COMP 652: Machine Learning

Lecture 20

Today

- Hidden Markov models
- Inference and learning problems
- Forward-backward algorithm
- Baum-Welch for parameter fitting

Hidden Markov models

- Hidden Markov models model situations in which we receive a sequence of observations that depend on a dynamical system, but we do not observe the state of the system itself.
- There are numerous applications, including:
  - Speech recognition
  - Robot localization
  - Gene finding
  - User modeling
  - Fetal heartrate monitoring

Hidden Markov models definition

A hidden Markov model (HMM) is defined by:

- A finite state set S
- State transition probabilities $p_{ss'}$ for $s, s' \in S$
- A finite observation set O
- Observation emission probabilities p_{so} for $s \in S$ and $o \in O$
- A start state distribution p_s for $s \in S$

The meaning of the model

- Let $S_1, S_2, S_3, \ldots$ be the state of the model at times $t = 1, 2, 3, \ldots$
- The initial state is chosen according to p_s

$$P(S_1 = s) = p_s$$

- Subsequent transitions are according to $p_{ss'}$.

$$P(S_{t+1} = s'|S_t = s) = p_{ss'} \text{ for all } t > 1$$

- At each time t, we make an observation o_t, the result of which depends (stochastically) on the state of the system:

$$P(O_t = o|S_t = s) = p_{so}$$

$\Rightarrow$ What's interesting is that we never know the S_t – we just observe the O_t, and have to make inferences about the S_t.

HMM Problems

- How likely is a given observation sequence, $o_1, o_2, \ldots, o_T$. I.e., assuming we know $p_{ss'}, p_{so}, p_s$, compute:

$$P(O_1 = o_1, O_2 = o_2, \ldots, O_T = o_T)$$

- Given an observation sequence, what is the probability distribution for the current state?

$$P(S_T = s | O_1 = o_1, O_2 = o_2, \ldots, O_T = o_T)$$

- What is the most likely state sequence for explaining a given observation sequence? ("Decoding problem")

$$\arg\max_{s_1,\ldots,s_T} P(S_1 = s_1, \ldots, S_T = s_T | O_1 = o_1, \ldots, O_T = o_T)$$

- Given one (or more) observation sequence(s), find the maximum likelihood model parameters.

$$\arg\max_{p_{ss'}, p_{so}, p_s} P(O_1 = o_1, \ldots, O_2 = o_2)$$

- Are these all machine learning problems? When do such problems arise?

Computing the probability of an observation sequence

If we know the model parameters, how do we compute $P(o_1, o_2, \ldots, o_T)$?

Computing the probability of an observation sequence

If we know the model parameters, how do we compute $P(o_1, o_2, \ldots, o_T)$?

- How not to do it:

$$\begin{aligned}
P(o_1, \ldots, o_T) &= \sum_{s_1, \ldots, s_T} P(o_1, \ldots, o_T, s_1, \ldots, s_T) \\
&= \sum_{s_1, \ldots, s_T} P(o_1, \ldots, o_T | s_1, \ldots, s_T) P(s_1, \ldots, s_T) \\
&= \sum_{s_1, \ldots, s_T} \left(\Pi_{t=1}^{T} P(o_t | s_t)\right) P(s_1) \left(\Pi_{t=2}^{T} P(s_t | s_{t-1})\right) \\
&= \sum_{s_1, \ldots, s_T} \left(\Pi_{t=1}^{T} p_{s_t o_t}\right) p_{s_1} \left(\Pi_{t=2}^{T} p_{s_{t-1} s_t}\right)
\end{aligned}$$

- Why not that way?

Computing the probability of an observation sequence

If we know the model parameters, how do we compute $P(o_1, o_2, \ldots, o_T)$?

- How not to do it:

$$\begin{aligned}
P(o_1, \ldots, o_T) &= \sum_{s_1,\ldots,s_T} P(o_1, \ldots, o_T, s_1, \ldots, s_T) \\
&= \sum_{s_1,\ldots,s_T} P(o_1, \ldots, o_T | s_1, \ldots, s_T) P(s_1, \ldots, s_T) \\
&= \sum_{s_1,\ldots,s_T} \left(\Pi_{t=1}^T P(o_t|s_t)\right) P(s_1) \left(\Pi_{t=2}^T P(s_t|s_{t-1})\right) \\
&= \sum_{s_1,\ldots,s_T} \left(\Pi_{t=1}^T p_{s_t o_t}\right) p_{s_1} \left(\Pi_{t=2}^T p_{s_{t-1} s_t}\right)
\end{aligned}$$

- Why not that way? Computational burden: $O(T|S|^T)$ operations

Computing the probability of an observation sequence

If we know the model parameters, how do we compute $P(o_1, o_2, \ldots, o_T)$?

