Hidden Markov Models

Robert Platt
Research Scientist
Computer Science and Artificial Intelligence Lab
MIT

Hidden Markov models are everywhere!

Used to model time-series data:

- Robot perception/control
- Speech recognition
- Video understanding
- ...

MIT DARPA grand challenge vehicle

Human speech

A Markov Process

N states: $S = \{s_1, s_2, \ldots, s_N\}$

At time *t*, state is: x_t

Discrete time steps: $t = \{1, 2, \ldots\}$

Stochastic transitions once per time step:

State transitions

Example

Suppose: $x_t = s_3$

Transition probabilities:

$$P(x_{t+1} = s_1 | x_t = s_3) = 1/6$$

$$P(x_{t+1} = s_2 | x_t = s_3) = 2/3$$

$$P(x_{t+1} = s_3 | x_t = s_3) = 1/6$$

State transitions

Notation: $a_{ij} = P(x_{t+1} = s_j | x_t = s_i)$

- conditional probability table (CPT)
- (process dynamics)

State transitions

Notation: $a_{ij} = P(x_{t+1} = s_j | x_t = s_i)$

- conditional probability table (CPT)
- (process dynamics)

Notice: $\forall i \in [1, N], \sum_{j=1}^{N} a_{ij} = 1$

Markov Property

$$P(x_{t+1}|x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1}|x_t)$$

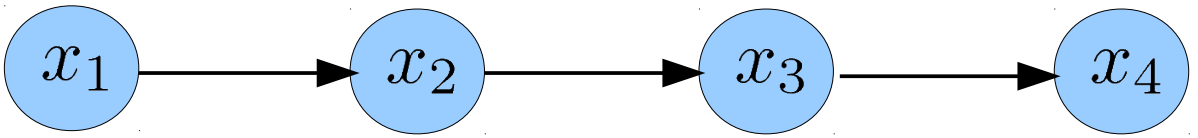
Implicit in CPT ...

Markov Property

$$P(x_{t+1}|x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1}|x_t)$$

Implicit in CPT ...

Bayes net for a Markov model:

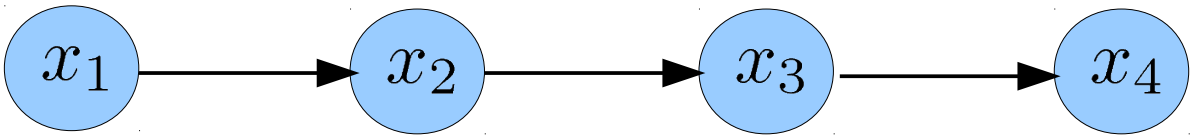


Markov Property

$$P(x_{t+1}|x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1}|x_t)$$

Implicit in CPT ...

Bayes net for a Markov model:



What does the Bayes net look like if the Markov property does not hold?

Gridworld Example

Robot on a 4x5 grid …

Define transition dynamics:

- 5 possible transitions
  (up, down, left, right, stay)

P(right) = 0.4
P(left) = 0.15
P(up) = 0.15
P(down) = 0.15
P(stay) = 0.15

- different dynamics at walls...

Warm up question

If robot starts in state $x_1 = s_s$

What is probability of landing on state $x_T = s_g$ at time T ?

Warm up question

If robot starts in state $x_1 = s_s$

What is probability of landing on state $x_T = s_g$ at time T ?

i.e. find: $P(x_T = s_g | x_1 = s_s)$

Naïve solution

Find: $P(x_T = s_g | x_1 = s_s)$

Naïve solution

Find: $P(x_T = s_g | x_1 = s_s)$

Sol'n: $P(x_T, \ldots, x_2 | x_1) = \prod_{t=1}^{T-1} p(x_{t+1} | x_t)$

$$P(x_T | x_1) = \sum_{x_{T-1}, \ldots, x_2} P(x_T, \ldots, x_2 | x_1)$$

Naïve solution

Find: $P(x_T = s_g | x_1 = s_s)$

Sol'n: $P(x_T, \ldots, x_2 | x_1) = \prod_{t=1}^{T-1} p(x_{t+1} | x_t)$

$$P(x_T | x_1) = \sum_{x_{T-1}, \ldots, x_2} P(x_T, \ldots, x_2 | x_1)$$

Cost: $O(N^{T-2})$

Dynamic Programming Sol'n

Find: $P(x_T = s_g | x_1 = s_s)$

Let: $p_t(i) = P(x_t = s_i | x_1 = s_s)$

Dynamic Programming Sol'n

Find: $P(x_T = s_g | x_1 = s_s)$

Let: $p_t(i) = P(x_t = s_i | x_1 = s_s)$

Sol'n:

1. $p_1(i) = \begin{cases} 1 \text{ if } x_1 = s_s \\ 0 \text{ else} \end{cases}$
2.
3.

