6.867 Machine learning and neural networks

Tommi Jaakkola
MIT AI Lab
tommi@ai.mit.edu

Lecture 18: viterbi, linear HMMs

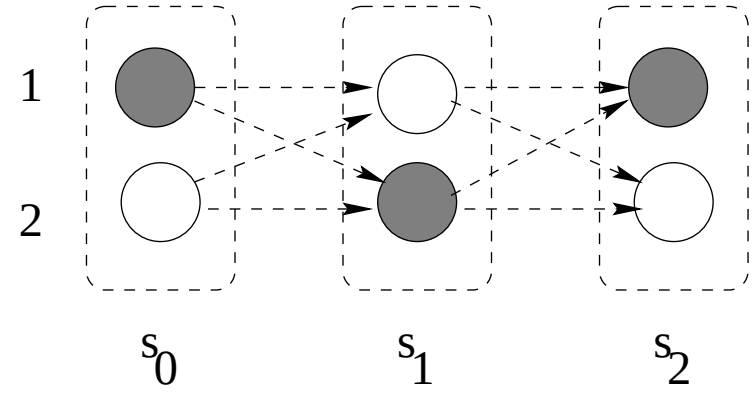
Topics

- Hidden markov models
  - dynamic programming
  - alignment examples
  - linear HMMs

HMM problems

- There are several problems we have to solve
  1. How do we evaluate the probability that our model generated the observation sequence $\{O_0, O_1, \ldots, O_n\}$?
     – forward-backward algorithm
  2. How do we uncover the most likely hidden state sequence corresponding to these observations?
     – dynamic programming
  3. How do we adapt the parameters of the HMM to better account for the observations?
     – the EM-algorithm

Dynamic programming (Viterbi)

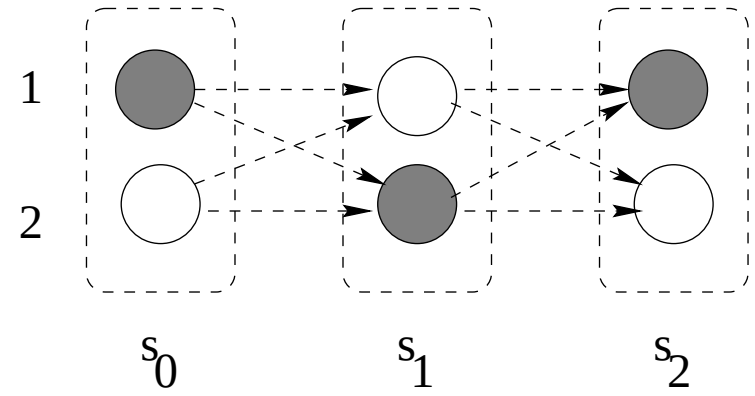


$$O_0 = heads,\ O_1 = tails,\ O_2 = heads$$

- The probability of generating a particular hidden state sequence $s_0 = 1$, $s_1 = 2$, $s_2 = 1$ and the observations is

$$\underset{\substack{\rightarrow\ s_0 \\ \downarrow \\ O_0}}{P_0(1)P_o(heads|1)} \times \underset{\substack{\rightarrow\ s_1 \\ \downarrow \\ O_1}}{P_1(2|1)P_o(tails|2)} \times \underset{\substack{\rightarrow\ s_2 \\ \downarrow \\ O_2}}{P_1(1|2)P_o(heads|1)}$$

Dynamic programming (Viterbi)



