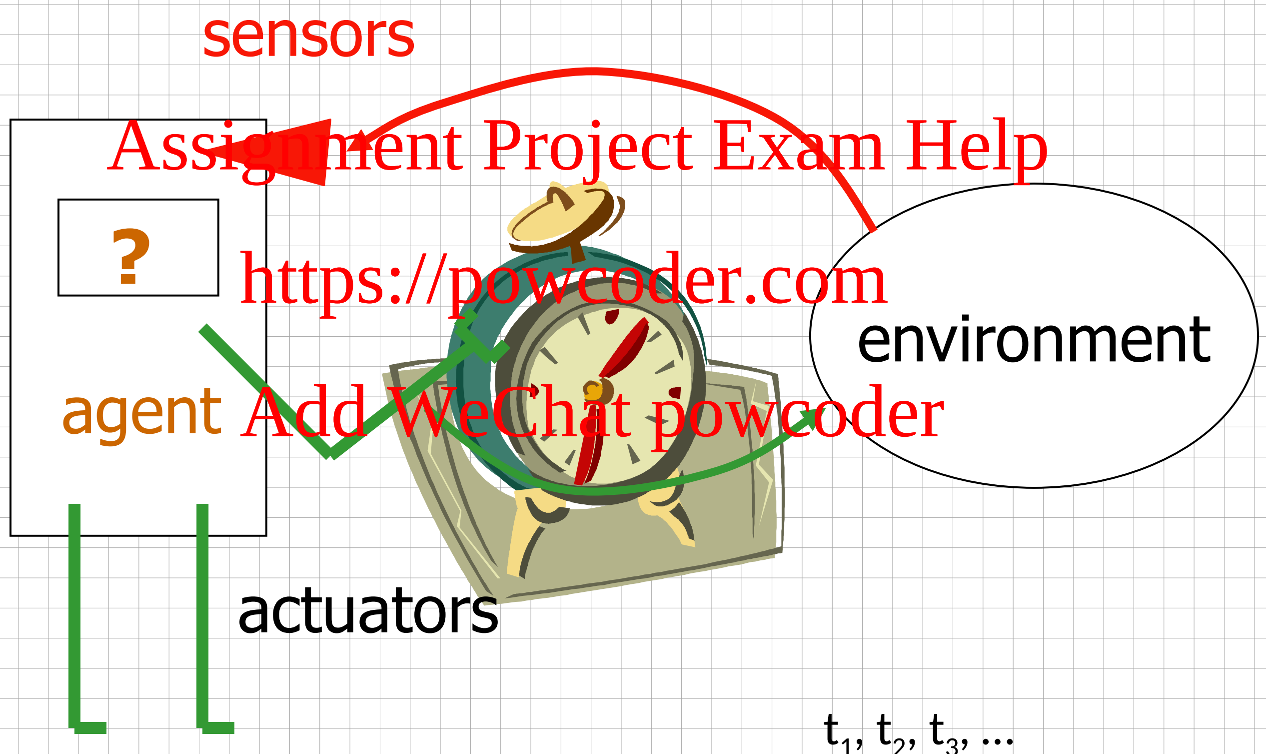
CISC 6525
Artificial Intelligence

Time and Uncertainty

Chapter 15

Temporal Probabilistic Agent



Time and Uncertainty

- The world changes, we need to track and predict it
- Examples: diabetes management, traffic monitoring

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

- X_t – set of unobservable state variables at time t
  - e.g., BloodSugar_t, StomachContents_t
- E_t – set of evidence variables at time t
  - e.g., $\text{MeasuredBloodSugar}_t$, PulseRate_t, FoodEaten_t
- Assumes discrete time steps

States and Observations

- Process of change is viewed as series of snapshots, each describing the state of the world at a particular time
- Each time slice involves a set of random variables indexed by t:
  1. the set of unobservable state variables X_t
  2. the set of observable evidence variable E_t
- The observation at time t is $E_t = e_t$ for some set of values e_t
- The notation $X_{a:b}$ denotes the set of variables from X_a to X_b

Stationary Process/Markov Assumption

- **Markov Assumption**: X_t depends on some previous X_is
- **First-order Markov process**: $P(X_t | X_{0:t-1}) = P(X_t | X_{t-1})$
- **kth order**: depends on previous k time steps
- Sensor Markov assumption: $P(E_t | X_{0:t}, E_{0:t-1}) = P(E_t | X_t)$
- Assume **stationary process**: transition model $P(X_t | X_{t-1})$ and sensor model $P(E_t | X_t)$ are the same for all t
- In a **stationary process**, the changes in the world state are governed by laws that do not themselves change over time

Complete Joint Distribution

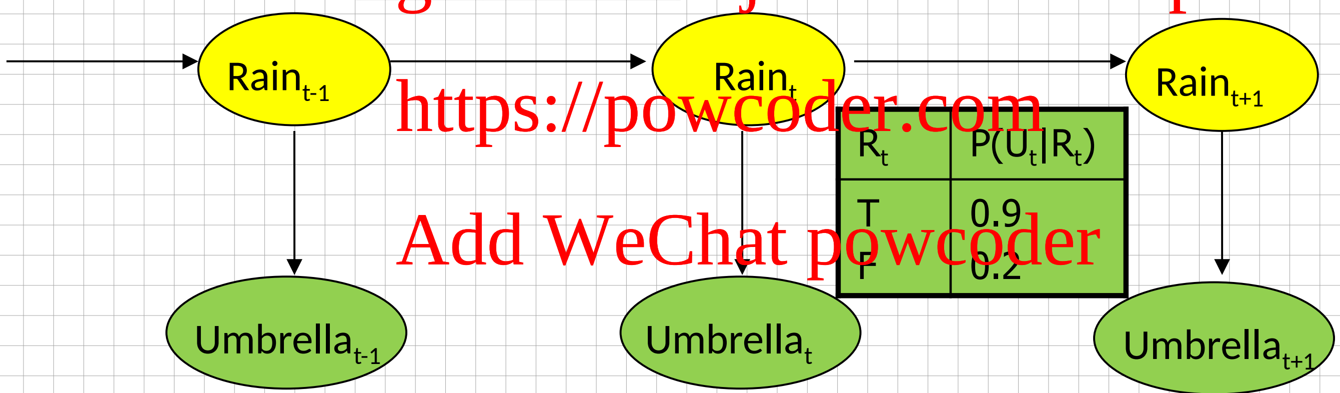
- Given:
  - Transition model: $P(X_t|X_{t-1})$
  - Sensor model: $P(E_t|X_t)$
  - Prior probability: $P(X_0)$
- Then we can specify complete joint distribution:

$$P(X_0, X_1, \ldots, X_t, E_1, \ldots, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i \mid X_{i-1}) P(E_i \mid X_i)$$

