Natural Language Processing

Lecture 9—9/22/2015

Jim Martin

Today

- More on HMMs
  - 3 HMM problems and algorithms
    - Decoding (Viterbi)
    - Forward/Backward
    - EM, (Forward-Backward or Baum-Welch)

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_N;$
- Observations $O = o_1, o_2 \ldots o_N;$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

- Observation likelihoods
  - Output probability matrix $B=\{b_i(k)\}$

$$b_i(k) = P(X_t = o_k \mid q_t = i)$$

- Special initial probability vector π

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

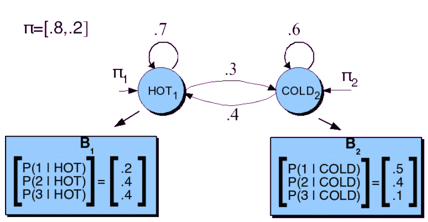
HMMs for Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Baltimore for summer of 2007
- But you find Jason Eisner' s diary which lists how many ice-creams Jason ate every day that summer
- Your job: figure out how hot it was each day

Eisner Task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Hidden Weather Sequence: H,C,H,H,H,C, C…

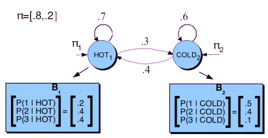
HMM for Ice Cream



Ice Cream HMM

- Let's just do 131 as the sequence
  - How many underlying state (hot/cold) sequences are there?

HHH
HHC
HCH
HCC
CCC
CCH
CHC
CHH



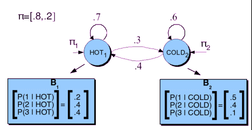
  - How do you pick the right one?

Argmax P(sequence | 1 3 1)

Ice Cream HMM

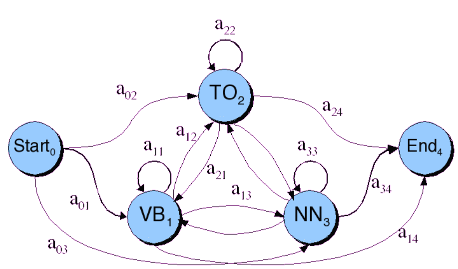
Let's just do 1 sequence: CHC

Cold as the initial state P(Cold|Start)	.2
Observing a 1 on a cold day P(1 | Cold)	.5
Hot as the next state P(Hot | Cold)	.4
Observing a 3 on a hot day P(3 | Hot)	.4
Cold as the next state P(Cold|Hot)	.3
Observing a 1 on a cold day P(1 | Cold)	.5

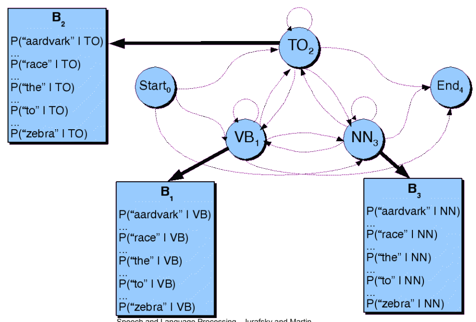


.0024

POS Transition Probabilities



Observation Likelihoods



Question

- If there are 30 or so tags in the Penn set
- And the average sentence is around 20 words...
- How many tag sequences do we have to enumerate to argmax over in the worst case scenario?

30^{20}

3 Problems

- Given this framework there are 3 problems that we can pose to an HMM
  - Given an observation sequence, what is the probability of that sequence given a model?
  - Given an observation sequence and a model, what is the most likely state sequence?
  - Given an observation sequence, find the best model parameters for a partially specified model

Problem 1

- The probability of a sequence given a model...

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

  - Used in model development... How do I know if some change I made to the model is making things better?
  - And in classification tasks
    - Word spotting in ASR, language identification, speaker identification, author identification, etc.
      - Train one HMM model per class
      - Given an observation, pass it to each model and compute P(seq|model).

