

Sequential Data

Oliver Schulte - CMPT 726

Bishop PRML Ch. 13
Russell and Norvig, AIMA

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

Temporal Models

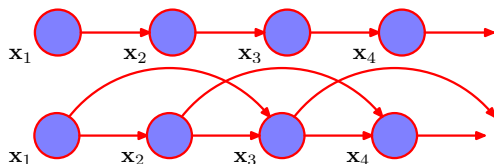
- The world changes over time
 - Explicitly model this change using Bayesian networks
 - Undirected models also exist (will not cover)
- Basic idea: copy state and evidence variables for each time step
- e.g. Diabetes management
- \mathbf{z}_t is set of **unobservable state variables** at time t
 - *bloodSugar_t, stomachContents_t, ...*
- \mathbf{x}_t is set of **observable evidence variables** at time t
 - *measuredBloodSugar_t, foodEaten_t, ...*
- Assume **discrete time step**, fixed
- Notation: $\mathbf{x}_{a:b} = \mathbf{x}_a, \mathbf{x}_{a+1}, \dots, \mathbf{x}_{b-1}, \mathbf{x}_b$

Markov Chain

- Construct Bayesian network from these variables
 - parents? distributions? for state variables z_t :

Markov Chain

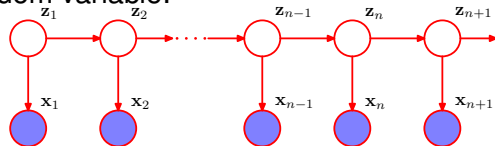
- Construct Bayesian network from these variables
 - parents? distributions? for state variables z_t :
- **Markov assumption**: z_t depends on **bounded** subset of $z_{1:t-1}$
 - **First-order Markov process**: $p(z_t | z_{1:t-1}) = p(z_t | z_{t-1})$
 - **Second-order Markov process**: $p(z_t | z_{1:t-1}) = p(z_t | z_{t-2}, z_{t-1})$



- **Stationary process**: $p(z_t | z_{t-1})$ fixed for all t

Hidden Markov Model (HMM)

- **Sensor Markov assumption:** $p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t | \mathbf{z}_t)$
- **Stationary process:** transition model $p(\mathbf{z}_t | \mathbf{z}_{t-1})$ and sensor model $p(\mathbf{x}_t | \mathbf{z}_t)$ fixed for all t (separate $p(\mathbf{z}_1)$)
- HMM special type of Bayesian network, \mathbf{z}_t is a **single discrete** random variable:

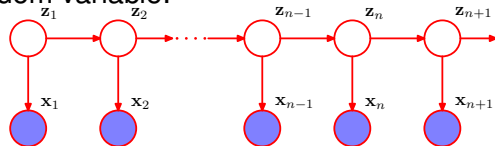


- **Joint distribution:**

$$p(\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) =$$

Hidden Markov Model (HMM)

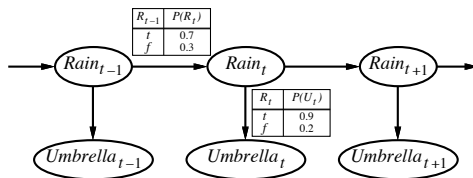
- **Sensor Markov assumption:** $p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t | \mathbf{z}_t)$
- **Stationary process:** transition model $p(\mathbf{z}_t | \mathbf{z}_{t-1})$ and sensor model $p(\mathbf{x}_t | \mathbf{z}_t)$ fixed for all t (separate $p(\mathbf{z}_1)$)
- HMM special type of Bayesian network, \mathbf{z}_t is a **single discrete** random variable:



- **Joint distribution:**

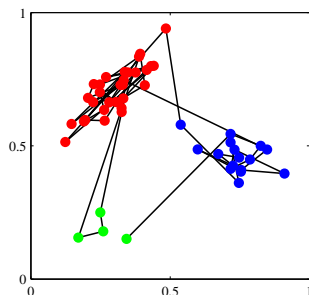
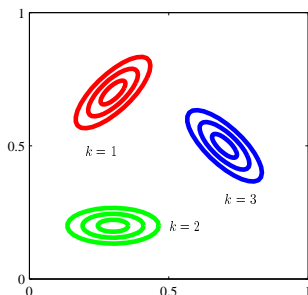
$$p(\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = p(\mathbf{z}_1) \prod_{i=2:t} p(\mathbf{z}_i | \mathbf{z}_{i-1}) \prod_{i=1:t} p(\mathbf{x}_i | \mathbf{z}_i)$$

HMM Example



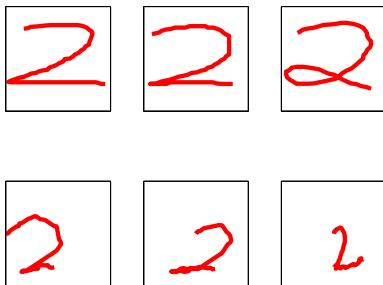
- First-order Markov assumption not true in real world
- Possible fixes:
 - **Increase order** of Markov process
 - **Augment state**, add $temp_t$, $pressure_t$

Generating Data with HMMs



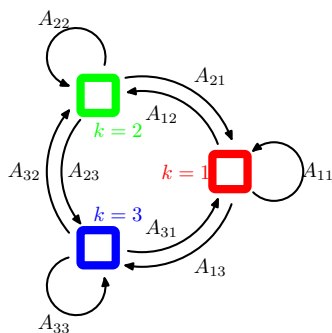
- z with 3 latent states, 2 dimensional observation x .
- left: contour map of emission probabilities.
- right: sample of 50 points.

Generating Sequences with HMMs



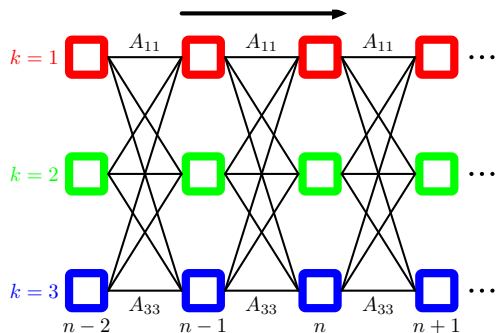
- Data are pen trajectory as it is writing the digit.
- Train HMM on 45 handwritten digits.
- Use HMM to randomly generate 2s.

Transition Diagram



- z_n takes one of 3 values
- Using one-of- K coding scheme, $z_{nk} = 1$ if in state k at time n
- **Transition matrix** A where $p(z_{nk} = 1 | z_{n-1,j} = 1) = A_{jk}$

Lattice / Trellis Representation



- The **lattice** or **trellis** representation shows possible paths through the latent state variables z_n

Applications, Pros and Cons

HMMs are widely applied. For example:

- Speech recognition
- Part-of-Speech tagging (e.g., John hit Mary \rightarrow NP VP NP).
- Gene sequence modelling.

Pros

- Conceptually simple.
- With small number of states, computationally tractable.

Cons

- Black box, states may not have interpretation.
- Complexity grows exponentially in number of states:
trade-off between expressiveness and complexity.

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

Inference Tasks

- **Filtering:** $p(z_t|x_{1:t})$
 - Estimate current unobservable state given all observations to date
- **Prediction:** $p(z_k|x_{1:t})$ for $k > t$
 - Similar to filtering, without evidence
- **Smoothing:** $p(z_k|x_{1:t})$ for $k < t$
 - Better estimate of past states
- **Most likely explanation:** $\arg \max_{z_{1:t}} p(z_{1:t}|x_{1:t})$
 - e.g. speech recognition, decoding noisy input sequence