Dynamic Programming Sol'n

Find: $P(x_T = s_g | x_1 = s_s)$

Let: $p_t(i) = P(x_t = s_i | x_1 = s_s)$

Sol'n:

1. $p_1(i) = \begin{cases} 1 \text{ if } x_1 = s_s \\ 0 \text{ else} \end{cases}$
2. $\forall t \in [1, T-1]$
3. $\forall j \in [1, N], p_{t+1}(j) = \sum_{i=1}^{N} a_{ij} p_t(i)$

Dynamic Programming Sol'n

Find: $P(x_T = s_g | x_1 = s_s)$

Let: $p_t(i) = P(x_t = s_i | x_1 = s_s)$

Sol'n:

1. $p_1(i) = \begin{cases} 1 \text{ if } x_1 = s_s \\ 0 \text{ else} \end{cases}$

2. $\forall t \in [1, T-1]$

3. $\forall j \in [1, N], p_{t+1}(j) = \sum_{i=1}^{N} a_{ij} p_t(i)$

Cost: $O((T-1)N^2)$

Dynamic Programming Sol'n

Find: $P(x_T = s_g | x_1 = s_s)$

Let: $p_t(i) = P(x_t = s_i | x_1 = s_s)$

Another way of writing the DP sol'n:

1. $A = \begin{pmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \dots & \dots & \dots \end{pmatrix}$

2. $\begin{pmatrix} p_\tau(1) \\ \vdots \\ p_\tau(N) \end{pmatrix} = \left(A^{\tau - t}\right)^T \begin{pmatrix} p_t(1) \\ \vdots \\ p_t(N) \end{pmatrix}$

Stationary distribution

Given a Markov process with:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

The process ultimately converges to a particular stationary distribution:

$$\exists \omega \in \Delta^{N-1} : \omega = A^T \omega$$

How do you calculate ω ?

A *Hidden* Markov Model

N states: $\{s_1, s_2, \ldots, s_N\}$

M observations: $\{o_1, o_2, \ldots, o_M\}$

Discrete time steps: $t = \{1, 2, \ldots\}$

At time *t*, state is: x_t

At time t, observation is: z_t

You never observe system state ...

Observations

Stochastic observations once per time step:

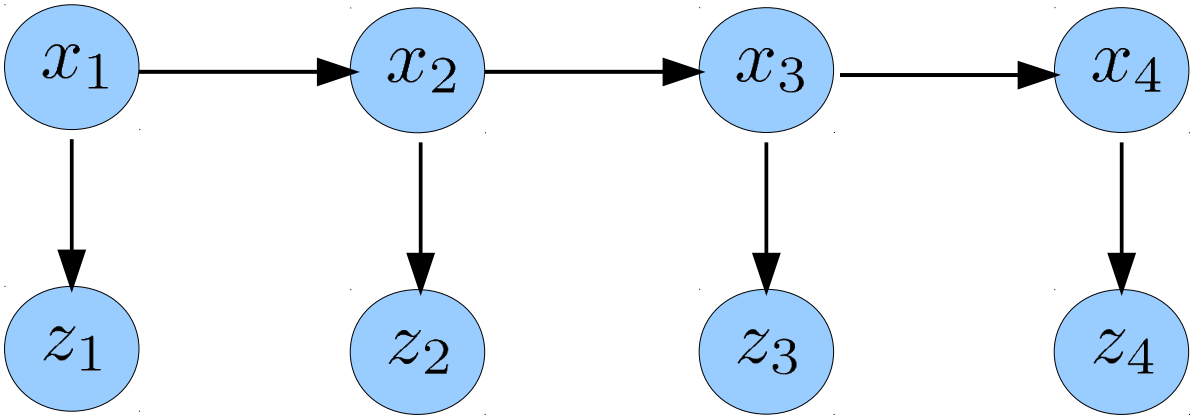
- when you first arrive in a state

Assume: $P(z_t|x_t) = P(z_t|x_t, \ldots, x_1)$

What does the Bayes net for an HMM look like?

HMM

What does the Bayes net for an HMM look like?



