CMPT-413
Computational Linguistics

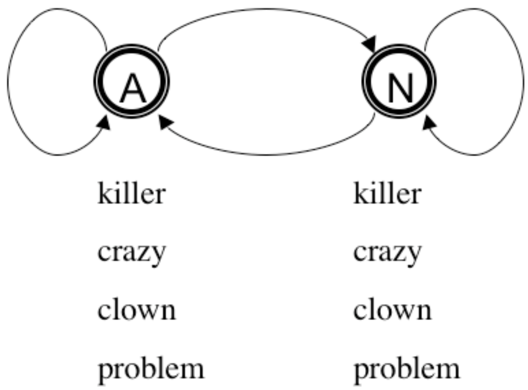
Anoop Sarkar
http://www.cs.sfu.ca/∼anoop

March 19, 2012

Outline

Hidden Markov Model

$$\text{Model } \theta = \begin{cases} \pi_i & \text{probability of starting at state } i \\ a_{i,j} & \text{probability of transition from state } i \text{ to state } j \\ b_i(o) & \text{probability of output } o \text{ at state } i \end{cases}$$

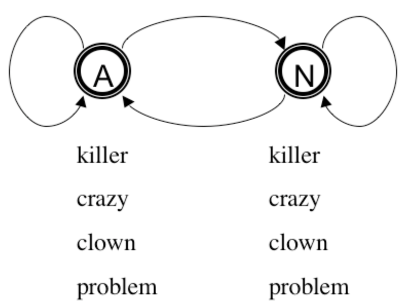


Hidden Markov Model Algorithms

- HMM as parser: compute the best sequence of states for a given observation sequence.
- HMM as language model: compute probability of given observation sequence.
- HMM as learner: given a corpus of observation sequences, learn its distribution, i.e. learn the parameters of the HMM from the corpus.
  - Learning from a set of observations with the sequence of states provided (states are not hidden) [Supervised Learning]
  - Learning from a set of observations without any state information. [Unsupervised Learning]

Outline

HMM as Parser



$\pi =$

A	N
0.25	0.75

$a =$

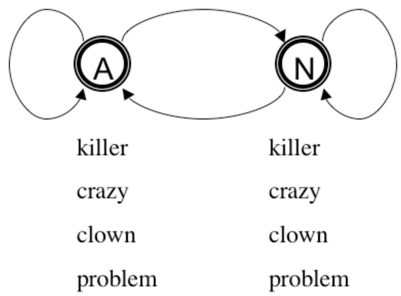
$a_{i,j}$	A	N
N	0.5	0.5
A	0.0	1.0

$b =$

$b_i(o)$	A	N
clown	0.0	0.4
killer	0.0	0.3
problem	0.0	0.3
crazy	1.0	0.0

The task: for a given observation sequence find the most likely state sequence.

HMM as Parser



- Find most likely sequence of states for *killer clown*
- Score every possible sequence of states: AA, AN, NN, NA
  - P(killer clown, AA) $= \pi_A \cdot b_A(killer) \cdot a_{A,A} \cdot b_A(clown) = 0.0$
  - P(killer clown, AN) $= \pi_A \cdot b_A(killer) \cdot a_{A,N} \cdot b_N(clown) = 0.0$
  - P(killer clown, NN) $= \pi_N \cdot b_N(killer) \cdot a_{N,N} \cdot b_N(clown) = 0.75 \cdot 0.3 \cdot 0.5 \cdot 0.4 = 0.045$
  - P(killer clown, NA) $= \pi_N \cdot b_N(killer) \cdot a_{N,A} \cdot b_A(clown) = 0.0$
- Pick the state sequence with highest probability (NN=0.045).