Probabilistic Temporal Models

- Hidden Markov Models (HMMs)
- Kalman Filters
- Dynamic Bayesian Networks (DBNs)

DBN Example



R_{t-1}	$P(R_t \mid R_{t-1})$
T	0.7
F	0.3

R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

Transition model: $P(Rain_t \mid Rain_{t-1})$
Sensor model: $P(Umbrella_t \mid Rain_t)$

Inference Tasks

- **Filtering or monitoring**: $P(X_t|e_1,\ldots,e_t)$

  computing current belief state, given all evidence to date

- **Prediction**: $P(X_{t+k}|e_1,\ldots,e_t)$

  computing prob. of some future state

- **Smoothing**: $P(X_k|e_1,\ldots,e_t)$

  computing prob. of past state (hindsight)

- **Most likely explanation**: $\arg\max_{x1,..xt} P(x_1,\ldots,x_t|e_1,\ldots,e_t)$

  given sequence of observation, find sequence of states that is most likely to have generated those observations.

Examples

- **Filtering:** What is the probability that it is raining today, given all the umbrella observations up through today?
- **Prediction:** What is the probability that it will rain the day after tomorrow, given all the umbrella observations up through today?
- **Smoothing:** What is the probability that it rained yesterday, given all the umbrella observations through today?
- **Most likely explanation:** if the umbrella appeared the first three days but not on the fourth, what is the most likely weather sequence to produce these umbrella sightings?

Filtering

- We use recursive estimation to compute $P(X_{t+1} \mid e_{1:t+1})$ as a function of e_{t+1} and $P(X_t \mid e_{1:t})$
- We can write this as follows:

$$
\begin{aligned}
P(X_{t+1} \mid e_{1:t+1}) &= P(X_{t+1} \mid e_{1:t}, e_{t+1}) \\
&= \alpha P(e_{t+1} \mid X_{t+1}, e_{1:t}) P(X_{t+1} \mid e_{1:t}) \\
&= \alpha P(e_{t+1} \mid X_{t+1}) P(X_{t+1} \mid e_{1:t}) \\
&= \alpha P(e_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t) P(x_t \mid e_{1:t})
\end{aligned}
$$

- This leads to a recursive definition
  - $f_{1:t+1} = \alpha \mathrm{FORWARD}(f_{1:t:t}, e_{t+1})$

Umbrella Example

Day 0 – no observations $P(R_0)=(0.5,0.5)$ *(note: using “(“ “)” instead of “<“ “>”)*

Day 1 – umbrella appears, U_1=true,

prediction t=0 to t=1 is

$P(r_1)=\Sigma_{r0}\ P(R_1|r_0)P(r_0)=(0.7,0.3)*0.5+(0.3,0.7)*0.5=(0.5.0.5)$

update of evidence for t=1

$P(R_1|u_1)=\alpha P(u_1|R_1)P(R_1)= \alpha(0.9,0.2)(0.5,0.5)= \alpha(0.45,0.1)\approx(0.818,0.182)$

Day 2 – umbrella appears, U2=true,

prediction t=1 to t=2,

$P(r_2|u_1)=\Sigma_{r0}\ P(R_2|r_1)P(r_1|u_1)=(0.7,0.3)*0.818+(0.3,0.7)*0.182 \approx(0.627,0.373)$

update of evidence for t=2

$P(R_2|u_1,u_2)= \alpha P(u_2|R_2)P(R_2|u_1)= \alpha(0.9,0.2)(0.627,0.373) \approx(0.883,0.117)$

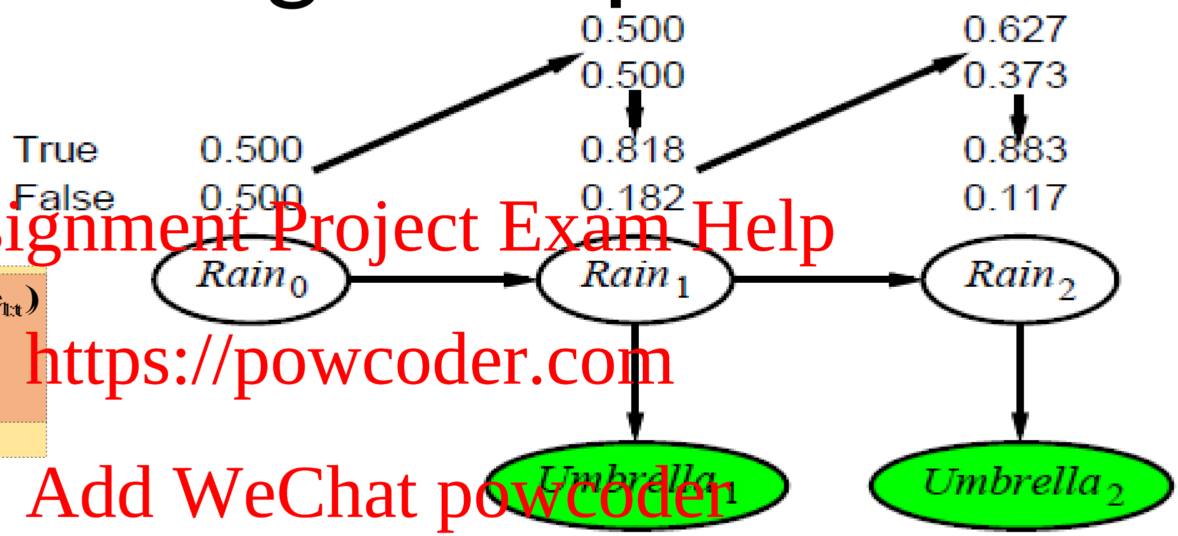
Forward Filtering Example

$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}, \mathbf{e}_{t+1})$$
$$= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$$
$$= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$$
$$= \alpha P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1} | \mathbf{x}_t) P(\mathbf{x}_t | \mathbf{e}_{1:t})$$