$O_0 = heads$, $O_1 = tails$, $O_2 = heads$

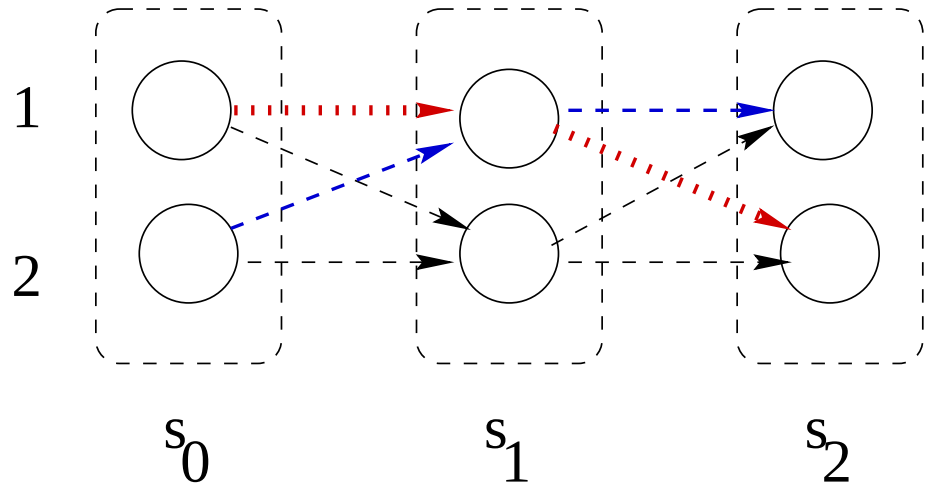
- The probability of the most likely (partial) state sequence and the corresponding observations:

$$\delta_t(i) = \max_{s_0,\ldots,s_{t-1}} \left\{ P(s_0) P_o(O_0|s_0) \cdots P_1(s_t = i|s_{t-1}) \right\} P_o(O_t|s_t = i)$$

- Recursive updates (cf. forward probabilities)

$$\begin{aligned}
\delta_0(j) &= P_0(j) P_o(heads|j), \;\; j = 1, 2 \\
\delta_1(1) &= \max\{\, \delta_0(1) P_1(1|1),\, \delta_0(2) P_1(1|2) \,\} \times P_o(tails|1) \\
\delta_1(2) &= \max\{\, \delta_0(1) P_1(2|1),\, \delta_0(2) P_1(2|2) \,\} \times P_o(tails|2) \\
&\ldots
\end{aligned}$$

Dynamic programming: properties

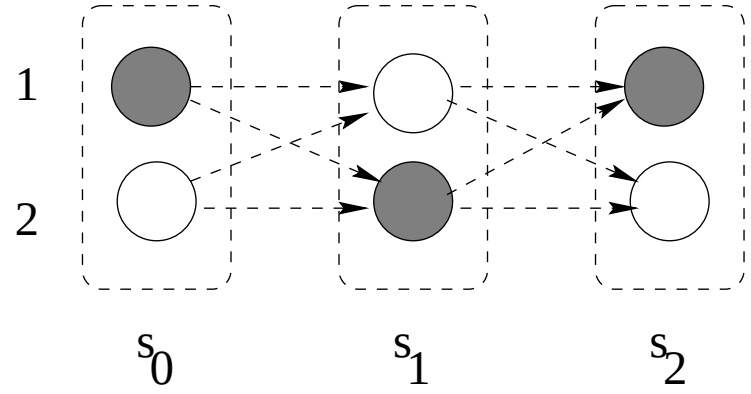


Red path (dotted): most likely path landing on $s_2 = 2$

Blue path (dashed): most likely path landing on $s_2 = 1$

- Possible?

Dynamic programming: backtracking



- The most likely value for state s_2 is the one that corresponds to the most likely path

$$s_2^* = \arg\max\{\, \delta_2(1),\, \delta_2(2)\,\}$$

(say $s_2^* = 1$ as in the figure)