Problem 2

- Most probable state sequence given a model and an observation sequence

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

  - Typically used in tagging problems, where the tags correspond to hidden states
    - As we'll see almost any problem can be cast as a sequence labeling problem

Problem 3

- Infer the best model parameters, given a partial model and an observation sequence...
  - That is, fill in the A and B tables with the right numbers...
    - The numbers that make the observation sequence most likely
  - Useful for getting an HMM without having to hire annotators...
    - That is, you tell me how many tags there are and give me a boatload of untagged text, and I can give you back a part of speech tagger.

Solutions

- Problem 2: Viterbi
- Problem 1: Forward
- Problem 3: Forward-Backward
  - An instance of EM

Problem 2: Decoding

- Ok, assume we have a complete model that can give us what we need. Recall that we need to get

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- We could just enumerate all paths (as we did with the ice cream example) given the input and use the model to assign probabilities to each.
  - Not a good idea.
  - Luckily dynamic programming helps us here

Intuition

- Consider a state sequence (tag sequence) that ends at some state j (i.e., has a particular tag T at the end)
- The probability of that tag sequence can be broken into parts
  - The probability of the BEST tag sequence up through j-1
  - Multiplied by
    - the transition probability from the tag at the end of the j-1 sequence to T.
    - And the observation probability of the observed word given tag T

Viterbi

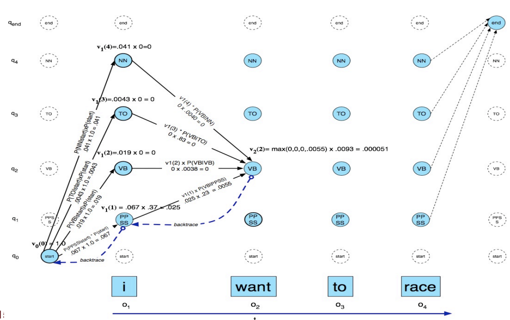
- Create an array
  - Columns corresponding to observations
  - Rows corresponding to possible hidden states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob and path to each cell, (not all paths)

The Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Viterbi Example



7

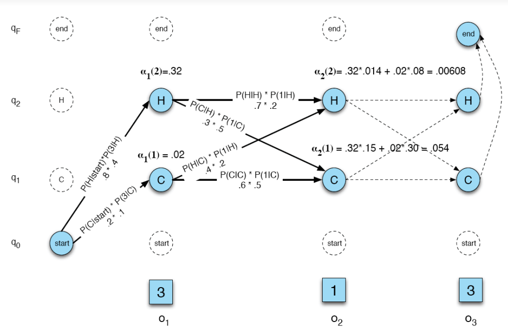
Problem 1: Forward

- Given an observation sequence return the probability of the sequence given the model...
  - Well in a normal Markov model, the states and the sequences are identical... So the probability of a sequence is the probability of the path sequence
  - But not in an HMM... Remember that any number of sequences might be responsible for any given observation sequence.

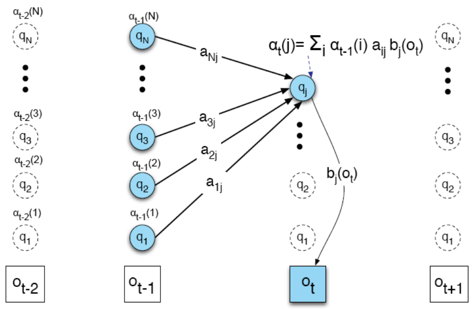
Forward

- Efficiently computes the probability of an observed sequence given a model
  - P(sequence|model)
- Nearly identical to Viterbi; replace the MAX with a SUM

Ice Cream Example



Ice Cream Example



Forward

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forward[q_F,T] \leftarrow \sum_{s=1}^{N} forward[s,T] * a_{s,q_F}$$; termination step

return $forward[q_F,T]$

Problem 3: Learning the Parameters

- First an example to get the intuition down
- We'll do Forward-Backward next time

Urn Example

- A genie has two urns filled with red and blue balls. The genie selects an urn and then draws a ball from it (and replaces it). The genie then selects either the same urn or the other one and then selects another ball…
  - The urns are hidden
  - The balls are observed

Urn

- Based on the results of a long series of draws...
  - Figure out the distribution of colors of balls in each urn
    - Observation probabilities (B table)
  - Figure out the genie's preferences in going from one urn to the next
    - Transition probabilities (A table)