- First observe that:

$$P(o_1, o_2, \ldots, o_T) = \sum_{s_T} P(o_1, o_2, \ldots, o_T, s_T)$$

- Let's work on the term inside the sum:

$$\begin{aligned}
& P(o_1, \ldots, o_T, s_T) \\
= \; & P(o_T | o_1, \ldots, o_{T-1}, s_T) P(o_1, \ldots, o_{T-1}, s_T) \\
= \; & P(o_T | s_T) \sum_{s_{T-1}} P(o_1, \ldots, o_{T-1}, s_{T-1}, s_T) \\
= \; & p_{s_T o_T} \sum_{s_{T-1}} P(s_T | o_1, \ldots, o_{T-1}, s_{T-1}) P(o_1, \ldots, o_{T-1}, s_{T-1}) \\
= \; & p_{s_T o_T} \sum_{s_{T-1}} p_{s_{T-1} s_T} P(o_1, \ldots, o_{T-1}, s_{T-1})
\end{aligned}$$

$\Rightarrow$ This suggests a dynamic programming solution!

The forward algorithm

- Given an HMM model and observation sequence $o_1, \ldots, o_T$.
- Define $\alpha_t(s) = P(o_1, \ldots, o_t, S_t = s)$
- In particular,

$$\alpha_1(s) = P(o_1, S_1 = s) = P(o_1 | S_1 = s) P(S_1 = s) = p_{so_1} p_s$$

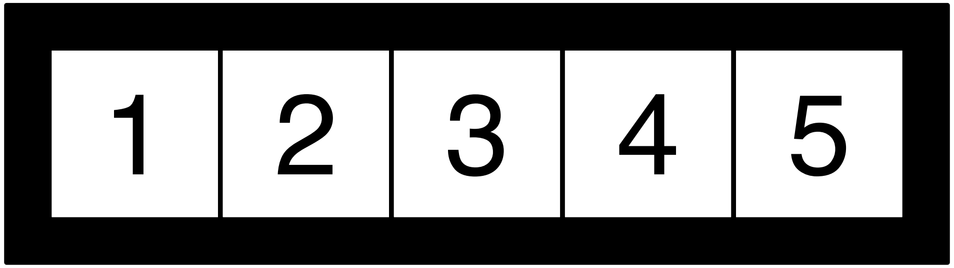
- For $t = 2, 3, \ldots, T$,

$$\alpha_t(s) = p_{so_t} \sum_{s'} p_{s's} \alpha_{t-1}(s')$$

- Solution is then

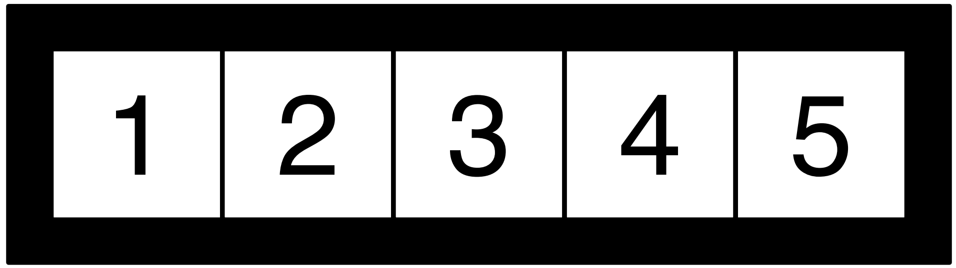
$$P(o_1, \ldots, o_T) = \sum_s \alpha_t(s)$$

Example



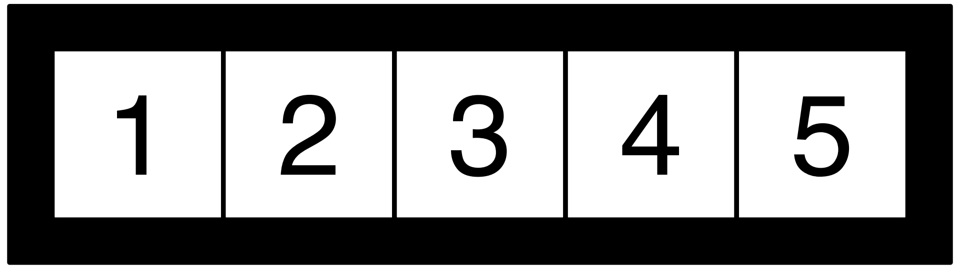
- Consider the 5-state hallway shown above
- Suppose we always start in state 3
- Suppose always observe how many walls (2 or 3) surrounding current state
- Suppose we try to step up, right, down or left with equal probability—moving to adjacent square if not blocked, else staying still