- The most likely previous state is

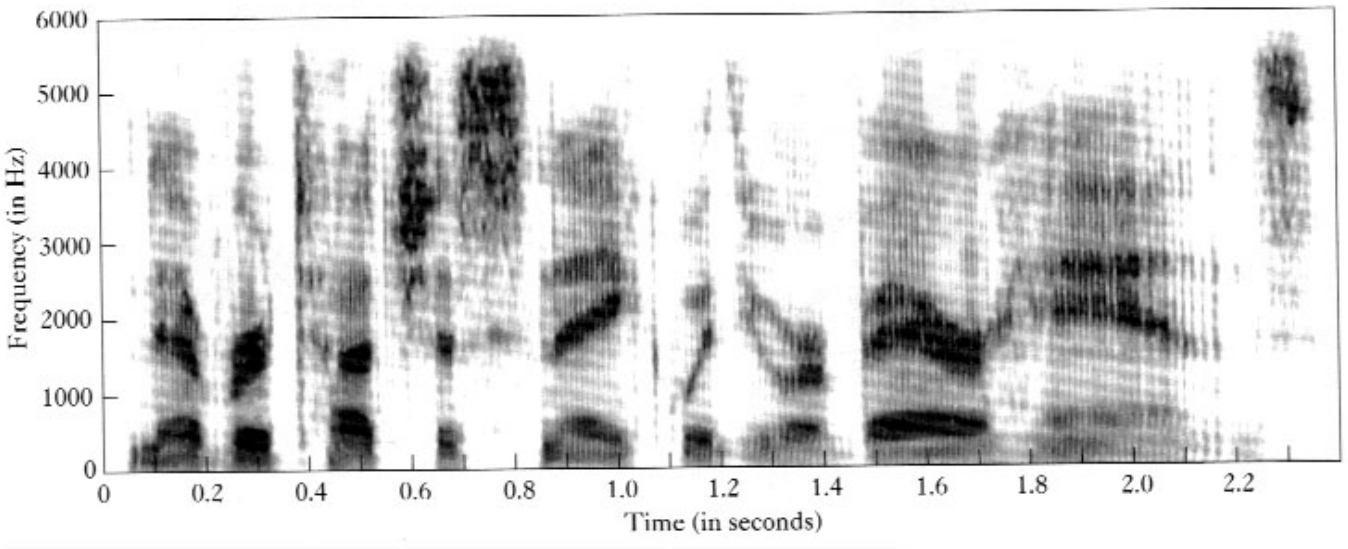
$$s_1^* \;=\; \arg\max\{\, \delta_1(1)P_1(1|1),\, \delta_1(2)P_1(1|2)\,\}$$

and so on...

- Why don't we have to worry about the observations here?

Uses of Dynamic programming

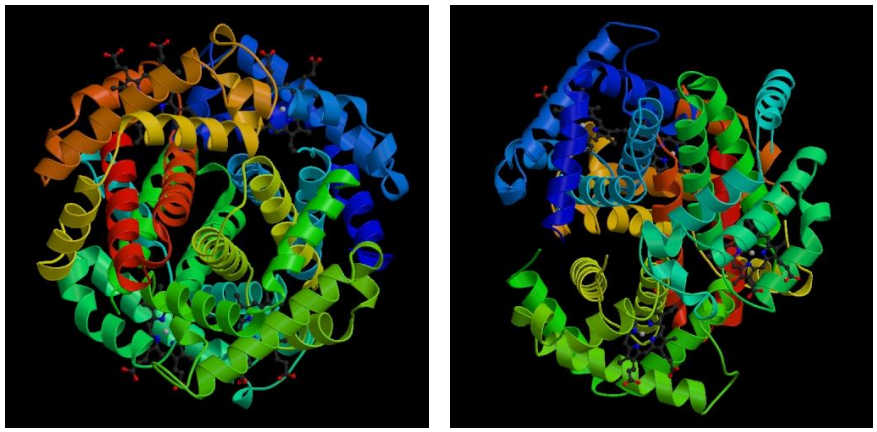
- Annotating or parsing a sequence of speech



Never touch a snake with your bare hands

Uses of Dynamic programming

- Annotating a protein sequence (sequence of amino acids) with markers of "conserved" regions



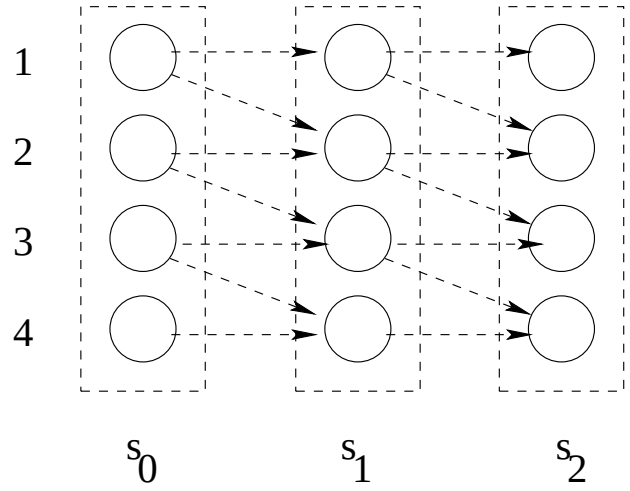
–	m_1	m_2	–	...
G	S	A	A	

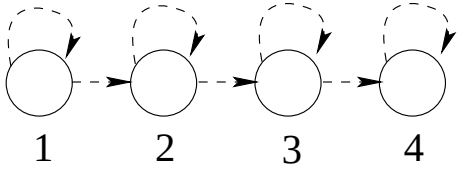
- Multiple alignment of sequences relative to the markers