HMM

Notation: $b_i(k) = P(z_t = o_k | x_t = s_i)$

- observation dynamics

Notice: $\sum_{k=1}^{M} b_i(k) = 1$

HMM Definition

N states: $\{s_1, s_2, \ldots, s_N\}$

M observations: $\{o_1, o_2, \ldots, o_M\}$

Process dynamics: $a_{ij} = P(x_{t+1} = s_j | x_t = s_i)$

Observation dynamics: $b_i(k) = P(z_t = o_k | x_t = s_i)$

or $b_i(z_t) = P(z_t | x_t = s_i)$

Prior distribution: $\pi_i = P(x_1 = s_i)$ or $\pi(x)$

Games you can play with an HMM

1. Inference:
   - where is the robot now?
   - where was the robot?
   - what path did it take?

Today

2. Learning:
   - what are the robot process and observation dynamics?

Some other time

Problem #1: Filtering

If the robot starts in prior distribution, π

and make a sequence of observations, $z_1, z_2, \ldots, z_T$

Then, where is the robot at time T?

In other words, what is: $P(x_T | z_1, \ldots, z_T)$?

Naïve solution

$$P(x_T|z_1,\ldots,z_T) = \frac{P(x_T,z_1,\ldots,z_T)}{P(z_1,\ldots,z_T)}$$

$$= \frac{\sum_{x_1,\ldots,x_{T-1}} P(x_1,\ldots,x_T,z_1,\ldots,z_T)}{\sum_{x_1,\ldots,x_T} P(x_1,\ldots,x_T,z_1,\ldots,z_T)}$$

$$P(x_1,\ldots,x_T,z_1,\ldots,z_T) = \underbrace{P(z_1,\ldots,z_T|x_1,\ldots,x_T)}_{?}\underbrace{P(x_1,\ldots,x_T)}_{?}$$

Naïve solution

$$P(x_T|z_1,\ldots,z_T) = \frac{P(x_T,z_1,\ldots,z_T)}{P(z_1,\ldots,z_T)}$$

$$= \frac{\sum_{x_1,\ldots,x_{T-1}} P(x_1,\ldots,x_T,z_1,\ldots,z_T)}{\sum_{x_1,\ldots,x_T} P(x_1,\ldots,x_T,z_1,\ldots,z_T)}$$

$$P(x_1,\ldots,x_T,z_1,\ldots,z_T) = \underbrace{P(z_1,\ldots,z_T|x_1,\ldots,x_T)}_{?}\underbrace{P(x_1,\ldots,x_T)}_{?}$$

Not cool … $O(N^{T-1})$

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1, \ldots, z_T)$

Define: $\alpha_t(i) = P(x_t = s_i, z_1, \ldots, z_t)$

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1,\ldots,z_T)$

Define: $\alpha_t(i) = P(x_t = s_i, z_1,\ldots,z_t)$

$$P(x_{t+1}, z_1,\ldots,z_{t+1}) = P(z_{t+1}|x_{t+1})P(x_{t+1}, z_1,\ldots,z_t)$$

$$= P(z_{t+1}|x_{t+1}) \sum_{x_t} P(x_t, x_{t+1}, z_1,\ldots,z_t)$$

$$= P(z_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t) P(x_t, z_1,\ldots,z_t)$$

Or ... $\alpha_{t+1}(j) = b_j(z_{t+1}) \sum_{i=1}^{N} a_{ij}\alpha_t(i)$

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1,\ldots,z_T)$

Define: $\alpha_t(i) = P(x_t = s_i, z_1,\ldots,z_t)$

1. $\alpha_1(i) = \pi_i b_i(z_1)$

2. $\forall t \in [1, T-1]$

3. $\forall j \in [1, N], \alpha_{t+1}(j) = b_j(z_{t+1}) \sum_{i=1}^{N} a_{ij}\alpha_t(i)$

Also called the *forward algorithm*

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1,\ldots,z_T)$

Define: $\alpha_t(i) = P(x_t = s_i, z_1,\ldots,z_t)$

1. $\alpha_1(i) = \pi_i b_i(z_1)$

2. $\forall t \in [1, T-1]$

3. $\forall j \in [1,N], \alpha_{t+1}(j) = b_j(z_{t+1}) \sum_{i=1}^{N} a_{ij}\alpha_t(i)$

Cost: $O((T-1)N^2)$

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1,\ldots,z_T)$

Define: $\alpha_t(i) = P(x_t, z_1,\ldots,z_t)$

Last step:

$$P(x_T = s_i|z_1,\ldots,z_T) = \frac{\alpha_T(i)}{\sum_{j=1}^{N}\alpha_T(j)}$$

Problem #2: Smoothing

If the robot starts in prior distribution, π

and make a sequence of observations, $z_1, z_2, \ldots, z_T$

Then, where is the robot at time t<T?

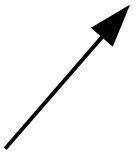
In other words, what is: $P(x_t|z_1, \ldots, z_T)$?

