CS 561 Artificial Intelligence
Lecture # 7-10

Probabilistic Reasoning Over Time

Rashmi Dutta Baruah

Department of Computer Science & Engineering

IIT Guwahati

Outline

- Time and Uncertainty
- Inference in Temporal Models
  - Filtering and Prediction
  - Smoothing
  - Most Likely explanation
- Hidden Markov Models
- Kalman filters
- Dynamic Bayesian Networks
  - Particle filters

Slides from 20 – 47 adapted from http://aima.eecs.berkeley.edu/

Time and Uncertainty

- The Bayesian networks we have seen so far describe static scenarios – each random variable has a single fixed value in a single problem instance.

- Now we will consider the problem of describing probabilistic environments that evolve over time –
  - Examples: target tracking, condition monitoring, making sense of spoken or written sequence of words, …

- World is viewed as a series of snapshots, or time slices, each of which contains a set of random variables, some observable and some not
  - same subset of variables is observable at each time slice

Time and Uncertainty

- Assumptions: discrete time steps, state sequence starts at $t = 0$ and evidence starts arriving at $t = 1$.
- Notations
  - $\mathbf{X}_t$: set of unobservable state variables at time t
  - $\mathbf{E}_t$: set of observable evidence variables at time t
  - $\mathbf{E}_t = \mathbf{e}_t$: observation at time t for some set of values
  - $\mathbf{X}_{a:b}$: the set of variables from $\mathbf{X}_a$ to $\mathbf{X}_b$
- Transition Model:
  - describes the probability distribution of the variables at time t, given the state of the world at past times (previous values)
- Sensor Model (or Observation Model):
  - describes the probability of each percept at time t, given the current state of the world

Transition and Sensor Models

- Transition model: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1})$
- Markov Assumption:
  - $\mathbf{X}_t$ depends on only finite fixed number of previous states
  - First-order Markov Process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
  - Second-order Markov Process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1},)$
- Sensor Model :
  - $\mathbf{E}_t$ could depend on previous variables as well as the current state variables, but we make sensor Markov assumption (any state should be capable of providing a precise sensor reading of itself)
  - $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$
- Assume stationary process: transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ and sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ are the same for all t

Complete Joint Distribution

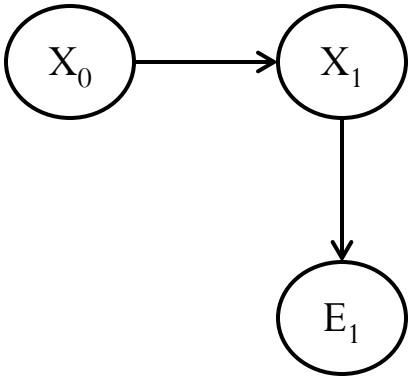
- Given:
  - Transition model: $\mathbf{P(X_t|X_{t-1})}$
  - Sensor model: $\mathbf{P(E_t|X_t)}$
  - Prior probability distribution at time $t = 0$: $\mathbf{P(X_0)}$
- Then we can specify complete joint distribution:

$$\mathbf{P(X_{0:t}, E_{1:t})} = \mathbf{P(X_0)} \prod_{i=1}^{t} \mathbf{P(X_i|X_{i-1})P(E_i|X_i)}$$

$$\mathbf{P(X_0, X_1, E_1)} = \mathbf{P(E_1|X_{1,}X_{0,})\ P(X_{1,}X_0)}$$

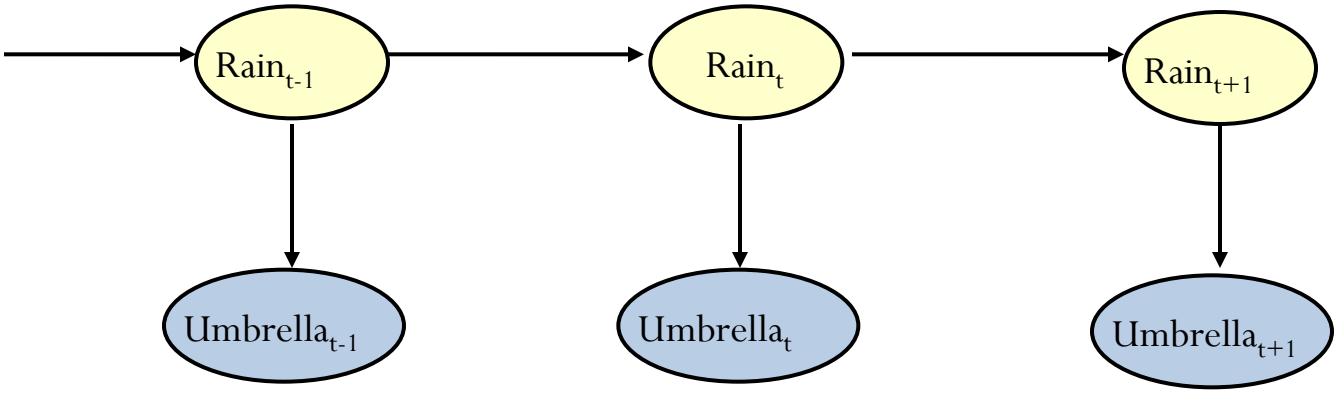
$$= \mathbf{P(E_1|X_1)P(X_1|X_0)\ P(X_0)}$$

$$= \mathbf{P(X_0)} \prod_{i=1}^{1} \mathbf{P(X_i|X_{i-1})P(E_i|X_i)}$$



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

Inference in Temporal Models

- Filtering (state estimation) $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$
  - computing current belief state (the posterior distribution over the most recent state) given all evidence to date;
- Prediction $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$
  - computing the posterior distribution over the future state, given all evidence to date
- Smoothing $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$
  - computing the posterior distribution over a past state, given all the evidence up to the present.
- Most likely explanation: $\underset{\mathbf{x}_{1:t}}{\text{argmax}}\, \mathbf{P}(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$
  - given sequence of observations, find sequence of states that is most likely to have generated those observations.

Filtering and Prediction

- Given the result of filtering up to time t, the result for $t + 1$ is computed from the new evidence i.e.
  - recursive estimation is used to compute $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1})$ as a function of $\mathbf{e}_{t+1}$ and $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$

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

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) = \frac{\mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t})}{\mathbf{P}(\mathbf{e}_{1:t})} = \frac{\sum_{x_t} \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{x_t}, \mathbf{e}_{1:t})}{\mathbf{P}(\mathbf{e}_{1:t})}$$

from transition model

from current state distribution

Filtering and Prediction

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

Update belief as new observation arrives

Propagate belief

- This leads to a recursive definition:
  - $\mathbf{f}_{1:t+1} = \alpha\, \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$
  - where $\mathbf{f}_{1:t}$ is $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ can be considered as "message" propagated along the sequence, modified by each transition and updated by each new observation, the process begins with $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$

- Example: Compute $\mathbf{P}(\mathrm{R}_2|u_{1:2})$
  - day 0, no observations $\mathbf{P}(\mathrm{R}_0) = \langle 0.5, 0.5 \rangle$ prior belief
  - day 1, $U_1 = true$, prediction from $t = 0$ to $t = 1$ is

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

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

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

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

Filtering and Prediction

- Then update step simply multiplies by the probability of the evidence for $t = 1$ and normalizes

$$\mathbf{P}(R_1|u_1) = \alpha\mathbf{P}(u_1|R_1)\sum_{r_0}\mathbf{P}(R_1|r_0)P(r_0)$$

$$\mathbf{P}(R_1|u_1) = \alpha\mathbf{P}(u_1|R_1)\mathbf{P}(R_1) = \alpha\langle 0.9,0.2\rangle\langle 0.5,0.5\rangle$$