Prediction

Update

$$\mathbf{f}_{1:t+1} = \alpha\, \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$$



prediction t=0 to t=1 is

$P(r_1)=\Sigma_{r0}\ P(R_1|r_0)P(r_0)$

$=(0.7,0.3)*0.5+(0.3,0.7)*0.5=(0.5.0.5)$

update of evidence for t=1

$P(R_1|u_1)=\alpha P(u_1|R_1)P(R_1)= \alpha(0.9,0.2)(0.5,0.5)$

$= \alpha(0.45,0.1)\approx(0.818,0.182)$

prediction t=1 to t=2,

$P(r_2|u_1)=\Sigma_{r0}\ P(R_2|r_1)P(r_1|u_1)$

$=(0.7,0.3)*0.818+(0.3,0.7)*0.182 \approx(0.627,0.373)$

update of evidence for t=2

$P(R_2|u_1,u_2)= \alpha P(u_2|R_2)P(R_2|u_1)$

$= \alpha(0.9,0.2)(0.627,0.373) \approx(0.883,0.117)$

Smoothing

- Compute $P(X_k | e_{1:t})$ for $0 <= k < t$
- Using a backward message $b_{k+1:t} = P(E_{k+1:t} | X_k)$, we obtain
  - $P(X_k | e_{1:t}) = \alpha f_{1:k} b_{k+1:t}$

- The backward message can be computed using

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

- This leads to a recursive definition
  - $B_{k+1:t} = \alpha \text{BACKWARD}(b_{k+2:t}, e_{k+1:t})$

Umbrella Example

Computed smoothed estimate prob rain at k=1 given umbrella observations on days 1 and 2:

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

First term is (0.818,0.182) from filtering example.

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

= (0.9*1*(0.7,0.3))+(0.2*1*(0.3,0.7))=(0.69,0.41)

giving us

$P(R_1|u_1,u_2)= \alpha(0.818,0.182)(0.69,0.41)\approx(0.883,0.117)$

Note – smoothed estimate higher than filtered estimate

Most Likely Explanation: Viterbi Algorithm,

Most likely sequence $\neq$ sequence of most likely states!!!!

Most likely path to each $\mathbf{x}_{t+1}$

$=$ most likely path to some $\mathbf{x}_t$ plus one more step

$$\max_{\mathbf{x}_1 \ldots \mathbf{x}_t} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{X}_{t+1} | \mathbf{e}_{1:t+1})$$

$$= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \left(\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{e}_{1:t}) \right)$$

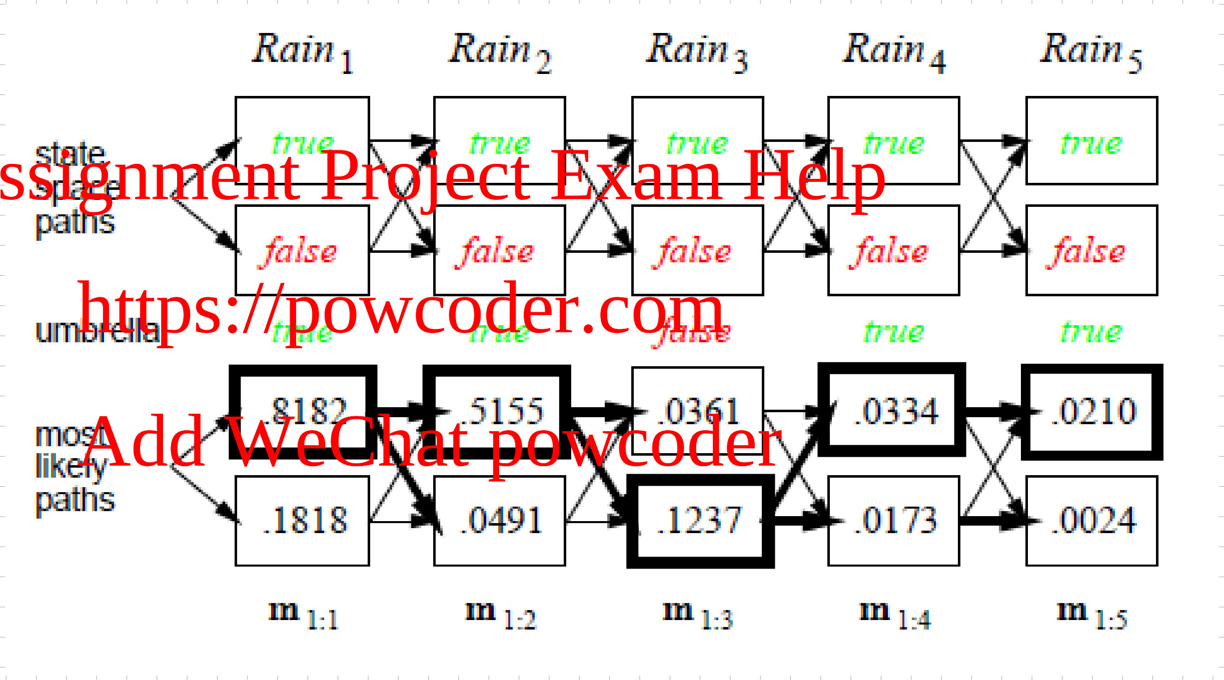
Identical to filtering, except $\mathbf{f}_{1:t}$ replaced by