Smoothing: DP solution

Find: $P(x_t|z_1,\ldots,z_T)$

Notice:

$$P(x_t = s_i, z_1,\ldots,z_T) = P(x_t = s_i, z_1,\ldots,z_t)P(z_{t+1},\ldots,z_T|x_t = s_i)$$



why?

Smoothing: DP solution

Find: $P(x_t|z_1,\ldots,z_T)$

Notice:

$$P(x_t = s_i, z_1,\ldots,z_T) = \underbrace{P(x_t = s_i, z_1,\ldots,z_t)}_{\alpha_t(i)}\underbrace{P(z_{t+1},\ldots,z_T|x_t = s_i)}_{\beta_t(i)}$$

Therefore: $P(x_t = s_i|z_1,\ldots,z_T) = \dfrac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}$

Smoothing: DP solution

Find recursion for:

$$\beta_t(i) = P(z_{t+1}, \ldots, z_T | x_t = s_i)$$

Smoothing: DP solution

Find recursion for:

$$\beta_t(i) = P(z_{t+1}, \ldots, z_T | x_t = s_i)$$

$$P(z_{t+1}, \ldots, z_T | x_t) = \sum_{x_{t+1}} P(z_{t+1}, \ldots, z_T, x_{t+1} | x_t)$$

$$= \sum_{x_{t+1}} P(z_{t+2}, \ldots, z_T | x_{t+1}) P(z_{t+1}, x_{t+1} | x_t)$$

$$= \sum_{x_{t+1}} P(z_{t+2}, \ldots, z_T | x_{t+1}) P(z_{t+1} | x_{t+1}) P(x_{t+1} | x_t)$$

Smoothing: DP solution

$$\overbrace{P(z_{t+1},\ldots,z_T|x_t)}^{\beta_t(i)} = \sum_{x_{t+1}} P(z_{t+1},\ldots,z_T,x_{t+1}|x_t)$$

$$= \sum_{x_{t+1}} P(z_{t+2},\ldots,z_T|x_{t+1})P(z_{t+1},x_{t+1}|x_t)$$

$$= \sum_{x_{t+1}} \underbrace{P(z_{t+2},\ldots,z_T|x_{t+1})}_{\beta_{t+1}(i)} P(z_{t+1}|x_{t+1})P(x_{t+1}|x_t)$$

Smoothing: DP solution

$$\overbrace{P(z_{t+1},\ldots,z_T|x_t)}^{\beta_t(i)} = \sum_{x_{t+1}} P(z_{t+1},\ldots,z_T,x_{t+1}|x_t)$$

$$= \sum_{x_{t+1}} P(z_{t+2},\ldots,z_T|x_{t+1})P(z_{t+1},x_{t+1}|x_t)$$

$$= \sum_{x_{t+1}} \underbrace{P(z_{t+2},\ldots,z_T|x_{t+1})}_{\beta_{t+1}(i)} P(z_{t+1}|x_{t+1})P(x_{t+1}|x_t)$$

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(i) b_j(z_{t+1}) a_{ij}$$

Smoothing: DP solution

Find: $P(x_t|z_1, \ldots, z_T)$

Define: $\beta_t(i) = P(z_{t+1}, \ldots, z_T|x_t = s_i)$

1. $\forall i \in [1, N], \beta_T(i) = 1$

2. $\forall t \in [1, T-1]$

3. $$\forall i \in [1, N], \beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) b_j(z_{t+1}) a_{ij}$$

Smoothing: DP solution

Find: $P(x_t|z_1, \ldots, z_T)$

Define: $\beta_t(i) = P(z_{t+1}, \ldots, z_T|x_t = s_i)$

1. $\forall i \in [1, N], \beta_T(i) = 1$

2. $\forall t \in [1, T-1]$

3. $$\forall i \in [1, N], \beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) b_j(z_{t+1}) a_{ij}$$

Cost: $O((T-1)N^2)$

Smoothing: DP solution

Find: $P(x_t|z_1,\ldots,z_T)$

1. Calculate all alphas and betas

2. $P(x_t = s_i|z_1,\ldots,z_T) = \dfrac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}$

Also called: *forward-backward* algorithm

Viterbi: finding the most likely path

Smoothing gives us $P(x_t|z_1, \ldots, z_T)$ for any t.

