Some slides adapted from Jason Eisner

Admin
- Readings
  - make sure you’re keeping up with them
- Assignment 1 due Sunday

Probability questions
- Monty hall
  - should you switch?
- The Coin game
  - HHH vs. THT
  - This is sort of like the language modeling task we’ll look at today
Independence

- Two variables are independent if they do not effect each other.
- For two independent variables, knowing the value of one does not change the probability distribution of the other variable.
  - The result of the toss of a coin is independent of a roll of a die.
  - Price of tea in England is independent of whether or not you get an A in NLP.

Independent or Dependent?

- Catching a cold and enjoying reading books.
- Miles per gallon and driving habits.
- Height and longevity of life.

Independent variables

- How does independence affect our probability equations/properties?

  - If A and B are independent, written $A \perp B$.
    - $P(A, B) = P(A)P(B)$
    - $P(A | B) = P(A)$
    - $P(B | A) = P(B)$

Conditional Independence

- Dependent events can become independent given certain other events.
- Examples:
  - Height and length of life.
  - "Correlation" studies
    - Size of your lawn and length of life.

http://xkcd.com/552/
Conditional Independence

- Dependent events can become independent given certain other events
  - Examples,
    - height and length of life
    - "correlation" studies
      - size of your lawn and length of life
  - If A, B are conditionally independent of C \( A \perp B \mid C \)
    - \( P(A,B \mid C) = P(A \mid C)P(B \mid C) \)
    - \( P(A \mid B,C) = P(A \mid C) \)
    - \( P(B \mid A,C) = P(B \mid C) \)
    - but \( P(A,B) \neq P(A)P(B) \)

Assume independence

- Sometimes we will assume two variables are independent (or conditionally independent) even though they’re not
  - Why?
    - Creates a simpler model
      - \( p(X,Y) \) many more variables than just \( p(X) \) and \( p(Y) \)
    - May not be able to estimate the more complicated model

Language modeling

- What does natural language look like?
  - More specifically in NLP, probabilistic model
  - Two related questions:
    - \( p(\text{sentence} ) \)
      - \( p(\text{“I like to eat pizza”}) \)
      - \( p(\text{“pizza like I eat”}) \)
    - \( p(\text{word} \mid \text{previous words} ) \)
      - \( p(\text{“pizza”} \mid \text{“I like to eat”} ) \)
      - \( p(\text{“garbage”} \mid \text{“I like to eat”} ) \)
      - \( p(\text{“run”} \mid \text{“I like to eat”} ) \)

Language modeling

- How might these models be useful?
  - Language generation tasks
    - machine translation
    - summarization
    - simplification
    - speech recognition
    - …
  - Text correction
    - spelling correction
    - grammar correction
Ideas?

- $p(\text{"I like to eat pizza"})$
- $p(\text{"pizza like I eat"})$
- $p(\text{"pizza" | "I like to eat"})$
- $p(\text{"garbage" | "I like to eat"})$
- $p(\text{"run" | "I like to eat"})$

Language modeling

I think today is a good day to be me

Google

"I think today is a good day to be me" Search

No results found for "I think today is a good day to be me".

Language modeling is about dealing with data sparsity!

Look at a corpus
Language modeling

- A language model is really a probabilistic explanation of how the sentence was generated
- Key idea:
  - break this generation process into smaller steps
  - estimate the probabilities of these smaller steps
  - the overall probability is the combined product of the steps

Language modeling

- Two approaches:
  - n-gram language modeling
    - Start at the beginning of the sentence
    - Generate one word at a time based on the previous words
  - syntax-based language modeling
    - Construct the syntactic tree from the top down
    - e.g. context free grammar
    - eventually at the leaves, generate the words

Pros/cons?

n-gram language modeling

I think today is a good day to be me

Google

"I think"

Web: Search

Google

"today is a good day"

Web: Search

Google

"to be me"