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{X}_t | \mathbf{e}_{1:t}),$$

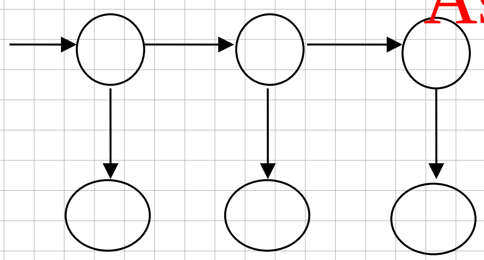
I.e., $\mathbf{m}_{1:t}(i)$ gives the probability of the most likely path to state i.
Update has sum replaced by max, giving the Viterbi algorithm:

$$\mathbf{m}_{1:t+1} = \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} (\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \mathbf{m}_{1:t})$$

Viterbi Example



Hidden Markov Model



$\mathbf{X}_t$ is a single, discrete variable (usually $\mathbf{E}_t$ is too)
Domain of X_t is $\{1, \ldots, S\}$

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

Sensor matrix $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t | X_t = i)$
e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages as column vectors:

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t}$$
$$\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t}$$

$$\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t}, \mathbf{e}_{t+1})$$
$$= \alpha \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$$
$$= \alpha \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{e}_{1:t})$$
$$= \alpha \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \mathbf{P}(\mathbf{x}_t | \mathbf{e}_{1:t})$$

$$\mathbf{f}_{1:t+1} = \alpha \, \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$$

HMM Example: Robot Localization

X_t - location of robot on grid $\{s1,\ldots,sn\}$

NEIGHBORS(s) = { empty squares adjacent to s }

N(s) = size of NEIGHBORS(s)

Transition Model

$$P(X_{t+1}=j \mid X_t=i) = \mathbf{T}_{ij} = (1/N(i) \text{ if } j \in \textsc{Neighbors}(i) \text{ else } 0)$$

Priors

$P(X_o=i)=1/n$ If n=42, T has 42x42=1764 entries

HMM Example: Robot Localization

E_t has 16 possible values – each giving the presence or absence of an obstacle in the N, S, E, W directions from the robot; e.g. e=NS

Assume error rate is ε, independent;

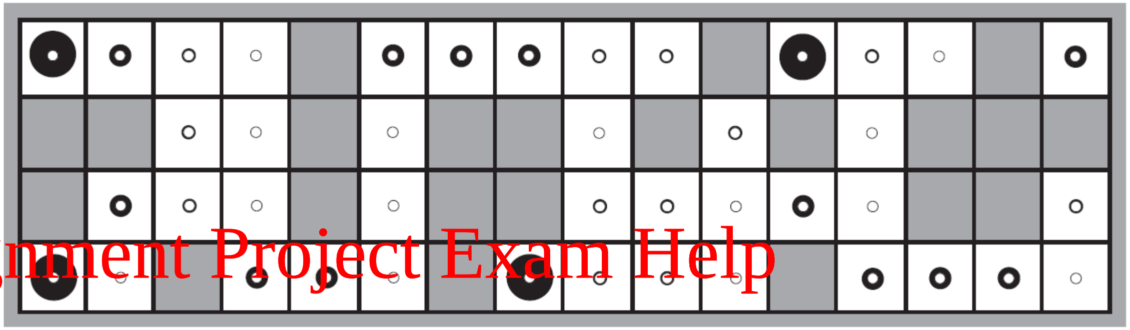
Probability of all four returns correct is $(1-\varepsilon)^4$ and ε^4 all wrong

$$P(E_t = e_t \mid X_t = i) = \mathbf{O}_{t_{ii}} = (1-\epsilon)^{4-d_{it}}\epsilon^{d_{it}}$$

Discrepancy d_{it} is number of different return bits.

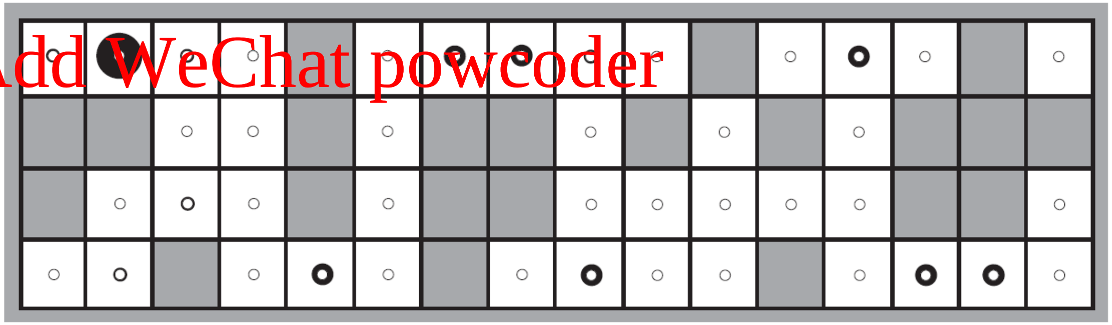
For example, the probability that a square with obstacles to the north and south would produce a sensor reading NSE is $(1-\varepsilon)^3\ \varepsilon^1$