- most likely state is not quite the same as the most likely path …

Viterbi will give us most likely path:

$$\arg \max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T|z_1, \ldots, z_T)$$

Viterbi

Find: $\arg \max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T|z_1,\ldots,z_T)$

Define:

$$\delta_t(i) = \max_{x_1,\ldots,x_{t-1}} P(x_1,\ldots,x_{t-1},x_t = s_i,z_1,\ldots,z_t)$$

Compare with: $\alpha_t(i) = P(x_t = s_i,z_1,\ldots,z_t)$

Filtering: dynamic programming sol'n

Find: $P(x_T|z_1,\ldots,z_T)$

Define: $\alpha_t(i) = P(x_t = s_i, z_1,\ldots,z_t)$

1. $\alpha_1(i) = \pi_i b_i(z_1)$

2. $\forall t \in [1, T-1]$

3. $\forall j \in [1, N], \alpha_{t+1}(j) = b_j(z_{t+1}) \sum_{i=1}^{N} a_{ij}\alpha_t(i)$

Viterbi

Find: $\arg \max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T|z_1,\ldots,z_T)$

Define: $\delta_t(i) = \max_{x_1,\ldots,x_{t-1}} P(x_1,\ldots,x_{t-1},x_t = s_i,z_1,\ldots,z_t)$

1. $\delta_1(i) = \pi_i b_i(z_1)$

2. $\forall t \in [1, T-1]$

3. $\forall j \in [1,N], \delta_{t+1}(j) = b_j(z_{t+1}) \max_{i\in[1,N]} a_{ij}\delta_t(i)$

4. $\forall j \in [1,N], \Psi_{t+1}(j) = \arg \max_{i\in[1,N]} a_{ij}\delta_t(i)$

Viterbi

Shorthand: $x_{1:t} \equiv x_1, \ldots, x_t$

Define: $\delta_t(i) = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t = s_i, z_{1:t})$

$$\begin{aligned}\delta_1(i) &= \pi_i b_i(z_1) \\ &= P(z_1 | x_1 = s_i) P(x_1 = s_i) = P(x_1 = s_i, z_1)\end{aligned}$$

Viterbi recursion

Recursion: $\delta_{t+1}(j) = b_j(z_{t+1}) \max_{i \in [1,N]} a_{ij} \delta_t(i)$

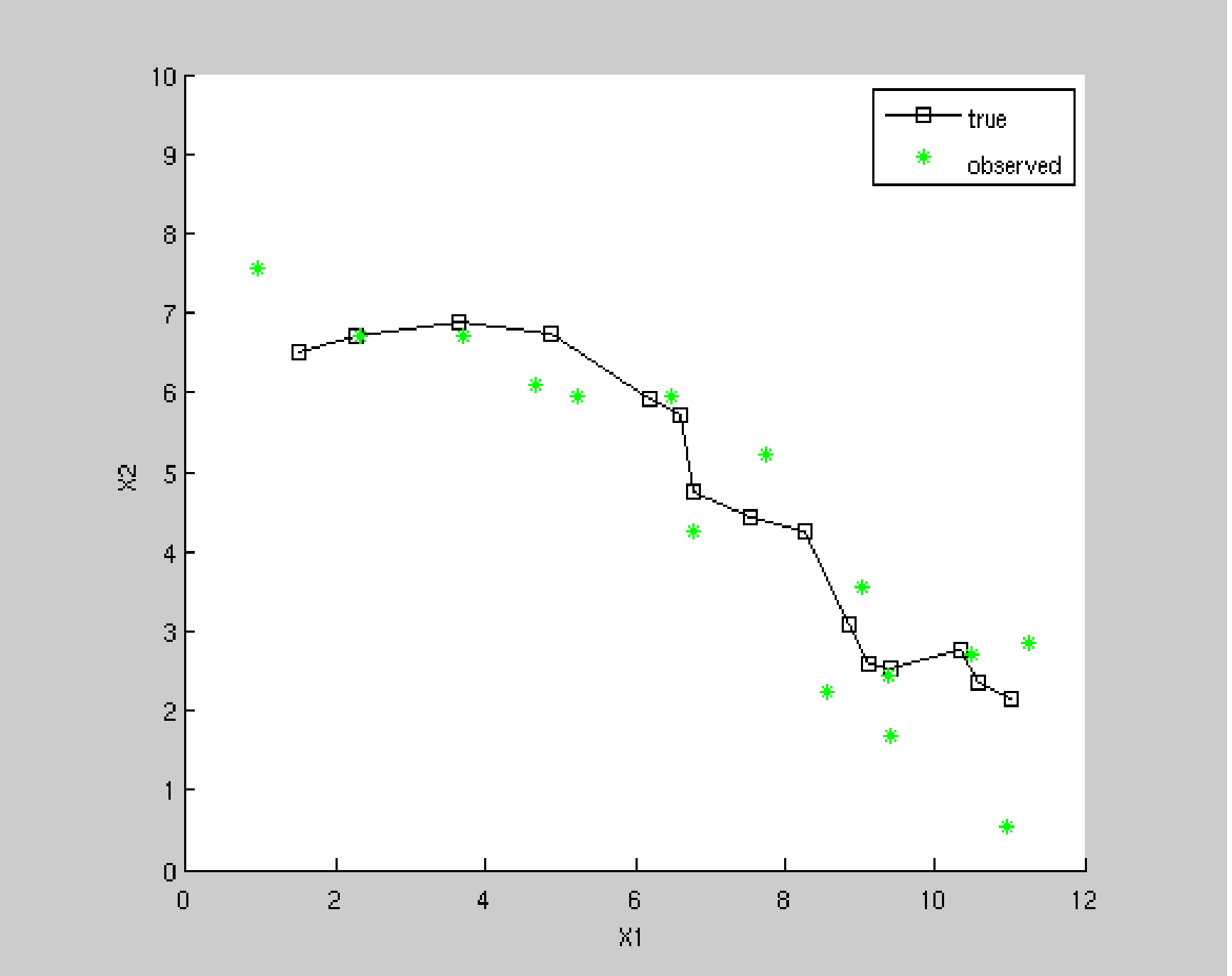
$$\max_{x_{1:t}} P(x_{1:t}, x_{t+1}, z_{1:t+1}) =$$

