Neural language models

- Skip-grams
- Continuous Bag of Words (CBOW)
- More details can be found at https://cs224d.stanford.edu/lecture_notes/notes1.pdf

Prediction-based models: An alternative way to get dense vectors

- Skip-gram (Mikolov et al. 2013a), CBOW (Mikolov et al. 2013b)
- Learn embeddings as part of the process of word prediction
- Train a neural network to predict neighboring words
- Advantages:
  - Fast, easy to train (much faster than SVD)
  - Available online in the word2vec package
  - Including sets of pretrained embeddings!

Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: predict rather than count

Word2vec

- Given a sentence:
  ... lemon, a tablespoon of apricot jam a pinch ...
- Instead of counting how often each word w occurs near "apricot"
- Train a classifier on a binary prediction task:
  - Is w likely to show up near "apricot"?
- We don’t actually care about this task
  - But we’ll take the learned weights (will be discussed later) as the word embeddings

Brilliant insight: Use running text as implicitly supervised training data!

- A word near apricot
  - Acts as gold ‘correct answer’ to the question
  - Ask questions: "Is word w likely to show up near apricot?"
- No need for hand-labeled supervision
- The idea comes from neural language modeling
  - Bengio et al. (2003)
  - Collobert et al. (2011)
Word2Vec: **Skip-Gram** Task

- Now we have positive samples.
- Where do the "negative samples" come from?

Word2Vec: **Skip-Gram** Task

- Word2vec provides a variety of options. Let's do "skip-gram with negative sampling" (SGNS)

**Skip-gram algorithm**

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples.
3. Use logistic regression (will discuss formulation later) to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

**Skip-gram Training Data**

- Training sentence:
  ...
  lemon, a **tablespoon of apricot jam**  a pinch ...
  c1  c2  target c3  c4

  Assume context words are those in +/- 2 word window

**Skip-gram Goal**

- Given a tuple \((t,c) = \text{target, context}\)
  - \((\text{apricot}, \text{jam})\)
  - \((\text{apricot}, \text{aardvark})\)
- Return probability that \(c\) is a real context word:
  - \(P(\dagger | t, c)\)
  - \(P(\nabla | t, c) = 1 - P(\dagger | t, c)\)
How to compute \( P(+) | t, c) \)?

- **Intuition:**
  - Words are likely to appear near similar words
  - Model similarity with dot-product!
  - \( \text{Similarity}(t, c) \propto t \cdot c \)

- **Problem:**
  - Dot product is not a probability!
  - *(Neither is cosine)*

Turning dot product into a probability

- The sigmoid lies between 0 and 1:
  \[
  \sigma(x) = \frac{1}{1 + e^{-x}}
  \]

For all the context words:

- Assume all context words are independent
  \[
  P(+) | t, c_{1:k} = \prod_{i=1}^{k} \frac{1}{1 + e^{-c_i}}
  \]
  \[
  \log P(+) | t, c_{1:k} = \sum_{i=1}^{k} \log \frac{1}{1 + e^{-c_i}}
  \]

Skip-gram Training Data

- Training sentence:
  ... lemon, a tablespoon of apricot jam a pinch ...
  c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Assume a +/- 2 word window

For all the context words:

- Assume all context words are independent
  \[
  P(+) | t, c_{1:k} = \prod_{i=1}^{k} \frac{1}{1 + e^{-c_i}}
  \]
  \[
  \log P(+) | t, c_{1:k} = \sum_{i=1}^{k} \log \frac{1}{1 + e^{-c_i}}
  \]

Skip-gram Training Data

- Training sentence:
  ... lemon, a tablespoon of apricot jam a pinch ...
  c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Assume a +/- 2 word window
### Skip-gram Training

- **Training sentence:**
  
  ... lemon, a tablespoon of apricot jam a pinch ...

  
  \[
  \begin{array}{c|ccccc}
  & t & c1 & c2 & c3 & c4 \\
  \hline
  \text{positive examples} & \text{apricot} & \text{tablespoon} & \text{apricot} & \text{of} & \text{apricot} \text{ preserves} \\
  \end{array}
  \]

  - For each positive example, we’ll create k negative examples.
  - Using noise words
  - Any random word that isn’t t

- **Negative examples**

  \[
  \begin{array}{c|ccccc}
  & t & c1 & c2 & c3 & c4 \\
  \hline
  \text{negative examples} & \text{apricot} & \text{aardvark} & \text{apricot} & \text{paddle} & \text{apricot} \text{ hello} \\
  \end{array}
  \]

### Choosing noise words

- Could pick \(w\) according to their unigram frequency \(P(w)\)
- More common to choose then according to \(p_\alpha(w)\)

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

- \(\alpha<1\) works well because it gives rare noise words slightly higher probability
- To show this, imagine two events \(p(a)\approx .99\) and \(p(b)\approx .01\):

  \[
  \begin{align*}
  p_\alpha(a) &= \frac{.99^\alpha}{.99^\alpha + .01^\alpha} = .97 \\
  p_\alpha(b) &= \frac{.01^\alpha}{.99^\alpha + .01^\alpha} = .03
  \end{align*}
  \]

### Setup

- Let’s represent words as vectors of some length (say 300), randomly initialized.
- So we start with \(300 \times V\) random parameters
- Over the entire training set, we’d like to adjust those word vectors such that we
  
  - Maximize the similarity of the target word, context word pairs \((t,c)\) drawn from the positive data
  - Minimize the similarity of the \((t,c)\) pairs drawn from the negative data

### Learning the classifier

- Iterative process on training data
- We’ll start with 0 or random weights
- Then adjust the word weights to
  
  - make the positive pairs more likely
  - and the negative pairs less likely
Formally

- We want to maximize the following objective:
  \[
  \sum_{(t,c) \in +} \log P(+|t,c) + \sum_{(t,c) \in -} \log P(-|t,c)
  \]
- Maximize the + label for the pairs from the positive training data, and the - label for the pairs sample from the negative data.

