Artificial Intelligence

Probabilistic Reasoning Over Time [1] [2]

Hidden Markov Models, Kalman Filters, Dynamic Bayesian Networks, Particle Filters

Jonathan Mwaura

[1] The materials contained here come from the AIMA book chapter 15. Please read the chapter.

[2] These particular slides have been prepared from AIMA book by Prof Paulo E. Santos who is based at the University de Fei, Sao Paulo, Brasil.

Outline I

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- X_t = set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- 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 the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- X_t = set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- 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 the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- X_t = set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- 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 the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- $X_t =$ set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- $E_t =$ set of observable evidence variables at time t
  e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$
- this assumes discrete time; step size depends on the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- $X_t =$ set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- $E_t =$ set of observable evidence variables at time t
  e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$
- this assumes discrete time; step size depends on the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- $X_t =$ set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- $E_t =$ set of observable evidence variables at time t
  e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$
- this assumes discrete time; step size depends on the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- $X_t =$ set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- $E_t =$ set of observable evidence variables at time t
  e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$
- this assumes discrete time; step size depends on the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Time and uncertainty

- We view the world as a series of snapshots (time slices), each of which contains a set of random variables, some observable and some not;
  - same subset of var is observable at each time slice
- Basic idea: copy state and evidence variables for each time step
- $X_t =$ set of unobservable state variables at time t.
  e.g. $BloodSugar_t$, $stomachContents_t$, etc
- $E_t =$ set of observable evidence variables at time t
  e.g. $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$
- this assumes discrete time; step size depends on the problem
- Notation: $X_{a:b} = X_a, X_{a+1}, ..., X_b$

Markov Processes (Markov Chains)

- what is the transition model (how the world evolves)? how the evidence variables are instantiated (sensor model)?
- Construct a Bayes net from these variables: parents?

Markov Processes (Markov Chains)

- what is the transition model (how the world evolves)? how the evidence variables are instantiated (sensor model)?
- Construct a Bayes net from these variables: parents?

Markov Processes (Markov Chains)

- what is the transition model (how the world evolves)? how the evidence variables are instantiated (sensor model)?
- Construct a Bayes net from these variables: parents?

Markov Processes (Markov Chains)

- Transition model: specifies the probability distribution over the latest state variables, given the previous values $\boldsymbol{P}(X_t|X_{0:t-1})$
- Markov Assumption: X_t depends on a **bounded** subset of $X_{a:t-1}$
- 1st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$(the state variables contain all the info. needed to characterize the probability distribution for the next time slice)
- 2st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-2}}, \boldsymbol{X_{t-1}})$

Markov Processes (Markov Chains)

- Transition model: specifies the probability distribution over the latest state variables, given the previous values $\boldsymbol{P}(X_t | X_{0:t-1})$
- Markov Assumption: X_t depends on a **bounded** subset of $X_{a:t-1}$
- 1st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t} | \boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t} | \boldsymbol{X_{t-1}})$(the state variables contain all the info. needed to characterize the probability distribution for the next time slice)
- 2st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t} | \boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t} | \boldsymbol{X_{t-2}}, \boldsymbol{X_{t-1}})$

Markov Processes (Markov Chains)

- Transition model: specifies the probability distribution over the latest state variables, given the previous values $\boldsymbol{P}(X_t|X_{0:t-1})$
- Markov Assumption: X_t depends on a **bounded** subset of $X_{a:t-1}$
- 1st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$(the state variables contain all the info. needed to characterize the probability distribution for the next time slice)
- 2st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-2}}, \boldsymbol{X_{t-1}})$

Markov Processes (Markov Chains)

- Transition model: specifies the probability distribution over the latest state variables, given the previous values $\boldsymbol{P}(X_t|X_{0:t-1})$
- Markov Assumption: X_t depends on a **bounded** subset of $X_{a:t-1}$
- 1st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$(the state variables contain all the info. needed to characterize the probability distribution for the next time slice)
- 2st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-2}}, \boldsymbol{X_{t-1}})$

Markov Processes (Markov Chains)

- Transition model: specifies the probability distribution over the latest state variables, given the previous values $\boldsymbol{P}(X_t|X_{0:t-1})$
- Markov Assumption: X_t depends on a **bounded** subset of $X_{a:t-1}$
- 1st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$(the state variables contain all the info. needed to characterize the probability distribution for the next time slice)
- 2st order Markov Process: $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-2}}, \boldsymbol{X_{t-1}})$

Markov Processes (Markov Chains)

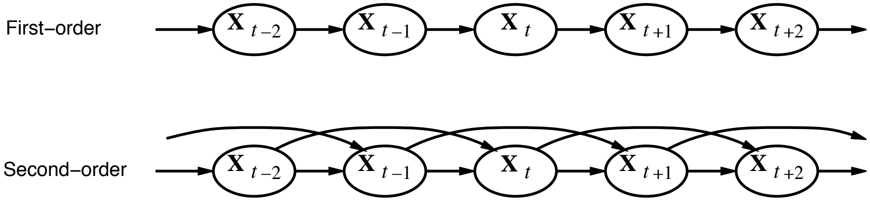


Figure:

Markov Processes (Markov Chains)

- the evidence variables E_t could depend on previous variables as well as the current state variables $\boldsymbol{P}(E_t|X_{0:t-1}, E_{0:t-1})$
- but any state should be capable of providing a precise sensor reading of itself
- Sensor Markov Assumption: $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_{0:t-1}}, \boldsymbol{E_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$ and sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$ fixed for all t

Markov Processes (Markov Chains)

- the evidence variables E_t could depend on previous variables as well as the current state variables $\boldsymbol{P}(E_t|X_{0:t-1}, E_{0:t-1})$
- but any state should be capable of providing a precise sensor reading of itself
- Sensor Markov Assumption: $\boldsymbol{P}(\boldsymbol{E}_t|\boldsymbol{X}_{0:t-1}, \boldsymbol{E}_{0:t-1}) = \boldsymbol{P}(\boldsymbol{E}_t|\boldsymbol{X}_t)$
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})$ and sensor model $\boldsymbol{P}(\boldsymbol{E}_t|\boldsymbol{X}_t)$ fixed for all t

Markov Processes (Markov Chains)

- the evidence variables E_t could depend on previous variables as well as the current state variables $\mathbf{P}(E_t | X_{0:t-1}, E_{0:t-1})$
- but any state should be capable of providing a precise sensor reading of itself
- Sensor Markov Assumption: $\mathbf{P}(\mathbf{E_t} | \mathbf{X_{0:t-1}}, \mathbf{E_{0:t-1}}) = \mathbf{P}(\mathbf{E_t} | \mathbf{X_t})$
- Stationary process: transition model $\mathbf{P}(\mathbf{X_t} | \mathbf{X_{t-1}})$ and sensor model $\mathbf{P}(\mathbf{E_t} | \mathbf{X_t})$ fixed for all t