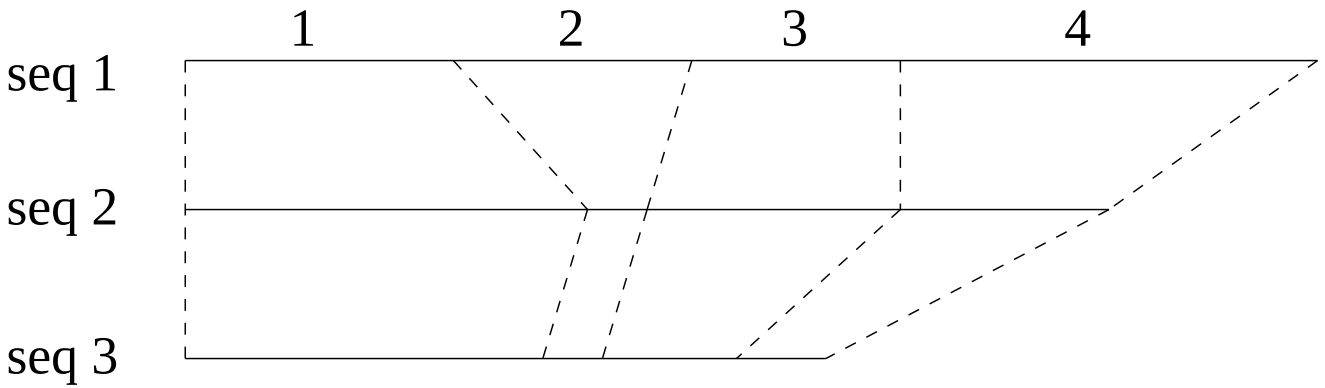
```
-VKGHGKKVADALTNAVAHVDD.....MPNALSALSDLHA...HKLRVDPV.NFKLLSHCLLVTLAAHLP
KVKAHGKKVLGAFSDGLAHLDN.....LKGTFATLSELHC...DKLHVDPE.NFRLLGNVLVCVLAHHFG
DLKKHGVTVLTALGAILKKKGH.....HEAELKPLAQSHA...TK-HKIPIkYLEFISEAIIHVLHSRHP
PFETHANRIVGFFSKIIGELPN.....IEADVNTFVASHK...PR-GVTHD.QLNNFRAGFVSYMKAH--
DVRWHAERIINAVNDAVASMDDtek..MSMKLRDLSGKHA...KSFQVDPQ.YFKVLAAVIADTVAA---
ELQAHAGKVFKLVYEAAIQLQVtgvvvTDATLKNLGSVHV...SK-GVADA.HFPVVKEAILKTIKEVVG
GVAALGAKVLAQIGVAVSHLGDegk..MVAQMKAVGVRHKgygNK-HIKAQ.YFEPLGASLLSAMEHRIG
```