Web: Search

Our friend the chain rule

Step 1: decompose the probability

\[ P(\text{I think today is a good day to be me}) = \]

\[ P(\text{I}) \mid \text{<start>}) \times \]

\[ P(\text{think}) \mid I \times \]

\[ P(\text{today}) \mid \text{I think}) \times \]

\[ P(\text{is}) \mid \text{I think today}) \times \]

\[ P(\text{a}) \mid \text{I think today is}) \times \]

\[ P(\text{good}) \mid \text{I think today is a}) \times \]

\[ \ldots \]

How can we simplify these?
The n-gram approximation

Assume each word depends only on the previous n-1 words (e.g. trigram: three words total)

- \( P(is \mid I\ think\ today) \approx P(is \mid think\ today) \)
- \( P(a \mid I\ think\ today\ is) = P(a \mid today\ is) \)
- \( P(good \mid I\ think\ today\ is\ a) = P(good \mid is\ a) \)

Estimating probabilities

- How do we find probabilities?
- Get real text, and start counting (MLE)!

\[
P(is \mid think\ today) = \frac{\text{count}(think\ today\ is)}{\text{count}(think\ today)}
\]

Estimating from a corpus

Corpus of sentences (e.g. gigaword corpus)

I am a happy Middlebury College student.

Why do we need \(<start>\) and \(<end>\)?

I am a happy Middlebury
Happy Middlebury College
Middlebury College student
College student
student

\(<start>\ <end>\)
Estimating from a corpus

I am a happy Middlebury College student.

- count all of the trigrams
  - `<start> <start> I`
  - `<start> I am`
  - `I am a`
  - `a happy Middlebury`
  - `happy Middlebury College`
  - `Middlebury College student`
  - `College student . <end>`
  - `<end> <end>`

- Do we need to count anything else?

Estimating from a corpus

- count all of the bigrams
  - `<start> <start>`
  - `<start> I`
  - `I am`
  - `am a`
  - `a happy`
  - `happy Middlebury`
  - `Middlebury College`
  - `College student . <end>`
  - `<end>`

- \[ p(c | a b) = \frac{\text{count}(a b c)}{\text{count}(a b)} \]

Applying a model

- Given a new sentence, we can apply the model
  - \[ p(\text{Middlebury College students are the best .}) = \]
    - \[ p(\text{Middlebury | <start> <start> }) \ast \]
    - \[ p(\text{College | <start> Middlebury }) \ast \]
    - \[ p(\text{students | Middlebury College }) \ast \]
    - \[ p(\text{the end | . <end> }) \ast \]
Some examples

Generating examples

- We can also use a trained model to generate a random sentence
- Ideas?

<start> <start>

We have a distribution over all possible starting words

Draw one from this distribution

p(A | <start> <start>)
p(Apples | <start> <start>)
p(I | <start> <start>)
p(The | <start> <start>)
p(Zebras | <start> <start>)

...

Generating examples

- Unigram

Unigram are were that áreas normally naturally built describes jazz territory heteromyid: film tenor prime live founding must on was feet negro legal gate in an beside. provincial son; stephenson simply spaces stretched performance double-entry grove replacing station across to burma; marinating áreas capital about double reached omilia el time believed what hotels parameter jurisprudence words syndrome to áreas profusely in administrators áreas offices hilario institutionalized remains writer royalty clients, áreas tycoon, and objective.

Instructs seem timekeeper has áreas valley áreas “ magnesium for love on áreas from ciklakent, once central enlightened.” so, áreas is belongs home they the corrected, on in pressure “NUMBER,” her flavored áreas derogatory is won metcard indirectly of crop duty learn northbound áreas áreas dancing similarity áreas named áreas berkeley, “off-scale overtime,” each manifield stripes died traffic assert and at alpha popularity town.
Generation examples

- Bigrams

  - the wikipedia county, mexico.
  - maurice ravel. it is require that is sparta, where functions most widely admired.
  - halogens chamioli cast jason against test site.

Generation examples

- Trigrams

  - is widespread in north africa in june. units were built by with.
  - jewish video spiritual are considered irad, this season was an extratropical cyclone.
  - the british railways' s strong and a spot.

Evaluation

- We can train a language model on some data
- How can we tell how well we’re doing?
  - for example
    - bigrams vs. trigrams
    - 100K sentence corpus vs. 100M
    - ...