Markov Processes (Markov Chains)

- the evidence variables E_t could depend on previous variables as well as the current state variables $\boldsymbol{P}(E_t|X_{0:t-1}, E_{0:t-1})$
- but any state should be capable of providing a precise sensor reading of itself
- Sensor Markov Assumption: $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_{0:t-1}}, \boldsymbol{E_{0:t-1}}) = \boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$ and sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$ fixed for all t

Markov Processes (Markov Chains)

- do we need to specify a different distribution for each time step?
- we avoid that by assuming a stationary distribution(note that it is distinct from a *static* process!)
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$ and sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$ fixed for all t

Markov Processes (Markov Chains)

- do we need to specify a different distribution for each time step?
- we avoid that by assuming a stationary distribution(note that it is distinct from a *static* process!)
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})$ and sensor model $\boldsymbol{P}(\boldsymbol{E}_t|\boldsymbol{X}_t)$ fixed for all t

Markov Processes (Markov Chains)

- do we need to specify a different distribution for each time step?
- we avoid that by assuming a stationary distribution(note that it is distinct from a *static* process!)
- Stationary process: transition model $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{X_{t-1}})$ and sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$ fixed for all t

Markov Processes (Markov Chains)

- In addition to transition and sensor models, we need to define a prior probability distribution at time 0: $\boldsymbol{P}(\boldsymbol{X_0})$
- now we have a specification of the complete joint distribution over all variables:
$P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i|X_{i-1})P(E_i|X_i))$

Markov Processes (Markov Chains)

- In addition to transition and sensor models, we need to define a prior probability distribution at time 0: $\boldsymbol{P}(\boldsymbol{X_0})$
- now we have a specification of the complete joint distribution over all variables:
  $P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i|X_{i-1})P(E_i|X_i))$

Example

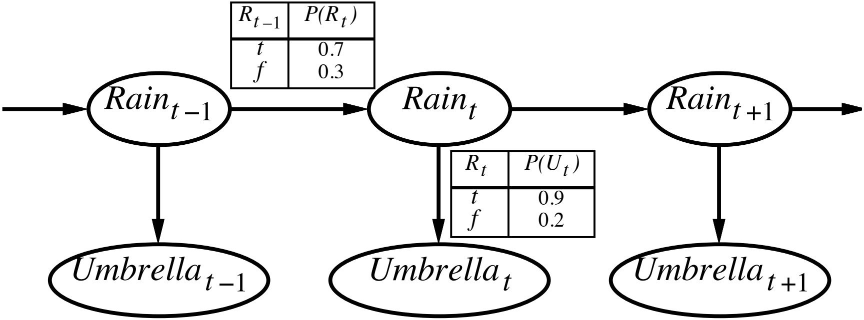


Figure:

First order Markov Assumption is only approximate.
Possible fixes:

- Increase the order
- augment state, e.g. add $Temp_t$, $Season_t$

Inference in temporal models

- Filtering $\boldsymbol{P}(\boldsymbol{X_t}|\boldsymbol{e_{1:t}})$: task of computing the belief state (the posterior distribution over the most recent state) given all evidence to date;
- Prediction $\boldsymbol{P}(\boldsymbol{X_{t+k}}|\boldsymbol{e_{1:t}})$ for $k > 0$: task of computing the posterior distribution over the future state, given all evidence to date;
- Smoothing $\boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:t}})$ for $0 =< k < t$, task of computing the posterior distribution over a past state, given all the evidence up to the present.
- Most likely explanation: $argmax_{X_{1:t}} P(X_{1:t}|e_{1:t})$: Given a sequence of observations, the task is to find the most likely sequence of states to have generated those observations

Outline I

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|X_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|X_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|X_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
  - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
  - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

Aim: devise a recursive state estimation algorithm:
$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t:+1}) = \boldsymbol{f}(\boldsymbol{e}_{t+1}, \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$

- first, the current state distribution is projected foward from t to $t+1$; then it is updated using the new evidence e_{t+1}.
- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1})$
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}, \boldsymbol{e}_{1:t}) \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (Bayes rule)
- $= \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t}) \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ (sensor Markov assumption)
    - $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$ represents a one step prediction of the next state, given evidence of the previous state;
    - $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{1+t})$ updates the prediction with the new evidence; this term comes from the sensor model

Filtering and prediction

We obtain a one-step prediction of the next state by conditioning on the hidden variable: the current state x_t:

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

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

- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t, \boldsymbol{e}_{1:t})$ comes from the transition model
- $\boldsymbol{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})$ comes from the current state distribution

Filtering and prediction

We obtain a one-step prediction of the next state by conditioning on the hidden variable: the current state x_t:

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

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

- $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t, \boldsymbol{e}_{1:t})$ comes from the transition model
- $\boldsymbol{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})$ comes from the current state distribution

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 0, no observations: $\boldsymbol{P}(R_0) = \langle 0.5, 0.5 \rangle$
- day 1, $U_1 = true$
  $\boldsymbol{P}(R_1) = \sum_{r_0} \boldsymbol{P}(R_1|r_0)P(r_0)$
  $\langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$
  Then the update step multiplies by the probability of the evidence (for t=1) and normalizes:
  $\boldsymbol{P}(R_1|u_1) = \alpha \boldsymbol{P}(u_1|R_1)\boldsymbol{P}(R_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle = \langle 0.818, 0.182 \rangle$

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 0, no observations: $\boldsymbol{P}(R_0) = \langle 0.5, 0.5 \rangle$
- day 1, $U_1 = true$
  $\boldsymbol{P}(R_1) = \sum_{r_0} \boldsymbol{P}(R_1|r_0)P(r_0)$
  $\langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$
  Then the update step multiplies by the probability of the evidence (for t=1) and normalizes:
  $\boldsymbol{P}(R_1|u_1) = \alpha \boldsymbol{P}(u_1|R_1)\boldsymbol{P}(R_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle = \langle 0.818, 0.182 \rangle$

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 0, no observations: $\boldsymbol{P}(R_0) = \langle 0.5, 0.5 \rangle$
- day 1, $U_1 = true$
  $$\boldsymbol{P}(R_1) = \sum_{r_0} \boldsymbol{P}(R_1|r_0)P(r_0)$$
  $\langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$
  Then the update step multiplies by the probability of the evidence (for t=1) and normalizes:
  $\boldsymbol{P}(R_1|u_1) = \alpha \boldsymbol{P}(u_1|R_1)\boldsymbol{P}(R_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle =$
  $\langle 0.818, 0.182 \rangle$

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 2, $U_2 = true$
  $\boldsymbol{P}(R_2|u_1) = \sum_{r_1} \boldsymbol{P}(R_2|r_1)P(r_1|u_1)$
  $\langle 0.7, 0.3\rangle \times 0.818 + \langle 0.3, 0.7\rangle \times 0.182 = \langle 0.627, 0.373\rangle$
  updating it with evidence from t = 2:
  $\boldsymbol{P}(R_2|u_1, u_2) = \alpha \boldsymbol{P}(u_2|R_2)\boldsymbol{P}(R_2|u_1) = \alpha\langle 0.9, 0.2\rangle\langle 0.627, 0.373\rangle = \langle 0.883, 0.117\rangle$
- the probability of rain increases from day 1 to day 2 because rain persists.

