Today 2/21

• Review HMMs
• EM Example
• Syntax
  • Context-Free Grammars

Review

• Parts of Speech
  • Basic syntactic/morphological categories that words belong to
• Part of Speech tagging
  • Assigning parts of speech to all the words in a sentence
Probabilities

- We want the best set of tags for a sequence of words (a sentence)
- W is a sequence of words
- T is a sequence of tags

So...

- We start with
- And get

HMMs

- This is an HMM
- The states in the model are the tags, and the observations are the words.
- The state to state transitions are driven by the bigram statistics
- The observed words are based solely on the state that you're currently in
Viterbi

- Efficiently return the most likely path
- Sweep through the columns multiplying the probabilities of one row, times the transition probabilities to the next row, times the appropriate observation probabilities
- And store the MAX
Forward

- Efficiently computes the probability of an observed sequence given a model
  - $P(\text{sequence}|\text{model})$
- Nearly identical to Viterbi; replace the MAX with a SUM
  - There is one complication there if you think about the logs that we've been using

EM

- Forward/Backward
  - Efficiently arrive at the right model parameters given a model structure and an observed sequence
  - So for POS tagging
    - Given a tag set
    - And an observed sequence
    - Fill the A, B and PI tables with the right numbers
      - Numbers that give a model that $\text{Argmax} P(\text{model} | \text{data})$

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
Based on the results of a long series of draws...
- Figure out the distribution of colors of balls in each urn
- Figure out the genie’s preferences in going from one urn to the next

**Urns and Balls**

- \( \pi: \text{Urn 1: 0.9; Urn 2: 0.1} \)
- \( A \)

<table>
<thead>
<tr>
<th></th>
<th>Urn 1</th>
<th>Urn 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urn 1</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>Urn 2</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- \( B \)

<table>
<thead>
<tr>
<th></th>
<th>Urn 1</th>
<th>Urn 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>0.7</td>
<td>0.4</td>
</tr>
<tr>
<td>Blue</td>
<td>0.3</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Urns and Balls**

- Let’s assume the input (observables) is Blue Blue Red (BBR)
- Since both urns contain red and blue balls
- Any path through this machine could produce this output
### Urns and Balls

#### Viterbi: Says 111 is the most likely state sequence

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>0.0204</td>
</tr>
<tr>
<td>1 1 2</td>
<td>0.0077</td>
</tr>
<tr>
<td>1 2 1</td>
<td>0.0136</td>
</tr>
<tr>
<td>1 2 2</td>
<td>0.0181</td>
</tr>
<tr>
<td>2 1 1</td>
<td>0.0052</td>
</tr>
<tr>
<td>2 1 2</td>
<td>0.0020</td>
</tr>
<tr>
<td>2 2 1</td>
<td>0.0052</td>
</tr>
<tr>
<td>2 2 2</td>
<td>0.0070</td>
</tr>
</tbody>
</table>

#### Forward: $P(BBR \text{ model}) = 0.792$

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>0.0204</td>
</tr>
<tr>
<td>1 1 2</td>
<td>0.0077</td>
</tr>
<tr>
<td>1 2 1</td>
<td>0.0136</td>
</tr>
<tr>
<td>1 2 2</td>
<td>0.0181</td>
</tr>
<tr>
<td>2 1 1</td>
<td>0.0052</td>
</tr>
<tr>
<td>2 1 2</td>
<td>0.0020</td>
</tr>
<tr>
<td>2 2 1</td>
<td>0.0052</td>
</tr>
<tr>
<td>2 2 2</td>
<td>0.0070</td>
</tr>
</tbody>
</table>
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?

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

• 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
  • But you said the numbers were wrong
    • Doesn’t matter; use the original numbers then replace the old ones with the new ones.
Let's re-estimate the Urn1->Urn2 transition and the Urn1->Urn1 transition (using Blue Blue Red as training data).