Example



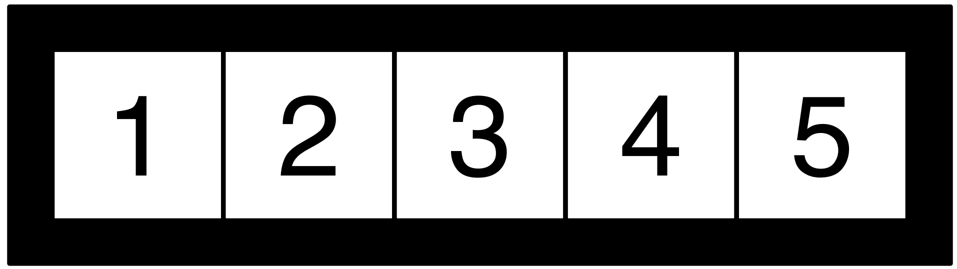
	start	to state					see walls				
state		1	2	3	4	5	0	1	2	3	4
1	0.00	0.75	0.25	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
2	0.00	0.25	0.50	0.25	0.00	0.00	0.00	0.00	1.00	0.00	0.00
3	1.00	0.00	0.25	0.50	0.25	0.00	0.00	0.00	1.00	0.00	0.00
4	0.00	0.00	0.00	0.25	0.50	0.25	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.25	0.75	0.00	0.00	0.00	1.00	0.00

Example: Forward algorithm



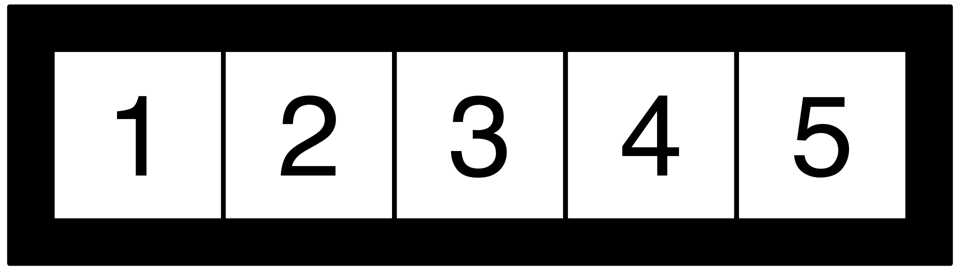
Time t	1
Obs	2
$\alpha_t(1)$	0.00000
$\alpha_t(2)$	0.00000
$\alpha_t(3)$	1.00000
$\alpha_t(4)$	0.00000
$\alpha_t(5)$	0.00000

Example: Forward algorithm



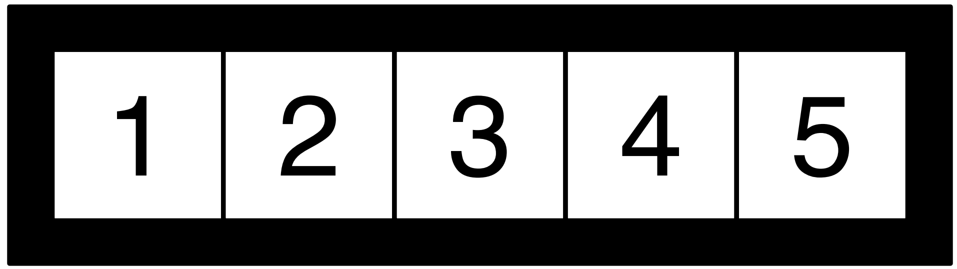
Time t	1
Obs	3
$\alpha_t(1)$	0.00000
$\alpha_t(2)$	0.00000
$\alpha_t(3)$	0.00000
$\alpha_t(4)$	0.00000
$\alpha_t(5)$	0.00000

Example: Forward algorithm



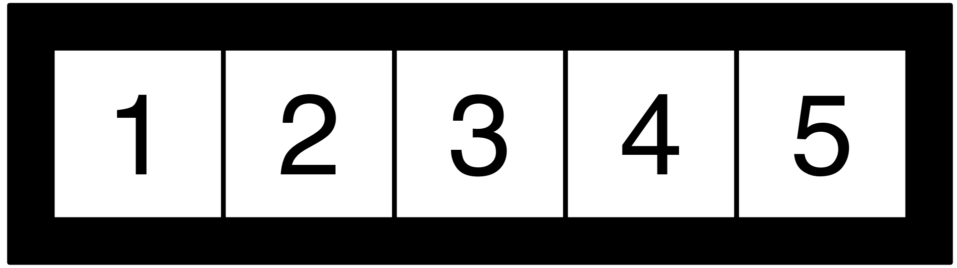
Time t	1	2
Obs	2	2
$\alpha_t(1)$	0.00000	0.00000
$\alpha_t(2)$	0.00000	0.25000
$\alpha_t(3)$	1.00000	0.50000
$\alpha_t(4)$	0.00000	0.25000
$\alpha_t(5)$	0.00000	0.00000