$$= \alpha\langle 0.45,0.1\rangle \approx \langle 0.818,0.182\rangle$$

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

- day 2, $U_2 = true$, the prediction from $t = 1$ *to* $t = 2$ *is*

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

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

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

- Update it with evidence for $t = 2$

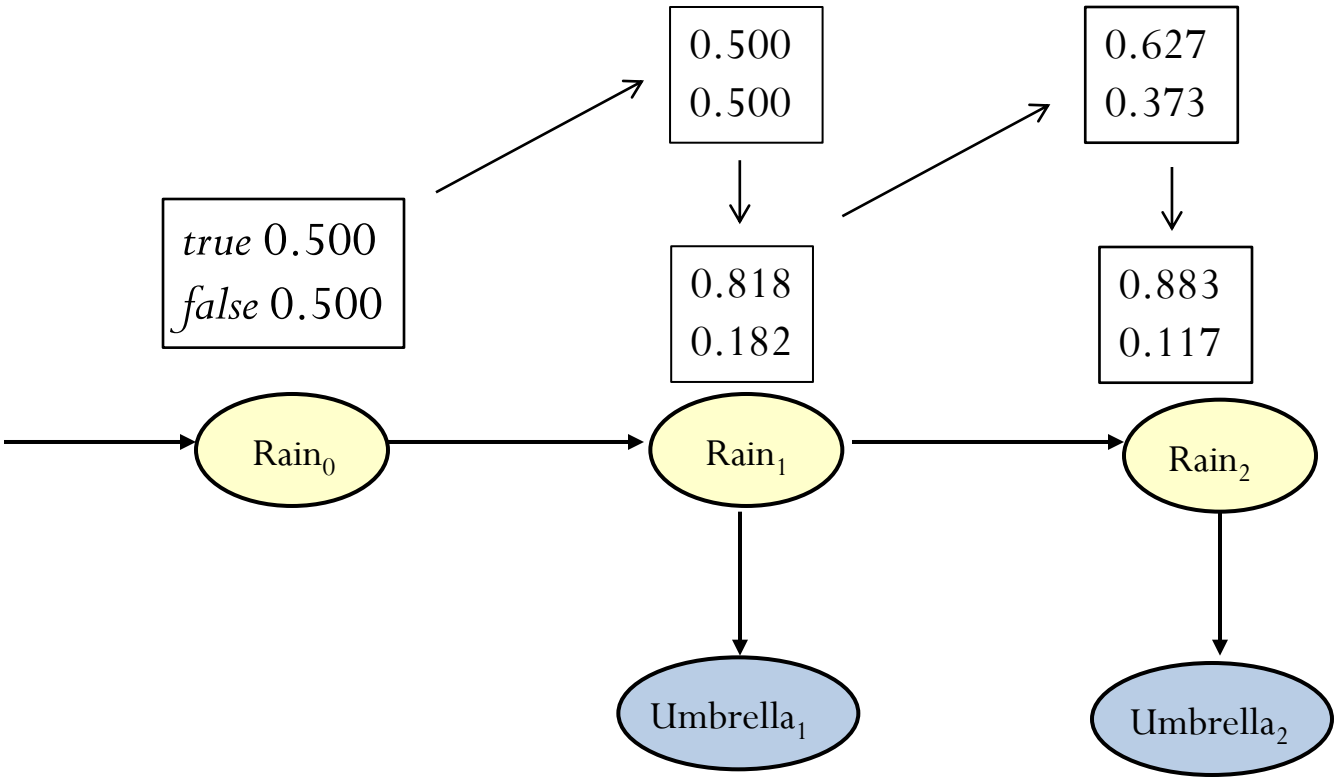
$$\mathbf{P}(R_2|u_1,u_2) = \alpha\mathbf{P}(u_2|R_2)\sum_{r_1}\mathbf{P}(R_2|r_1)P(r_1|u_1)$$

$$\mathbf{P}(R_2|u_1,u_2) = \alpha\mathbf{P}(u_2|R_2)\mathbf{P}(R_2|u_1) = \alpha\langle 0.9,0.2\rangle\langle 0.627,0.373\rangle$$

$$= \alpha\langle 0.565,0.075\rangle \approx \langle 0.883,0.117\rangle$$

Filtering and Prediction

- the probability of rain increases from day 1 to day 2 because it persists



Filtering and Prediction

- Prediction is filtering without the addition of new evidence:

$$\mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{e}_{1:t}) = \sum_{x_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{x}_{t+k}) P(\mathbf{x}_{t+k}|\mathbf{e}_{1:t})$$

- Note that this computation involves only the transition model and not the sensor model.

Smoothing

- Compute $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}), 0 \leq k < t$: distribution over past states given evidence up to the present

$$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}, \mathbf{e}_{k+1:t})$$
$$= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{e}_{1:k}) \text{ (using Bayes' rule)}$$
$$= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \text{ (using conditional independence)}$$
$$= \alpha\mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t} \text{ (" × " pointwise multiplication of vectors)}$$

$\mathbf{b}_{k+1:t}$

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k, \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} \mathrm{P}(\boldsymbol{e}_{k+1:t}|\mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} \mathrm{P}(\boldsymbol{e}_{k+1}, \boldsymbol{e}_{k+2:t}|\mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} \mathrm{P}(\boldsymbol{e}_{k+1}|\mathbf{x}_{k+1})\mathrm{P}(\boldsymbol{e}_{k+2:t}|\mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$$

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\boldsymbol{X}_k) = \frac{\mathbf{P}(\mathbf{e}_{k+1:t}, \boldsymbol{X}_k)}{\mathbf{P}(\boldsymbol{X}_k)} = \frac{\sum_{x_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}, \boldsymbol{X}_k, \mathbf{x}_{k+1})}{\mathbf{P}(\mathbf{X}_k)}$$

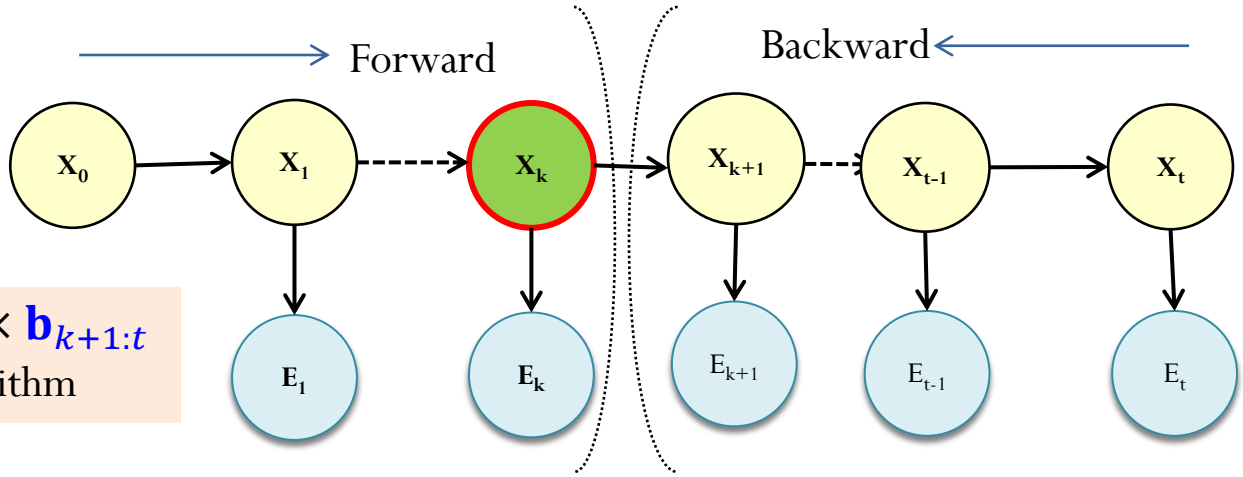
Recursively computed

$\mathbf{b}_{k+2:t}$

Apply conditional independence of $\boldsymbol{e}_{k+1}$ and $\boldsymbol{e}_{k+2:t}$ given $\boldsymbol{X}_{k+1}$

Smoothing

- This leads to a recursive definition
  - $\mathbf{b}_{k+1:t} = \alpha \text{BACKWARD}(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1:t})$ where BACKWARD implements the update $\sum_{\mathbf{x}_{k+1}} \mathbf{P}(\boldsymbol{e}_{k+1}|\mathbf{x}_{k+1})\mathbf{b}_{k+2:t}\mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$
  - time and space needed for each update are constant and independent of t
  - Backward phase is initialised with $b_{t+1:t} = \mathbf{P}(e_{t+1:t}|X_t) = \mathbf{P}(\_|X_t) = \mathbf{1}$ (vector of 1s)



$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$

Forward-backward algorithm

Smoothing

- Example: Estimate the probability of rain at time $k = 1$, given umbrella observations on days 1 and 2.

