Today 2/19

- Review HMMs for POS tagging
- Entropy intuition
- Statistical Sequence classifiers
  - HMMs
  - MaxEnt
  - MEMMs

Statistical Sequence Classification

- Given an input sequence, assign a label (or tag) to each element of the tape
  - Or... Given an input tape, write a tag out to an output tape for each cell on the input tape
- Can be viewed as a classification task if we view
  - The individual cells on the input tape as things to be classified
  - The tags written on the output tape as the class labels
POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words \(w_1 \ldots w_n\).

Statistical Sequence Classification

- We want, out of all sequences of n tags \(t_1 \ldots t_n\), the single tag sequence such that \(P(t_1 \ldots t_n | w_1 \ldots w_n)\) is highest.

\[
\hat{t}_1^n = \arg \max_{t_i^n} P(t_i^n | w_1^n)
\]

- Hat more means "our estimate of the best one"
- Argmax, \(f(x)\) means "the \(x\) such that \(f(x)\) is maximized"

Road to HMMs

- This equation is guaranteed to give us the best tag sequence
  
  \[
  \hat{t}_1^n = \arg \max_{t_i^n} P(t_i^n | w_1^n)
  \]
- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform into a set of other probabilities that are easier to compute
Using Bayes Rule

\[ P(x|y) = \frac{P(y|x)P(x)}{P(y)} \]

\[ \hat{t}_1^n = \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} \]

\[ \hat{t}_1^n = \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n) \]

Likelihood and Prior

\[ \hat{t}_1^n = \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(t_1^n)} \approx \frac{\prod_{i=1}^{n} P(w_i|t_i)}{\prod_{i=1}^{n} P(t_i|t_{i-1})} \]

\[ P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \]

\[ \hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) \]

Transition Probabilities

- Tag transition probabilities \(p(t_1|t_{i-1})\)
- Determiners likely to precede adjs and nouns
  - That/DT flight/NN
  - The/DT yellow/JJ hat/NN
- So we expect \(P(\text{NN|DT})\) and \(P(\text{JJ|DT})\) to be high
- Compute \(P(\text{NN|DT})\) by counting in a labeled corpus:

\[ P(t_1|t_{i-1}) = \frac{C(t_1|t_{i-1})}{C(t_{i-1})} \]

\[ P(\text{NN|DT}) = \frac{C(\text{DT, NN})}{C(\text{DT})} = \frac{56,509}{116,454} = .49 \]
Observation Probabilities

- Word likelihood probabilities $p(w_i|t_i)$
- VBZ (3sg Pres verb) likely to be “is”
- Compute $P(is|VBZ)$ by counting in a labeled corpus:
  \[ P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} \]
  \[ P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47 \]

An Example: the verb “race”

- Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating “race”
Example

- \( P(\text{NN|TO}) = .00047 \)
- \( P(\text{VB|TO}) = .83 \)
- \( P(\text{race|NN}) = .00057 \)
- \( P(\text{race|VB}) = .00012 \)
- \( P(\text{NR|VB}) = .0027 \)
- \( P(\text{NR|NN}) = .0012 \)
- \( P(\text{VB|TO})P(\text{NR|VB})P(\text{race|VB}) = .00000027 \)
- \( P(\text{NN|TO})P(\text{NR|NN})P(\text{race|NN}) = .000000032 \)
- So we (correctly) choose the verb reading.

Markov chain for words

Markov chain = “First-order Observable Markov Model”

- A set of states
  - \( Q = q_1, q_2, ..., q_N \); the state at time \( t \) is \( q_t \)
- Transition probabilities:
  - A set of probabilities \( A = a_{01} \ldots a_{0n} \ldots a_{nn} \)
  - Each \( a_{ij} \) represents the probability of transitioning from state \( i \) to state \( j \)
  - The set of these is the transition probability matrix \( A \)
- Current state only depends on previous state
  - \( \pi_{t+1} | \pi_t, \ldots, \pi_1 \) - \( \pi_{t+1} | \pi_t \)
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 \( A = \{a_{ij}\} \)
- Observation likelihoods
  - Output probability matrix \( B = \{b_{ik}\} \)
- Special initial probability vector \( \pi \)

Transitions between the hidden states of HMM, showing A probs

B observation likelihoods for POS HMM
The A matrix for the POS HMM

<table>
<thead>
<tr>
<th></th>
<th>VB</th>
<th>TO</th>
<th>NN</th>
<th>PPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;S&gt;</td>
<td>0.019</td>
<td>0.0043</td>
<td>0.041</td>
<td>0.0067</td>
</tr>
<tr>
<td>VB</td>
<td>0.0038</td>
<td>0.0835</td>
<td>0.047</td>
<td>0.0070</td>
</tr>
<tr>
<td>TO</td>
<td>0.0040</td>
<td>0.016</td>
<td>0.087</td>
<td>0.0045</td>
</tr>
<tr>
<td>PPSS</td>
<td>0.0023</td>
<td>0.00079</td>
<td>0.0012</td>
<td>0.00014</td>
</tr>
</tbody>
</table>

Figure 4.15 Tag transition probabilities (the a array, p(t|t-1)) computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event, thus P(OS/VB) is 0.0070. The symbol <S> is the start-of-sentence symbol.

The B matrix for the POS HMM

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>race</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>0</td>
<td>0.0095</td>
<td>0</td>
<td>0.0012</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>0</td>
<td>0.59</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
<td>0.00054</td>
<td>0</td>
<td>0.00057</td>
</tr>
<tr>
<td>PPSS</td>
<td>0.37</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.16 Observation likelihoods (the b array) computed from the 87-tag Brown corpus without smoothing.

