Vector semantics and embeddings unit

- Lexical semantics, words as vectors (last week Monday)
- Catch-up day (last week Wednesday)
- Word2Vec (last week Friday)
- Finish Word2Vec (today)
- Overview of neural nets (Wednesday)
- Neural-net language models, in lab (Friday)

Today:

- Review of premise and elements
- Details of the training algorithm
- Observing results
- Problems with data-driven methods
Goal: Find word embeddings, vectors that represent words in a semantic space.

Word2vec premise: Train a classifier on a “fake” task and use that classifier’s weights/parameters as word embeddings.

Word2vec algorithm outline: Given a corpus,

- Find the vocabulary of the corpus
- Collect training data from the corpus
- Train a classifier on that data
- Return the classifier’s weights
The classification task: Given target word $w$ and potential context word $c$, is $c$ likely (or, how likely is $c$) to be used near $w$? That is, is $c$ a true context word for $w$?

Alice looked all around her at the flowers

$c_0$  $c_1$  $w$  $c_2$  $c_3$

The training data: For every token $w$ in the corpus, pair it with the tokens found $L$ positions before it and $L$ positions after it (positive examples). For every positive example, find $k$ randomly chosen negative examples.

<table>
<thead>
<tr>
<th>$w$</th>
<th>$c_{pos}$</th>
<th>$c_{neg_0}$</th>
<th>$c_{neg_1}$</th>
<th>$c_{neg_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>around</td>
<td>looked</td>
<td>pineapple</td>
<td>earnestly</td>
<td>asleep</td>
</tr>
<tr>
<td>around</td>
<td>all</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>around</td>
<td>her</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>around</td>
<td>at</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Algorithm parameters:
$L$, window size ($L = 2$ above)
$k$, number of negative samples ($k = 3$ above)
Choosing negative samples:

For training a binary classifier, we also need negative examples. . . [For each training instance,] we’ll create \( k \) negative samples, each consisting of a target word \( w \) plus a “noise word” \( c_{neg} \). A noise word is a random word from the lexicon, constrained not to be the target word \( w \). . .

The noise words are chosen according to their weighted unigram frequency \( p_{\alpha}(w) \), where \( \alpha \) is a weight. . .

\[
p_{\alpha}(w) = \frac{\text{count}(w)^{\alpha}}{\sum_{w'} \text{count}(w')^{\alpha}}
\]

[Weighting \( p \), for example at \( \alpha = .75 \)] gives better performance because it gives rare noise words slightly higher probability: for rare words, \( P_{\alpha}(w) > P(w) \).

Jurafsky and Martin, 6.8.2, pg 20–21
The classifier: Logistic regression

\[ P(+ \mid w, c) = \sigma(w \cdot c) \]

where

- \( P(\pm \mid w, c) \) is the probability \( c \) is a context word for \( w \).
- \( w \) is a vector for \( w \) as a target word.
- \( c \) is a vector for \( c \) as a context word.
- \( w \cdot c \) is the dot product
- \( \sigma x = \frac{1}{1 + \exp(-x)} \) is the logistic (sigmoid) function.

Parameters to the classifier: \( W \) and \( C \), each \( V \times D \) matrices.

Parameter to the algorithm: \( D \), length of embedding
The loss function: The cross-entropy (negative log likelihood) loss. For a given data point \((w, c_{pos}, c_{neg_0}, \ldots c_{neg_{k-1}})\), the loss is

\[
- \left( \log \sigma(c_{pos} \cdot w) + \sum_{i=0}^{k-1} \log \sigma(-c_{neg_i} \cdot w) \right)
\]

The training algorithm: Stochastic gradient descent. For each data point \((w, c_{pos}, c_{neg_0}, \ldots c_{neg_{k-1}})\),

\[
c_{pos} = \eta(\sigma(c_{pos} \cdot w) - 1)w
\]

\[
c_{neg_i} = \eta(\sigma(c_{neg_i} \cdot w))w
\]

\[
w = \eta \left( (\sigma(c_{pos} \cdot w) - 1)c_{pos} + \sum_{i=0}^{k-1} \sigma(c_{neg_i} \cdot w)c_{neg_i} \right)
\]

Parameter to the algorithm: \(\eta\), the learning rate
Coming up:

- Read J&M chapter 7 (Mon, Nov 13)
- Work on stylo assignment

Word2Vec assignment coming...