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

$$\mathbf{P}(\mathrm{R}_1|u_1, u_2) = \alpha\mathbf{P}(\mathrm{R}_1|u_1)\mathbf{P}(u_2|\mathrm{R}_1)$$

$\mathbf{P}(\mathrm{R}_1|u_1) = \langle 0.818, 0.182\rangle$, already computed using filtering

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathrm{x}_{k+1}} \mathrm{P}(\boldsymbol{e}_{k+1}|\mathbf{x}_{k+1})\mathrm{P}(\boldsymbol{e}_{k+2:t}|\mathbf{x}_{k+1})\,\mathbf{P}(\mathbf{x}_{k+1}|\boldsymbol{X}_k)$$

$$\mathbf{P}(u_2|\mathrm{R}_1) = \sum_{r_2} P(u_2|r_2)P(e_{3:2}|r_2)\mathbf{P}(r_2|R_1)$$

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

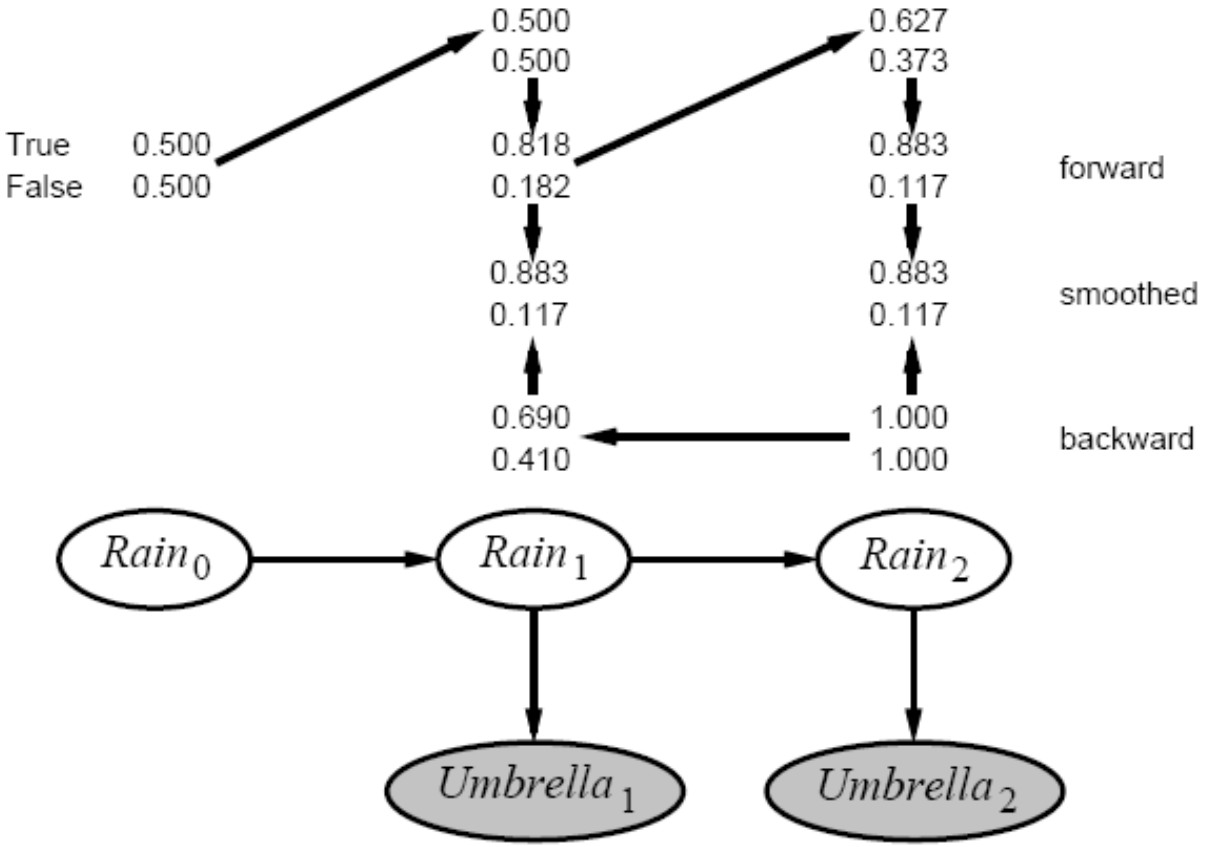
$$\mathbf{P}(\mathrm{R}_1|u_1, u_2) = \alpha\langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \approx \langle 0.883, 0.117\rangle$$

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

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

Smoothed estimate for rain on day 1is more than filtered estimate

Smoothing Example



Most likely Explanation

- Suppose that $[true, true, false, true, true]$ is the observed umbrella sequence for the first five days. What is the weather sequence most likely to explain this?
  - there are 2^5 possible weather sequences – which is the most likely
- Most likely sequence ≠ sequence of most likely states
  - the latter can be obtained by a combination of smoothing and filtering, the former cannot!
- Most likely path to each $\mathbf{x}_{t+1}$ = most likely path to some $\mathbf{x}_t$ plus one more step (recursive definition, due to the Markov property)

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

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

  - identical to filtering except $\mathbf{f}_{1:t}$ replaced by this term and summation is replaced by maximization over $\mathbf{x}_t$

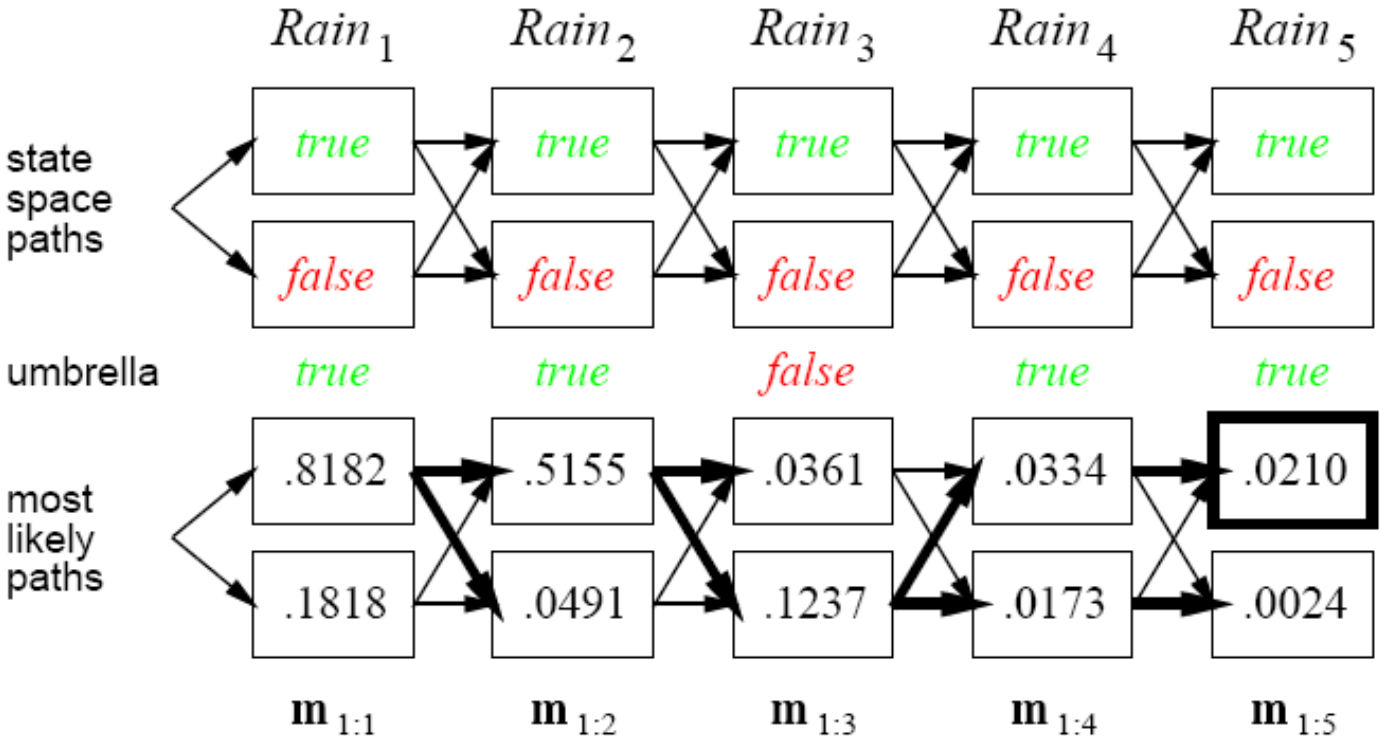
Most likely Explanation

- computing most likely sequence is similar to filtering
  - runs forward along the sequence, computing the $\mathbf{m}$ messages at each time step where $\mathbf{m}$ is given as

$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{X}_t | \mathbf{e}_{1:t})$ i.e. the probabilities of the most likely path to each state $\mathbf{x}_t$

- at the end this procedure will give the probability for the most likely sequence reaching each of the final states:
  - we can just select the most likely sequence over all
- in order to identify the actual sequence (as opposed to just computing the probability) the algorithm will also need to record, for each state, the best state that leads to it.

