Announcements

• **Homeworks:**
  – HW3 released. Start early!

• **Class roadmap:**

<table>
<thead>
<tr>
<th>Tuesday, Sep 28</th>
<th>NLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, Sep 30</td>
<td>ML Intro</td>
</tr>
<tr>
<td>Tuesday, Oct 5</td>
<td>ML Unsupervised I</td>
</tr>
<tr>
<td>Thursday, Oct 7</td>
<td>ML Unsupervised I</td>
</tr>
<tr>
<td>Tuesday, Oct 12</td>
<td>ML Linear Regression</td>
</tr>
</tbody>
</table>
Homework Review: PCA Recursion

• Once we have $k$-1 components, next?

$$\hat{X}_k = X - \sum_{i=1}^{k-1} X v_i v_i^T$$

• Then do the same thing

$$v_k = \arg \max_{\|v\|=1} \|\hat{X}_k v \|^2$$
Homework Review: Eigendecomposition

- Recall e-val/e-vectors: $A\nu = \lambda\nu$
- Eigendecomposition:
  $$A = Q\Lambda Q^{-1}$$

- **Equivalency**: $\nu_k$ computed by recursion is the $k$-th e-vector of sample covariance matrix $\Sigma$
- HW3: replace PCA recursion with eigendecomposition
Homework Review: Covariance

- Recall variance: $\mathbb{E}[(X - \mathbb{E}[X])^2]$ 
- Now, for a **random vector** $X = (X_1, \ldots, X_d)^T$
- **Covariance matrix** (size $d \times d$):
  \[
  \begin{bmatrix}
  \mathbb{E}[(X_1 - \mathbb{E}[X_1])^2] & \cdots & \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_d - \mathbb{E}[X_d])] \\
  \vdots & \ddots & \vdots \\
  \mathbb{E}[(X_d - \mathbb{E}[X_d])(X_1 - \mathbb{E}[X_1])] & \cdots & \mathbb{E}[(X_d - \mathbb{E}[X_d])^2]
  \end{bmatrix}
  \]

  - **Cross-variance**
  - **Diagonals**: **Variance**

- **Sample covariance matrix**: estimate of above
Last Time: CNFs

• Recall the form:
  \[(\neg A \lor B \lor C) \land (\neg B \lor A) \land (\neg C \lor A)\]

• Any sentence can be transformed to a CNF
  – Use equivalences for connectives we don’t use (i.e., ⇒)
  – Move negatives inside (DeMorgan’s laws)
  – Push \(\lor\) inside \(\land\) by distributing

• Note: in general, not unique. In canonical form, unique. Not guaranteed to be satisfiable
What is NLP?

Combining computing with human language. Want to:

- Answer questions
- Summarize or extract information
- Translate between languages
- Generate dialogue/language
- Write stories automatically
Why is it hard?

Many reasons:

– Ambiguity: “We saw her duck”. Several meanings.
– Non-standard use of language
– Segmentation challenges
– Understanding of the world
  • “Bob and Joe are brothers”.
  • “Bob and Joe are fathers”.


Approaches to NLP

A brief history

– Symbolic NLP: 50’s to 90’s
– Statistical/Probabilistic: 90’s to present
  • Neural: 2010’s to present

Lots of progress!

Lots more to work to do

ELIZA program
Outline

• Introduction to language models
  – n-grams, training, improving issues, evaluation

• Classic NLP tasks
  – Part-of-speech tagging, parsing, dependencies

• Word representations
  – One-hot, word embeddings, transformer-based
Language Models

• Basic idea: use probabilistic models to assign a probability to a sentence

\[ P(W) = P(w_1, w_2, \ldots, w_n) \text{ or } P(w_{\text{next}} | w_1, w_2 \ldots) \]

• Goes back to Shannon
  – Information theory: letters
Training The Model

Recall the chain rule

\[ P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_{n-1}\ldots w_1) \]

• How do we estimate these probabilities
  – Same thing as “training”

• From data?
  – Yes, but not directly: too many sentences.
  – Can’t estimate reliably.
Training: Make Assumptions

• Markov-type assumptions:

\[ P(w_i | w_{i-1} w_{i-2} \ldots w_1) = P(w_i | w_{i-1} w_{i-2} \ldots w_{i-k}) \]

• Present doesn’t depend on whole past
  – Just recent past
  – Markov chains have \( k=1 \). \text{(Present only depends on immediate past).}
  – What’s \( k=0 \)?
k=0: **Unigram Model**

- **Full independence assumption:**
  - (Present doesn’t depend on the past)

\[
P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2)\ldots P(w_n)
\]

- **Example (from Dan Jurafsky’s notes)**

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass thrift, did, eighty, said, hard, 'm, july, bullish that, or, limited, the
k=1: **Bigram Model**

- **Markov Assumption:**
  - (Present depends on immediate past)

\[
P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1})
\]

- **Example:**

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen outside, new, car, parking, lot, of, the, agreement, reached this, would, be, a, record, november
k=n-1: n-gram Model

Can do trigrams, 4-grams, and so on

• More expressive as $n$ goes up
• Harder to estimate

Training: just count? I.e, for bigram:

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$
n-gram Training

Issues:

1. Multiply tiny numbers?
   - **Solution**: use logs; add instead of multiply

2. n-grams with zero probability?
   - **Solution**: smoothing

\[
P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}
\]

\[
P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i) + 1}{\text{count}(w_{i-1}) + V}
\]
Other Solutions: Backoff & Interpolation