Filtering

- Aim: devise a **recursive** state estimation algorithm:

$$p(z_{t+1}|x_{1:t+1}) = f(x_{t+1}, p(z_t|x_{1:t}))$$

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &= p(z_{t+1}|x_{1:t}, x_{t+1}) \\ &\propto p(x_{t+1}|x_{1:t}, z_{t+1})p(z_{t+1}|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1})p(z_{t+1}|x_{1:t}) \end{aligned}$$

- I.e. **prediction** + **estimation**. Prediction by summing out z_t :

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &\propto p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}, z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t, x_{1:t})p(z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)p(z_t|x_{1:t}) \end{aligned}$$

Filtering

- Aim: devise a **recursive** state estimation algorithm:

$$p(z_{t+1}|x_{1:t+1}) = f(x_{t+1}, p(z_t|x_{1:t}))$$

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &= p(z_{t+1}|x_{1:t}, x_{t+1}) \\ &\propto p(x_{t+1}|x_{1:t}, z_{t+1})p(z_{t+1}|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1})p(z_{t+1}|x_{1:t}) \end{aligned}$$

- I.e. **prediction** + **estimation**. Prediction by summing out z_t :

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &\propto p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}, z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t, x_{1:t})p(z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)p(z_t|x_{1:t}) \end{aligned}$$

Filtering

- Aim: devise a **recursive** state estimation algorithm:

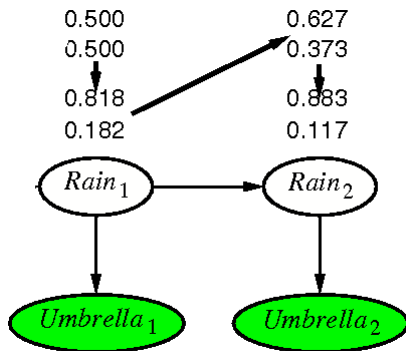
$$p(z_{t+1}|x_{1:t+1}) = f(x_{t+1}, p(z_t|x_{1:t}))$$

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &= p(z_{t+1}|x_{1:t}, x_{t+1}) \\ &\propto p(x_{t+1}|x_{1:t}, z_{t+1})p(z_{t+1}|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1})p(z_{t+1}|x_{1:t}) \end{aligned}$$

- I.e. **prediction** + **estimation**. Prediction by summing out z_t :

$$\begin{aligned} p(z_{t+1}|x_{1:t+1}) &\propto p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}, z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t, x_{1:t})p(z_t|x_{1:t}) \\ &= p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)p(z_t|x_{1:t}) \end{aligned}$$

Filtering Example



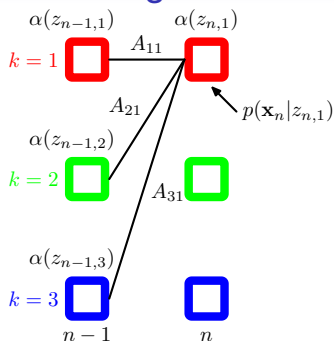
R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

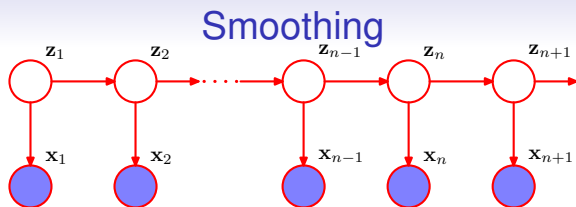
$$p(\text{rain}_1 = \text{true}) = 0.5$$

$$p(z_{t+1}|x_{1:t+1}) \propto p(x_{t+1}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)p(z_t|x_{1:t})$$

Filtering - Lattice



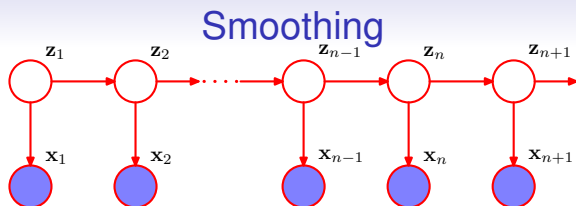
- Using notation in PRML, forward message is $\alpha(z_n)$, updated probability of time- n state.
- Compute $\alpha(z_{n,i})$ using sum over k of $\alpha(z_{n-1,k})$ multiplied by A_{ki} , then multiplying in evidence $p(x_t | z_{ni})$
- Each step, computing $\alpha(z_n)$ takes $O(K^2)$ time, with K values for z_n