$$\mathbf{P}(X_1 \mid E_1 = NSW)$$



(a) Posterior distribution over robot location after $E_1 = NSW$

$$\mathbf{P}(X_2 \mid E_1 = NSW, E_2 = NS)$$



(b) Posterior distribution over robot location after $E_1 = NSW, E_2 = NS$

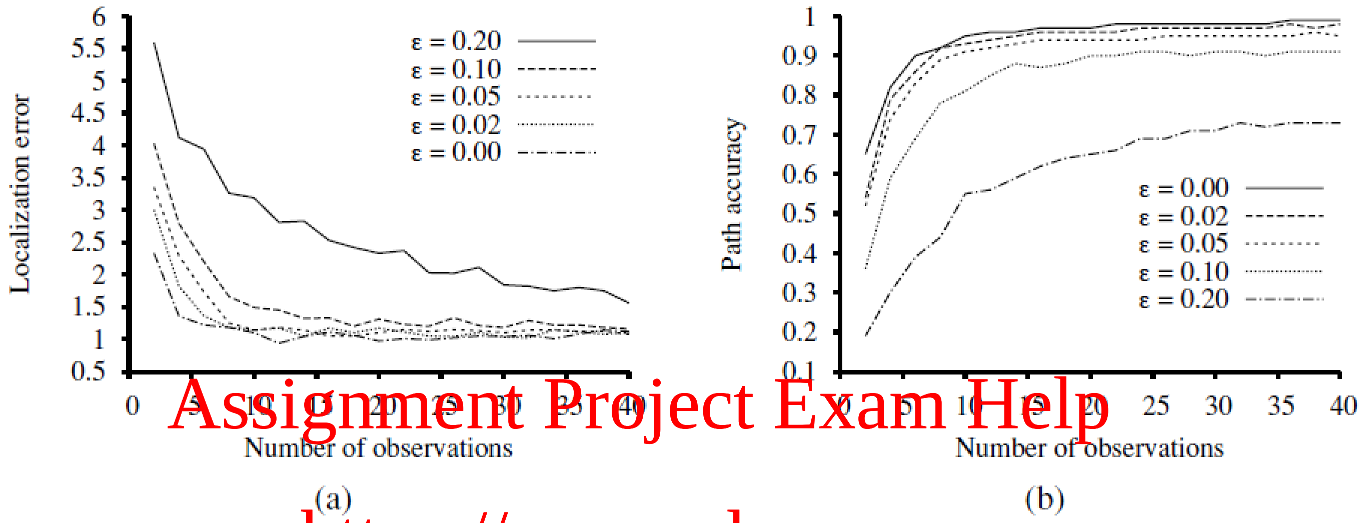


Figure 15.8 Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path accuracy, defined as the fraction of correct states on the Viterbi path.

- Even when ε is 20%—which means that the overall sensor reading is wrong 59% of the time—the robot is usually able to work out its location within two squares after 25 observations.

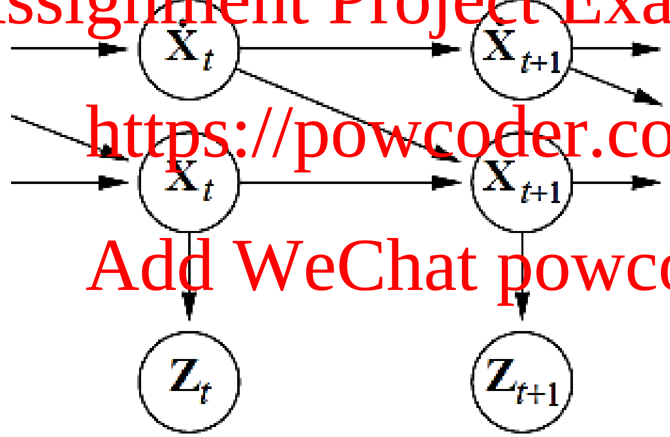
- When ε is 10%, the performance after a half-dozen observations is hard to distinguish from the performance with perfect sensing.

Kalman Filters

Modelling systems described by a set of continuous variables,

e.g., tracking a bird flying—$\mathbf{X}_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$.

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



Linear Gaussian
Next state is a linear function of the current state PLUS Gaussian noise

Bayes network structure for a linear dynamical system

Gaussian prior, linear Gaussian transition model and sensor model

$$P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_t + \dot{x}_t\,\Delta, \sigma^2)(x_{t+\Delta})$$

Updating a Gaussian distribution

Prediction step: if $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ is Gaussian, then prediction

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) = \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t})\, d\mathbf{x}_t$$

is Gaussian. If $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ is Gaussian, then the updated distribution

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

is Gaussian

Hence $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all t

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

Simple 1D Example: Random Walk

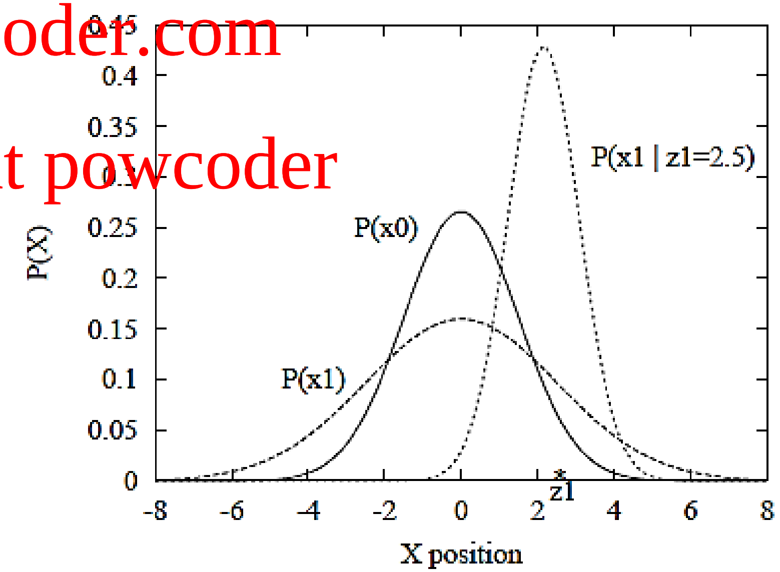
$$P(x_0) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}.$$

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

$$P(z_t \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)}.$$

Gaussian random walk on X–axis, s.d. σ_x, sensor s.d. σ_z

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



General (Multivariate) Kalman Update

$$N(\mu, \Sigma)(\mathbf{x}) = \alpha\, e^{-\frac{1}{2}\left((\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)\right)}$$

Transition and sensor models:

$$\begin{aligned} P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{F}\mathbf{x}_t, \Sigma_x)(\mathbf{x}_{t+1}) \\ P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{H}\mathbf{x}_t, \Sigma_z)(\mathbf{z}_t) \end{aligned}$$