$$P(z_{t+1}|x_{t+1}) \max_{x_t} P(x_{t+1}|x_t) \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, z_{1:t})$$

$$b_j(z_{t+1}) \qquad a_{ij} \qquad \delta_t(i)$$

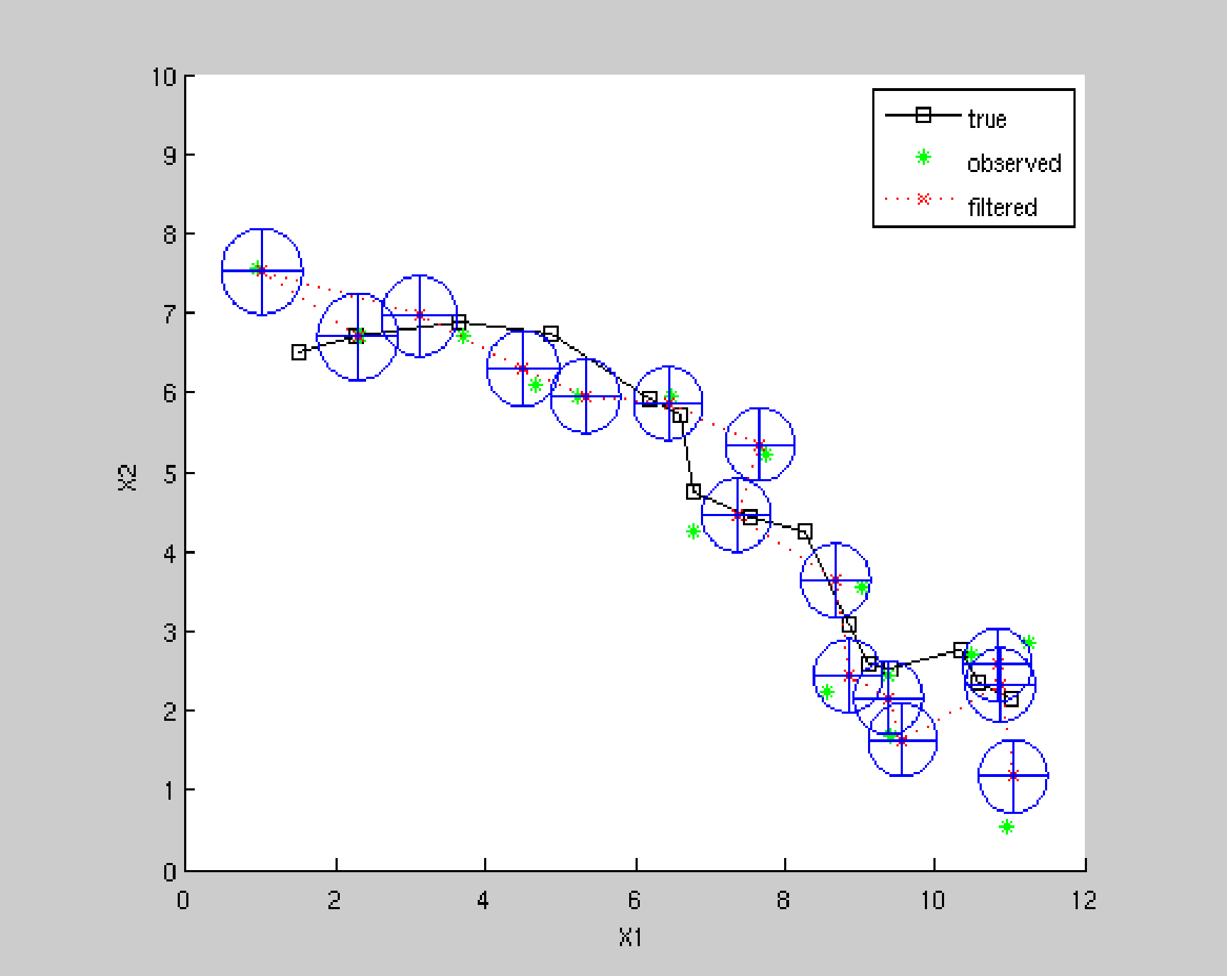
probability of most likely path to x_{t-1} followed by x_t

