6.891 Machine learning and neural networks

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

LECTURE 15 (November 2, 2000)

Problem set #3 due tomorrow (Friday)

Topics

- Hidden markov models
  - forward-backward algorithm (summary from last lecture)
  - estimation via EM
  - dynamic programming
- Biosequence modeling with HMMs
  - proteins, structure etc.
  - pairwise/multiple alignment of sequences
  - linear HMMs for multiple alignment

Hidden Markov models: notation

A hidden Markov model (HMM) is model where we generate a sequence of outputs in addition to the Markov state sequence

$$\begin{array}{ccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow & \ldots \\ \downarrow & & \downarrow & & \downarrow & & \\ O_0 & & O_1 & & O_2 & & \end{array} \tag{1}$$

- To fully specify an HMM, we need to know
  1. the number of states m
  2. the initial state distribution $P_0(s_0)$
  3. the hidden state transition probabilities $P_1(s_{t+1}|s_t)$
  4. the output distribution $P_o(O_t|s_t)$ (discrete or continuous)

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

Propagation algorithms

$$\begin{array}{ccccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow \ldots \rightarrow & s_{n-1} & \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_1 & & O_2 & & O_{n-1} & & O_n \end{array}$$

- Forward probabilities $\alpha_t(i)$:

$$\alpha_t(i) = P(O_0, \ldots, O_t, s_t = i) \tag{2}$$
$$\frac{\alpha_t(i)}{\sum_j \alpha_t(j)} = P(s_t = i | O_0, \ldots, O_t) \tag{3}$$

  (tracking etc; discrete state Kalman filter)

- Backward propabilities $\beta_t(i)$:

$$\beta_t(i) = P(O_{t+1}, \ldots, O_n | s_t = i) \tag{4}$$

  (evidence about the current state from future observations)

- Both can be updated recursively

Recursive updates

$$\begin{array}{ccccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow \ldots \rightarrow & s_{n-1} & \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_1 & & O_2 & & O_{n-1} & & O_n \end{array}$$

Forward recursion:

$$\begin{aligned} \alpha_0(i) &= P_0(s_0 = i)\, P_o(O_0 | s_0 = i) \\ \alpha_t(i) &= \sum_j \alpha_{t-1}(j)\, P_1(s_t = i | s_{t-1} = j) P_o(O_t | s_t = i) \end{aligned}$$

Backward recursion:

$$\begin{aligned} \beta_n(i) &= 1 \\ \beta_{t-1}(i) &= \sum_j P_1(s_t = j | s_{t-1} = i) P_o(O_t | s_t = j)\, \beta_t(j) \end{aligned}$$

- What happens to these if there are no observations?

Properties of the forward/backward probabilities

$$\begin{array}{ccccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow \ldots \rightarrow & s_{n-1} & \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_1 & & O_2 & & O_{n-1} & & O_n \end{array}$$

- The complementary forward/backward probabilities

$$\alpha_t(i) = P(O_0, \ldots, O_t, s_t = i) \tag{5}$$
$$\beta_t(i) = P(O_{t+1}, \ldots, O_n | s_t = i) \tag{6}$$

  permit us to evaluate various posterior probabilities

- The probability of generating the observations and going through state i at time t is

$$P(O_0, \ldots, O_n, s_t = i) = \alpha_t(i)\beta_t(i) \tag{7}$$

  Summing over the possible states at time t gives back

$$P(O_0, \ldots, O_n) = \sum_i \alpha_t(i)\beta_t(i), \text{ for any } t = 0, \ldots, n \tag{8}$$

Forward/backward probabilities cont'd

$$\begin{array}{ccccccc} s_0 & \rightarrow \ldots \rightarrow & s_t & \rightarrow & s_{t+1} & \rightarrow \ldots \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_t & & O_{t+1} & & O_n \end{array}$$

Recall the definitions:

$$\begin{aligned} \alpha_t(i) &= P(O_0, \ldots, O_t, s_t = i) \quad \text{current estimate about } s_t \\ \beta_t(i) &= P(O_{t+1}, \ldots, O_n | s_t = i) \quad \text{future evidence about } s_t \end{aligned}$$

- Using these probabilities we can compute the posterior probability that the HMM was in a particular state i at time t

$$P(s_t = i | O_0, \ldots, O_n) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \stackrel{def}{=} \gamma_t(i) \tag{9}$$

Forward/backward probabilities cont'd

$$\begin{array}{ccccccccc} s_0 & \rightarrow \ldots \rightarrow & s_t & \rightarrow & s_{t+1} & \rightarrow \ldots \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_t & & O_{t+1} & & O_n \end{array}$$

$$\begin{aligned} \alpha_t(i) &= P(O_0,\ldots,O_t,s_t=i) \quad \text{current estimate about } s_t \\ \beta_{t+1}(j) &= P(O_{t+2},\ldots,O_n|s_{t+1}=j) \quad \text{future evidence about } s_{t+1} \end{aligned}$$