Urns and Balls

- Pi: Urn 1: 0.9; Urn 2: 0.1
- A

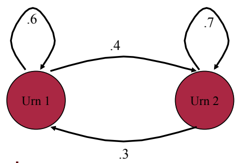
	Urn 1	Urn 2
Urn 1	0.6	0.4
Urn 2	0.3	0.7

- B

	Urn 1	Urn 2
Red	0.7	0.4
Blue	0.3	0.6

Urns and Balls

- Let's assume the input (observables) is Blue Blue Red (BBR)
- Since both urns contain red and blue balls any path of length 3 through this machine could produce this output



Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

Viterbi: Says 111 is the most likely state sequence

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

Forward: P(BBR| model) = .0792 ∑

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- EM
  - What if I told you I lied about the numbers in the model (Priors,A,B). I just made them up.
  - Can I get better numbers just from the input sequence?

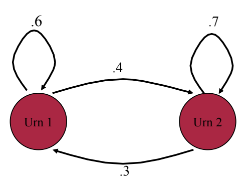
Urns and Balls

- Yup
  - Just count up and prorate the number of times a given transition is traversed while processing the observations inputs.
  - Then use that pro-rated count to re-estimate the transition probability for that transition

Urns and Balls

- But… we just saw that don't know the actual path the input took, its hidden!
  - So prorate the counts from all the possible paths based on the path probabilities the model gives you
    - Basically do what Forward does
- But you said the numbers were wrong
  - Doesn't matter; use the original numbers then replace the old ones with the new ones.

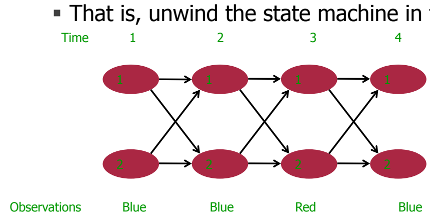
Urn Example



Let's re-estimate the Urn1->Urn2 transition and the Urn1->Urn1 transition (using Blue Blue Red as training data).

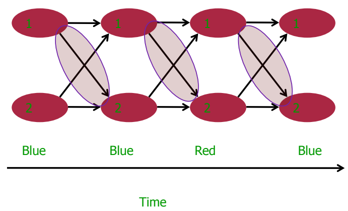
Another View

- We can view all those products as a lattice
  - That is, unwind the state machine in time



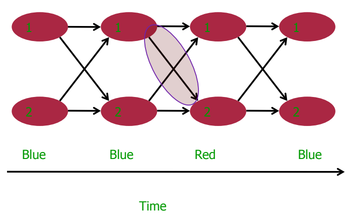
Another View

- Re-estimating the 1-> 2 transitions...



Another View

- Re-estimating the 1-> 2 transitions...



Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

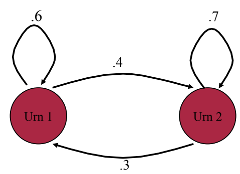
First, what exactly is this probability?

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- So the probability of passing through 1->2 is the weighted sum of the paths taken through that transition given the observations
  - (.0077*1)+(.0136*1)+(.0181*1)+(.0020*1)
  - = .0414
- But, that's not the probability we want, it needs to be divided by the probability of leaving Urn 1 total.
- There's only one other way out of Urn 1 (going back to urn1)
  - So let's reestimate Urn1-> Urn1

Urn Example



Let's re-estimate the Urn1->Urn1 transition

Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- That's
  - (2*.0204)+(1*.0077)+(1*.0052) = .0537
- Again, not what we need but we're closer… we just need to normalize using those two numbers.

Urns and Balls

- The 1->2 transition probability is
  .0414/(.0414+.0537) = 0.435
- The 1->1 transition probability is
  .0537/(.0414+.0537) = 0.565
- So in re-estimation the 1->2 transition went up from .4 to .435 and the 1->1 transition went down from .6 to .565

EM Re-estimation

- Not done yet. No reason to think those values are right. They're just more right than they used to be.
  - So do it again, and again and....
  - Until convergence
  - Convergence does not guarantee a global optima, just a local one
- As with Problems 1 and 2, you wouldn't actually compute it this way. Enumerating all the paths is infeasible.