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 2, $U_2 = true$
  $\mathbf{P}(R_2|u_1) = \sum_{r_1} \mathbf{P}(R_2|r_1)P(r_1|u_1)$
  $\langle 0.7, 0.3\rangle \times 0.818 + \langle 0.3, 0.7\rangle \times 0.182 = \langle 0.627, 0.373\rangle$
  updating it with evidence from t = 2:
  $\mathbf{P}(R_2|u_1, u_2) = \alpha \mathbf{P}(u_2|R_2)\mathbf{P}(R_2|u_1) = \alpha\langle 0.9, 0.2\rangle\langle 0.627, 0.373\rangle = \langle 0.883, 0.117\rangle$
- the probability of rain increases from day 1 to day 2 because rain persists.

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 2, $U_2 = true$
  $\mathbf{P}(R_2|u_1) = \sum_{r_1} \mathbf{P}(R_2|r_1)P(r_1|u_1)$
  $\langle 0.7, 0.3\rangle \times 0.818 + \langle 0.3, 0.7\rangle \times 0.182 = \langle 0.627, 0.373\rangle$
  updating it with evidence from t = 2:
  $\mathbf{P}(R_2|u_1, u_2) = \alpha \mathbf{P}(u_2|R_2)\mathbf{P}(R_2|u_1) = \alpha\langle 0.9, 0.2\rangle\langle 0.627, 0.373\rangle = \langle 0.883, 0.117\rangle$
- the probability of rain increases from day 1 to day 2 because rain persists.

Example: umbrella network, compute $P(R_2|u_{1:2})$

- day 2, $U_2 = true$
  $\boldsymbol{P}(R_2|u_1) = \sum_{r_1} \boldsymbol{P}(R_2|r_1)P(r_1|u_1)$
  $\langle 0.7, 0.3\rangle \times 0.818 + \langle 0.3, 0.7\rangle \times 0.182 = \langle 0.627, 0.373\rangle$
  updating it with evidence from t = 2:
  $\boldsymbol{P}(R_2|u_1, u_2) = \alpha \boldsymbol{P}(u_2|R_2)\boldsymbol{P}(R_2|u_1) = \alpha\langle 0.9, 0.2\rangle\langle 0.627, 0.373\rangle = \langle 0.883, 0.117\rangle$
- the probability of rain increases from day 1 to day 2 because rain persists.

Example: unbrella network, compute $P(R_2|u_{1:2})$

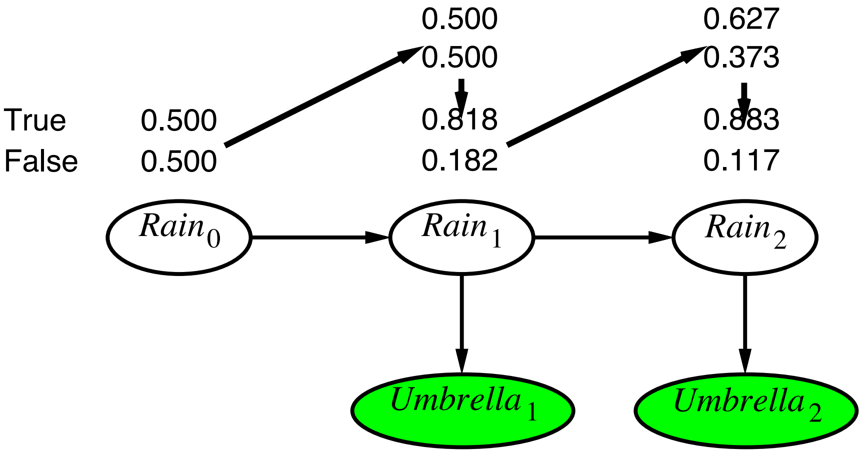


Figure:

Prediction

Prediction is filtering without the addition of new evidence:

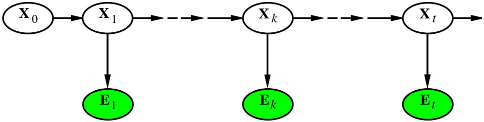
$$\boldsymbol{P}(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} \boldsymbol{P}(X_{t+k+1}|x_{t+k})P(x_{t+k}|e_{1:t})$$

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

Outline I

Smoothing

Process of computing the distribution over the past states given the evidence up to the present, that is $P(X_k|e_{1:t})$ for 0=< k < t.



Smoothing

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

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

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

$= \alpha \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}})\boldsymbol{P}(\boldsymbol{e_{k+1}}|\boldsymbol{X_k})$

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

Smoothing

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

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

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

$= \alpha \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}}) \boldsymbol{P}(\boldsymbol{e_{k+1}}|\boldsymbol{X_k})$

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

Smoothing

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

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

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

$= \alpha \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}})\boldsymbol{P}(\boldsymbol{e_{k+1}}|\boldsymbol{X_k})$

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

Smoothing

Divide evidence $e_{1:t}$ into $e_{1:k}$, $e_{k+1:t}$:
$\boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:t}}) = \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}}, \boldsymbol{e_{k+1:t}})$
$= \alpha \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}})\boldsymbol{P}(\boldsymbol{e_{k+1}}|\boldsymbol{X_k}, \boldsymbol{e_{1:k}})$
$= \alpha \boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:k}})\boldsymbol{P}(\boldsymbol{e_{k+1}}|\boldsymbol{X_k})$
$= \alpha \boldsymbol{f_{1:k}} \times \boldsymbol{b_{k+1}}$