Focusing on one target word \( t \):

\[
L(\theta) = \log P(+|t,c) + \sum_{i=1}^{k} \log P(-|r_i, a_i)
\]

\[
= \log \sigma(c \cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i \cdot t)
\]

\[
= \log \frac{1}{1 + e^{-\sigma}} + \sum_{i=1}^{k} \log \frac{1}{1 + e^{\sigma}}
\]

Train using gradient descent (not required)

- Idea: gradually changing \( W \) and \( C \)
- Finally learns two separate embedding matrices \( W \) and \( C \)
- Can use \( W \) and throw away \( C \), or merge them

Summary: How to learn skip-gram embeddings

- Start with \( V \) random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naïve bayes
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don’t co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
  - Throw away the classifier code and keep the embeddings.

(Dense) Word embeddings you can download!

- **Word2vec** (Mikolov et al.)
  [https://code.google.com/archive/p/word2vec/](https://code.google.com/archive/p/word2vec/)
- **Fasttext** [http://www.fasttext.cc/](http://www.fasttext.cc/)
- **Glove** (Pennington, Socher, Manning)
Evaluating embeddings

- Compare to human scores on word similarity-type tasks:
  - WordSim-353 (Finkelstein et al., 2002)
  - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset:
  - 

Properties of embeddings

- Nearest words to some embeddings (Mikolov et al. 2013)

<table>
<thead>
<tr>
<th>Target</th>
<th>Redmond</th>
<th>Havel</th>
<th>president</th>
<th>graffiti</th>
<th>capitulate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Redmond Washington Microsoft</td>
<td>Vaclav Havel Velvet Revolution</td>
<td>martial arts</td>
<td>graffiti</td>
<td>capitalizing</td>
</tr>
</tbody>
</table>

Properties of embeddings

- Similarity depends on window size C

- \( C = \pm 2 \) The nearest words to \textit{Hogwarts}:
  - Sunnydale
  - Evernight

- \( C = \pm 5 \) The nearest words to \textit{Hogwarts}:
  - Dumbledore
  - Malfoy
  - halfblood

Analogy: Embeddings capture relational meaning!

- \( \text{vector('king')} - \text{vector('man')} + \text{vector('woman')} \approx \text{vector('queen')} \)
- \( \text{vector('Paris')} - \text{vector('France')} + \text{vector('Italy')} \approx \text{vector('Rome')} \)
Embeddings can help study word history!

• Train embeddings on old books to study changes in word meaning!

Diachronic word embeddings for studying language change!

Visualizing changes

Project 300 dimensions down into 2

The evolution of sentiment words

Negative words change faster than positive words

Embeddings and bias

Embeddings reflect cultural bias


• Ask “Paris : France :: Tokyo : x”
  • x = Japan
• Ask “father : doctor :: mother : x”
  • x = nurse
• Ask “man : computer programmer :: woman : x”
  • x = homemaker
Embeddings reflect cultural bias

- Implicit Association test (Greenwald et al 1998): How associated are concepts (flowers, insects) & attributes (pleasantness, unpleasantness)?
- Studied by measuring timing biases for categorization.
- Psychological findings on US participants:
  - African-American names are associated with unpleasant words (more than European-American names)
  - Male names associated more with math, female names with arts
  - Old people’s names with unpleasant words, young people with pleasant words.
- Caliskan et al. replication with embeddings:
  - African-American names (Jenny, Shanique) had a higher GloVe cosine with unpleasant words (abuse, stink, ugly)
  - European American names (Brend, Greg, Courtney) had a higher cosine with pleasant words (love, peace, minute)
- Embeddings reflect and replicate all sorts of pernicious biases.

Embeddings as a window onto history

- The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names
- Find its correlation with the actual percentage of women teachers in decade X

History of biased framings of women

- Embeddings for competence adjectives are biased toward men
  - Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.
- This bias is slowly decreasing

Embeddings reflect ethnic stereotypes over time

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - industrious, superstitious, nationalistic, etc
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

Change in linguistic framing 1910-1990

Change in association of Chinese names with adjectives framed as “othering” (barbaric, monstrous, bizarre)

Changes in framing: adjectives associated with Chinese

<table>
<thead>
<tr>
<th>1910</th>
<th>1950</th>
<th>1990</th>
</tr>
</thead>
<tbody>
<tr>
<td>irresponsible</td>
<td>Divorced</td>
<td>Inhibited</td>
</tr>
<tr>
<td>Envious</td>
<td>Outrageous</td>
<td>Passive</td>
</tr>
<tr>
<td>Barbaric</td>
<td>Pompous</td>
<td>Dissolute</td>
</tr>
<tr>
<td>Aggresive</td>
<td>Unstable</td>
<td>Haughty</td>
</tr>
<tr>
<td>Transparent</td>
<td>Effeminate</td>
<td>Complacent</td>
</tr>
<tr>
<td>Morosous</td>
<td>Unprincipled</td>
<td>Forceful</td>
</tr>
<tr>
<td>Hateful</td>
<td>Venomous</td>
<td>Fixed</td>
</tr>
<tr>
<td>Cruel</td>
<td>Disobedient</td>
<td>Active</td>
</tr>
<tr>
<td>Greedy</td>
<td>Predatory</td>
<td>Sensitive</td>
</tr>
<tr>
<td>Bizarre</td>
<td>Boisterous</td>
<td>Hearty</td>
</tr>
</tbody>
</table>
Directions

- Debiasing algorithms for embeddings
- Use embeddings as a historical tool to study bias