Filtering smoothing example: Kalman filter



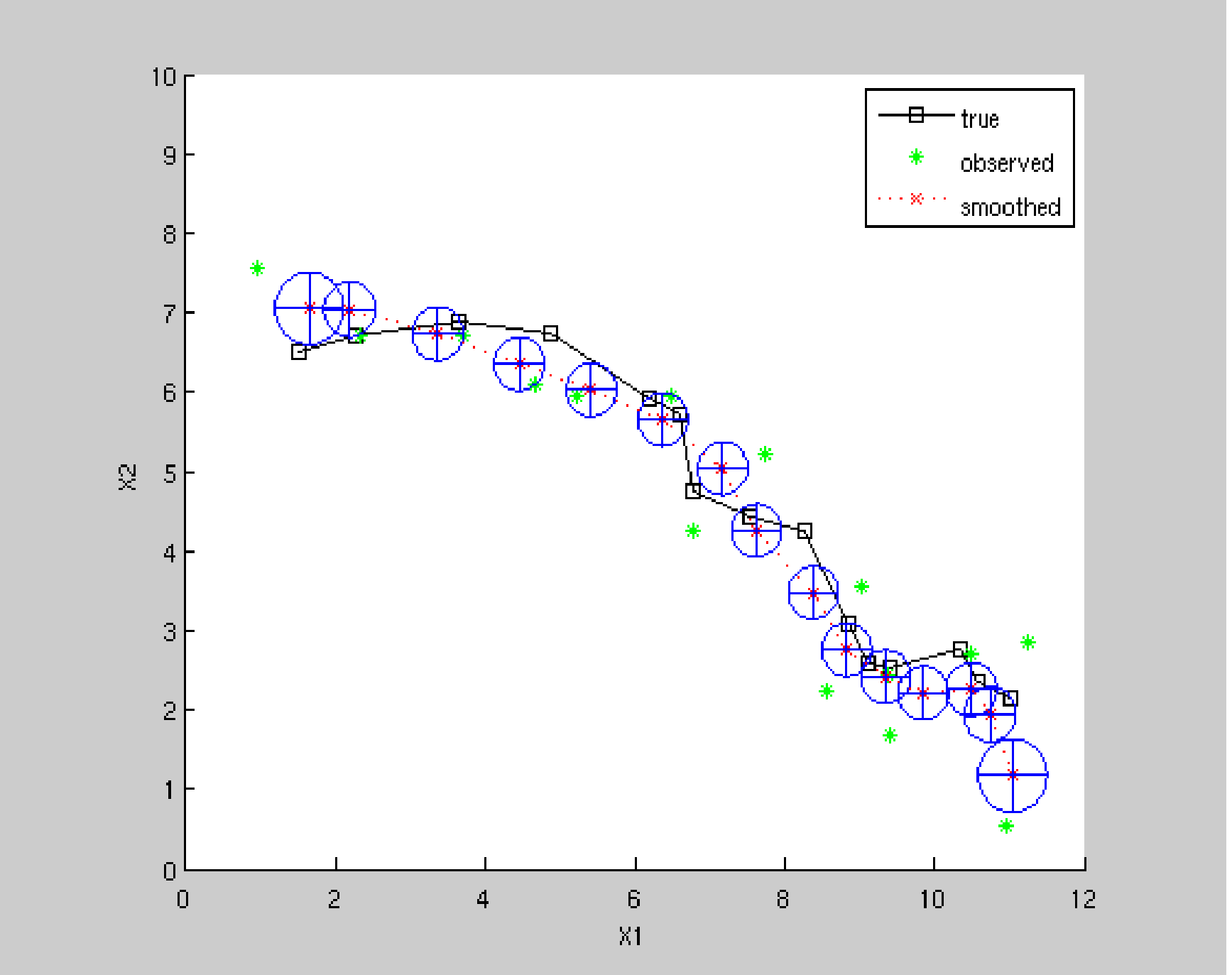
Kevin Murphy, Kalman filter toolbox

Filtering smoothing example: Kalman filter



Kevin Murphy, Kalman filter toolbox

Filtering smoothing example: Kalman filter



Kevin Murphy, Kalman filter toolbox

HMM Learning (overview)