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:
  $$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$
  (conditioning on the x_{k+1} possible states)
  $$= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k) \text{ (by conditional independence)}$$
  $$= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k)$$
  $$= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k)$$
  where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:
  $$\boldsymbol{P}(\boldsymbol{e_{k+1:t}}|\boldsymbol{X_k}) = \sum_{\boldsymbol{x_{K+1}}} \boldsymbol{P}(\boldsymbol{e_{k+1:t}}|\boldsymbol{X_k}, \boldsymbol{x_{k+1}})\boldsymbol{P}(\boldsymbol{x_{k+1}}|\boldsymbol{X_k})$$
  (conditioning on the x_{k+1} possible states)
  $$= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k) \text{ (by conditional independence)}$$
  $$= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$$
  $$= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$$
  where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:
  $$\boldsymbol{P}(\boldsymbol{e_{k+1:t}}|\boldsymbol{X_k}) = \sum_{\boldsymbol{x_{K+1}}} \boldsymbol{P}(\boldsymbol{e_{k+1:t}}|\boldsymbol{X_k}, \boldsymbol{x_{k+1}})\boldsymbol{P}(\boldsymbol{x_{k+1}}|\boldsymbol{X_k})$$
  (conditioning on the x_{k+1} possible states)
  $$= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k) \text{ (by conditional independence)}$$
  $$= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$$
  $$= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$$
  where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:

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

(conditioning on the x_{k+1} possible states)

$$= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k) \text{ (by conditional independence)}$$

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

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

where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:
  $\boldsymbol{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k) = \sum_{\boldsymbol{x}_{k+1}} \boldsymbol{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k, \boldsymbol{x}_{k+1})\boldsymbol{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k)$
  (conditioning on the x_{k+1} possible states)
  $= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$ (by conditional independence)
  $= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$
  $= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\boldsymbol{P}(x_{k+1}|X_k)$
  where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Smoothing

- the forward message can be computed by filtering
- the backward message can ne computed by backwards recursion:

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

(conditioning on the x_{k+1} possible states)

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

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

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

where the first and last terms are obtained directly from the model, whereas the middle term is a recursive call.

Example: umbrella network, compute $P(R_1|u_{1:2})$

$\boldsymbol{P}(R_1|u_{1:2}) = \alpha \boldsymbol{P}(R_1|u_1)\boldsymbol{P}(u_2|R_1)$
$\boldsymbol{P}(R_1|u_1) = \langle 0.818, 0.182 \rangle$
$\boldsymbol{P}(u_2|R_1) = \sum_{r_2} P(u_2|r_2)P(|r_2)\boldsymbol{P}(r_2|R_1) =$
$(0.9x1x\langle 0.7, 0.3\rangle) + (0.2x1x\langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle$
therefore,
$\boldsymbol{P}(R_1|u_{1:2}) = \langle 0.883, 0.117 \rangle$

Example: umbrella network, compute $P(R_1|u_{1:2})$

$\boldsymbol{P}(R_1|u_{1:2}) = \alpha \boldsymbol{P}(R_1|u_1)\boldsymbol{P}(u_2|R_1)$
$\boldsymbol{P}(R_1|u_1) = \langle 0.818, 0.182 \rangle$
$\boldsymbol{P}(u_2|R_1) = \sum_{r_2} P(u_2|r_2)P(|r_2)\boldsymbol{P}(r_2|R_1) =$
$(0.9x1x\langle 0.7, 0.3 \rangle) + (0.2x1x\langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle$
therefore,
$\boldsymbol{P}(R_1|u_{1:2}) = \langle 0.883, 0.117 \rangle$

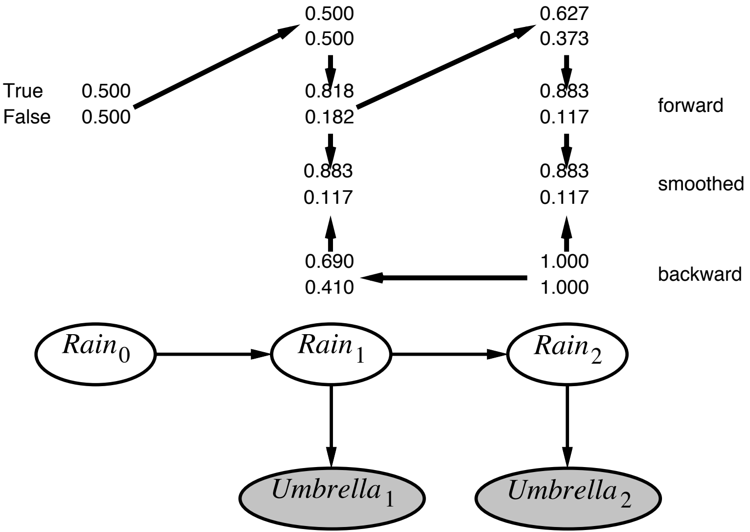
Example: umbrella network, compute $P(R_1|u_{1:2})$

$\boldsymbol{P}(R_1|u_{1:2}) = \alpha \boldsymbol{P}(R_1|u_1)\boldsymbol{P}(u_2|R_1)$
$\boldsymbol{P}(R_1|u_1) = \langle 0.818, 0.182\rangle$
$\boldsymbol{P}(u_2|R_1) = \sum_{r_2} P(u_2|r_2)P(|r_2)\boldsymbol{P}(r_2|R_1) =$
$(0.9x1x\langle 0.7, 0.3\rangle) + (0.2x1x\langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle$
therefore,
$\boldsymbol{P}(R_1|u_{1:2}) = \langle 0.883, 0.117\rangle$

Example: umbrella network, compute $P(R_1|u_{1:2})$

$\boldsymbol{P}(R_1|u_{1:2}) = \alpha \boldsymbol{P}(R_1|u_1)\boldsymbol{P}(u_2|R_1)$
$\boldsymbol{P}(R_1|u_1) = \langle 0.818, 0.182 \rangle$
$\boldsymbol{P}(u_2|R_1) = \sum_{r_2} P(u_2|r_2)P(|r_2)\boldsymbol{P}(r_2|R_1) =$
$(0.9x1x\langle 0.7, 0.3\rangle) + (0.2x1x\langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle$
therefore,
$\boldsymbol{P}(R_1|u_{1:2}) = \langle 0.883, 0.117\rangle$

Example: umbrella network, compute $P(R_1|u_{1:2})$



Outline I

Most likely explanation

- Most likely sequence $\neq$ 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 x_{t+1} = most likely path to some x_t plus one more step (recursive definition, due to the Markov property)
  - $max_{x_1 \ldots x_t} \boldsymbol{P}(x_1 \ldots x_t, X_{t+1} | e_{1:t+1})$
  - $= \boldsymbol{P}(e_{t+1} | X_{t+1}) max_{x_1 \ldots x_t} (\boldsymbol{P}(X_{t+1} | x_t) max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_t | e_{t+1}))$
  - Identical to filtering, except $f_{1:t}$ replaced by $m_{1:t} = max_{x_1 \ldots x_{t-1}} (\boldsymbol{P}(x_1 \ldots x_{t-1}, X_{t+1} | e_{1:t}))$