Viterbi algorithm



Viterbi algorithm : time and space complexity $o(t), t$ is the length of sequence

Hidden Markov Models

- HMM is a temporal probabilistic model in which the state of the process is described by a single discrete random variable

$\mathbf{X}_t$ 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:

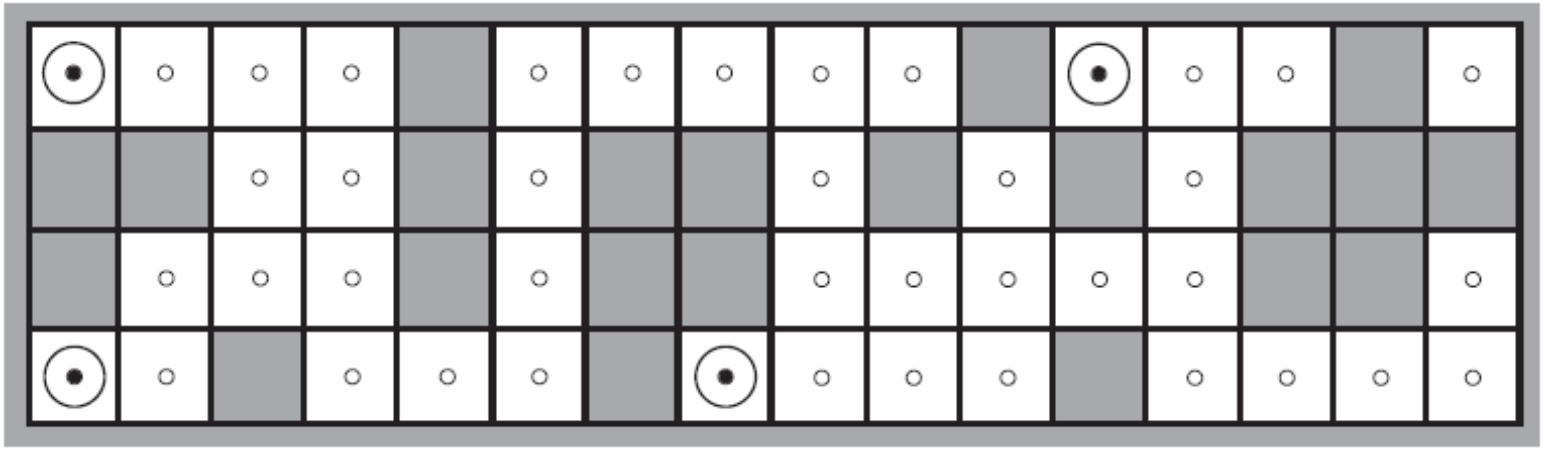
$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t}$$
$$\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t}$$

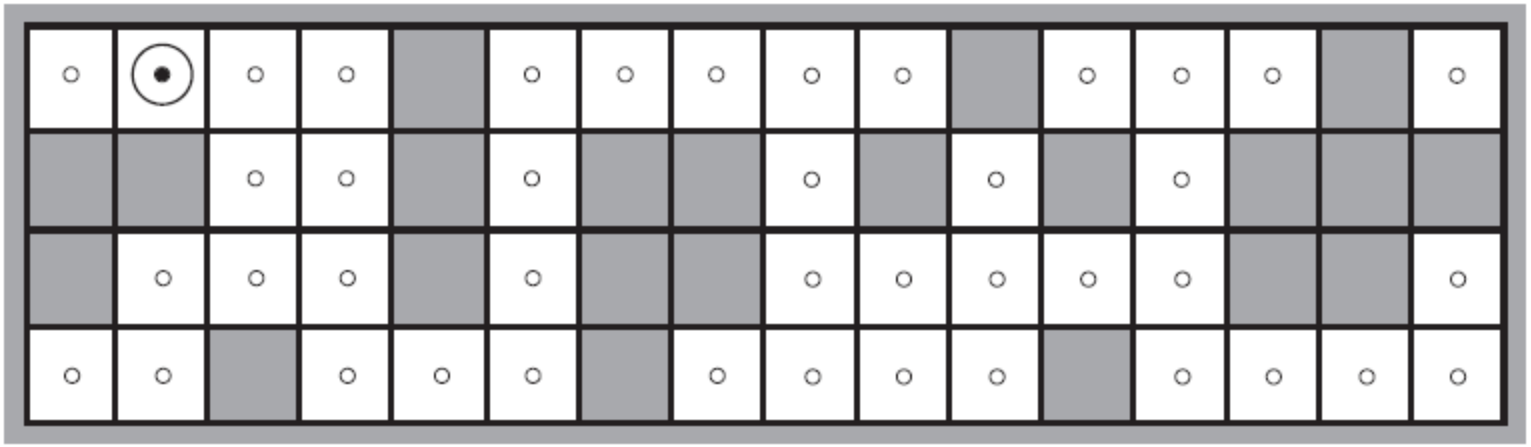
$$\boldsymbol{P}(\boldsymbol{X}_{t+1} | e_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} | \boldsymbol{X}_{t+1}) \sum_{x_t} P(\boldsymbol{X}_{t+1} | \boldsymbol{X}_t) P(\boldsymbol{x}_t | \boldsymbol{e}_{1:t})$$

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

HMM Example

- Robot localization





(b) Possible locations of robot After $E_1 = NSW, E_2 = NS$

Kalman Filter

- State estimation in linear dynamically systems(LDS), a Bayesian model (Dynamic Bayesian Network) similar to HMM but
  - state space of hidden variables is continuous
  - hidden and observed variables have a Gaussian distribution (i,e. dependencies are linear Gaussian)

- LDS represents a system of one or more real-valued variables that evolve linearly over time, with some Gaussian noise

- LDs are often used to model the dynamics of moving objects and to track their current positions given noisy measurements

Kalman Filter

- Kalman Filter model assumes that the state of a system at a time $t + 1$ evolved from the prior state at time t according to

$$\mathbf{X}_{t+1} = \mathbf{F}\boldsymbol{X}_t + \mathbf{B}\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1}$$

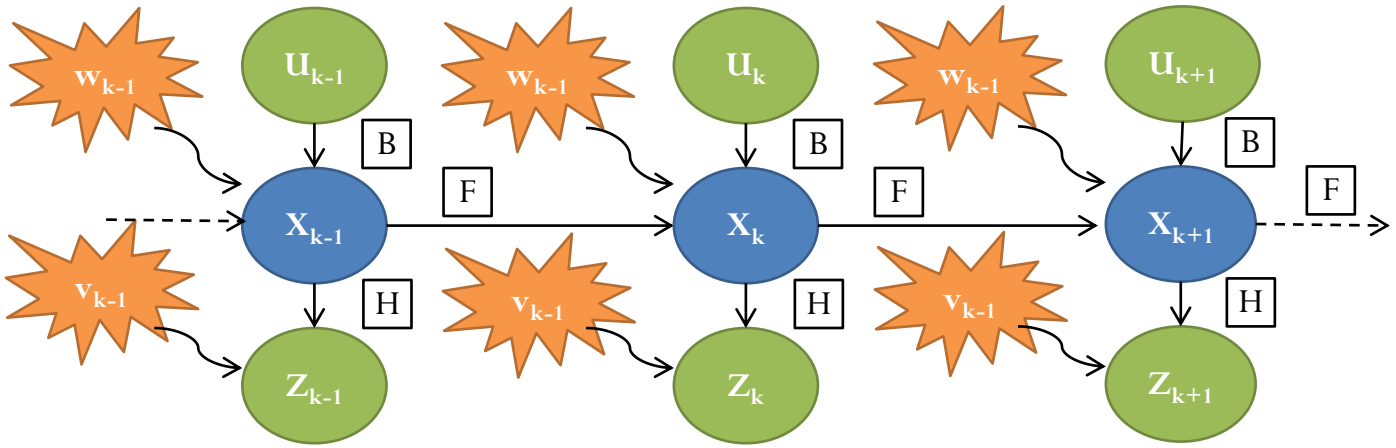
  - where $\mathbf{X}_{t+1}$ is the state vector containing the terms of interest (e.g. position, velocity, heading at time t+1),
  - $\boldsymbol{u}_{t+1}$ vector of control inputs (steering angle, braking force)
  - $\mathbf{F}$ is the state transition matrix,
  - $\mathbf{B}$ is the control input matrix that applies the effect of each input parameter in the vector $\boldsymbol{u}_{t+1}$ on the state vector (e.g. applies the effect of break force on the system velocity and position)
  - $\boldsymbol{w}_{t+1}$ noise vector containing noise terms for each parameter in the state vector-multivariate normal distribution with zero mean and $\mathbf{Q}_t$ covariance matrix

Kalman Filter

- System measurements (observations) can be made with the following model

$$\mathbf{Z}_{t+1} = \mathbf{H}\mathbf{X}_{t+1} + \mathbf{v}_{t+1}$$

  - Where $\mathbf{Z}_{t+1}$ vector of measurements (observations), $\mathbf{H}$ is the transformation matrix that maps the state vector parameters into the measurement domain (sensor model), $\mathbf{v}_{t+1}$ is the vector containing the measurement noise terms for each observation in the measurement vector- assumed to be zero mean Gaussian with covariance $\mathbf{R}_t$