Evaluation

- A very good option: extrinsic evaluation

- If you’re going to be using it for machine translation
  - build a system with each language model
  - compare the two based on their approach for machine translation

- Sometimes we don’t know the application
- Can be time consuming
- Granularity of results
Evaluation

- Common NLP/machine learning/AI approach

- All sentences
  - Training sentences
  - Testing sentences

Evaluation

- A good model should do a good job of predicting actual sentences

- Test sentences

Perplexity

- View the problem as trying to predict the test corpus one word at a time in sequence
- A perfect model would always know the next word with probability 1 (like people who finish each other's sentences)

- Test sentences

I like to eat banana peels.
Perplexity

- A reasonable measure of how well our model is doing would be the average probability:
  \[ \sqrt[n]{\prod_{i=1}^{n} P(w_i | w_{1:i-1})} \]

- Perplexity is a related measure that is commonly used and is 1 over this value and often done in log space
  \[ \sqrt[n]{\frac{1}{\prod_{i=1}^{n} P(w_i | w_{1:i-1})}} = \frac{-1}{n} \sum_{i=1}^{n} \log P(w_i | w_{1:i-1}) \]

Another view of perplexity

- Weighted average branching factor
  - number of possible next words that can follow a word or phrase
  - measure of the complexity/uncertainty of text (as viewed from the language models perspective)

Smoothing

What if our test set contains the following sentence, but one of the trigrams never occurred in our training data?

```
P(\text{think today is a good day to be me}) =
P(\text{I}) \cdot \text{<start> <start>} \cdot x
P(\text{think} | \text{<start>} \text{I}) \cdot x
P(\text{today} | \text{think} \text{I}) \cdot x
P(\text{is} | \text{think today} \text{I}) \cdot x
P(\text{good} | \text{is a} \text{I}) \cdot x
... If any of these has never been seen before, prob = 0!
```

A better approach

- \( p(z | x, y) = ? \)
- Suppose our training data includes
  - \( \ldots x y a \ldots \)
  - \( \ldots x y d \ldots \)
  - but never: \( xyz \)
- We would conclude
  - \( p(\text{a} | x y) = 1/3? \)
  - \( p(\text{d} | x y) = 2/3? \)
  - \( p(z | x y) = 0/3? \)
- Is this ok?
- Intuitively, how should we fix these?
Smoothing the estimates

- Basic idea:
  - $p(a | x y) = 1/3$: reduce
  - $p(d | x y) = 2/3$: reduce
  - $p(z | x y) = 0/3$: increase

- Discount the positive counts somewhat
- Reallocate that probability to the zeroes
- Remember, it needs to stay a probability distribution

Other situations

- $p(z | x y) = ?$
- Suppose our training data includes
  - $... x y a ...$ (100 times)
  - $... x y d ...$ (100 times)
  - but never: $x y z$

- Suppose our training data includes
  - $... x y a ...$
  - $... x y d ...$
  - $... x y z$ (300 times)
  - but never: $x y z$

  Is this the same situation as before?

Smoothing the estimates

- Should we conclude:
  - $p(a | x y) = 1/3$: reduce
  - $p(d | x y) = 2/3$: reduce
  - $p(z | x y) = 0/3$: increase

- Readjusting the estimate is particularly important if:
  - the denominator is small ...
  - numerator is small ...

  1/3 probably too high, 100/300 probably about right
  1/300 probably too high, 100/300 probably about right

Add-one (Laplacian) smoothing

<table>
<thead>
<tr>
<th></th>
<th>xya</th>
<th>xyb</th>
<th>xyc</th>
<th>xyd</th>
<th>xye</th>
<th>xyz</th>
<th>Total xy</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>prob</td>
<td>1/3</td>
<td>0/3</td>
<td>0/3</td>
<td>2/3</td>
<td>0/3</td>
<td>0/3</td>
<td>3/3</td>
</tr>
<tr>
<td>prob</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>prob</td>
<td>2/29</td>
<td>1/29</td>
<td>1/29</td>
<td>3/29</td>
<td>1/29</td>
<td>1/29</td>
<td>29/29</td>
</tr>
</tbody>
</table>