Most likely explanation

- Most likely sequence $\neq$ 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 x_{t+1} = most likely path to some x_t plus one more step (recursive definition, due to the Markov property)
  - $max_{x_1 \ldots x_t} \boldsymbol{P}(x_1 \ldots x_t, X_{t+1} | e_{1:t+1})$
  - $= \boldsymbol{P}(e_{t+1} | X_{t+1}) max_{x_1 \ldots x_t} (\boldsymbol{P}(X_{t+1} | x_t) max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_t | e_{t+1}))$
  - Identical to filtering, except $f_{1:t}$ replaced by $m_{1:t} = max_{x_1 \ldots x_{t-1}} (\boldsymbol{P}(x_1 \ldots x_{t-1}, X_{t+1} | e_{1:t}))$

Most likely explanation

- Most likely sequence $\neq$ 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 x_{t+1} = most likely path to some x_t plus one more step (recursive definition, due to the Markov property)
  - $max_{x_1 \ldots x_t} \boldsymbol{P}(x_1 \ldots x_t, X_{t+1} | e_{1:t+1})$
  - $= \boldsymbol{P}(e_{t+1} | X_{t+1}) max_{x_1 \ldots x_t} (\boldsymbol{P}(X_{t+1} | x_t) max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_t | e_{t+1}))$
  - Identical to filtering, except $f_{1:t}$ replaced by $m_{1:t} = max_{x_1 \ldots x_{t-1}} (\boldsymbol{P}(x_1 \ldots x_{t-1}, X_{t+1} | e_{1:t}))$

Most likely explanation

- Most likely sequence $\neq$ 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 x_{t+1} = most likely path to some x_t plus one more step (recursive definition, due to the Markov property)
  - $max_{x_1 \ldots x_t} \boldsymbol{P}(x_1 \ldots x_t, X_{t+1} | e_{1:t+1})$
  - $= \boldsymbol{P}(e_{t+1} | X_{t+1}) max_{x_1 \ldots x_t} (\boldsymbol{P}(X_{t+1} | x_t) max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_t | e_{t+1}))$
  - Identical to filtering, except $f_{1:t}$ replaced by $m_{1:t} = max_{x_1 \ldots x_{t-1}} (\boldsymbol{P}(x_1 \ldots x_{t-1}, X_{t+1} | e_{1:t}))$

Most likely explanation

- Most likely sequence $\neq$ 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 x_{t+1} = most likely path to some x_t plus one more step (recursive definition, due to the Markov property)
  - $max_{x_1 \ldots x_t} \boldsymbol{P}(x_1 \ldots x_t, X_{t+1} | e_{1:t+1})$
  - $= \boldsymbol{P}(e_{t+1} | X_{t+1}) max_{x_1 \ldots x_t} (\boldsymbol{P}(X_{t+1} | x_t) max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_t | e_{t+1}))$
  - Identical to filtering, except $f_{1:t}$ replaced by $m_{1:t} = max_{x_1 \ldots x_{t-1}} (\boldsymbol{P}(x_1 \ldots x_{t-1}, X_{t+1} | e_{1:t}))$

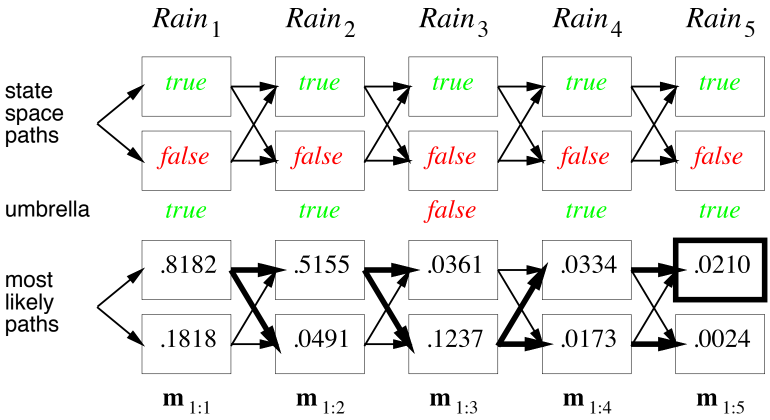
Most likely explanation

- this is the probabilities of the most likely path to each state x_t and the summation of the filtering algorithm is replaced by a maximization over 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 pick 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.

Most likely explanation

- this is the probabilities of the most likely path to each state x_t and the summation of the filtering algorithm is replaced by a maximization over 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 pick 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.

Viterby example



Hidden Markov models

HMM is a temporal probabilistic model in which the state of the process is described by a *single discrete* random variable:
X_t is a single, discrete variable (usually E_t is too)
Domain of X_t is $\{1, \ldots, S\}$

- **Transition matrix:** $T_{ij} = P(X_t = j | X_{t-1} = i)$, e.g. umbrella example: $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$
- **Sensor matrix:** O_t for each time step, diagonal elements (diagonal matrix, for mathematical convenience) $P(e_t | X_t = i)$, e.g. umbrella example, day1 $U_1 = \textit{true}$, day 3, $U_3 = \textit{false}$
  $O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$ $O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$

Hidden Markov models

Forward and backward messages as column vectors:

- forward equation: $f_{1:t+1} = \alpha O_{t+1} T^{\top} f_{1:t}$
- backward equation: $b_{k+1:t} = T O_{k+1} b_{k+2:t}$

Hidden Markov models

Advantages of matrix representation:

- compact representation;
- simpler formulae;
- efficiency (smoothing in constant space, independently of the length of the sequence)

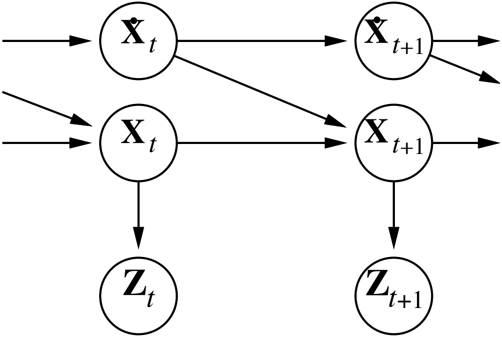
Hidden Markov models

Check the robot localization example on section 15.3.2 (3rd edition)

Kalman filters

Modelling systems described by a set of continuous variables, e.g. tracking a bird flying - $X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$

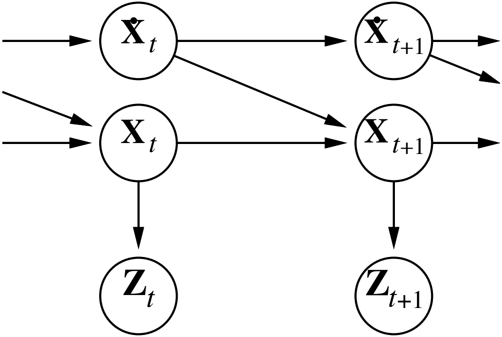
Airplanes, robots, ecosystems, economies, chemical plants,, ...