Easy once you know filtering/smoothing

- given a series of observations, learn the parameters of the HMM, $a_{ij}, b_j(k)$

Find the parameters, $\lambda = \{a_{ij}, b_j(k); i, j \in [1, N], k \in [1, M]\}$, that maximize: $P(z_1, z_T|\lambda)$

HMM Learning (overview)

1. Given, $a_{ij}, b_k(k)$, do forward/backward to calculate α, β

HMM Learning (overview)

2. Given α, β, calculate expected $a_{ij}, b_j(k)$:

- expected value of a_{ij}:

$$\bar{a}_{ij} = \sum_{t=1}^{T-1} \frac{P(x_t = s_i, x_{t+1} = s_j | z_1, \ldots, z_T)}{P(x_t = s_i | z_1, \ldots, z_T)}$$

- expected value of b_j:

$$\bar{b}_j(k) = \sum_{t=1}^{T-1} \frac{P(x_t = s_i, z_t = o_k | z_1, \ldots, z_T)}{P(x_t = s_i | z_1, \ldots, z_T)}$$

HMM Learning (overview)

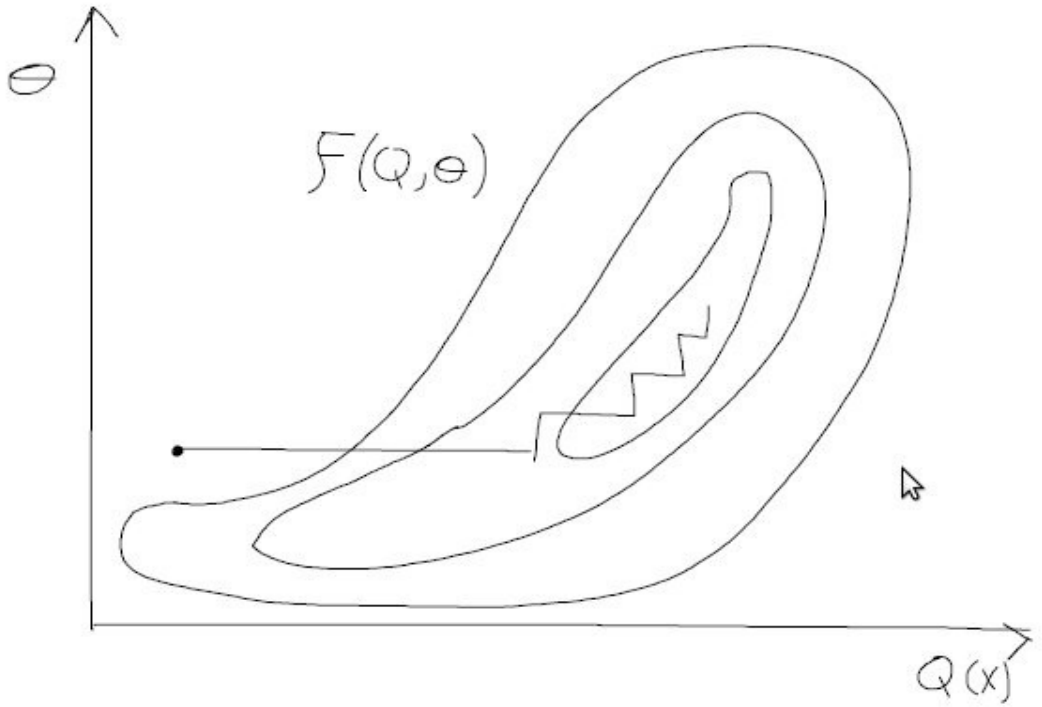
3. Goto step 1, using new values of $a_{ij}, b_j(k)$

This procedure is an application of EM (expectation maximization) known as the Baum-Welch algorithm.

- procedure converges to local optimum
- not necessarily globally optimal

HMM Learning (overview)

Baum-Welch can be viewed as coordinate ascent:



Roweis, Ghahramani, 2001

Summary

1. Markov Processes
2. Hidden Markov Models
3. Filtering (forward algorithm)
4. Smoothing (backward algorithm)
5. Viterbi (most likely path)
6. Parameter learning

References

1. Rabiner, *A tutorial on hidden Markov models and selected applications in speech Recognition,* Proc. of the IEEE, 1989

2. Bishop, *Pattern recognition and machine Learning*, Springer, Ch 13, 2006