Example: Forward algorithm



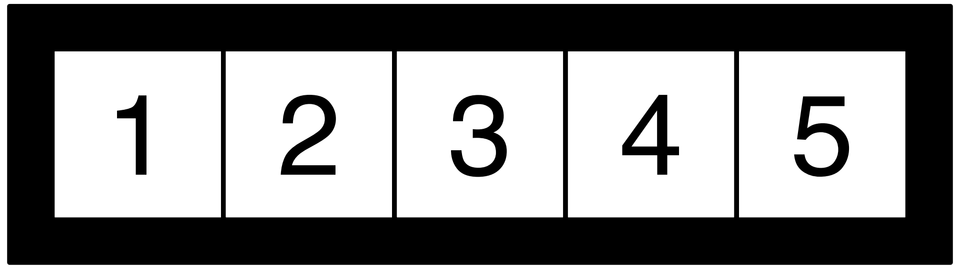
Time t	1	2	3
Obs	2	2	2
$\alpha_t(1)$	0.00000	0.00000	0.00000
$\alpha_t(2)$	0.00000	0.25000	0.25000
$\alpha_t(3)$	1.00000	0.50000	0.37500
$\alpha_t(4)$	0.00000	0.25000	0.25000
$\alpha_t(5)$	0.00000	0.00000	0.00000

Example: Forward algorithm



Time t	1	2	3
Obs	2	2	3
$\alpha_t(1)$	0.00000	0.00000	0.06250
$\alpha_t(2)$	0.00000	0.25000	0.00000
$\alpha_t(3)$	1.00000	0.50000	0.00000
$\alpha_t(4)$	0.00000	0.25000	0.00000
$\alpha_t(5)$	0.00000	0.00000	0.06250

Example: Forward algorithm



Time t	1	2	3	4	5	6	7
Obs	2	2	3	2	3	2	2
$\alpha_t(1)$	0.00000	0.00000	0.06250	0.00000	0.00391	0.00000	0.00000
$\alpha_t(2)$	0.00000	0.25000	0.00000	0.01562	0.00000	0.00098	0.00049
$\alpha_t(3)$	1.00000	0.50000	0.00000	0.00000	0.00000	0.00000	0.00049
$\alpha_t(4)$	0.00000	0.25000	0.00000	0.01562	0.00000	0.00098	0.00049
$\alpha_t(5)$	0.00000	0.00000	0.06250	0.00000	0.00391	0.00000	0.00000

Computing state probabilities

If we know the model parameters and an observation sequence, how do we compute $P(S_t = s | o_1, o_2, \ldots, o_T)$?

Computing state probabilities

If we know the model parameters and an observation sequence, how do we compute $P(S_t = s|o_1, o_2, \ldots, o_T)$?

- First, we rewrite as follows

$$
\begin{aligned}
& P(S_t = s|o_1, \ldots, o_T) \\
= \quad & \frac{P(o_1, \ldots, o_T, S_t = s)}{P(o_1, \ldots, o_T)} \\
= \quad & \frac{P(o_{t+1}, \ldots, o_T|o_1, \ldots, o_t, S_t = s)P(o_1, \ldots, o_t, S_t = s)}{P(o_1, \ldots, o_T)} \\
= \quad & \frac{P(o_{t+1}, \ldots, o_T|S_t = s)P(o_1, \ldots, o_t, S_t = s)}{P(o_1, \ldots, o_T)}
\end{aligned}
$$

- The denominator and second term in the numerator can be computed using the forward algorithm. The leaves the first term in the numerator.

Computing state probabilities (II)

- Working on the remaining term:

$$
\begin{aligned}
& P(o_{t+1}, \ldots, o_T | S_t = s) \\
= & \sum_{s'} P(o_{t+1}, \ldots, o_T, S_{t+1} = s' | S_t = s) \\
= & \sum_{s'} P(o_{t+1}, \ldots, o_T | S_{t+1} = s', S_t = s) P(S_{t+1} = s' | S_t = s) \\
= & \sum_{s'} P(o_{t+1} | S_{t+1} = s') P(o_{t+2}, \ldots, o_T | S_{t+1} = s') P(S_{t+1} = s' | S_t = s) \\
= & \sum_{s'} p_{ss'} p_{s' o_{t+1}} P(o_{t+2}, \ldots, o_T | S_{t+1} = s')
\end{aligned}
$$

The backward algorithm

- Define $\beta_t(s) = P(o_{t+1}, \ldots, o_T | S_t = s)$
- Then we can compute the β_t by the following (backwards-in-time) dynamic program:

$$\beta_T(s) = 1$$

$$\beta_t(s) = \sum_{s'} p_{ss'} p_{s' o_{t+1}} \beta_{t+1}(s') \text{ for } t = T-1, T-2, T-3, \ldots$$