In all these cases we're doing filtering: estimating state variables form noisy observation over time.

Kalman filters

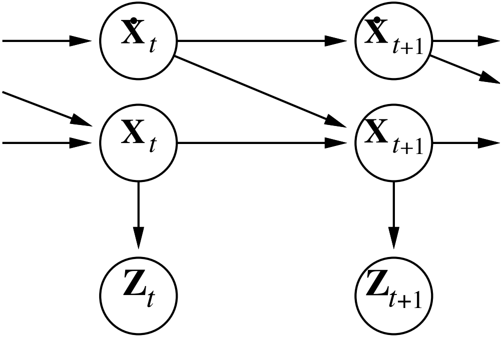
Modelling systems described by a set of continuous variables, e.g. tracking a bird flying - $X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$
Airplanes, robots, ecosystems, economies, chemical plants„ ....



In all these cases we're doing filtering: estimating state variables form noisy observation over time.

Kalman filters

Modelling systems described by a set of continuous variables, e.g. tracking a bird flying - $X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$
Airplanes, robots, ecosystems, economies, chemical plants„ ....



In all these cases we're doing filtering: estimating state variables form noisy observation over time.

Kalman filters

The transition and sensor models are modeled as linear Gaussian distributions

I.e., the next state X_{t+1} must be a linear function of the current state X_t, plus some Gaussian noise

Updating Gaussian distributions

- Prediction step: if $\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t}}|\boldsymbol{e}_{\boldsymbol{1:t}})$ (current distrib.) is Gaussian and the transition model $\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t+1}}|\boldsymbol{x}_{\boldsymbol{t}})$ is Gaussian, then the one-step prediction distribution
  $\boldsymbol{P}(X_{t+1}|e_{1:t}) = \int_{X_t} \boldsymbol{P}(X_{t+1}|x_{1+t})P(x_t|e_{1:t})dx_t$
  is Gaussian
- if the prediction $\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t+1}}|\boldsymbol{e}_{\boldsymbol{1:t}})$ is Gaussian and the senso model $\boldsymbol{P}(\boldsymbol{e}_{\boldsymbol{t+1}}|\boldsymbol{X}_{\boldsymbol{t+1}})$ is Gaussian, then the updated distribution
  $\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t+1}}|\boldsymbol{e}_{\boldsymbol{1:t+1}}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{\boldsymbol{t+1}}|\boldsymbol{X}_{\boldsymbol{1+t}})\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t+1}}|\boldsymbol{e}_{\boldsymbol{1:t}})$
  is Gaussian.
- Hence, $\boldsymbol{P}(\boldsymbol{X}_{\boldsymbol{t}}|\boldsymbol{e}_{\boldsymbol{1:t}})$ is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all t
- General (nonlinear, non-Gaussian) process: description of posterior grows unboundedly in time.

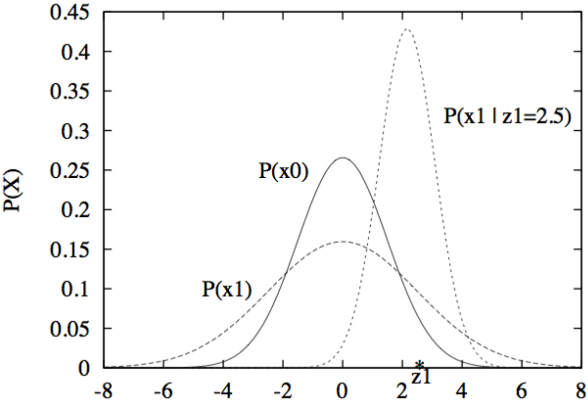
Updating Gaussian distributions

- the FORWARD operator for Kalman filtering takes a Gaussian forward message $f_{1:t}$ specified by a mean μ_t and covariance matrix Σ_t, and produces a new multivariate Gaussian forward message $f_{1:t+1}$, specified by a mean μ_{t+1} and covariance matrix Σ_{t+1}.

Simple 1D example

Gaussian random walk on X-axis (s.d σ_X), with a noisy observation X_t (s.d. σ_Z);

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



Simple 1D example

- The equations for μ_{t+1} and σ_{t+1} play the same role as the general filtering equation and the HMM filtering equation.
- additional properties:
  - the calculation of the new mean μ_{t+1} can be viewed as a weighted mean of the new observation z_{t+1} and the old mean μ_t ;
  - the update of the variance σ^2_{t+1} is independent of the observation. Therefore, it can be computed in advance

Simple 1D example

- The equations for μ_{t+1} and σ_{t+1} play the same role as the general filtering equation and the HMM filtering equation.
- additional properties:
  - the calculation of the new mean μ_{t+1} can be viewed as a weighted mean of the new observation z_{t+1} and the old mean μ_t ;
  - the update of the variance σ_{t+1}^2 is independent of the observation. Therefore, it can be computed in advance

Simple 1D example

- The equations for μ_{t+1} and σ_{t+1} play the same role as the general filtering equation and the HMM filtering equation.
- additional properties:
  - the calculation of the new mean μ_{t+1} can be viewed as a weighted mean of the new observation z_{t+1} and the old mean μ_t ;
  - the update of the variance σ^2_{t+1} is independent of the observation. Therefore, it can be computed in advance

General Kalman update

The full multivariate Gaussian distribution has the form:
$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})(\boldsymbol{x}) = \alpha e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$

General Kalman update

Transition and sensor models:

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

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

Filter computes the following update:

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

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

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

General Kalman update

- The term $F\mu_t$ is the predicted state at $t+1$
- $HF\mu_t$ is the predicted observation
- $z_{t+1} - HF\mu_t$ represents the error in the predicted observation
  - this is multiplied by K_{t+1} to correct the predicted state
  - hence, K_{t+1} is a measure of how seriously to take the new observation wrt the prediction
- as before the variance update is independent of the observations, thus Σ_t and K_t can be computed offline

General Kalman update

- The term $F\mu_t$ is the predicted state at $t+1$
- $HF\mu_t$ is the predicted observation
- $z_{t+1} - HF\mu_t$ represents the error in the predicted observation
  - this is multiplied by K_{t+1} to correct the predicted state
  - hence, K_{t+1} is a measure of how seriously to take the new observation wrt the prediction
- as before the variance update is independent of the observations, thus Σ_t and K_t can be computed offline

General Kalman update

- The term $F\mu_t$ is the predicted state at $t+1$
- $HF\mu_t$ is the predicted observation
- $z_{t+1} - HF\mu_t$ represents the error in the predicted observation
  - this is multiplied by K_{t+1} to correct the predicted state
  - hence, K_{t+1} is a measure of how seriously to take the new observation wrt the prediction
- as before the variance update is independent of the observations, thus Σ_t and K_t can be computed offline

