6.891 Machine learning and neural networks

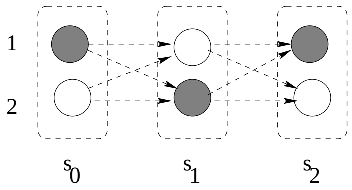
Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

LECTURE 16 (November 7, 2000)

Topics

- Dynamic programming (again)
- Biosequence modeling with HMMs
  - proteins, structure etc.
  - pairwise/multiple alignment of sequences
  - linear HMMs for multiple alignment
- Representations of model structure
  - properties of representations
  - state diagrams vs. graph models
  - Bayesian networks, Markov random fields

Dynamic programming (Viterbi)

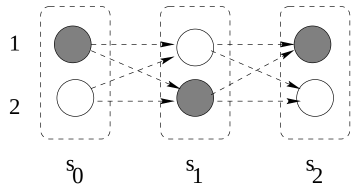


$$O_0 = heads,\ O_1 = tails,\ O_2 = heads$$

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

$$\begin{array}{ccccc} P_0(1)P_o(heads|1) & \times & P_1(2|1)P_o(tails|2) & \times & P_1(1|2)P_o(heads|1) \\ \rightarrow\ s_0 & & \rightarrow\ s_1 & & \rightarrow\ s_2 \\ \downarrow & & \downarrow & & \downarrow \\ O_0 & & O_1 & & O_2 \end{array} \tag{1}$$

Dynamic programming (Viterbi)



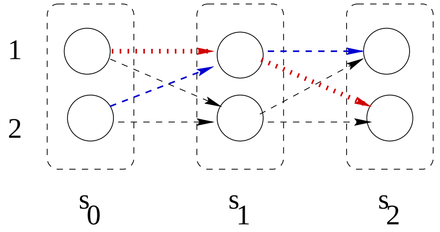
$$O_0 = heads,\ O_1 = tails,\ O_2 = heads$$

- Optimal partial path costs:

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

Dynamic programming: properties

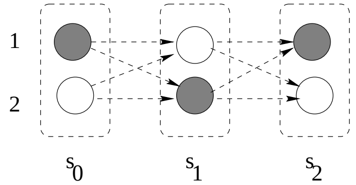
- Why is this *impossible*?



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

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

Dynamic programming: backtracking



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

$$s_2^* = \text{argmax}\,\{\,\delta_2(1),\, \delta_2(2)\,\}$$

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

- The most likely previous state is

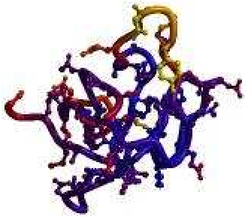
$$s_1^* \;=\; \text{argmax}\,\{\,\delta_1(1)P_1(1|1),\, \delta_1(2)P_1(1|2)\,\}$$

and so on...

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

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 $C(A, \varnothing)$
- 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

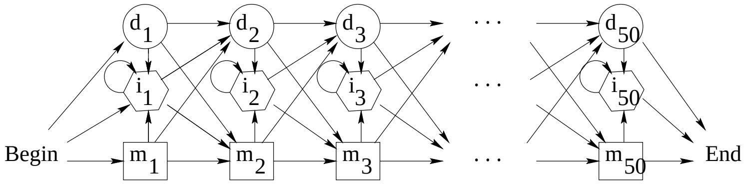
Multiple alignment

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

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

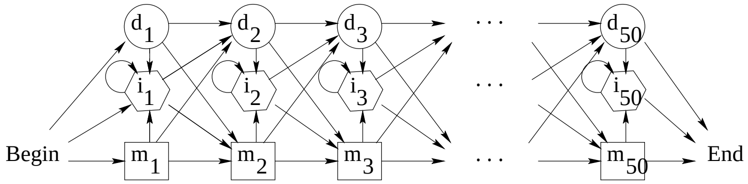
- Possible solution: we align each sequence separately to some (fixed) reference sequence
- Better solution: we align each sequence to a model (HMM)
  - the associated hidden state sequence annotates the sequence

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 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)
- 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.