For issue 2, back-off methods

• Use n-gram where there is lots of information, r-gram (with $r << n$) elsewhere. (trigrams / bigrams)

Interpolation

• Mix different models: (tri- + bi- + unigrams)

$$\hat{P}(w_i | w_{i-1}, w_{i-2}) = \lambda_1 P(w_i | w_{i-1}, w_{i-2}) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i)$$
n-gram Training Issues

Issues:

• **1. Multiply tiny numbers?**
  – **Solution**: use logs; add instead of multiply

• **2. Sparse n-grams**
  – **Solution**: smoothing, backoff, interpolation

• **3. Vocabulary: open vs closed**
  – **Solution**: use <UNK> unknown word token
Vocabulary: open vs closed

• Possible to estimate size of unknown vocabulary
  – Good-Turing estimator

• Originally developed to crack the Enigma machine
Break & Quiz

**Q 1.1:** Which of the below are bigrams from the sentence “It is cold outside today”.

- A. It is
- B. cold today
- C. is cold
- D. A & C
Break & Quiz

Q 1.1: Which of the below are bigrams from the sentence “It is cold outside today”.

• A. It is
• B. cold today
• C. is cold
• D. A & C
Break & Quiz

Q 1.2: Smoothing is increasingly useful for n-grams when

- A. n gets larger
- B. n gets smaller
- C. always the same
- D. n larger than 10
Q 1.2: Smoothing is increasingly useful for n-grams when

• **A. n gets larger**
• B. n gets smaller
• C. always the same
• D. n larger than 10
Evaluating Language Models

How do we know we’ve done a good job?

• Observation
• Train/test on separate data & measure metrics
• **Metrics:**
  – 1. Extrinsic evaluation
  – 2. Perplexity
Extrinsic Evaluation

How do we know we’ve done a good job?

• **Pick a task** and use the model to do the task

• For two models, $M_1$, $M_2$, compare the accuracy for each task
  - **Ex**: Q/A system: how many questions right. Translation: how many words translated correctly

• Downside: slow; may change relatively
Intrinsic Evaluation: Perplexity

Perplexity is a **measure of uncertainty**

\[
PP(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}
\]

**Lower is better!** Examples:

- WSJ corpus; 40 million words for training:
  - Unigram: 962, Bigram 170, Trigram 109
Further NLP Tasks

Language modeling is not the only task. Two further types:

1. **Auxiliary** tasks:
   - Part-of-speech tagging, parsing, etc.

2. **Direct** tasks:
   - Question-answering, translation, summarization, classification (e.g., sentiment analysis)
Part-of-speech Tagging

Tag words as nouns, verbs, adjectives, etc.

- Tough part: ambiguous, even for people.
- Needs:
  - Getting neighboring word parts right
  - Knowledge of words (“man” is used as a noun, rarely as verb)

<table>
<thead>
<tr>
<th>Model</th>
<th>Features</th>
<th>Token</th>
<th>Unknown</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>56,805</td>
<td>93.69%</td>
<td>82.61%</td>
<td>26.74%</td>
</tr>
<tr>
<td>3Words</td>
<td>239,767</td>
<td>96.57%</td>
<td>86.78%</td>
<td>48.27%</td>
</tr>
</tbody>
</table>

Chris Manning
Parsing

Get the grammatical structure of sentences

- Which words depend on each other? Note: input a sentence, output a tree (dependency parsing)

Chris Manning
Break & Quiz

Q 2.1: What is the perplexity for a sequence of $n$ digits 0-9? All occur with equal probability.

• A. 10
• B. $\frac{1}{10}$
• C. $10^n$
• D. 0

$$\text{PP}(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$
Break & Quiz

Q 2.1: What is the perplexity for a sequence of $n$ digits 0-9? All occur with equal probability.

• A. 10
• B. 1/10
• C. $10^n$
• D. 0

$$PP(W) = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}}$$
Representing Words

Recall random variables *(RVs)*: real valued

- Easier to work with than objects like ‘dog’

Traditional representation: *one-hot vectors*

\[
dog = [0 \ 0 \ 0 \ 0 \ 1 \ 0]
\]

- Dimension: # of words in vocabulary
- Relationships between words?
Smarter Representations

Distributional semantics: account for relationships

• Reps should be close/similar to other words that appear in a similar context

Dense vectors:

\[
\text{dog} = [0.13 \ 0.87 \ -0.23 \ 0.46 \ 0.87 \ -0.31]^T
\]

\[
\text{cat} = [0.07 \ 1.03 \ -0.43 \ -0.21 \ 1.11 \ -0.34]^T
\]

AKA word embeddings
Training Word Embeddings

Many approaches (super popular 2010-present)

• Word2vec: a famous approach
• What’s our likelihood?

\[
L(\theta) = \prod_{t=1}^{T} \prod_{-a \leq j \leq a} P(w_{t+j} | w_t, \theta)
\]

Windows of length 2a

Our word vectors (variables/hypotheses)
All positions
Training Word Embeddings

Word2vec likelihood

\[ L(\theta) = \prod_{t=1}^{T} \prod_{-a \leq j \leq a} P(w_{t+j} | w_t, \theta) \]

• Maximize this; what’s the probability?
  – Two vectors per word. \( v_w, u_w \) for center/context (\( o \) is context word, \( c \) is center)

\[
P(o | c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}
\]
Beyond “Shallow” Embeddings

- Transformers: special model architectures based on attention
  - Sophisticated types of neural networks
- Pretrained models
  - Based on transformers: BERT
  - Include context!
- **Fine-tune** for desired task

Vaswani et al. 17