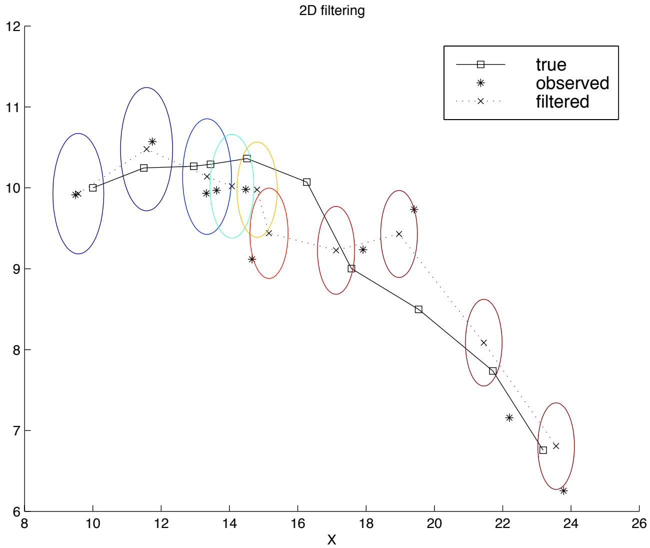
General Kalman update

- The term $F\mu_t$ is the predicted state at $t + 1$
- $HF\mu_t$ is the predicted observation
- $z_{t+1} - HF\mu_t$ represents the error in the predicted observation
  - this is multiplied by K_{t+1} to correct the predicted state
  - hence, K_{t+1} is a measure of how seriously to take the new observation wrt the prediction
- as before the variance update is independent of the observations, thus Σ_t and K_t can be computed offline

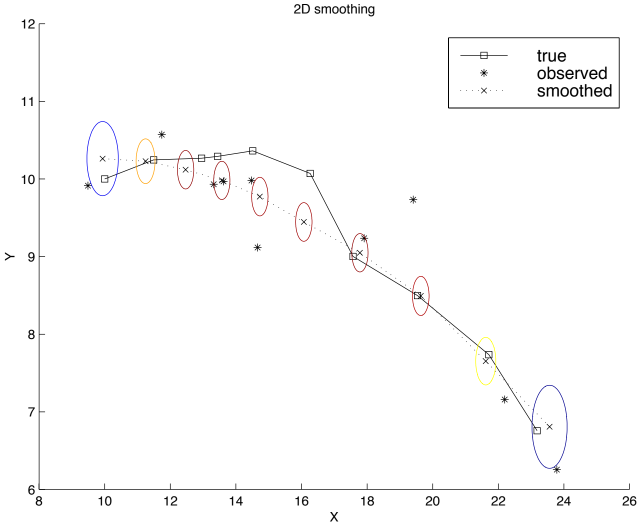
General Kalman update

- The term $F\mu_t$ is the predicted state at $t+1$
- $HF\mu_t$ is the predicted observation
- $z_{t+1} - HF\mu_t$ represents the error in the predicted observation
  - this is multiplied by K_{t+1} to correct the predicted state
  - hence, K_{t+1} is a measure of how seriously to take the new observation wrt the prediction
- as before the variance update is independent of the observations, thus Σ_t and K_t can be computed offline

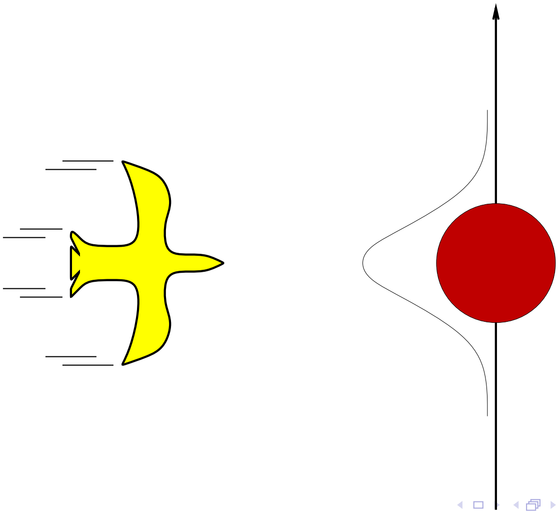
2D tracking example: filtering



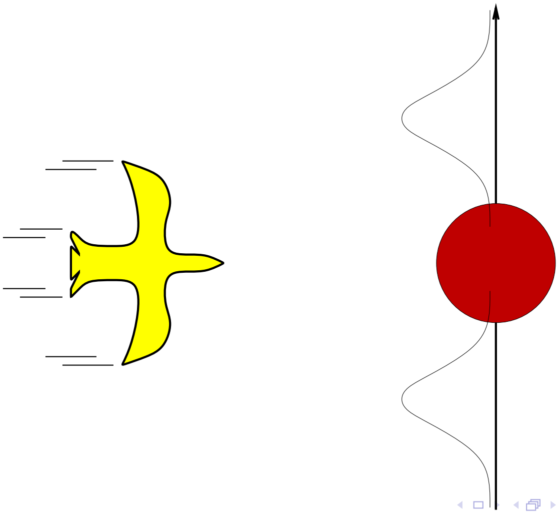
2D tracking example: smoothing



Where it breaks



Where it breaks



Kalman filter

- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

Kalman filter

- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

Kalman filter

- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

Kalman filter

- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

Kalman filter

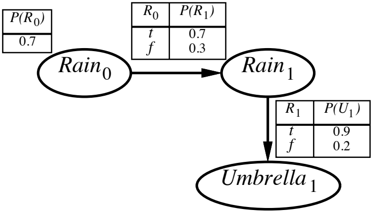
- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

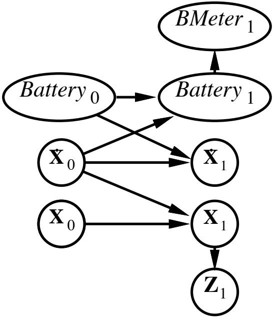
Kalman filter

- KF can be applied to a vast array of domains
- but the results are not always valid or useful
- strong assumption: linear Gaussian transition ans sensor models
- EKF extends a KF towards non-linear systems (i.e. systems that the transition model cannot be modeled as a multiplication of the state vector)
- EKF works by modeling the system as locally linear in x_t in the region of $x_t = \mu_t$ (the mean of the current state distribution)
- other solution: Switching Kalman filter: multiple KF run an parallel, each using a different model of the system

Dynamic Bayesian Networks

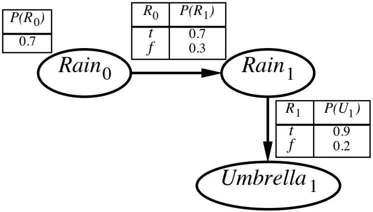
- DBN is a BN that represents a temporal probability model
- each slice of a DBN can contain any number of variables X_t and E_t
- assumption: variables and their links are replicated from slice to slice and a DBN is a first-order Markov process.