HMM as Parser

- As we have seen, for input of length 2, and a HMM with 2 states there are 2^2 possible state sequences.
- In general, if we have q states and input of length T there are q^T possible state sequences.
- Using our example HMM, for input *killer crazy clown problem* we will have 2^4 possible state sequences to score.
- Our naive algorithm takes exponential time to find the best state sequence for a given input.
- The **Viterbi algorithm** uses dynamic programming to provide the best state sequence with a time complexity of $q^2 \cdot T$

Outline

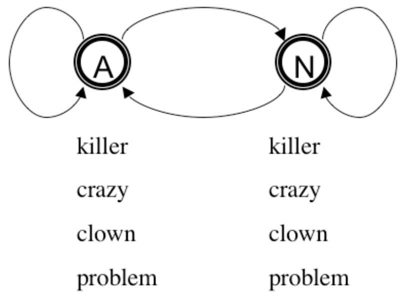
Viterbi Algorithm for HMMs

- For input of length T: $o_1, \ldots, o_T$, we want to find the sequence of states $s_1, \ldots, s_T$
- Each s_t in this sequence is one of the states in the HMM.
- So the task is to find the most likely sequence of states:

$$\operatorname*{argmax}_{s_1,\ldots,s_T} P(o_1, \ldots, o_T, s_1, \ldots, s_T)$$

- The Viterbi algorithm solves this by creating a table $V[s, t]$ where s is one of the states, and t is an index between $1, \ldots, T$.

Viterbi Algorithm for HMMs



- Consider the input *killer crazy clown problem*
- So the task is to find the most likely sequence of states:

$$\underset{s_1,s_2,s_3,s_4}{\operatorname{argmax}} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, s_4)$$

- A sub-problem is to find the most likely sequence of states for *killer crazy clown*:

$$\underset{s_1,s_2,s_3}{\operatorname{argmax}} P(\textit{killer crazy clown}, s_1, s_2, s_3)$$

Viterbi Algorithm for HMMs

- In our example there are two possible values for s_4:

$$\max_{s_1,\ldots,s_4} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, s_4) =$$

$$\max \left\{ \max_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N), \right.$$

$$\left. \max_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, A) \right\}$$

- Similarly:

$$\underset{s_1,\ldots,s_3}{\operatorname{argmax}} P(\textit{killer crazy clown}, s_1, s_2, s_3) =$$

$$\underset{N,V}{\operatorname{argmax}} \left\{ \max_{s_1,s_2} P(\textit{killer crazy clown}, s_1, s_2, N), \right.$$

$$\left. \max_{s_1,s_2} P(\textit{killer crazy clown}, s_1, s_2, A) \right\}$$

Viterbi Algorithm for HMMs

- Putting them together:

$$P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N) = \max\{P(\textit{killer crazy clown}, s_1, s_2, N) \cdot a_{N,N} \cdot b_N(\textit{problem}), P(\textit{killer crazy clown}, s_1, s_2, A) \cdot a_{A,N} \cdot b_N(\textit{problem})\}$$

$$P(\textit{killer crazy clown problem}, s_1, s_2, s_3, A) = \max\{P(\textit{killer crazy clown}, s_1, s_2, N) \cdot a_{N,A} \cdot b_A(\textit{problem}), P(\textit{killer crazy clown}, s_1, s_2, A) \cdot a_{A,A} \cdot b_A(\textit{problem})\}$$

- The best score is given by:

$$\max_{s_1,\ldots,s_4} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, s_4) = \max_{N,A} \left\{ \max_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N), \max_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, A) \right\}$$

Viterbi Algorithm for HMMs

- Provide an index for each input symbol:
  1:killer 2:crazy 3:clown 4:problem

$$V[N,3] = \max_{s_1,s_2} P(\textit{killer crazy clown}, s_1, s_2, N)$$
$$V[N,4] = \max_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N)$$

- Putting them together:

$$V[N,4] = \max\{V[N,3] \cdot a_{N,N} \cdot b_N(\textit{problem}), V[A,3] \cdot a_{A,N} \cdot b_N(\textit{problem})\}$$

$$V[A,4] = \max\{V[N,3] \cdot a_{N,A} \cdot b_A(\textit{problem}), V[A,3] \cdot a_{A,A} \cdot b_A(\textit{problem})\}$$

- The best score for the input is given by: $\max\{V[N,4], V[A,4]\}$
- To extract the best sequence of states we backtrack (same trick as obtaining alignments from minimum edit distance)