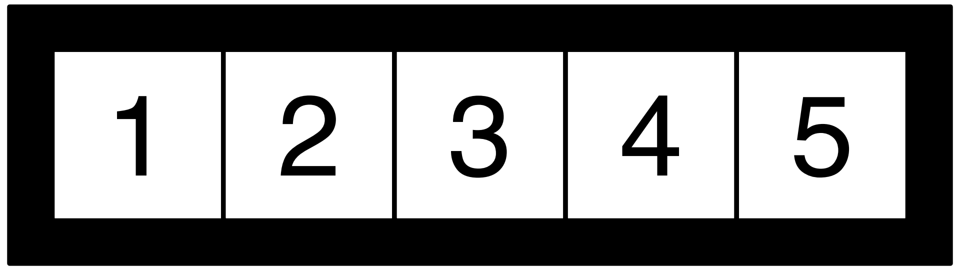
The forward-backward algorithm

Given the observation sequence, $o_1, \ldots, o_T$ we can then compute the probability of any state at any time by:

- Computing all the $\alpha_t(s)$
- Computing all the $\beta_t(s)$
- Then, for any $s \in S$ and $t \in \{1, \ldots, T\}$:

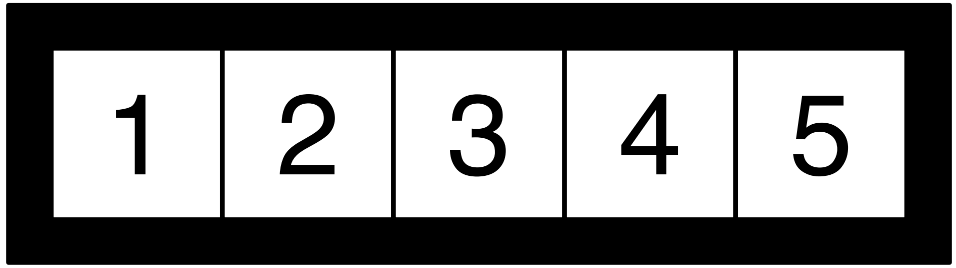
$$
\begin{aligned}
& P(S_t = s | o_1, \ldots, o_T) \\
= \quad & \frac{P(o_1, \ldots, o_t, S_t = s) P(o_{t+1}, \ldots, o_T | S_t = s)}{P(o_1, \ldots, o_T)} \\
= \quad & \frac{\alpha_t(s)\beta_t(s)}{\sum_{s'} \alpha_T(s')}
\end{aligned}
$$

Example



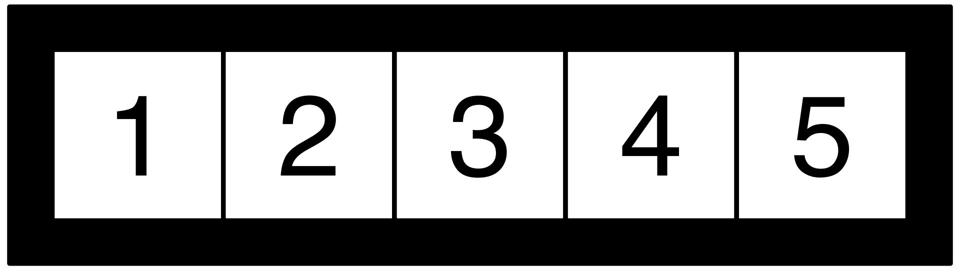
Time t	1	2	3	4	5	6
Obs	2	2	3	2	3	3
$\beta_t(1)$	0.00293	0.03516	0.04688	0.56250	0.75000	1.00000
$\beta_t(2)$	0.00586	0.01172	0.09375	0.18750	0.25000	1.00000
$\beta_t(3)$	0.00586	0.00000	0.09375	0.00000	0.00000	1.00000
$\beta_t(4)$	0.00586	0.01172	0.09375	0.18750	0.25000	1.00000
$\beta_t(5)$	0.00293	0.03516	0.04688	0.56250	0.75000	1.00000

Example



Time t	1	2	3	4	5	6
Obs	2	2	3	2	3	3
$\alpha_t(1)$	0.00000	0.00000	0.06250	0.00000	0.00391	0.00293
$\alpha_t(2)$	0.00000	0.25000	0.00000	0.01562	0.00000	0.00000
$\alpha_t(3)$	1.00000	0.50000	0.00000	0.00000	0.00000	0.00000
$\alpha_t(4)$	0.00000	0.25000	0.00000	0.01562	0.00000	0.00000
$\alpha_t(5)$	0.00000	0.00000	0.06250	0.00000	0.00391	0.00293

Example



Time t	1	2	3	4	5	6
Obs	2	2	3	2	3	3
$P(S_t = 1 \mid o_1, \ldots, o_6)$	0.00000	0.00000	0.50000	0.00000	0.50000	0.50000
$P(S_t = 2 \mid o_1, \ldots, o_6)$	0.00000	0.50000	0.00000	0.50000	0.00000	0.00000
$P(S_t = 3 \mid o_1, \ldots, o_6)$	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
$P(S_t = 4 \mid o_1, \ldots, o_6)$	0.00000	0.50000	0.00000	0.50000	0.00000	0.00000
$P(S_t = 5 \mid o_1, \ldots, o_6)$	0.00000	0.00000	0.50000	0.00000	0.50000	0.50000

HMM Parameter estimation

Suppose we have an observation sequence $o_1, \ldots, o_T$ (or a bunch of such sequences). How can we find HMM parameters $\lambda = (p_{ss'}, p_{so}, p_s)$ that maximize the likelihood of these observations?