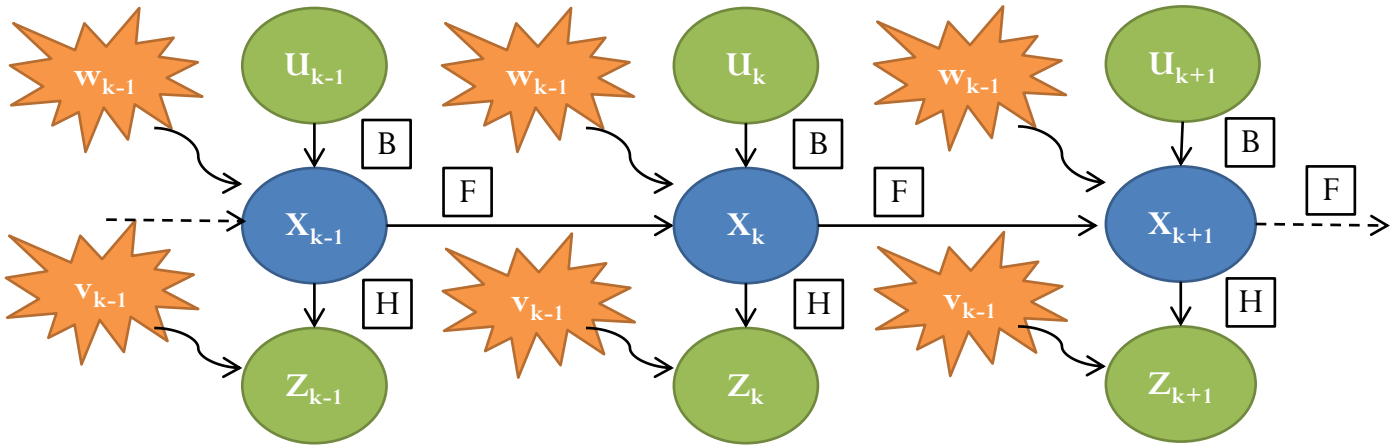


Kalman Filter

- System measurements (observations) can be made with the following model

$$\mathbf{Z}_{t+1} = \mathbf{H}\mathbf{X}_{t+1} + \mathbf{v}_{t+1}$$

  - Where $\mathbf{Z}_{t+1}$ vector of measurements (observations), $\mathbf{H}$ is the transformation matrix that maps the state vector parameters into the measurement domain (sensor model), $\mathbf{v}_{t+1}$ is the vector containing the measurement noise terms for each observation in the measurement vector- assumed to be zero mean Gaussian with covariance $\mathbf{R}_t$



Kalman Filter

- Example: 1-D Tracking

$$\mathbf{X}_{t+1} = \begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix}$$ State vector: position and velocity

Control input: driver may apply braking or accelerating input to the system, which is considered here as a function of an applied force f_t and mass of vehicle m.

$$\mathbf{u}_t = \frac{f_t}{m}$$

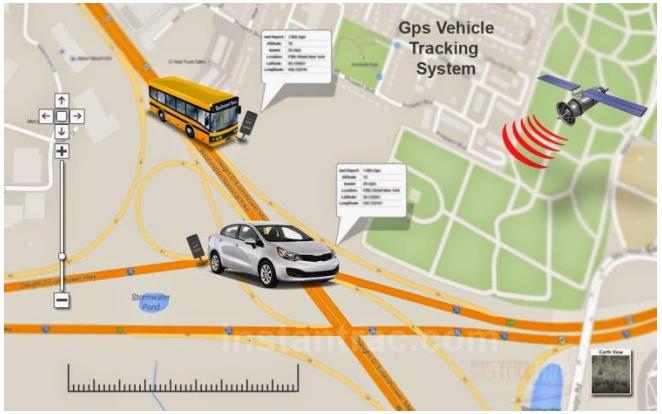


Image source: http://www.iconcox.com/

Position and velocity of the vehicle at t+1 can be given as :

$$x_{t+1} = x_t + \dot{x}_t \Delta t + \frac{f_t (\Delta t)^2}{2m} \qquad \dot{x}_{t+1} = \dot{x}_t + \frac{f_t \Delta t}{m}$$

The above linear equations can be written in matrix form:

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} + \begin{bmatrix} \frac{(\Delta t)^2}{2} \\ \Delta t \end{bmatrix} \frac{f_t}{m}$$

$$\mathbf{X}_{t+1} = \mathbf{F}\boldsymbol{X}_t + \mathbf{B}\boldsymbol{u}_{t+1}$$

Using kalman filter we can estimate the state $\hat{x}_{t+1}$ by combing models of the system and noisy measurements of certain parameters or linear functions of parameters.
Estimates are provided by probability density functions rather then discrete values.

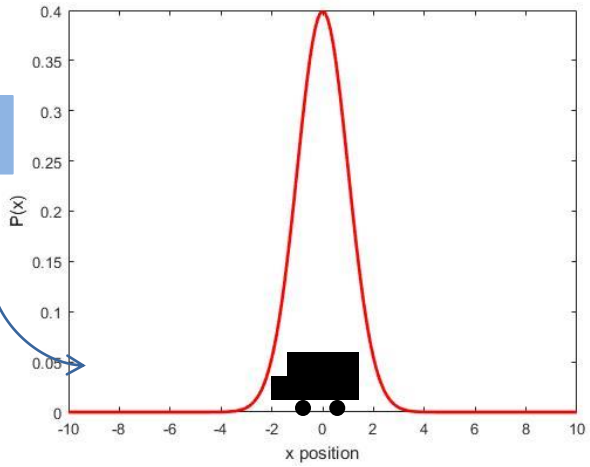
Kalman Filter

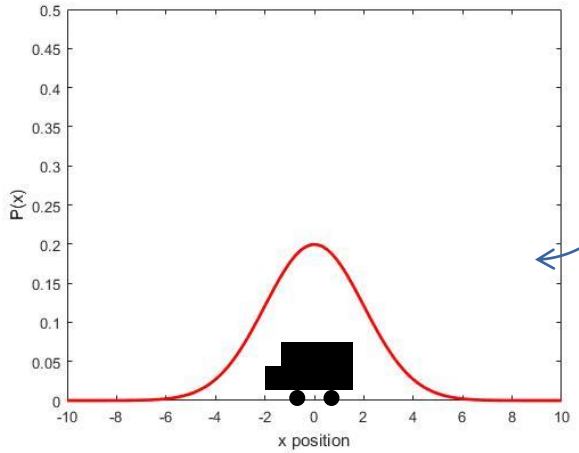
- allows recursive estimation of $\hat{\mathbf{x}}_{t+1}$ by combining prior knowledge, predictions from systems models, and noisy measurements
- Involves two stages:
  - time update (prediction)
  - measurement update (correction/update)
- 1-D tracking Example: Estimate the position of the vehicle when information is available from two sources(i) predictions based on the last known position and velocity of the vehicle (ii) measurement from some positioning system.
  - These two are combined to give the best possible estimate of the position of the vehicle.

Kalman Filter

- 1-D tracking Example:

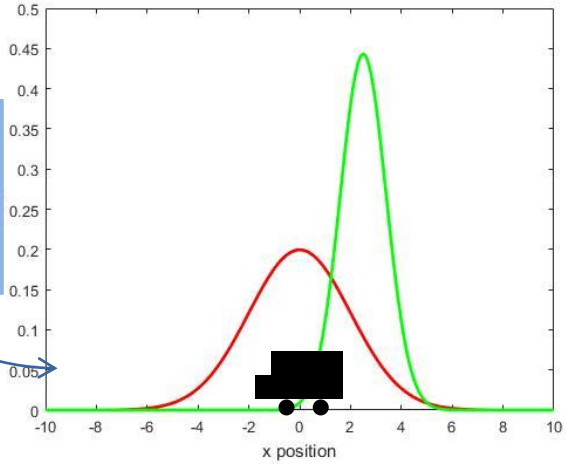
Prior belief

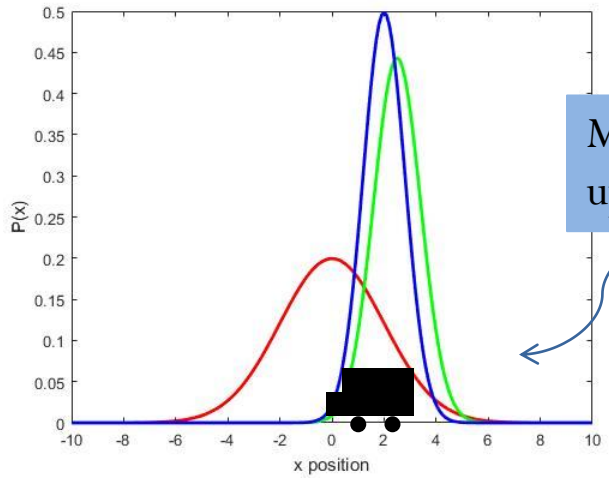




Time update

Measurement /observation available





Measurement update

Kalman Filter

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

Gaussian Prior

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

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

Sensor model assumes Gaussian noise with variance σ_z^2

Given the prior, the one step predicted distribution can be given as :