A linear HMM model

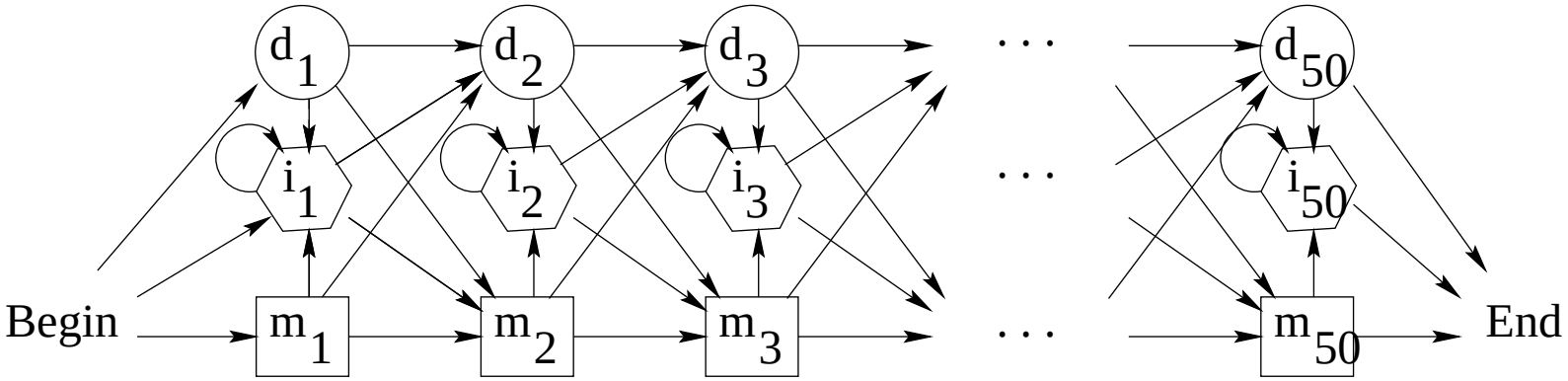
- To align sequences to a model we want the model to be "linear"
  Example: two representations of a "linear" Markov model





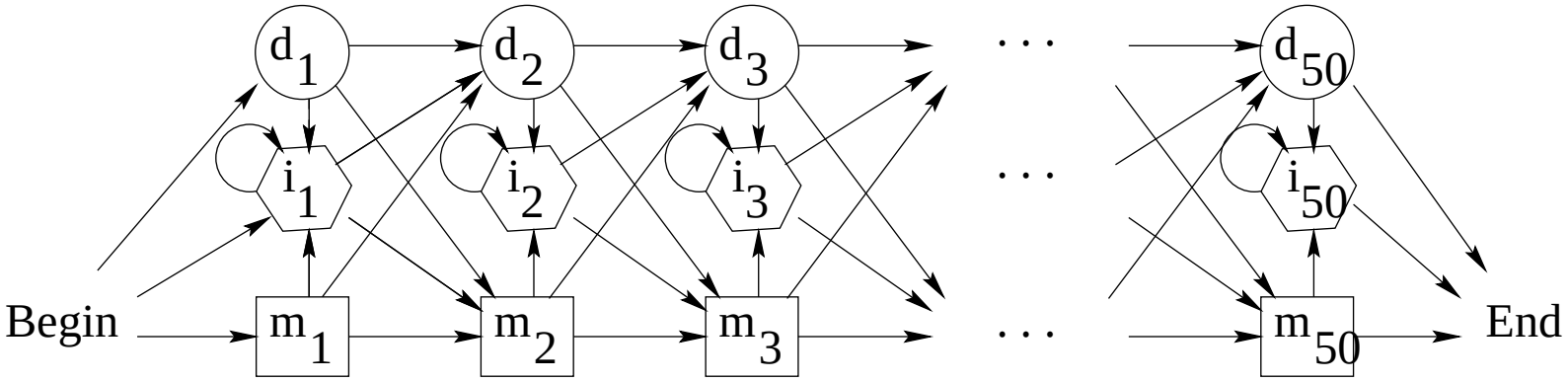


A linear HMM model for protein sequences



- There are three types of states:
  1. *Match states* $m_1, m_2, \ldots$. These try to capture conserved pieces of the sequences
  2. *Insert states* $i_1, i_2, \ldots$. These model inserted amino acid residues between the conserved regions
  3. *Delete states* $d_2, d_3, \ldots$. These permit us to skip a match state
- Only *insert* and *match* states can generate any output (one of 20 possible amino acid letters)

Properties of linear HMMs



- This is a linear architecture in the sense that, for example, you will never come back to a match state m_1 once you have visited it or skipped it.