- Divide evidence $x_{1:t}$ into $x_{1:n-1}$, $x_{n:t}$.
- Intuitively: what is probability of getting to a state by time $n - 1$ given the previous observations, and what is the probability of continuing with the future observations?

$$\begin{aligned}
 p(z_{n-1} | x_{1:t}) &= p(z_{n-1} | x_{1:n-1}, x_{n:t}) \\
 &\propto p(z_{n-1} | x_{1:n-1}) p(x_{n:t} | z_{n-1}, x_{1:n-1}) \\
 &= p(z_{n-1} | x_{1:n-1}) p(x_{n:t} | z_{n-1}) \\
 &\equiv \alpha(z_{n-1}) \beta(z_{n-1})
 \end{aligned}$$

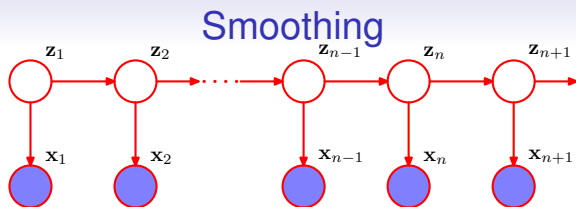
- Backwards message $\beta(z_{n-1})$ another recursion:



- Divide evidence $x_{1:t}$ into $x_{1:n-1}$, $x_{n:t}$.
- Intuitively: what is probability of getting to a state by time $n - 1$ given the previous observations, and what is the probability of continuing with the future observations?

$$\begin{aligned}
 p(z_{n-1} | x_{1:t}) &= p(z_{n-1} | x_{1:n-1}, x_{n:t}) \\
 &\propto p(z_{n-1} | x_{1:n-1}) p(x_{n:t} | z_{n-1}, x_{1:n-1}) \\
 &= p(z_{n-1} | x_{1:n-1}) p(x_{n:t} | z_{n-1}) \\
 &\equiv \alpha(z_{n-1}) \beta(z_{n-1})
 \end{aligned}$$

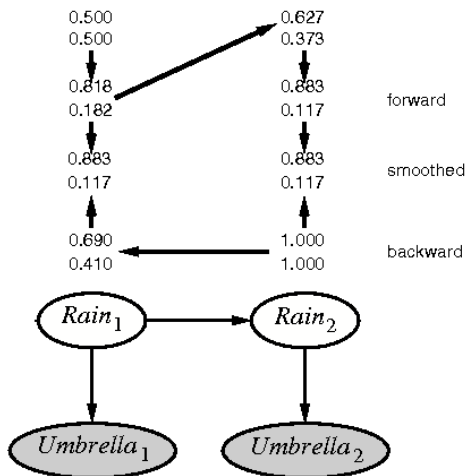
- Backwards message $\beta(z_{n-1})$ another recursion:



- Divide evidence $x_{1:t}$ into $x_{1:n-1}$, $x_{n:t}$,
 $p(z_{n-1}|x_{1:t}) \propto \alpha(z_{n-1})\beta(z_{n-1})$
- Backwards message another recursion:

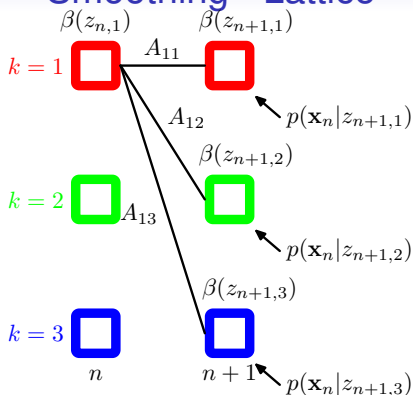
$$\begin{aligned}
 p(x_{n:t}|z_{n-1}) &= \sum_{z_n} p(x_{n:t}, z_n | z_{n-1}) \\
 &= \sum_{z_n} p(x_{n:t} | z_n, z_{n-1}) p(z_n | z_{n-1}) \\
 &= \sum_{z_n} p(x_{n:t} | z_n) p(z_n | z_{n-1}) \\
 &= \sum_{z_n} p(x_n | z_n) \mathbf{p}(x_{n+1:t} | z_n) p(z_n | z_{n-1})
 \end{aligned}$$

Smoothing Example



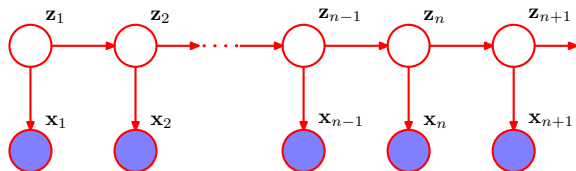
change 0.410 to 0.310

Smoothing - Lattice



- Using notation in PRML, backward message is $\beta(z_n)$
- Compute $\beta(z_{n,i})$ using sum over k of $\beta(z_{n+1,k})$ multiplied by A_{ik} and evidence $p(x_{n+1} | z_{n+1,k})$
- Each step, computing $\beta(z_n)$ takes $O(K^2)$ time, with K values for z_n

Forward-Backward Algorithm for Smoothing



- Filter from time 1 to N , and cache forward messages $\alpha(z_n)$
- Cache backward messages $\beta(z_n)$ from N to 1.
- Smooth: now compute $p(z_n | x_1, x_2, \dots, x_N)$ for all n
- Total complexity $O(NK^2)$
- a.k.a Baum-Welch algorithm
- Demo: <http://cmble.com/ForwardBackwardAlgorithm.jsp>

Outline

Hidden Markov Models

Inference for HMMs

Learning for HMMs

HMM Parameters

- The **parameters** of an HMM are:
 - **Transition matrix** A where $p(z_{nk} = 1 | z_{n-1,j} = 1) = A_{jk}$
 - **Sensor model** ϕ_k parameters to each $p(\mathbf{x}_n | z_{nk} = 1, \phi_k)$ (e.g. ϕ_k could be mean and variance of Gaussian)
 - **Prior for initial state** z_1 , model as multinomial
 $p(z_{1k} = 1) = \pi_k$, parameters π
- Call these parameters $\theta = (A, \pi, \phi)$
- **Learning problem: given one sequence x , find best θ**
 - Extension to multiple sequences straight-forward (assume independent, log of product is sum)

Maximum Likelihood for HMMs

- We can use maximum likelihood to choose the best parameters:

$$\theta_{ML} = \arg \max p(\mathbf{x}|\theta)$$

- Unfortunately this is hard to do: we can get $p(\mathbf{x}|\theta)$ by summing out from the joint distribution:

$$\begin{aligned} p(\mathbf{x}|\theta) &= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_N} p(\mathbf{x}, z_1, z_2, \dots, z_N | \theta) \\ &\equiv \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z} | \theta) \end{aligned}$$

- But this sum has K^N terms in it
- And, as in the mixture distribution case, no simple closed-form solution
- Instead, use expectation-maximization (EM)

EM for HMMs

- Start with initial guess for parameters $\theta^{old} = (\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- **E-step**: Calculate posterior on latent variables $p(\mathbf{z}|\mathbf{x}, \theta^{old})$
- **M-step**: Maximize $Q(\boldsymbol{\theta}, \theta^{old}) = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \theta^{old}) \ln p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$
- The details are covered in the book.