Viterbi Algorithm for HMMs

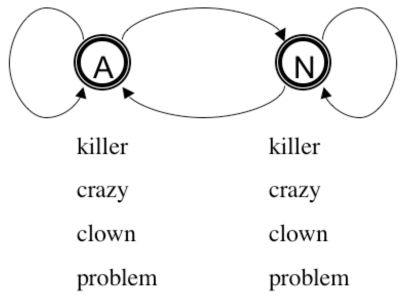
- For input of length T: $o_1, \ldots, o_T$, we want to find the sequence of states $s_1, \ldots, s_T$
- Each s_t in this sequence is one of the states in the HMM.
- For each state q we initialize our table: $V[q, 1] = \pi_q \cdot b_q(o_1)$
- Then compute recursively for $t = 1 \ldots T - 1$ for each state q:

$$V[q, t+1] = \max_{q'} \left\{ V[q', t] \cdot a_{q',q} \cdot b_q(o_{t+1}) \right\}$$

- After the loop terminates, the best score is $\max_q V[q, T]$

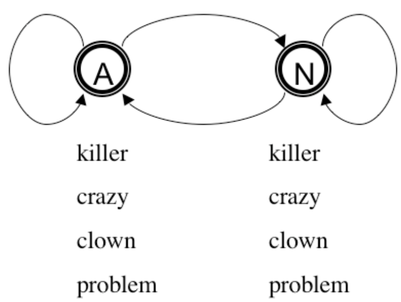
Outline

HMM as Language Model



- Find $P(\textit{killer clown}) = \sum_y P(y, \textit{killer clown})$
- $P(\textit{killer clown}) = P(AA, \textit{killer clown}) + P(AN, \textit{killer clown}) + P(NN, \textit{killer clown}) + P(NA, \textit{killer clown})$

HMM as Language Model



- Consider the input *killer crazy clown problem*
- So the task is to find the sum over all sequences of states:

$$\sum_{s_1, s_2, s_3, s_4} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, s_4)$$

- A sub-problem is to find the most likely sequence of states for *killer crazy clown*:

$$\sum_{s_1, s_2, s_3} P(\textit{killer crazy clown}, s_1, s_2, s_3)$$

HMM as Language Model

- In our example there are two possible values for s_4:

$$\sum_{s_1,\ldots,s_4} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, s_4) =$$
$$\sum_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N) +$$
$$\sum_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, A)$$

- Very similar to the Viterbi algorithm. Sum instead of max, and that's the only difference!

HMM as Language Model

- Provide an index for each input symbol:
  1:killer 2:crazy 3:clown 4:problem

$$\begin{aligned} V[N,3] &= \sum_{s_1,s_2} P(\textit{killer crazy clown}, s_1, s_2, N) \\ V[N,4] &= \sum_{s_1,s_2,s_3} P(\textit{killer crazy clown problem}, s_1, s_2, s_3, N) \end{aligned}$$

- Putting them together:

$$\begin{aligned} V[N,4] &= V[N,3]\cdot a_{N,N}\cdot b_N(\textit{problem}) + \\ &\quad V[A,3]\cdot a_{A,N}\cdot b_N(\textit{problem}) \end{aligned}$$

$$\begin{aligned} V[A,4] &= V[N,3]\cdot a_{N,A}\cdot b_A(\textit{problem}) + \\ &\quad V[A,3]\cdot a_{A,A}\cdot b_A(\textit{problem}) \end{aligned}$$

- The best score for the input is given by: $V[N,4] + V[A,4]$

HMM as Language Model

- For input of length T: $o_1, \ldots, o_T$, we want to find $P(o_1, \ldots, o_T) = \sum_{y_1,\ldots,y_T} P(y_1, \ldots, y_T, o_1, \ldots, o_T)$
- Each y_t in this sequence is one of the states in the HMM.
- For each state q we initialize our table: $V[q, 1] = \pi_q \cdot b_q(o_1)$
- Then compute recursively for $t = 1 \ldots T - 1$ for each state q:

$$V[q, t+1] = \sum_{q'} \left\{ V[q', t] \cdot a_{q',q} \cdot b_q(o_{t+1}) \right\}$$

- After the loop terminates, the best score is $\sum_q V[q, T]$
- So: Viterbi with sum instead of max gives us an algorithm for HMM as a language model.
- This algorithm is sometimes called the *forward algorithm*.