HMM Parameter estimation

Suppose we have an observation sequence $o_1, \ldots, o_T$ (or a bunch of such sequences). How can we find HMM parameters $\lambda = (p_{ss'}, p_{so}, p_s)$ that maximize the likelihood of these observations?

- ☐ Use analytical/numerical method to optimize $P(o_1, \ldots, o_T|\lambda)$ directly.
- ☐ Baum-Welch algorithm (an instance of Expectation-Maximization)
- ☐ Cheat! Get trajectories of form $s_1, o_1, s_2, o_2, \ldots, s_T, o_T$ and maximize $P(s_1, o_1, \ldots, s_T, o_T|\lambda)$!

Learning with state information

- HMM models are useful in situations where we do not directly observe the state of a sequential system.
- However, we can often make special arrangements to obtain state information. For example:
  - In speech recognition, we can have human listeners determine exactly what word or phoneme is being spoken at each moment
  - In gene identification, we can have biological experiments or wetlab experiments verify what are actually genes
  - In robot localization, we can collect data in a controlled environment where robot's location is verified by other means
- Thus, at some extra "cost", it is reasonable to assume we can obtain trajectories that include true system state and observations: $s_1, o_1, \ldots, s_T, o_T$.
- It is *much, much, much, much* easier to train HMMs with such data than with observation data alone! (Or at least to prime model parameters.)

Learning with state information (II)

- Suppose we have a set of m trajectories, with the i^{th} trajectory of the form:

$$\tau^i = (s_1^i, o_1^i, s_2^i, o_2^i, \ldots, s_{T^i}^i, o_{T^i}^i)$$

- Maximum likelihood estimates of the HMM parameters are:

Learning with state information (II)

- Suppose we have a set of m trajectories, with the i^{th} trajectory of the form:

$$\tau^i = (s_1^i, o_1^i, s_2^i, o_2^i, \ldots, s_{T^i}^i, o_{T^i}^i)$$

- Maximum likelihood estimates of the HMM parameters are:

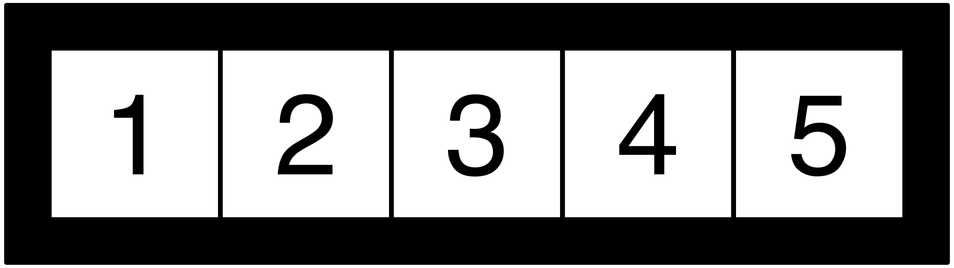
$$p_s = \frac{\#\text{ traj's starting at } s}{m} = \frac{|\{i : s_1^i = s\}|}{m}$$

$$p_{ss'} = \frac{\#\ s\text{-to-}s' \text{ transitions}}{\#\ s\text{-to-anything transitions}} = \frac{|\{(i,t) : s_t^i = s \text{ and } s_{t+1}^i = s'\}|}{|\{(i,t) : s_t^i = s \text{ and } t < T^i\}|}$$

$$p_{so} = \frac{\#\ o \text{ observations in state } s}{\#\text{ times in state } s} = \frac{|\{(i,t) : s_t^i = s \text{ and } o_t^i = o\}|}{|\{(i,t) : s_t^i = s\}|}$$

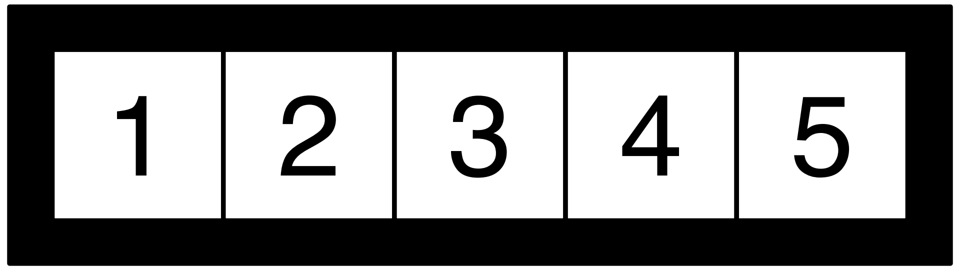
- Question: How would you prove the above are correct?