- We can also compute the posterior probability that the system was in state i at time t AND transitioned to state j at time $t+1$:

$$\begin{aligned} & P(s_t=i,s_{t+1}=j|O_0,\ldots,O_n) \\ &= \frac{\alpha_t(i)\ \overbrace{P_1(s_{t+1}=j|s_t=i)P_o(O_{t+1}|s_{t+1}=j)}^{\text{fixed } i\to j \text{ transition, one observation}}\ \beta_{t+1}(j)}{\sum_j \alpha_t(j)\beta_t(j)} \\ &\stackrel{def}{=} \xi_t(i,j), \end{aligned} \tag{10}$$

where $t=0,\ldots,n-1$.

The EM algorithm for HMMs

Assume we have L observation sequences $O_0^{(l)},\ldots,O_{n_l}^{(l)}$

E-step: compute the posterior probabilities

$$\gamma_t^{(l)}(i) \quad \text{for all } l,\ i, \text{ and } t\ (t=0,\ldots,n_l) \tag{11}$$

$$\xi_t^{(l)}(i,j) \quad \text{for all } l,\ i, \text{ and } t\ (t=0,\ldots,n_l-1) \tag{12}$$

M-step:

The initial state distribution can be updated according to the expected fraction of times the sequences started from a specific state i

$$\hat{P}_0(i) \leftarrow \frac{1}{L}\sum_{l=1}^{L}\gamma_0^{(l)}(i) \tag{13}$$

M-step cont'd

To update the transtion probabilities, we first define the expected number of transitions from i to j

$$\hat{N}(i,j) = \sum_{l=1}^{L}\sum_{t=0}^{n-1}\xi_t^{(l)}(i,j)$$

and the expecped number of visits to state i

$$\hat{N}(i) = \sum_{l=1}^{L}\sum_{t=0}^{n}\gamma_t^{(l)}(i)$$

- The maximum likelihood estimate of the transition probabilities is the ratio of these two

$$\hat{P}_1(j|i) \leftarrow \frac{\hat{N}(i,j)}{\hat{N}(i)} \tag{14}$$

- What about the output distributions?

M-step cont'd

- If the outputs are discrete, we define the expected number of times a particular observations say $O=k$ was generated from a specific state i

$$\hat{N}_o(i,k) = \sum_{l=1}^{L}\sum_{t=0}^{n_l}\gamma_t^{(l)}(i)\,\delta(O_t^{(l)},k) \tag{15}$$

The ML estimate is the ratio of this to the expected number of visits to i

$$\hat{P}_o(k|i) \leftarrow \frac{\hat{N}_o(i,k)}{\hat{N}(i)} \tag{16}$$

- If the outputs are continuous (e.g., multi-variate Gaussian), we have to solve a weighted maximum likelihood estimation problem

Separately for each i we maximize:

$$\sum_{l=1}^{L}\sum_{t=0}^{n_l}\gamma_t^{(l)}(i)\,\log P(O_t^{(l)}|i) \tag{17}$$

Dynamic programming (Viterbi)

- We wish to find the sequence of hidden states $s_0, \ldots, s_n$ that has the highest probability of generating the sequence of observations $O_0, \ldots, O_n$:

$$s_0^*, \ldots, s_n^* = \arg\max_{s_0,\ldots,s_n} P(O_0, \ldots, O_n, s_0, \ldots, s_n) \tag{18}$$

- Let's define the probability of generating the first t observations along the best path that ends with $s_t = i$

$$\delta_t(i) = \max_{s_0,\ldots,s_{t-1}} P(O_0, \ldots, O_t, s_0, \ldots, s_{t-1}, s_t = i) \tag{19}$$

- We can find these recursively (this is forward recursion with summation replaced with maximum)

$$\begin{aligned} \delta_0(i) &= P_0(s_0 = i) P(O_0 | s_0 = i) \\ \delta_t(i) &= \max_j \{\, \delta_{t-1}(j)\, P_1(s_t = i | s_{t-1} = j)\, P(O_t | s_t = i)\, \} \end{aligned}$$

Dynamic programming: backtracking

$$\begin{array}{ccccccccc} s_0 & \rightarrow \ldots \rightarrow & s_t & \rightarrow & s_{t+1} & \rightarrow \ldots \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_t & & O_{t+1} & & O_n \end{array}$$

$$\delta_t(i) = \max_{s_0,\ldots,s_{t-1}} P(O_0, \ldots, O_t, s_0, \ldots, s_{t-1}, s_t = i)$$