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

Product of two Gaussian Functions is another Gaussian function

Now, when z_1 is available do the correction or measurement update step:

$$P(x_1|z_1) = \alpha\, P(z_1|x_1)P(x_1) = \alpha e^{-\frac{1}{2}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2}\right)}$$

μ_1 or μ_{t+1} is in slide 32

Updating Gaussian distributions

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

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

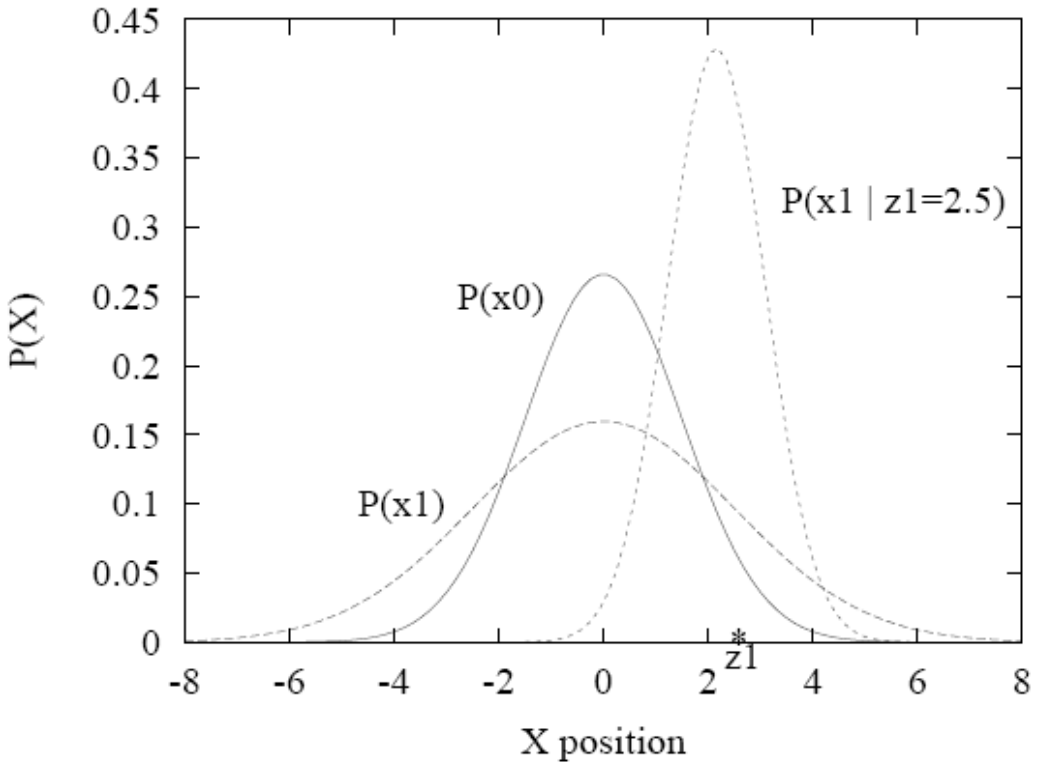
is Gaussian

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

Simple 1-D example

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 Kalman update

Transition and sensor models:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = N(\mathbf{F}\mathbf{x}_t, \boldsymbol{\Sigma}_x)(\mathbf{x}_{t+1})$$
$$P(\mathbf{z}_t|\mathbf{x}_t) = N(\mathbf{H}\mathbf{x}_t, \boldsymbol{\Sigma}_z)(\mathbf{z}_t)$$

$\mathbf{F}$ is the matrix for the transition; $\boldsymbol{\Sigma}_x$ the transition noise covariance
$\mathbf{H}$ is the matrix for the sensors; $\boldsymbol{\Sigma}_z$ the sensor noise covariance

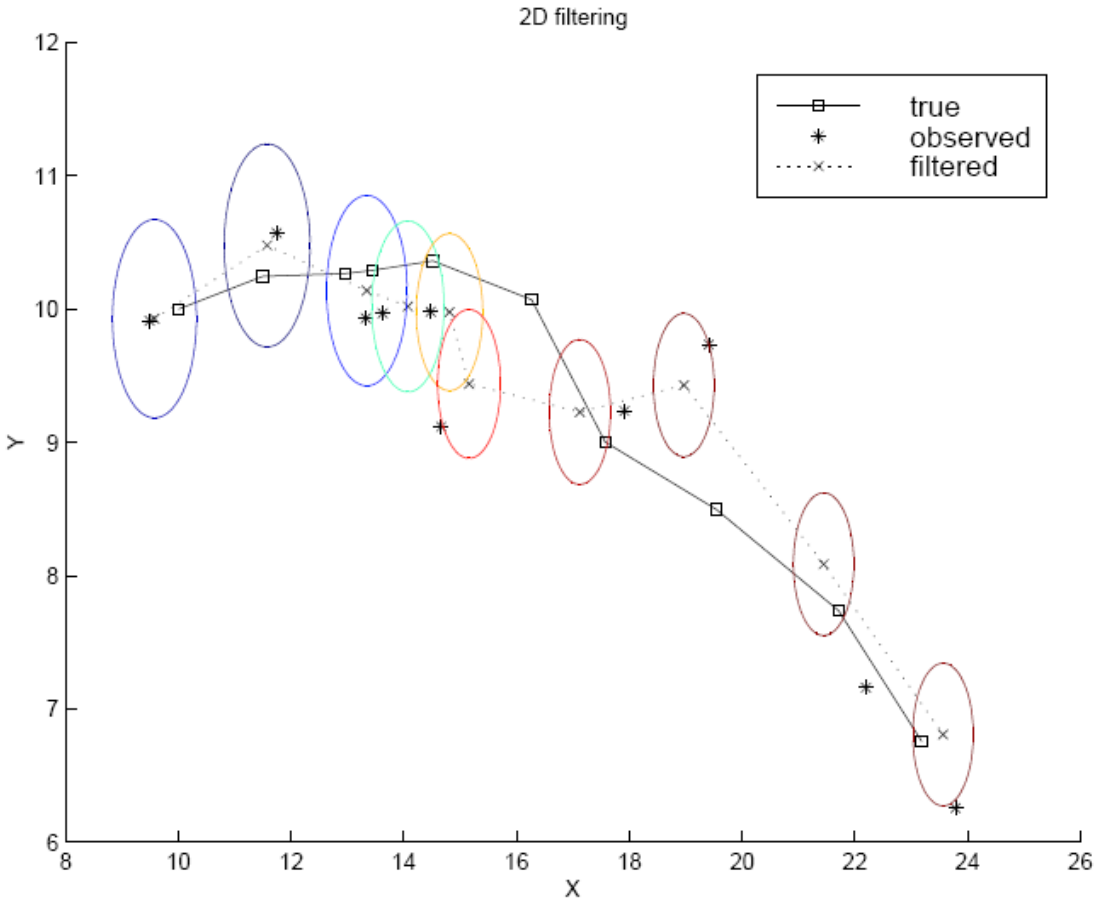
Filter computes the following update:

$$\boldsymbol{\mu}_{t+1} = \mathbf{F}\boldsymbol{\mu}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\boldsymbol{\mu}_t)$$
$$\boldsymbol{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)$$