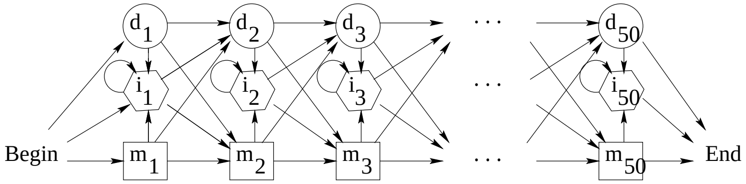
$$\begin{array}{ccccccc} s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow \ldots \rightarrow \\ \downarrow & & \downarrow & & \downarrow & \\ O_0 & & O_1 & & O_2 & \end{array}$$

- How many possible values are there for the state variables $s_0, s_1, \ldots$?

  The state s_0 can be any of $\{i_1, i_2, \ldots, i_{50}, m_1, m_2, \ldots, m_{50}\}$

  The state s_1 can be any of $\{i_2, i_3, \ldots, i_{50}, m_2, m_3, \ldots, m_{50}\}$

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 in character (for pretty alignment only)

Topics

- Representations of model structure
  - properties of representations
  - state diagrams vs. graph models
  - Bayesian networks, Markov random fields

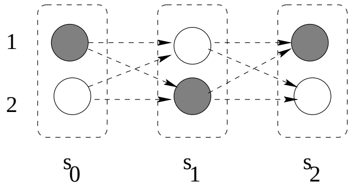
What is a good representation?

- We can represent our *model* in various different ways
- How should we represent it?
- Properties of good representations
  1. Explicit
  2. Compact
  3. Modular
  4. Permits efficient computation
  5. etc.

  Roman numbers, vector representation of words, ...

Representing the model structure

- Two possible representations (of HMMs):
  1. in terms of state diagrams (nodes in the graph correspond to the possible values of the states)



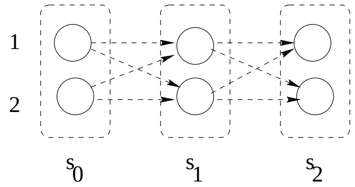
  2. in terms of variables (nodes in the graph are variables):

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

- The representations differ in terms of what aspects of the model are made *explicit*

Model structure cont'd

- Suppose the successive hidden states are *independent of each other*
  1. Representation in terms of a state diagram (the relevant information is hidden)



  2. Representation in terms of variables (the relevant information is explicit)

$$\begin{array}{cccc} s_0 & s_1 & s_2 & \ldots \\ \downarrow & \downarrow & \downarrow & \\ O_0 & O_1 & O_2 & \end{array}$$

Model structure cont'd

- Many time series signals such as music involve multiple relatively independent underlying processes that operate at different *time scales*

  Can we capture the resulting observation sequences with HMMs?
  1. in terms of a state diagram (argh #$& ...)
  2. in terms of variables (graph model)

$$\begin{array}{ccccccccccc} s_0^{(1)} & \rightarrow & & & s_2^{(1)} & \rightarrow & & & s_4^{(1)} & & \\ \downarrow & & & & \downarrow & & & & \downarrow & & \\ s_0^{(2)} & \rightarrow & s_1^{(2)} & \rightarrow & s_2^{(2)} & \rightarrow & s_3^{(2)} & \rightarrow & s_4^{(2)} & \rightarrow & \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ O_0 & & O_1 & & O_2 & & O_3 & & O_4 & & \end{array} \tag{2}$$

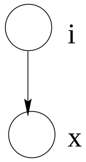
Graphical models

- Graph represention of probability models in terms of *variables* are known as *graphical models*

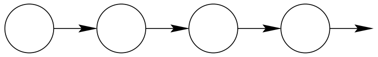
  (a marriage between graph theory and probability theory)
- There are multiple types of graphical models
  1. Bayesian networks (natural for "causal" relations)
  2. Markov random fields (natural for physical or symmetric relations)
  3. etc.
- These graph models differ in terms of how we represent *dependencies* and *independencies* among the variables

Bayesian networks: examples

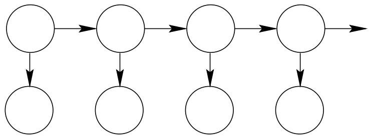
A mixture model: mixture index i, output $\mathbf{x}$



A Markov chain:

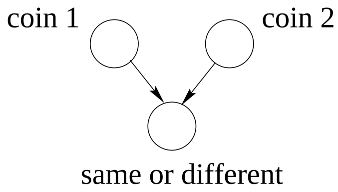


A hidden Markov model:



Bayesian networks: examples

- The graphs try to capture various dependence and independence properties
- *Marginal independence* of the two coins



- *Induced dependence* between the coins