- To recover the most likely hidden state sequence from $\delta_t(i)$ values, we have to instantiate the states in the backwards direction

$$s_n^* = \arg\max_i \delta_n(i) \tag{20}$$
$$s_t^* = \arg\max_i \delta_t(i)\, P_1(s_{t+1} = s_{t+1}^* | s_t = i) \tag{21}$$

- Note: while $\delta_t(i)$ are computed in the forward direction, the states are instantiated in the backwards direction

Example: biosequence analysis

- We construct statistical models for biosequences (proteins, DNA, RNA)
- Proteins are composed of (long) sequences of amino acids (20 possibilities)

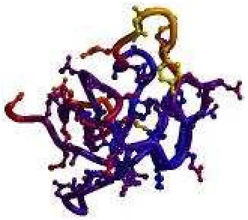
  `VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP`

  where 'V' = valine, 'H' = histidine, etc. This is a member of the globin family of proteins.

- Modeling the statistics of amino acid residues in such sequences permits us to
  1. recognize proteins
  2. uncover *conserved* properties in a particular protein family

Protein structure

- A primary sequence of amino acids "`GLSAAQRQVIAATWKD...`" can fold into a complex 3-D structure



- Proteins exhibit structure at different levels (primary, secondary, tertiary, ...)
- This structure is *very hard* to predict from the primary sequence alone (there are also external factors)
- The properties of the proteins are determined by highly concerved *active sites*, these are sites that, e.g., bind to other molecules such as DNA

Multiple alignment

- Given a set of protein sequences in the same family we'd like to uncover the concerved regions corresponding to active sites or structurally important regions
- We want a *multiple alignment*:

```
--------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP
---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASE
----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQFAG-KDLESIKGTA
PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSA
--------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-FLK-GTSEVPQNNP
---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-FSG----AS---DP
```

- Problems to deal with:
  - variable length
  - additional residues between conserved regions
  - parts of conserved regions missing or altered

Pairwise alignment

- We assume that we have some cost of aligning any two amino acid residues, e.g., $C(A,G)$, including the cost of aligning an amino acid with an empty letter $C(A,\emptyset)$
- The optimal pairwise alignment is computed by finding the minimum cost path from top left to bottom right along the grid points:

	V	H	L	T	P	E	E	K	...	P
V										
L										
S										
E										
G										
E										
...										
E										

- state = grid point, transitions = right, down, diagonal down

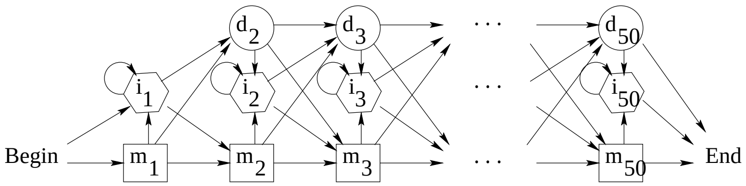
Multiple alignment

- While pairwise alignment can be done efficiently, multiple alignment is exponentially costly in the number of sequences!

```
--------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP
---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASE
----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQFAG-KDLESIKGTA
...
```

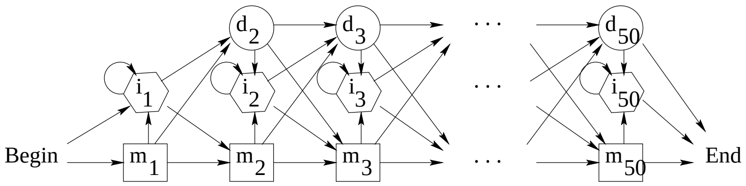
- Possible solution: we align each sequence separately to some (fixed) reference sequence
- Better solution: we align each sequence to a HMM
  - the associated hidden state sequence gives a "parse"

Linear HMM model for protein sequences



- There are three types of states:
  1. *Match states* $m_1, m_2, \ldots$. These try to capture concerved 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 (you have to go through either a match or a delete state)
- 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.

Properties of linear HMMs



$$\begin{array}{ccccccccc} s_0 & \rightarrow \ldots \rightarrow & s_t & \rightarrow & s_{t+1} & \rightarrow \ldots \rightarrow & s_n \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_t & & O_{t+1} & & O_n \end{array}$$

- The linear architecture confers a computational advantage for long sequences
  - cost linear in the number of states, not quadratic
  - necessary to efficiently model proteins with thousands of amino acids
- The sequences are aligned based on the match states

Model based multiple alignment: example

- A single HMM trained on 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
```

where

uppercase letters = match states

lowercase letter = insert states

'-' = delete states

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