Viterbi intuition: we are looking for the best ‘path’
The Viterbi Algorithm

Function Viterbi(observations, state-graph, state-hmm) returns best path

- create a path probability matrix viterbi[N][T]
- for each state i from 1 to N do
  - viterbi[i][1] = a_{i} \cdot \pi_{i}
- backpointer[i][1] = 0
- for each time step t from 2 to T do
  - for each state i from 1 to N do
    - viterbi[i][t] = \max_{j \in S} viterbi[j][t-1] \cdot a_{ij} \cdot b_{i}(o_{t})
    - backpointer[i][t] = \arg\max_{j \in S} viterbi[j][t-1] \cdot a_{ij}
- viterbi[best][T] = \max viterbi[i][T] \cdot a_{i, \text{start}}
- for each time step t from T-1 to 1 do
    - viterbi[best][t] = \arg\max_{i \in S} viterbi[i][t+1] \cdot \pi_{i}
- return the best path by following backpointers to states back in time from backpointer[best][1]

Viterbi example

Information Theory

- Who is going to win the World Series next year?
- Well there are 30 teams. Each has a chance, so there’s a 1/30 chance for any team…? No.
  - Rockies? Big surprise, lots of information
  - Yankees? No surprise, not much information
Information Theory

• How much uncertainty is there when you don’t know the outcome of some event (answer to some question)?
• How much information is to be gained by knowing the outcome of some event (answer to some question)?

Aside on logs

• Base doesn’t matter. Unless I say otherwise, I mean base 2.
• Probabilities lie between 0 an 1. So log probabilities are negative and range from 0 (log 1) to –infinity (log 0).
• The – is a pain so at some point we’ll make it go away by multiplying by -1.

Entropy

• Let’s start with a simple case, the probability of word sequences with a unigram model
• Example
  • $S = \text{“One fish two fish red fish blue fish”}$
  • $P(S) = P(\text{One})P(\text{fish})P(\text{two})P(\text{fish})P(\text{red})P(\text{fish})P(\text{blue})P(\text{fish})$
  • $\log P(S) = \log P(\text{One}) + \log P(\text{fish}) + \ldots + \log P(\text{fish})$
In general that’s...

But note that:
- the order doesn’t matter
- that words can occur multiple times
- and that they always contribute the same each time
- so rearranging...

• One fish two fish red fish blue fish
• Fish fish fish fish one two red blue

Now let’s divide both sides by N, the length of the sequence:

That’s basically an average of the log probs
Entropy

Now assume the sequence is really really long.
Moving the N into the summation you get

Rewriting and getting rid of the minus sign

Entropy

Think about this in terms of uncertainty or surprise.
The more likely a sequence is, the lower the entropy. Why?

Model Evaluation

Remember the name of the game is to come up with statistical models that capture something useful in some body of text or speech.
There are precisely a gazzilion ways to do this
  • N-grams of various sizes
  • Smoothing
  • Backoff…
Model Evaluation

- Given a collection of text and a couple of models, how can we tell which model is best?
- Intuition… the model that assigns the highest probability to a set of withheld text
  - Withheld text? Text drawn from the same distribution (corpus), but not used in the creation of the model being evaluated.

Model Evaluation

- The more you’re surprised at some event that actually happens, the worse your model was.
- We want models that minimize your surprise at observed outcomes.
- Given two models and some training data and some withheld test data… which is better?

Three HMM Problems

- Given a model and an observation sequence
  - Compute Argmax P(states | observation seq)
    - Viterbi
  - Compute P(observation seq | model)
    - Forward
  - Compute P(model | observation seq)
    - EM (magic)
Viterbi

- Given a model and an observation sequence, what is the most likely state sequence?
  - The state sequence is the set of labels assigned
  - So using Viterbi with an HMM solves the sequence classification task

Forward

- Given an HMM model and an observed sequence, what is the probability of that sequence?
  - \( P(\text{sequence} \mid \text{Model}) \)
  - Sum of all the paths in the model that could have produced that sequence
  - So...
    - How do we change Viterbi to get Forward?

Who cares?

- Suppose I have two different HMM models extracted from some training data.
- And suppose I have a good-sized set of held-out data (not used to produce the above models).
- How can I tell which model is the better model?
Learning Models

• Now assume that you just have a single HMM model (pi, A, and B tables)
• How can I produce a second model from that model?
  • Rejigger the numbers... (in such a way that the tables still function correctly)
  • Now how can I tell if I’ve made things better?

EM

• Given an HMM structure and a sequence, we can learn the best parameters for the model without explicit training data.
  • In the case of POS tagging all you need is unlabelled text.
  • Huh? Magic. We’ll come back to this.

Generative vs. Discriminative Models

• For POS tagging we start with the question... P(tags | words) but we end up via Bayes at
  • P(words|tags)P(tags)
  • That’s called a generative model
  • We’re reasoning backwards from the models that could have produced such an output
Disambiguating “race”

(a) NNP VBP VBN TO VB NR
Secretariat is expected to race tomorrow

(b) NNP VBP VBN TO NN NR
Secretariat is expected to race tomorrow

Discriminative Models

- What if we went back to the start to Argmax \( P(\text{tags}|\text{words}) \) and didn’t use Bayes?
- Can we get a handle on this directly?
- First let’s generalize to \( P(\text{tags}|\text{evidence}) \)
  - Let’s make some independence assumptions and consider the previous state and the current word as the evidence. How does that look as a graphical model?

MaxEnt Tagging

- Secretariat is expected to race tomorrow
MaxEnt Tagging

• This framework allows us to throw in a wide range of “features”. That is, evidence that can help with the tagging.

Statistical Sequence Classification

Corresponding feature representation