Total $xy$: 29/29
Add-one (Laplacian) smoothing

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xy</td>
<td>100</td>
<td>100/300</td>
<td>101</td>
<td>101/326</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>xyd</td>
<td>200</td>
<td>200/300</td>
<td>201</td>
<td>201/326</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>300/300</td>
<td>326</td>
<td>326/326</td>
</tr>
</tbody>
</table>

Total xy

Add-one (Laplacian) smoothing

What happens if we're now considering 20,000 word types?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
<td>2/29</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
<td>3/29</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>29</td>
<td>29/29</td>
</tr>
</tbody>
</table>

Add-one (Laplacian) smoothing

20,000 word types, not 26 letters

- An "unseen event" is a 0-count event
- The probability of an unseen event is 19998/20003
- Add one smoothing thinks it is very likely to see a novel event
- The problem with add-one smoothing is it gives too much probability mass to unseen events

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
<td>2/20003</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
<td>3/20003</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>20003</td>
<td>20003/20003</td>
</tr>
</tbody>
</table>

Any problem with this?
The general smoothing problem

<table>
<thead>
<tr>
<th></th>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
</tr>
</tbody>
</table>

Add-lambda smoothing

- A large dictionary makes novel events too probable.
- Instead of adding 1 to all counts, add $\lambda = 0.01$.
- This gives much less probability to novel events.

<table>
<thead>
<tr>
<th></th>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3 1.01 1.01/203</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3 2.01 2.01/203</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3 203</td>
</tr>
</tbody>
</table>

Add-lambda smoothing

How should we pick $\lambda$?

<table>
<thead>
<tr>
<th></th>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3 1.01 1.01/203</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3 2.01 2.01/203</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3 0.01 0.01/203</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3 203</td>
</tr>
</tbody>
</table>

Setting smoothing parameters

- Idea 1: try many $\lambda$ values & report the one that gets best results?

Is this fair/appropriate?
Correct experimentation

- **General rules:**
  - Test data should only be used for evaluation
  - No peeking! Only use it for your final results.
  - Never skew anything in your favor

- **Other ideas?**

Setting smoothing parameters

- Training
- Test
- Training
- Dev.
- Dev.

**Training**
- Collect counts from 80% of the data
- Pick $\lambda$ that gets best results on 20%
- Now use that $\lambda$ to get smoothed counts from all 100%
- And report results of that final model on test data

Concerns

- 20% may not be enough to reliably determine $\lambda$.
- We’re maximizing lambda for only 80% of our data (will not be the same as the optimal for 100%)
- We’re losing 20% of our data for calculating counts

**Ideas?**

Cross-validation (aka “jackknifing”)

- If 20% too little: try 5 training/test splits as below
  - Pick $\lambda$ that gets best average performance

<table>
<thead>
<tr>
<th>Dev.</th>
<th>Dev.</th>
<th>Dev.</th>
<th>Dev.</th>
<th>Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- This tests on all 100% (in turn), so we can more reliably assess $\lambda$.
- Unfortunately, still picks a $\lambda$ that does well on 80% training.
N-fold Cross-Validation and “Leave One Out”

- Test each sentence with smoothed model from other N-1 sentences
- Still tests on all 100% (in turn), so we can reliably assess λ.
- Tests if λ is good for smoothing (N-1)/N = 100% of training data, which matches our actual test conditions
- Surprisingly fast: why?
  - Usually easy to change model by adding/subtracting 1 sentence’s counts

Discussion

- In a Race to Out-Rave, 5-Star Web Reviews Go for $5
- Summary
- Have you ever been misled or tricked by fake product reviews? Do you trust online reviews?
- Are fake reviews easy for you to spot?
- Is it a sound investment for companies to fund this type of research? Do fake reviews hurt business?

- In a Race to Out-Rave, 5-Star Web Reviews Go for $5
- Summary
- Have you ever been misled or tricked by fake product reviews? Do you trust online reviews?
- Are fake reviews easy for you to spot?
- Is it a sound investment for companies to fund this type of research? Do fake reviews hurt business?