$\mathbf{F}$ is the matrix for the transition; Σ_x the transition noise covariance
$\mathbf{H}$ is the matrix for the sensors; Σ_z the sensor noise covariance

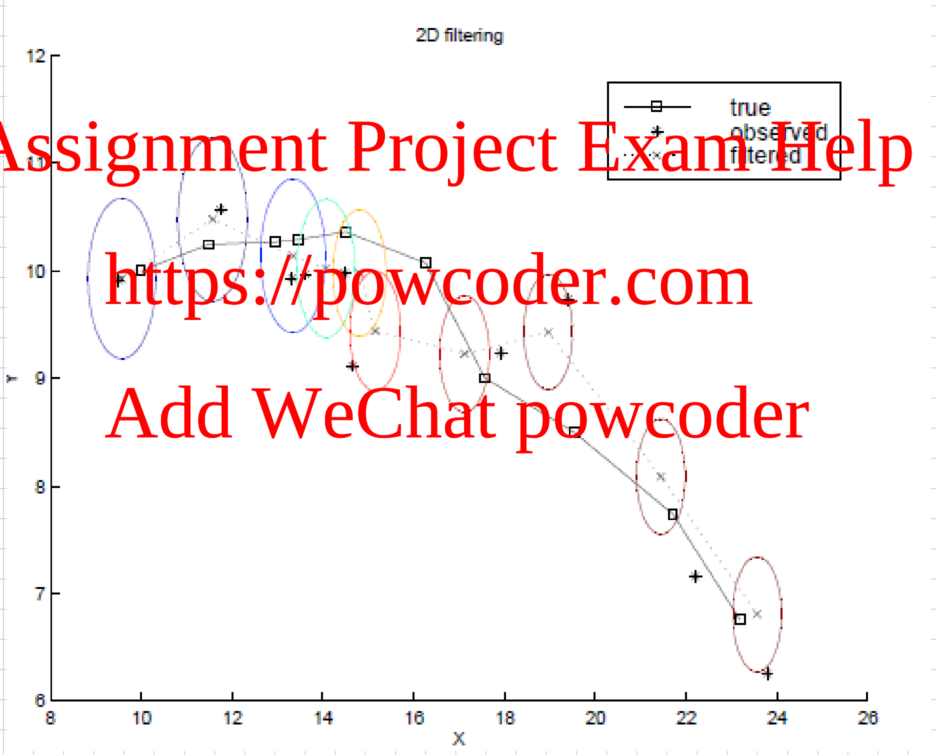
Filter computes the following update:

$$\begin{aligned} \mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\ \Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x) \end{aligned}$$

where $\mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}$
is the Kalman gain matrix

Σ_t and $\mathbf{K}_t$ are independent of observation sequence, so compute offline

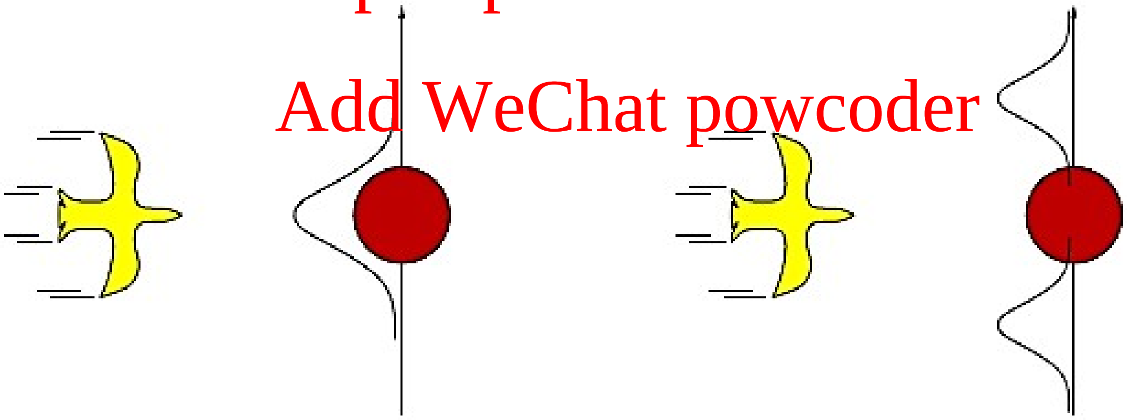
Tracking Example: Filtering



Where it Breaks!

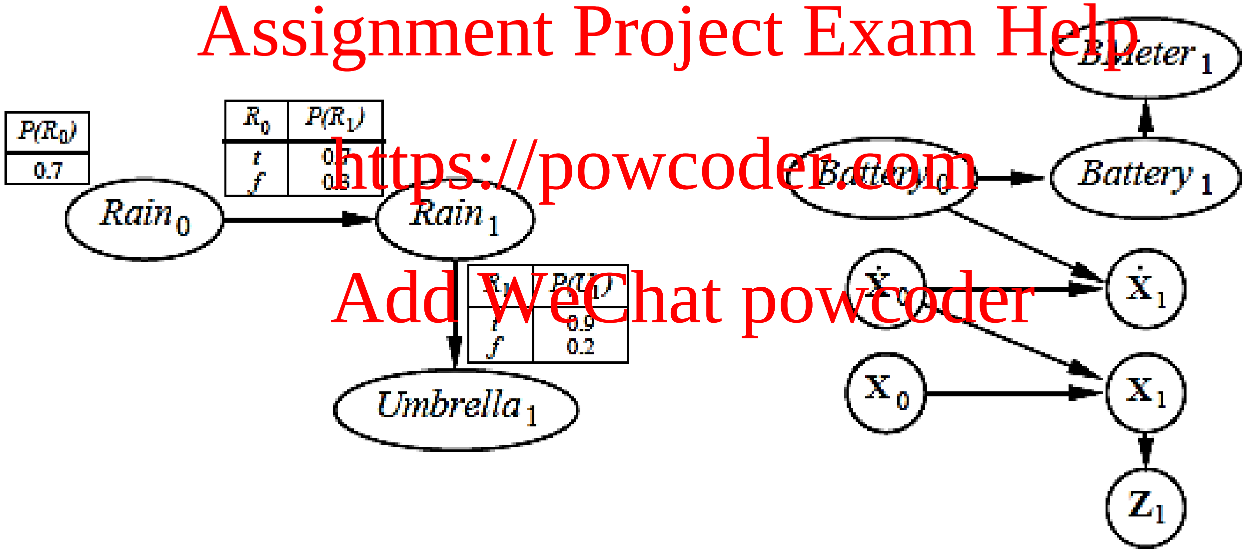
Cannot be applied if the transition model is nonlinear