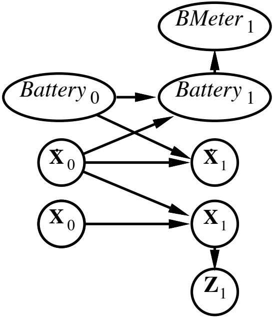




Dynamic Bayesian Networks

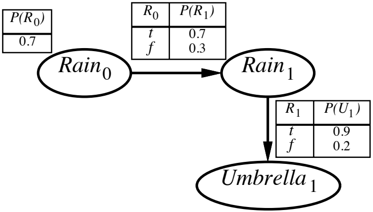
- DBN is a BN that represents a temporal probability model
- each slice of a DBN can contain any number of variables X_t and E_t
- assumption: variables and their links are replicated from slice to slice and a DBN is a first-order Markov process.

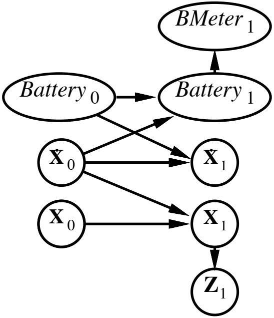




Dynamic Bayesian Networks

- DBN is a BN that represents a temporal probability model
- each slice of a DBN can contain any number of variables X_t and E_t
- assumption: variables and their links are replicated from slice to slice and a DBN is a first-order Markov process.





Dynamic Bayesian Networks

to construct a DBN we need the following:

- the prior distribution over the state variables, $\boldsymbol{P}(\boldsymbol{X_0})$
- the transition model $\boldsymbol{P}(\boldsymbol{X_{t+1}}|\boldsymbol{X_t})$
- the sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$
- to specify the transition and sensor models we need the topology of the network
- due to our assumption of a *stationary distribution* we just need to specify them for the first slice

Dynamic Bayesian Networks

to construct a DBN we need the following:

- the prior distribution over the state variables, $\boldsymbol{P}(\boldsymbol{X_0})$
- the transition model $\boldsymbol{P}(\boldsymbol{X_{t+1}}|\boldsymbol{X_t})$
- the sensor model $\boldsymbol{P}(\boldsymbol{E_t}|\boldsymbol{X_t})$
- to specify the transition and sensor models we need the topology of the network
- due to our assumption of a *stationary distribution* we just need to specify them for the first slice

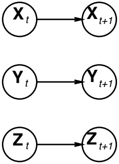
Dynamic Bayesian Networks

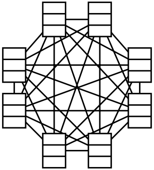
to construct a DBN we need the following:

- the prior distribution over the state variables, $\boldsymbol{P}(\boldsymbol{X_0})$
- the transition model $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t)$
- the sensor model $\boldsymbol{P}(\boldsymbol{E}_t|\boldsymbol{X}_t)$
- to specify the transition and sensor models we need the topology of the network
- due to our assumption of a *stationary distribution* we just need to specify them for the first slice

DBNs vs. HMMs

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

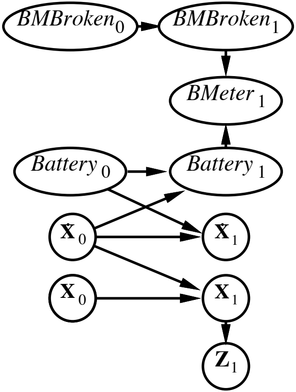


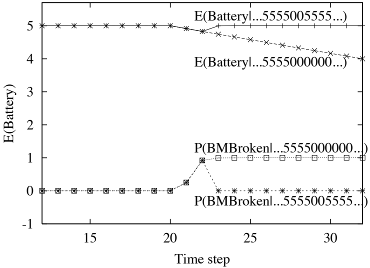


Sparse dependencies -> exponentially fewer parameters
e.g., 20 state variables, three parents each
DBN has $20 \times 2^3 = 160$ parameters, HMM has $2 \times 2^{20} \approx 10^{12}$

DBNs vs. Kalman filters

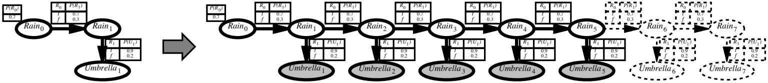
Every Kalman filter model is a DBN, but few DBNs are KFs;
real world requires non-Gaussian posteriors
E.g. Where are my keys? What's the battery charge?





Exact Inference in DBNs

Naive method: unroll the network and run any exact algorithm



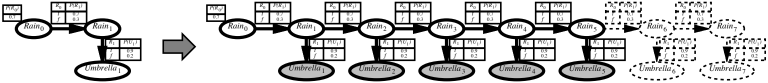
Problem: inference cost for each update grows with t

Rollup filtering: add slice t_1, “sum out” slice t (as done in the filtering equation before)

blows up in complexity as time goes by!

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 (as done in the filtering equation before)
blows up in complexity as time goes by!

Likelihood weighting for DBNs

- LW works by sampling the nonevidence nodes of the network in topological order , weighting each sample by the likelihood it accords to the observed variable.
- use the samples themselves as an approximate representation of the current state distribution
- no need to unroll the entire net, only needs current and next slices

Likelihood weighting for DBNs

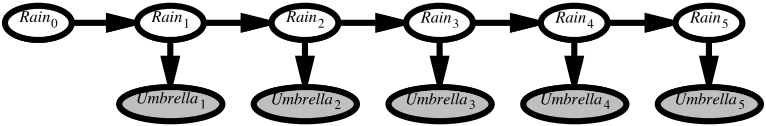
- LW works by sampling the nonevidence nodes of the network in topological order , weighting each sample by the likelihood it accords to the observed variable.
- use the samples themselves as an approximate representation of the current state distribution
- no need to unroll the entire net, only needs current and next slices

Likelihood weighting for DBNs

- LW works by sampling the nonevidence nodes of the network in topological order , weighting each sample by the likelihood it accords to the observed variable.
- use the samples themselves as an approximate representation of the current state distribution
- no need to unroll the entire net, only needs current and next slices

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
- number of samples required grows exponentially with t

Particle filtering

Basic idea: ensure that the population of samples (particles) tracks the high-likelyhood regions of the state-space
Replicate particles proportional to likelihood for e_t

- widely used for tracking nonlinear systems, esp. vision
- used for SLAM - Simultaneous localization and map building

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 performance

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