### Urns and Balls

<table>
<thead>
<tr>
<th></th>
<th>Blue</th>
<th>Blue</th>
<th>Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>(0.9<em>0.3)</em>(0.6<em>0.3)</em>(0.6*0.7) = 0.0204</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 2</td>
<td>(0.9<em>0.3)</em>(0.6<em>0.3)</em>(0.4*0.4) = 0.0077</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 1</td>
<td>(0.9<em>0.3)</em>(0.4<em>0.6)</em>(0.3*0.7) = 0.0136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 2</td>
<td>(0.9<em>0.3)</em>(0.4<em>0.6)</em>(0.7*0.4) = 0.0181</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1 1</td>
<td>(0.1<em>0.6)</em>(0.3<em>0.7)</em>(0.6*0.7) = 0.0052</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1 2</td>
<td>(0.1<em>0.6)</em>(0.3<em>0.7)</em>(0.4*0.4) = 0.0020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 2 1</td>
<td>(0.1<em>0.6)</em>(0.7<em>0.6)</em>(0.3*0.7) = 0.0052</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 2 2</td>
<td>(0.1<em>0.6)</em>(0.7<em>0.6)</em>(0.7*0.4) = 0.0070</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

That's

- (.0077*1)+(.0136*1)+(.0181*1)+(.0020*1) = .0414

Of course, that's not a probability, 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
Let’s re-estimate the Urn\(1\rightarrow\)Urn\(1\) transition.

<table>
<thead>
<tr>
<th>Urn Example</th>
<th>Urn 1</th>
<th>Urn 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

Let’s re-estimate the Urn\(1\rightarrow\)Urn\(1\) transition.

<table>
<thead>
<tr>
<th>Urns and Balls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Blue Red</td>
</tr>
<tr>
<td>1 1 1</td>
</tr>
<tr>
<td>1 1 2</td>
</tr>
<tr>
<td>1 2 1</td>
</tr>
<tr>
<td>1 2 2</td>
</tr>
<tr>
<td>2 1 1</td>
</tr>
<tr>
<td>2 1 2</td>
</tr>
<tr>
<td>2 2 1</td>
</tr>
<tr>
<td>2 2 2</td>
</tr>
</tbody>
</table>

Blue Blue Red

\((0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204\)
\((0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077\)
\((0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136\)
\((0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181\)
\((0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052\)
\((0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020\)
\((0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052\)
\((0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070\)

\(\text{Blue Blue Red}\)

\(\text{That’s just}\)
\(\text{2*.0204)+(1*.0077)+(1*.0052) = .0537}\)
\(\text{Again not what we need but we’re closer… we just need to normalize using those two numbers.}\)
Urns and Balls

• The 1->2 transition probability is 
  \( \frac{0.0414}{0.0414 + 0.0537} = 0.435 \)
• The 1->1 transition probability is 
  \( \frac{0.0537}{0.0414 + 0.0537} = 0.565 \)
• So in re-estimation the 1->2 transition went from .4 to .435 and the 1->1 transition went from .6 to .565

EM Re-estimation

• As with Problems 1 and 2, you wouldn’t actually compute it this way. The Forward-Backward algorithm re-estimates these numbers in the same dynamic programming way that Viterbi and Forward do.

EM Re-estimation

• With a long enough training string, completely random initial model parameters will converge to the right parameters
• In real systems, you try to get the initial model parameters as close to correct as possible
• Then you use a small amount of training material to home in on the right parameters
Break

• Next HW
  • I’ll give you a training corpus
    • You build a bigram language model for that corpus
    • Use it to assign a log prob to withheld data
    • We’ll use to implement the author identification task
  • To get started
    • Alter your code to count acquire unigram and bigram counts from a corpus.
• Due 3/4

Syntax

• By syntax (or grammar) I mean the kind of implicit knowledge of your native language that you had mastered by the time you were 2 or 3 years old without explicit instruction
• Not the kind of stuff you were later taught in school.

Syntax

• Why should you care?
  • Grammar checkers
  • Question answering
  • Information extraction
  • Machine translation
On Friday, PARC is announcing a deal that underscores that strategy. It is licensing a broad portfolio of patents and technology to a well-financed start-up with an ambitious and potentially lucrative goal: to build a search engine that could some day rival Google. The start-up, Powerset, is licensing PARC’s natural language technology - the art of making computers understand and process languages like English... Powerset hopes the technology will be the basis of a new search engine that allows users to type queries in plain English, rather than using keywords.

For a lot of things, keyword search works well, said Barney Pell, chief executive of Powerset. But I think we are going to look back in 10 years and say, remember when we used to search using keywords.