Example



- Generated length-100 state-observation sequence
- Fit maximum likelihood parameters

Example



	start	to state					see walls				
state		1	2	3	4	5	0	1	2	3	4
1	0.00	0.64	0.36	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
2	0.00	0.18	0.59	0.23	0.00	0.00	0.00	0.00	1.00	0.00	0.00
3	1.00	0.00	0.25	0.35	0.40	0.00	0.00	0.00	1.00	0.00	0.00
4	0.00	0.00	0.00	0.20	0.63	0.17	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.45	0.55	0.00	0.00	0.00	1.00	0.00

Learning without state information

- Suppose we have a single observation trajectory $o_1, o_2, \ldots, o_T$
- The problem is:

$$\begin{aligned}
\max \quad & P(o_1, \ldots, o_T) \\
w.r.t. \quad & p_s, p_{ss'}, p_{so} \\
s.t. \quad & p_s, p_{ss'}, p_{so} \in [0,1] \\
& \sum_s p_s = 1 \\
& \sum_{s'} p_{ss'} = 1 \text{ for all } s \\
& \sum_o p_{so} = 1 \text{ for all } s
\end{aligned}$$

Learning without state information – direct approach

- The most straightforward thing is to write out $P(o_1, \ldots, o_T)$ as:

$$
\begin{aligned}
P(o_1, \ldots, o_T) &= \sum_{s_1, \ldots, s_T} P(o_1, \ldots, o_T | s_1, \ldots, s_T) P(s_1, \ldots, s_T) \\
&= \sum_{s_1, \ldots, s_T} \left(\Pi_{t=1}^{T} p_{s_t o_t}\right) p_{s_1} \left(\Pi_{t=2}^{T} p_{s_{t-1} s_t}\right)
\end{aligned}
$$

- In principle, we can compute the gradient, or do local search, or any other nonlinear optimization approach
- (We have to respect the constraints, though, e.g. through Lagrange multipliers, or feasibility checking.)
- We could also write out $P(o_1, \ldots, o_T)$ in terms of the α's of the forward algorithm:

$$
P(o_1, \ldots, o_T) = \sum_s P(o_1, \ldots, o_T, S_t = s) = \sum_s \alpha_T(s)
$$

Differentiating w.r.t. the parameters yields a backprop-like algorithm.

Learning without state information – Baum-Welch

- The Baum-Welch algorithm is an Expectation-Maximization algorithm algorithm for fitting HMM parameters.
- Recall:
  - We last saw EM in the context of fitting a mixture of Gaussians.
  - In general, EM is useful when we're dealing with "missing data".
- In this case, the missing data is the state sequence.
- If we knew the state sequence, parameter estimation would be easy. (See several slides ago.)
- Instead, we will estimate the state sequence, fit model parameters, reestimate state sequence, refit model parameters, etc.

Baum-Welch overview

- Given observation sequence $o_1, \dots, o_T$
- Given initial parameters $\lambda = (p_s, p_{ss'}, p_{so})$
- Repeat the following until convergence:
  - E-Step:
    - For every s, t compute: $P(S_t = s | o_1, \dots, o_T)$
    - For every s, s', t compute: $P(S_t = s, S_{t+1} = s' | o_1, \dots, o_T)$
  - M-Step:
    - $p_s = P(S_1 = s | o_1, \dots, o_T)$
    - $$p_{ss'} = \frac{\text{Expected } \# \ s \text{ to } s' \text{ transitions}}{\text{Expected } \# \ s \text{ to anything transitions}} = \frac{\sum_{t<T} P(S_t = s, S_{t+1} = s' | o_1, \dots, o_T)}{\sum_{t<T} P(S_t = s | o_1, \dots, o_T)}$$
    - $$p_{so} = \frac{\text{Expected } \# \text{ of } o \text{ observations in } s \text{ state}}{\text{Expected } \# \text{ times in } s \text{ state}} = \frac{\sum_{t: o_t = o} P(S_t = s | o_1, \dots, o_T)}{\sum_t P(S_t = s | o_1, \dots, o_T)}$$

Details of E-Step

- We've already seen how to compute $P(S_t = s|o_1, \ldots, o_T)$... using the forward-backward algorithm.
- How about $P(S_t = s, S_{t+1} = s'|o_1, \ldots, o_T)$?
- It turns out that:

$$P(S_t = s, S_{t+1} = s'|o_1, \ldots, o_T) = \frac{\alpha_t(s) p_{ss'} p_{s' o_{t+1}} \beta_{t+1}(s')}{P(o_1, \ldots, o_T)}$$

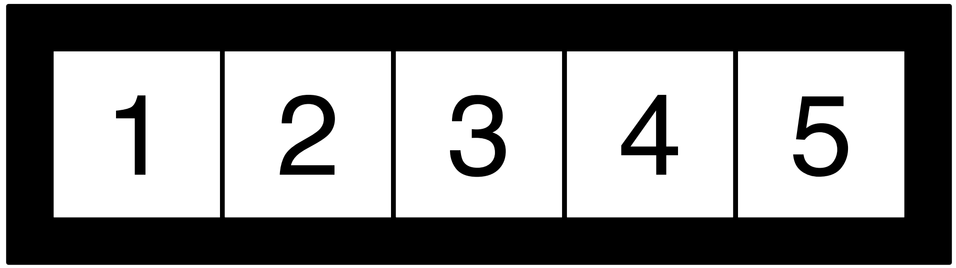
where the α's and β's are from the forward-backward algorithm. (Proof on board.)