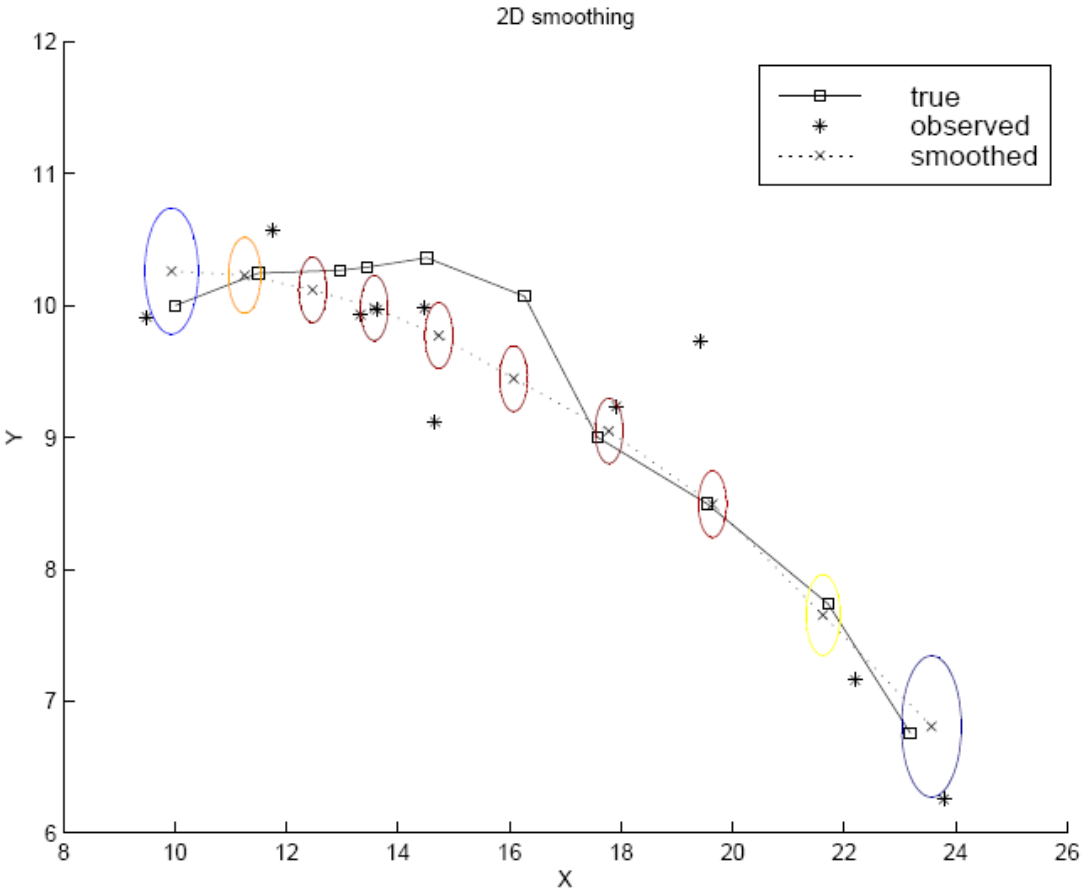
where $\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top + \boldsymbol{\Sigma}_z)^{-1}$
is the Kalman gain matrix

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

2-D tracking example: Filtering



2-D tracking example: smoothing



Dynamic Bayesian Networks

- Dynamic Bayesian Networks (DBN)
  - Each slice of DBN can have any number of state $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$, variables and their links same from slice to slice
  - First order Markov process – each variable can have parents only in its own slice or the immediately preceding slice

- Vehicle localization task: track current location using data obtained from possibly faulty sensor.
- System state can be encoded using:
  - Location, Velocity, Weather, Failure (failure status of the sensor), and Obs (the current observation)**: one such set of variables for every point t**

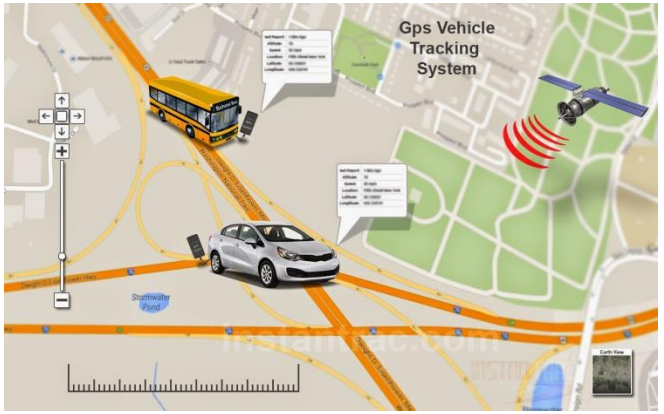
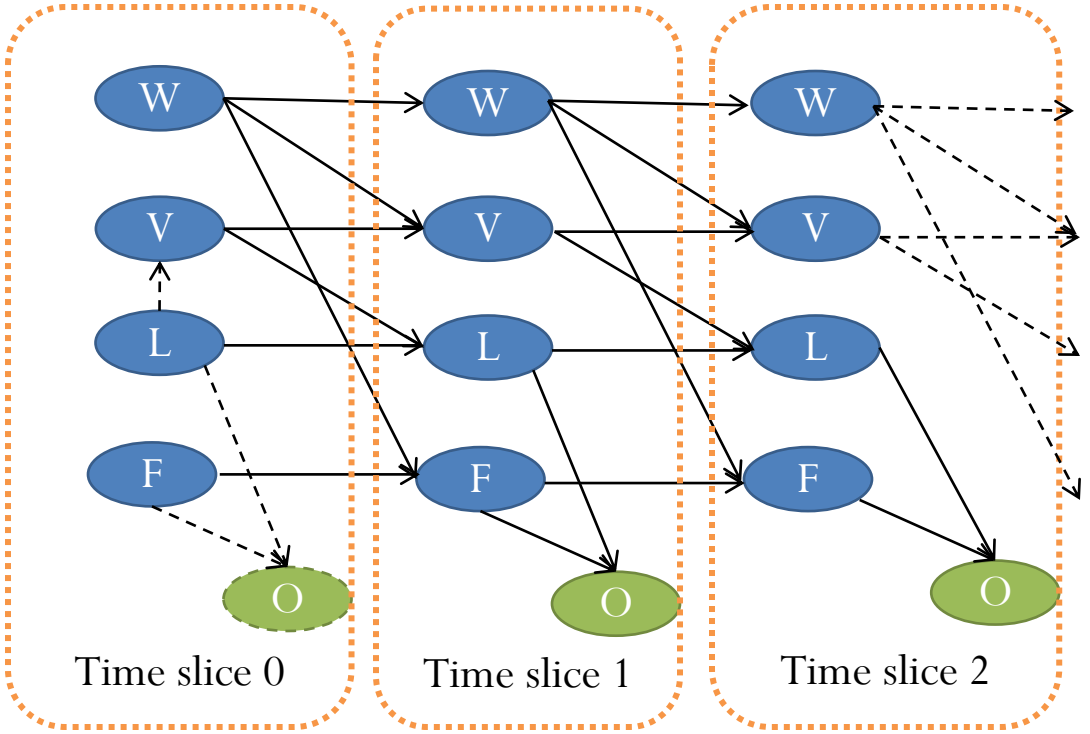


Image source: http://www.iconcox.com/

Given a sequence of observations about the car, where is it now?
Where is it likely to be in 10 minutes?
Did it stop at red light?

Dynamic Bayesian Networks

- current observation depends on vehicle's location (and the map, which is not explicitly modeled), and on the error status of the sensor
- bad weather makes the sensor more likely to fail
- vehicle's location depends on the previous position and velocity



W: Weather
V: Velocity
L: Location
F: Failure
O: Observation

DBN unrolled over three time steps

Dynamic Bayesian Networks

- HMM- simplest DBN with a single state variable and a single observation variable
- Discrete DBN- HMM with all the state variables in the DBN as a single state variable whose values are all possible tuples of values of the individual state variables

- When the state of a complex system is decomposed into its constituent variables, it can take advantage of sparseness in the temporal probability model
  - Sparse dependencies → exponentially fewer parameters
  - Example: 20 state variables, three parents each
  - DBN has $20 \times 2^3 = 160$ probabilities and corresponding HMM has 2^{20} states and so 2^{40} probabilities in transition matrix

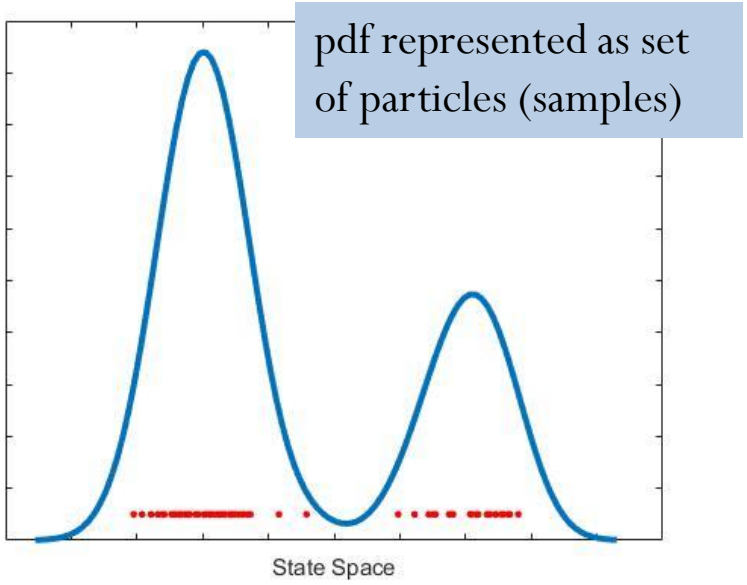
- Every Kalman Filter model is a DBN, but not all DBNs are KFs; real world requires non-Gaussian posteriors

Inference in DBNs

- DBNs are Bayesian networks – inference algorithms for Bayesian networks can be used by unrolling
- Unrolling the DBN- given a sequence of observations, replicate slices until the network accommodates the observations
- Recursive computations (e.g. in filtering and smoothing) require constant time update (independent of t), however, the "constant" time is exponential in the number of state variables.
- Approximate inference preferred over exact inference due to computational costs in networks with large number of variables.