In a November interview, Marissa Mayer, Google's vice president for search and user experience, said: “Natural language is really hard. I don’t think it will happen in the next five years.”
Context-Free Grammars

• Capture constituency and ordering
  • Ordering is easy
    What are the rules that govern the ordering of words and bigger units in the language
  • What’s constituency?
    How words group into units and how the various kinds of units behave wrt one another

CFG Examples

• S -> NP VP
• NP -> Det NOMINAL
• NOMINAL -> Noun
• VP -> Verb
• Det -> a
• Noun -> flight
• Verb -> left

CFGs

• S -> NP VP
  • This says that there are units called S, NP, and VP in this language
  • That an S consists of an NP followed immediately by a VP
  • Doesn’t say that that’s the only kind of S
  • Nor does it say that this is the only place that NPs and VPs occur
Generativity

- As with FSAs and FSTs you can view these rules as either analysis or synthesis machines
  - Generate strings in the language
  - Reject strings not in the language
  - Impose structures (trees) on strings in the language

Derivations

- A derivation is a sequence of rules applied to a string that accounts for that string
  - Covers all the elements in the string
  - Covers only the elements in the string

Derivations as Trees
Parsing

• Parsing is the process of taking a string and a grammar and returning a (many?) parse tree(s) for that string
• It is completely analogous to running a finite-state transducer with a tape
  • It’s just more powerful
    • Remember this means that there are languages we can capture with CFGs that we can’t capture with finite-state methods

Other Options

• Regular languages (expressions)
  • Too weak
• Context-sensitive or Turing equiv
  • Too powerful (maybe)

Context?

• The notion of context in CFGs has nothing to do with the ordinary meaning of the word context in language.
• All it really means is that the non-terminal on the left-hand side of a rule is out there all by itself (free of context)
  A → B C
  Means that
    • I can rewrite an A as a B followed by a C regardless of the context in which A is found
    • Or when I see a B followed by a C I can infer an A regardless of the surrounding context
Key Constituents (English)

- Sentences
- Noun phrases
- Verb phrases
- Prepositional phrases

Sentence-Types

- Declaratives: A plane left
  \[ S \rightarrow NP \ VP \]
- Imperatives: Leave!
  \[ S \rightarrow VP \]
- Yes-No Questions: Did the plane leave?
  \[ S \rightarrow Aux \ NP \ VP \]
- WH Questions: When did the plane leave?
  \[ S \rightarrow WH \ Aux \ NP \ VP \]

Recursion

- We’ll have to deal with rules such as the following where the non-terminal on the left also appears somewhere on the right (directly).
  Nominal \rightarrow Nominal PP  [[flight] [to Boston]]
  VP \rightarrow VP PP  [[departed Miami] [at noon]]
Recursion

- Of course, this is what makes syntax interesting
  -Flights from Denver
  -Flights from Denver to Miami
  -Flights from Denver to Miami in February
  -Flights from Denver to Miami in February on a Friday
  -Flights from Denver to Miami in February on a Friday under $300
  -Flights from Denver to Miami in February on a Friday under $300 with lunch

Recursion

- Of course, this is what makes syntax interesting
  -[[flights] [from Denver]]
  -[[[Flights] [from Denver]] [to Miami]]
  -[[[[Flights] [from Denver]] [to Miami]] [in February]]
  -[[[[[Flights] [from Denver]] [to Miami]] [in February]] [on a Friday]]
  -Etc.

The Point

- If you have a rule like
  -VP -> V NP
  -It only cares that the thing after the verb is an NP. It doesn’t have to know about the internal affairs of that NP
The Point

Conjunctive Constructions

- S -> S and S
  - John went to NY and Mary followed him
- NP -> NP and NP
- VP -> VP and VP
- ...
  - In fact the right rule for English is
    X -> X and X

Problems

- Agreement
- Subcategorization
- Movement (for want of a better term)
Agreement

- This dog
- Those dogs
- This dog eats
- Those dogs eat
- *This dogs
- *Those dog
- *This dog eat
- *Those dogs eats

Subcategorization

- Sneeze: John sneezed
- Find: Please find [a flight to NY]\[NP
- Give: Give [me]\[NP[a cheaper fare]\[NP
- Help: Can you help [me]\[NP[with a flight]\[PP
- Prefer: I prefer [to leave earlier]\[TO-VP
- Told: I was told [United has a flight]\[S
- ...

- *John sneezed the book
- *I prefer United has a flight
- *Give with a flight

Subcat expresses the constraints that a predicate (verb for now) places on the number and syntactic types of arguments it wants to take (occur with).
So?

• So the various rules for VPs overgenerate.
  • They permit the presence of strings containing verbs and arguments that don’t go together
  • For example
  • VP -> V NP therefore
  Sneezed the book is a VP since “sneeze” is a verb and “the book” is a valid NP

Next Time

• We’re now into Chapters 12 and 13.
• Finish reading all of 12.
• Get through the CKY discussion in 13