Extended Kalman Filter models transition as locally linear around $\mathrm{x}_t = \mu_t$
Fails if systems is locally unsmooth



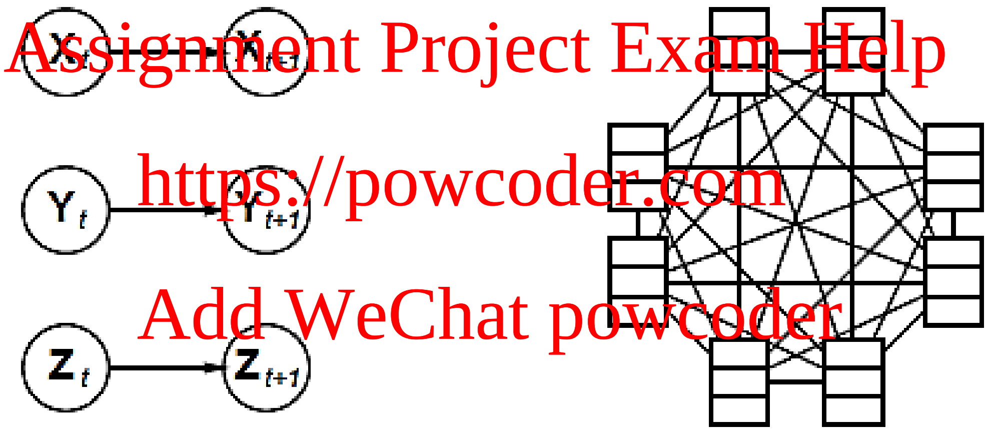
Remember
Dynamic Bayesian Networks?

$\mathbf{X}_t$, $\mathbf{E}_t$ contain arbitrarily many variables in a replicated Bayes net



DBN versus HMM ??

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



Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

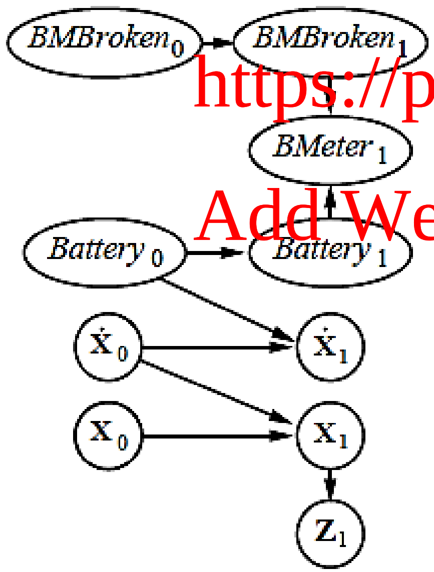
e.g., 20 state variables, three parents each

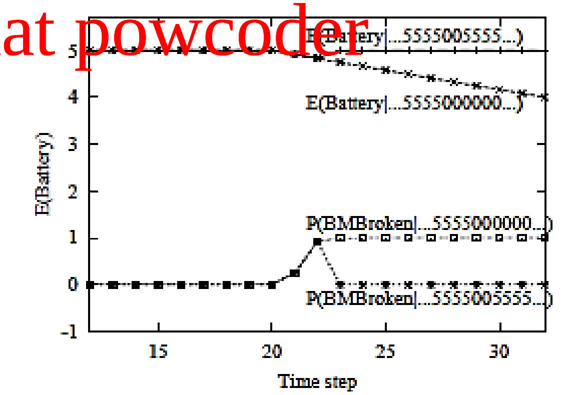
DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

DBN versus KF ??

Every Kalman filter model is a DBN, but few DBNs are KFs;
real world requires non-Gaussian posteriors

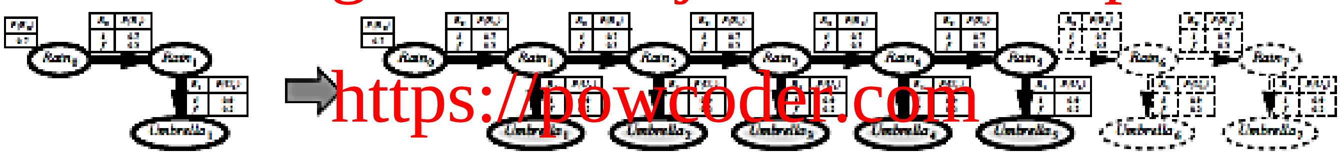
E.g., where are bin Laden and my keys? What's the battery charge?





