Lecture 16:

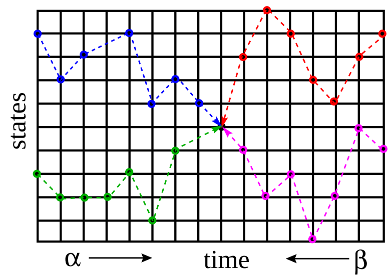
Constrained HMMs

Sam Roweis

March 11, 2002

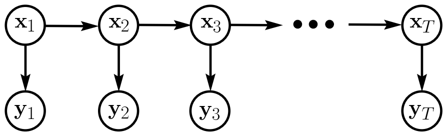
Costs in HMMs

- The number of parameters in the model was $O(K^2 + KM)$ for M output symbols or dimensions.
- Recall the forward-backward algorithm for inference of state probabilities $p(x_t|\{\mathbf{y}\})$.
- The storage cost of this procedure was $O(KT + K^2)$ for K states and a sequence of length T.
- The time complexity was $O(K^2T)$.



HMM Graphical Model

Recall the basic graphical model for HMMs:



- Hidden states $\{x_t\}$, outputs $\{\mathbf{y}_t\}$
  Joint probability factorizes:

$$
\begin{aligned}
\mathsf{P}(\{\mathbf{x}\},\{\mathbf{y}\}) &= \prod_{t=1}^{T} \mathsf{P}(x_t|\mathbf{x}_{t-1})\mathsf{P}(\mathbf{y}_t|x_t) \\
&= \pi_{\mathbf{x}_1} \prod_{t=1}^{\tau-1} T_{x_t,x_{t+1}} \prod_{t=1}^{\tau} A_{x_t}(\mathbf{y}_t)
\end{aligned}
$$

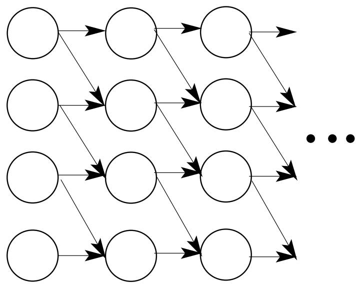
- NB: Data are *not* i.i.d.

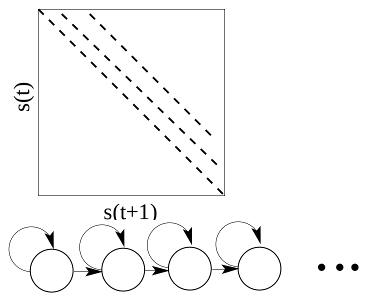
Constraining Large Models

- For large Hidden Markov Models, the quadratic dependence on the number of states can be problematic.
- One way to fix this problem is to *constrain* the possible state transitions to be relatively *sparse*.
- Instead of allowing any state to transition to any other state, each state has only a few possible successor states.
- For example if each state has at most p possible next states then the cost of inference is $O(pKT)$ and the number of parameters is $O(pK + KM)$ which are both *linear* in the number of states.

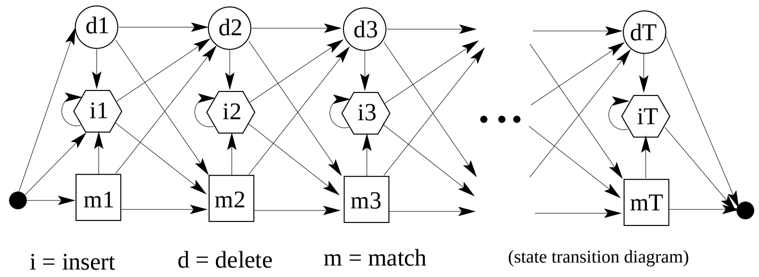
Linear HMMs

- An extremely effective way to constrain the transitions is to *order* the states in the HMM and allow transitions only to *states that come later in the ordering*.
- Such models are known as "linear HMMs", "chain HMMs" or "left-to-right HMMs".
- Transition matrix is upper-diagonal (ususally only has a few bands).