HMM EM Summary

- Start with initial guess for parameters $\theta^{old} = (\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- Run forward-backward algorithm to get all messages $\alpha(\mathbf{z}_n)$, $\beta(\mathbf{z}_n)$ (E-step)
 - $O(NK^2)$ time complexity
 - Can use these to compute any smoothed posterior $p(z_{nk} = 1 | \mathbf{x}, \theta^{old})$
 - Also can compute any $p(z_{n-1,j} = 1, z_{n,k} = 1 | \mathbf{x}, \theta^{old})$
- Using these, update values for parameters (M-step)
 - π_k is smoothed probability of being in state k at time 1
 - A_{jk} is smoothed probability of transitioning from state j to k averaged over all time steps
 - ϕ is weighted sensor parameters using smoothed probabilities (e.g. similar to mixture of Gaussians)
- Repeat until convergence

HMM EM Summary

- Start with initial guess for parameters $\theta^{old} = (\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- Run forward-backward algorithm to get all messages $\alpha(\mathbf{z}_n)$, $\beta(\mathbf{z}_n)$ (E-step)
 - $O(NK^2)$ time complexity
 - Can use these to compute any smoothed posterior $p(z_{nk} = 1 | \mathbf{x}, \theta^{old})$
 - Also can compute any $p(z_{n-1,j} = 1, z_{n,k} = 1 | \mathbf{x}, \theta^{old})$
- Using these, update values for parameters (M-step)
 - π_k is smoothed probability of being in state k at time 1
 - A_{jk} is smoothed probability of transitioning from state j to k averaged over all time steps
 - ϕ is weighted sensor parameters using smoothed probabilities (e.g. similar to mixture of Gaussians)
- Repeat until convergence

HMM EM Summary

- Start with initial guess for parameters $\theta^{old} = (\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- Run forward-backward algorithm to get all messages $\alpha(\mathbf{z}_n)$, $\beta(\mathbf{z}_n)$ (E-step)
 - $O(NK^2)$ time complexity
 - Can use these to compute any smoothed posterior $p(z_{nk} = 1 | \mathbf{x}, \theta^{old})$
 - Also can compute any $p(z_{n-1,j} = 1, z_{n,k} = 1 | \mathbf{x}, \theta^{old})$
- **Using these, update values for parameters (M-step)**
 - π_k is smoothed probability of being in state k at time 1
 - A_{jk} is smoothed probability of transitioning from state j to k averaged over all time steps
 - $\boldsymbol{\phi}$ is weighted sensor parameters using smoothed probabilities (e.g. similar to mixture of Gaussians)
- Repeat until convergence

HMM EM Summary

- Start with initial guess for parameters $\theta^{old} = (\mathbf{A}, \boldsymbol{\pi}, \boldsymbol{\phi})$
- Run forward-backward algorithm to get all messages $\alpha(\mathbf{z}_n)$, $\beta(\mathbf{z}_n)$ (E-step)
 - $O(NK^2)$ time complexity
 - Can use these to compute any smoothed posterior $p(z_{nk} = 1 | \mathbf{x}, \theta^{old})$
 - Also can compute any $p(z_{n-1,j} = 1, z_{n,k} = 1 | \mathbf{x}, \theta^{old})$
- Using these, update values for parameters (M-step)
 - π_k is smoothed probability of being in state k at time 1
 - A_{jk} is smoothed probability of transitioning from state j to k averaged over all time steps
 - ϕ is weighted sensor parameters using smoothed probabilities (e.g. similar to mixture of Gaussians)
- Repeat until convergence

Conclusion

- Readings: Ch. 13.2, 13.2.1, 13.2.2
- HMM - Probabilistic model of temporal data
 - Discrete hidden (unobserved, latent) state variable at each time
 - Continuous (next week)
 - Observation (can be discrete / continuous) at each time
 - Conditional independence assumptions (Markov)
 - Assumptions on distributions (stationary)
- Inference
 - Filtering
 - Smoothing
 - Most likely sequence (next week)
- Maximum likelihood learning
 - EM – efficient computation $O(NK^2)$ time using forward-backward smoothing