  $$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$$

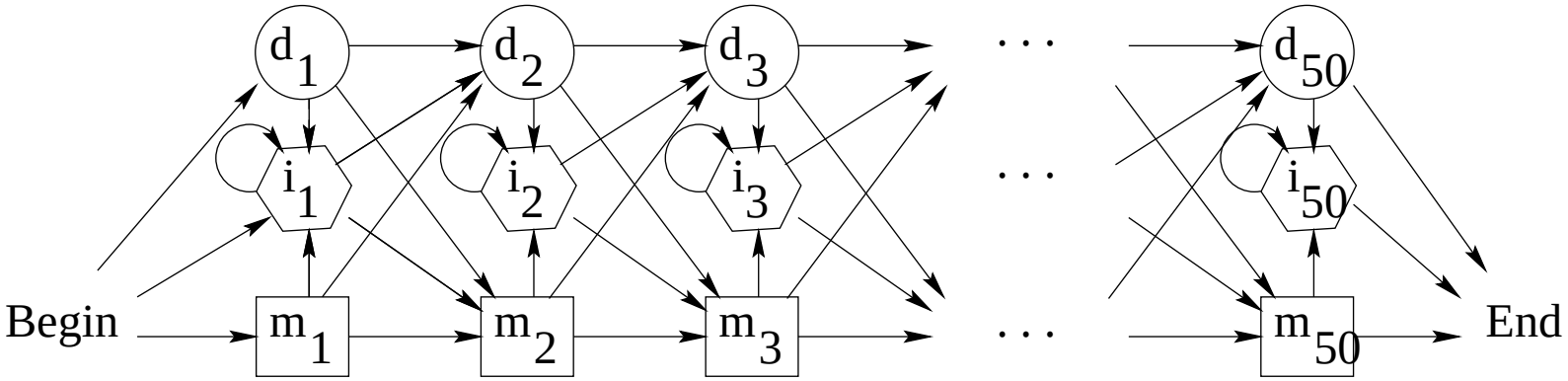
  The state s_0 can be any of $\{i_1, \ldots, i_{50}, d_1, \ldots, d_{50}, m_1, \ldots, m_{50}\}$

  If $s_t = m_{49}$, s_{t+1} can be any of $\{i_{50}, d_{50}, m_{50}\}$

- State variables and observations are no longer in correspondence

  $$\begin{array}{ccccccc} (s_0 = m_1) & \rightarrow & (s_1 = d_2) & \rightarrow & (s_2 = i_2) & \rightarrow \\ \downarrow & & \downarrow & & \downarrow & \\ O_0 & & & & O_1 & \end{array}$$

Properties of linear HMMs



- The linear architecture has a computational advantage: cost is linear in the number of states in the model, not quadratic
- Given any protein (sequence of amino acid letters) we can compute the corresponding most likely hidden state sequence:

G	L	S	A	A	...
i_1	i_2	m_3	m_4	i_5	...
V	K	G	H	G	...
m_4	m_5	i_8	i_9	m_{10}	...

- Multiple sequences can be aligned based on the associated match states

Model based multiple alignment: example

- A single HMM trained for the globin family
- Resulting alignments to the model (match states)

```
-VKGHGKKVADALTNAVAHVDD.....MPNALSALSDLHA...HKLRVDPV.NFKLLSHCLLVTLAAHLP
KVKAHGKKVLGAFSDGLAHLDN.....LKGTFATLSELHC...DKLHVDPE.NFRLLGNVLVCVLAHHFG
DLKKHGVTVLTALGAILKKKGH.....HEAELKPLAQSHA...TK-HKIPIkYLEFISEAIIHVLHSRHP
PFETHANRIVGFFSKIIGELPN.....IEADVNTFVASHK...PR-GVTHD.QLNNFRAGFVSYMKAH--
DVRWHAERIINAVNDAVASMDDtek..MSMKLRDLSGKHA...KSFQVDPQ.YFKVLAAVIADTVAA---
ELQAHAGKVFKLVYEAAIQLQVtgvvvTDATLKNLGSVHV...SK-GVADA.HFPVVKEAILKTIKEVVG
GVAALGAKVLAQIGVAVSHLGDegk..MVAQMKAVGVRHKgygNK-HIKAQ.YFEPLGASLLSAMEHRIG
```

uppercase letters = match states

lowercase letter = insert states

'-' = delete states

'.' = fill character (for pretty alignment only)