$\Rightarrow$ The E-step requires the forward-backward algorithm, plus a bit of extra computation.

Remarks on Baum-Welch

- Each iteration increases $P(o_1, \ldots, o_T)$
- Each iteration computationally efficient
- The algorithm converges to a local maximum
- There can be <u>*tons*</u> of local maxima that are not globally optimal
- Reasonable initial guesses for parameters and/or some amount of state-observation data are a big help
- Baum-Welch optimizes the HMM parameters, p_s, $p_{ss'}$, p_{so}. What if we don't even know the (size of the) state set S?

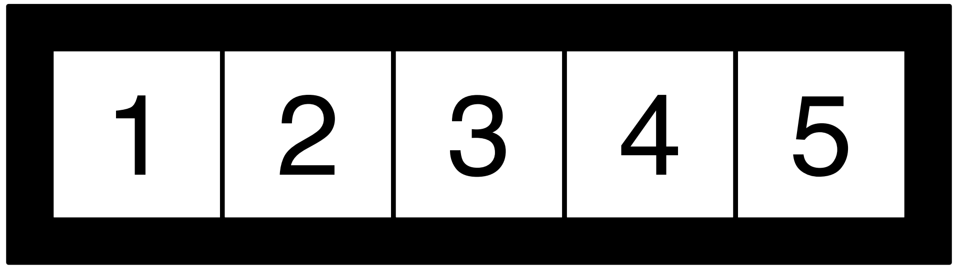
Baum-Welch from correct parameters



state	start	to state 1	to state 2	to state 3	to state 4	to state 5	see walls 0	see walls 1	see walls 2	see walls 3	see walls 4
1	0.00	0.59	0.41	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
2	0.00	0.35	0.01	0.65	0.00	0.00	0.00	0.00	1.00	0.00	0.00
3	1.00	0.00	0.20	0.60	0.20	0.00	0.00	0.00	1.00	0.00	0.00
4	0.00	0.00	0.00	0.65	0.01	0.35	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.41	0.59	0.00	0.00	0.00	1.00	0.00

Likelihood of data: 3.8645e-19

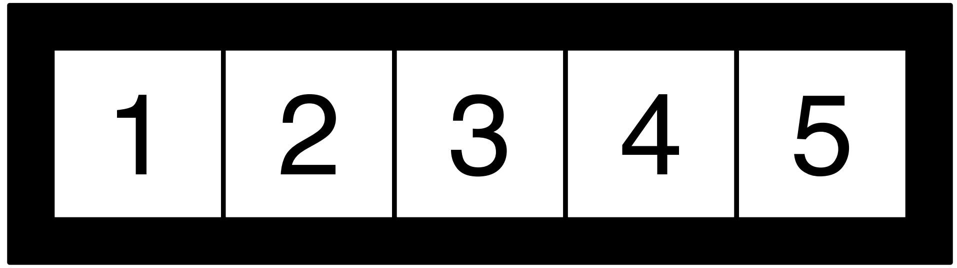
Baum-Welch from uniform parameters



	start	to state					see walls				
state		1	2	3	4	5	0	1	2	3	4
1	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.00	0.77	0.23	0.00
2	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.00	0.77	0.23	0.00
3	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.00	0.77	0.23	0.00
4	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.00	0.77	0.23	0.00
5	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.00	0.77	0.23	0.00

Likelihood of data: 3.7977e-24

Baum-Welch from random parameters



	start	to state					see walls				
state		1	2	3	4	5	0	1	2	3	4
1	0.00	0.07	0.04	0.16	0.00	0.73	0.00	0.00	0.00	1.00	0.00
2	1.00	0.00	0.22	0.31	0.47	0.00	0.00	0.00	1.00	0.00	0.00
3	0.00	0.00	0.79	0.21	0.00	0.00	0.00	0.00	1.00	0.00	0.00
4	0.00	0.48	0.05	0.47	0.00	0.00	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.01	0.59	0.40	0.00	0.00	0.00	1.00	0.00

Likelihood of data: 1.7665e-17

Summary

- Hidden Markov models formalize sequential observation of a system without perfect access to state (i.e., state is "hidden")
- Many applications
- A variety of inference problems can be solved using straightforward dynamic programs
- The parameter fitting problem is best done with "supervised" data – i.e., state & observation trajectories
- Parameter fitting can also be solved purely from observation data by the Baum-Welch algorithm, though results are only locally optimal