Approximate inference in DBNs

- Particle Filter
  - **key idea**: at any given time, belief represented with a collection of particles (samples) drawn from the distribution
- We need to estimate the state i.e $P(\mathbf{X}_t|\boldsymbol{e}_{1:t})$ where $\boldsymbol{e}_{1:t}$ is the observation sequence.
  - Each particle contains one set of values for the state variables
  - We want to draw the samples from the posterior $P(\mathbf{X}_t|\boldsymbol{e}_{1:t})$, but we don't have explicit representation of the posterior
  - But we have prior belief, sample from prior and then weight each sample based on observations made.



Particle Filtering

Particle filter

Input: **e**-the incoming evidence, N-the number of samples to be maintained, a DBN with prior $\mathbf{P(X_0)}$, transition model $\mathbf{P(X_1|X_0)}$, sensor model $\mathbf{P(E_1|X_1)}$, **S** vector of samples of size N, **W**-vector of weights of size N

Output: a set of samples for the next time step

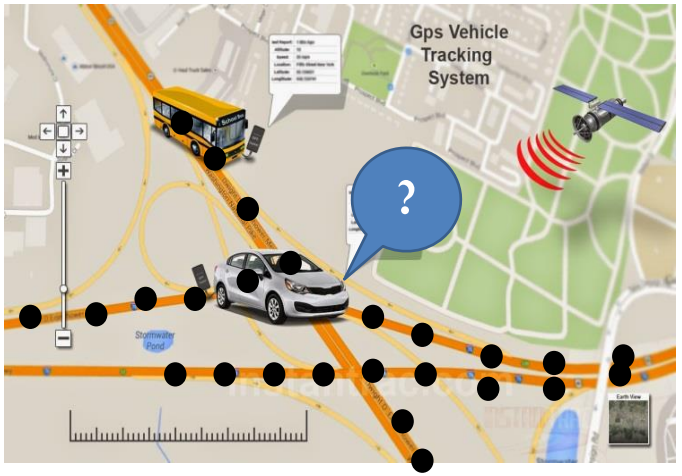
Start with sampling from initial pdf $\mathbf{P(X_0)}$

For each time step, repeat the following

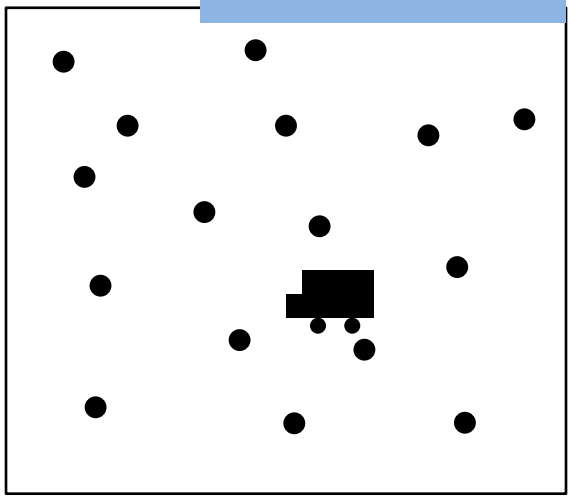
- time update (prediction)
- measurement update (correction/update)
- Resample

```
Initialize S by drawing samples from P(X0) and W ← 1
for i = 1 to N do
    S[i] ← sample from P(X1|X0 = S[i])
    W[i] ← P(e|X1=S[i])
for j = 1 to N do
    temp_S[j] ← sample from S with probability W[i]
S ← temp_S
W ← 1
return S
```

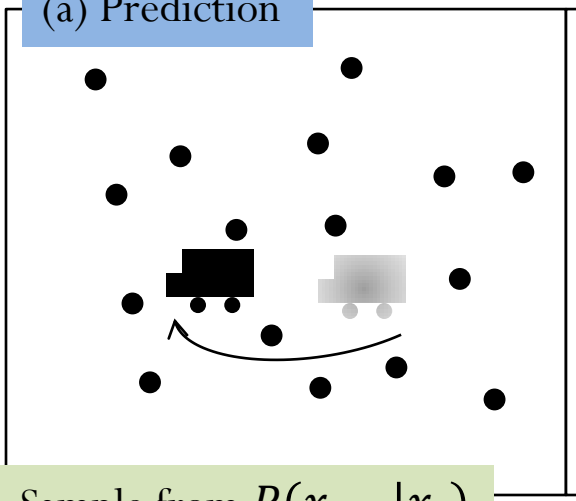
Particle Filtering



Initial distribution

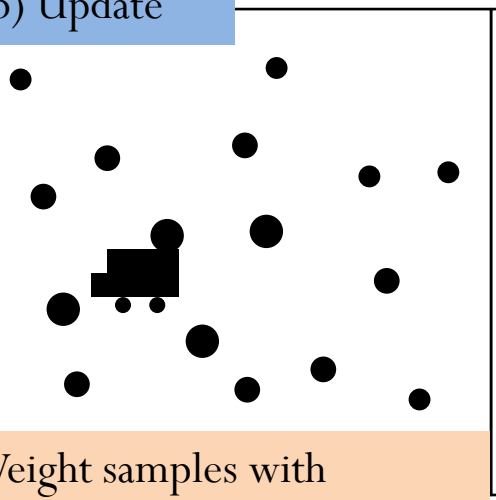


(a) Prediction



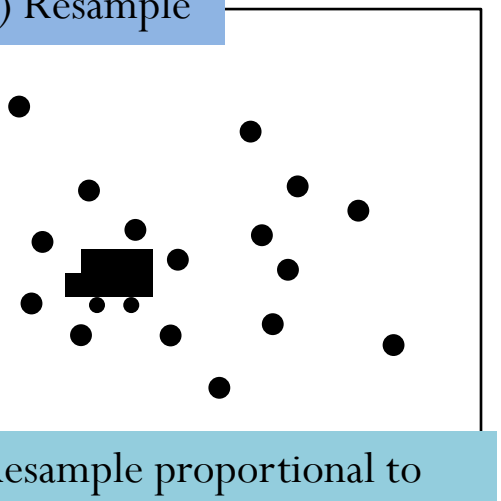
Sample from $P(x_{t+1}|x_t)$

(b) Update



Weight samples with $P(z_{t+1}|x_{t+1})$

(c) Resample

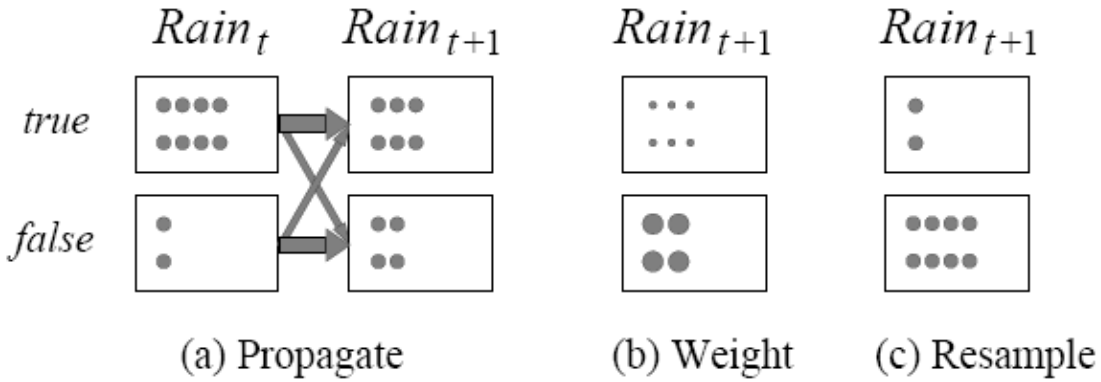


Resample proportional to weight

Particle Filtering

The particle filtering update cycle for the Umbrella DB with N = 10, showing the sample population of each state.

(a) At time t, 8 samples indicate *rain* and 2 indicate *~rain*. Each is propagated forward by sampling the next state through the transition model. At t+1, 6 samples indicate *rain* and 4 indicate *~rain*.



(b) *~umbrella* is observed at t+1. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles.

(c) A new set of samples is generated by weighted random selection from the current set ,resulting in 2 samples that indicate *rain* and 8 that indicate *~rain*

What did we discuss in L7-10?

- What are temporal models and how to represent them?
- How Markov assumption simplifies the transition and observation models?
- How to perform Filtering, Prediction, Smoothing, and Most likely explanation (inference tasks) recursively with constant time per step?

- What are Dynamic Bayesian Networks (DBNs), HMMs, Kalman Filtering for Linear Dynamical systems and how to represent them and how to infer from such models?

- What is Particle filtering and how is it useful to DBNs when exact inference in such networks is intractable?