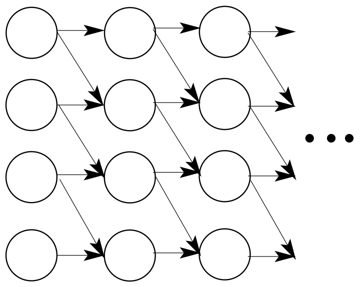
String-Edit HMMs



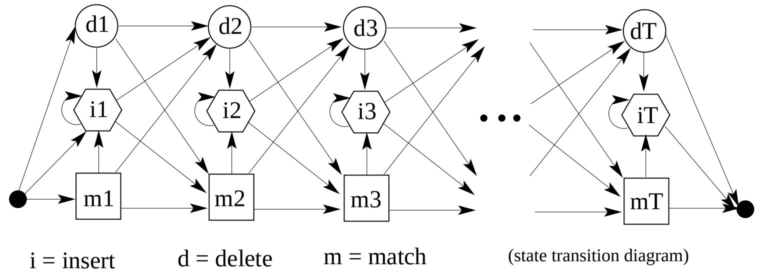
- Another linear architecture is the string-edit HMM.
- Used for probabilistically matching an observed input string to a stored template pattern with possible insertions and deletions.
- Three kinds of states:
  m_j: use position j in the template to match an observed symbol
  i_j: insert extra symbol(s) observations after template position j
  d_j: delete (skip) template position j

Dynamic Programming for Linear HMMs

- What does the DP grid look like for linear HMMs?
- Exactly the same, except that now we only need to check for incoming messages (bugs) from nodes (states) that come earlier in the ordering.
- For example if states can only self-transition or transition to the next state ("skip-1") then each node receives only 2 messages.



String-Edit HMMs

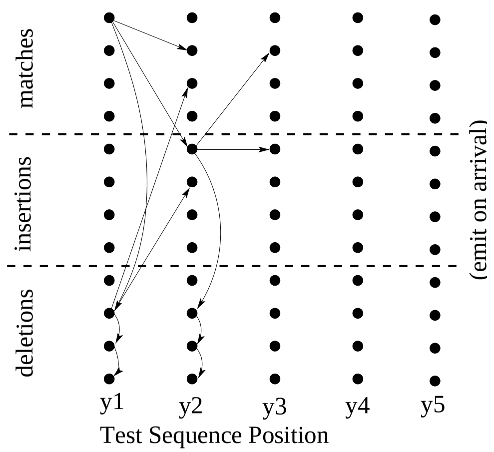


- number of states = 3(length_template)
- Only insert and match states can generate output symbols.
- Why linear? Because once you visit or skip a match state you can never return to it.
- At most 3 destination states from any state.
- State variables and observations no longer in sync.
  (e.g. y1:s1=m1 ; y2:s2=d2 ; y3:s2=i2 ; y4:s2=m2 ; ...)

String-Edit HMMs

- How do we fill in the costs for a DP grid using a string-edit HMM?
- Almost the same as normal except:
  - Now the grid is 3 times its normal height.
  - It is possible to move down without moving right if you move into a deletion state.

eg: template length=4, test sequence length=5



Applications of HMMs

- Speech recognition.
- Language modeling.
- Information retrieval.
- Motion video analysis/tracking.
- Protein sequence and genetic sequence alignment and analysis.
- Financial time series prediction.
- ...

String-Edit HMM Grid Costs

$C_{x_t \to x_{t+1}} = -\log T_{x_t, x_{t+1}} - \log A_{x_{t+1}}(\mathbf{y}_t)$ if x_{t+1} is match or insert

$C_{x_t \to x_{t+1}} = -\log T_{x_t, x_{t+1}}$ if x_{t+1} is a delete state

State $x_t \in \{m_j, i_j, d_j\}$ has nonzero transition probabilities only to states $x_{t+1} \in \{m_{j+1}, i_j, d_{j+1}\}$.