Exact Inference in DBNs

Naive method: unroll the network and run any exact algorithm



Problem: inference cost for each update grows with t

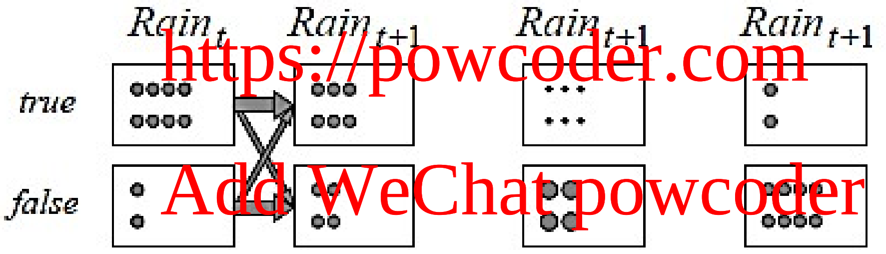
Rollup filtering: add slice $t + 1$, "sum out" slice t using variable elimination

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

Inexact Inference: 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



(a) Propagate (b) Weight (c) Resample

Widely used for tracking nonlinear systems, esp. in vision

Also used for simultaneous localization and mapping in mobile robots

10^5-dimensional state space

Particle Filtering

Assume consistent at time t: $N(\mathbf{x}_t|\mathbf{e}_{1:t})/N = P(\mathbf{x}_t|\mathbf{e}_{1:t})$

Propagate forward: populations of $\mathbf{x}_{t+1}$ are

$$N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) = \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t)N(\mathbf{x}_t|\mathbf{e}_{1:t})$$

Weight samples by their likelihood for $\mathbf{e}_{t+1}$:

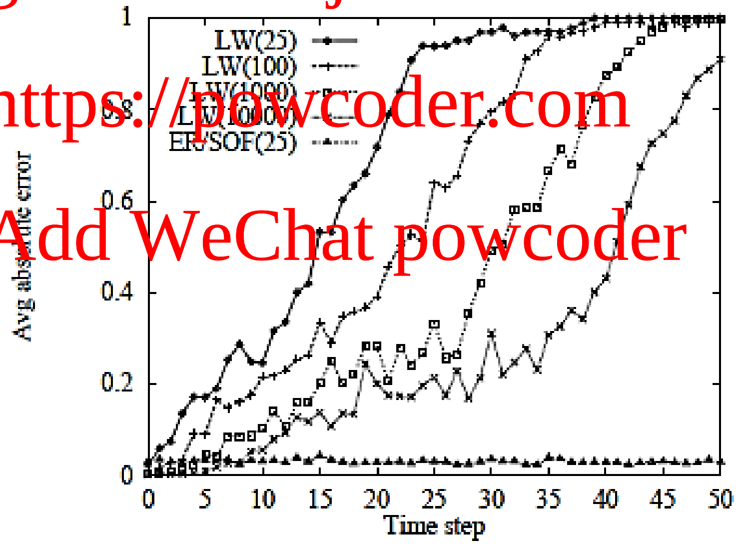
$$W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1})N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t})$$

Resample to obtain populations proportional to W:

$$\begin{aligned}
N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})/N &= \alpha W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1})N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) \\
&= \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1})\Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t)N(\mathbf{x}_t|\mathbf{e}_{1:t}) \\
&= \alpha' P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1})\Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\
&= P(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})
\end{aligned}$$

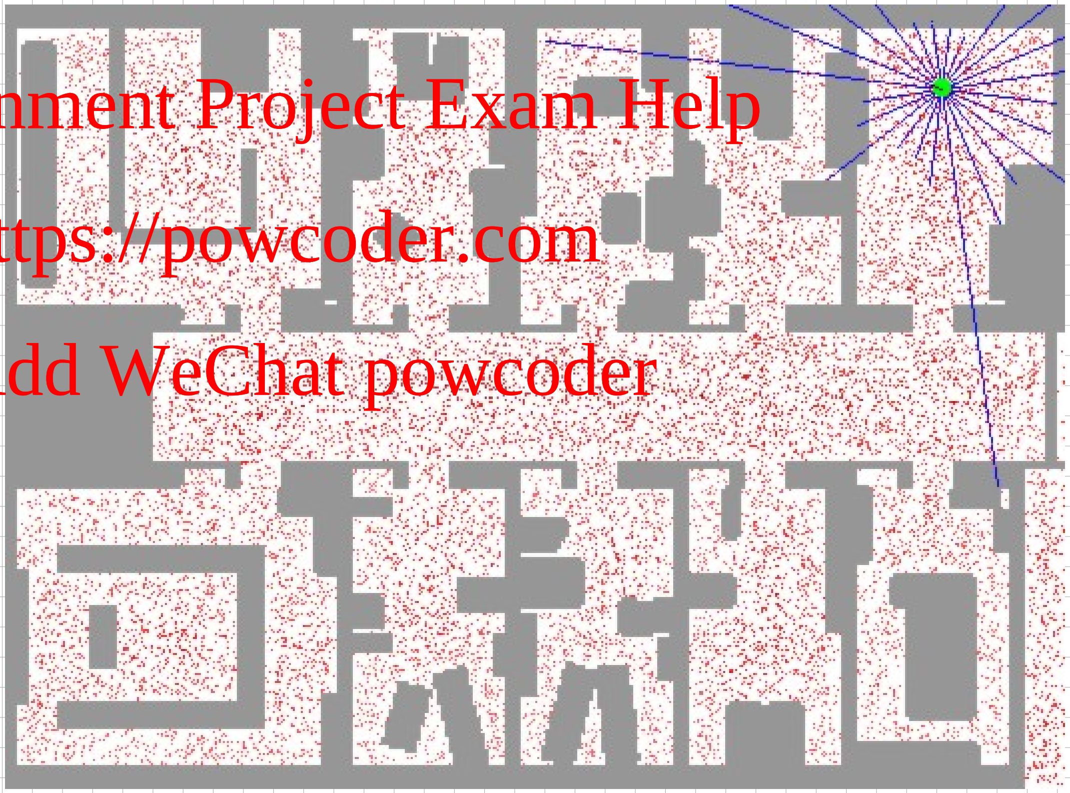
Particle Filtering

Approximation error of particle filtering remains bounded over time, at least empirically—theoretical analysis is difficult



Localization using PF

- Localization using Sonar
- Red dots are possible poses



Summary

Temporal models use state and sensor variables replicated over time

Markov assumptions and stationarity assumption, so we need
- transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
- sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

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 for speech recognition

Kalman filters allow n state variables, linear Gaussian, $O(n^3)$ update

Dynamic Bayes nets subsume HMMs, Kalman filters; exact update intractable

Particle filtering is a good approximate filtering algorithm for DBNs