Vector Semantics and Embeddings
Tasks

- Vectors which represent words or sequences

how?

- Dimensionality Reduction
- Recurrent Neural Network and Sequence Models
Objective

To embed: convert a token (or sequence) to a vector that represents meaning.
Objective

To embed: convert a token (or sequence) to a vector that represents meaning, or is useful to perform downstream NLP application.
Objective

port $\xrightarrow{\text{embed}}$ [ ]
Objective

\[ \text{port} \xrightarrow{\text{embed}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix} \]
Objective

One-hot is sparse vector

Port $\xrightarrow{\text{embed}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}$

Prefer dense vectors
- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms

For deep learning, in practice, they work better. Why? Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.
Objective

Prefer dense vectors
- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms

For deep learning, in practice, they work better. Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.

(Jurafsky, 2012)
Objective

One-hot encoding vector

Prefer dense vectors

- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms for deep learning, in practice, they work better.

One-hot encoding vector is sparse vector (Jurafsky, 2012)

Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.
Objective

To embed: convert a token (or sequence) to a vector that represents meaning.
Objective

To embed: convert a token (or sequence) to a vector that represents meaning.

Wittgenstein, 1945: “The meaning of a word is its use in the language”

Distributional hypothesis -- A word’s meaning is defined by all the different contexts it appears in (i.e. how it is “distributed” in natural language).

Firth, 1957: “You shall know a word by the company it keeps”
Objective

To embed: convert a token (or sequence) to a vector that represents meaning.

Wittgenstein, 1945: “The meaning of a word is its use in the language”

Distributional hypothesis -- A word’s meaning is defined by all the different contexts it appears in (i.e. how it is “distributed” in natural language).

Firth, 1957: “You shall know a word by the company it keeps”

The nail hit the beam behind the wall.
The nail hit the beam behind the wall.
Objective

\[ \text{port} \xrightarrow{\text{embed}} \begin{bmatrix} 0.53 \\ 1.5 \\ 3.21 \\ -2.3 \\ 0.76 \end{bmatrix} \]
Objective

port n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, port n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, port n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))
How?

1. One-hot representation
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

Tf-IDF: Term Frequency, Inverse Document Frequency,

PMI: Point-wise mutual information, ...etc...
How?

1. One-hot representation
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

..., word1, word2, bill, word3, word4, ...

4. Word2vec
5. Fasttext
6. Glove
7. Bert
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
SVD-Based Embeddings

Singular Value Decomposition...
Concept, In Matrix Form:

<table>
<thead>
<tr>
<th></th>
<th>o1</th>
<th>o2</th>
<th>o3</th>
<th>…</th>
<th>o_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1, f2, f3, f4, …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>columns: p features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rows: n observations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SVD-Based Embeddings

f1, f2, f3, f4, ...  fp

o1
o2
o3
...

On
SVD-Based Embeddings

Dimensionality reduction
-- try to represent with only $p'$ dimensions
Data (or, at least, what we want from the data) may be accurately represented with less dimensions.
Concept: Dimensionality Reduction in 3-D, 2-D, and 1-D

Data (or, at least, what we want from the data) may be accurately represented with less dimensions.

Concept: Dimensionality Reduction

Rank: Number of linearly independent columns of A. (i.e. columns that can’t be derived from the other columns through addition).

Q: What is the rank of this matrix?

\[
\begin{pmatrix}
1 & -2 & 3 \\
2 & -3 & 5 \\
1 & 1 & 0
\end{pmatrix}
\]
Concept: Dimensionality Reduction

Rank: Number of linearly independent columns of A.
(i.e. columns that can’t be derived from the other columns through addition).

Q: What is the rank of this matrix?

\[
\begin{pmatrix}
1 & -2 & 3 \\
2 & -3 & 5 \\
1 & 1 & 0
\end{pmatrix}
\]

A: 2. The 1st is just the sum of the second two columns

\[
\begin{pmatrix}
1 \\
2 \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
-2 \\
-3 \\
1
\end{pmatrix}
\]

\ldots we can represent as linear combination of 2 vectors:
SVD-Based Embeddings

context words are features
f1, f2, f3, f4, ...

co-occurrence counts are cells.

target words are observations
o1, o2, o3, ...

Dimensionality reduction
-- try to represent with only p' dimensions

c1, c2, c3, c4, ...
cp'

SVD-Based Embeddings
Dimensionality Reduction - PCA

Linear approximates of data in $r$ dimensions.

Found via *Singular Value Decomposition*:

$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

$X$: original matrix, $U$: “left singular vectors”, $D$: “singular values” (diagonal), $V$: “right singular vectors”
Dimensionality Reduction - PCA

Linear approximates of data in $r$ dimensions.

Found via *Singular Value Decomposition*:

$$X_{[nxp]} = U_{[nxr]}D_{[rxr]}V_{[pxr]}^T$$

$X$: original matrix,
$U$: “left singular vectors”,
$D$: “singular values” (diagonal),
$V$: “right singular vectors”
Dimensionality Reduction - PCA - Example

\[ X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T \]

Word co-occurrence counts:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
= \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
\]
Dimensionality Reduction - PCA - Example

\[ X_{[n \times p]} \cong U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T \]

Observation: “beam.”

- count(beam, hit) = 100 -- horizontal dimension
- count(beam, nail) = 80 -- vertical dimension
Dimensionality Reduction - PCA

Linear approximates of data in \( r \) dimensions.

Found via *Singular Value Decomposition*:

\[
X_{[n \times p]} \approx U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T
\]

\( X \): original matrix, \( U \): “left singular vectors”,
\( D \): “singular values” (diagonal), \( V \): “right singular vectors”

Projection (dimensionality reduced space) in 3 dimensions:

\[
(U_{[n \times 3]} D_{[3 \times 3]} V_{[p \times 3]}^T)
\]

To reduce features in new dataset, \( A \):

\[
A_{[m \times p]} V D = A_{\text{small}[m \times 3]}
\]
Dimensionality Reduction - PCA

Linear approximates of data in $r$ dimensions.

Found via *Singular Value Decomposition*:

$$X_{[nxp]} \approx U_{[nxr]} D_{[r.rx]} V_{[pxr]}^T$$

- **X**: original matrix,
- **U**: “left singular vectors”,
- **D**: “singular values” (diagonal),
- **V**: “right singular vectors”

To check how well the original matrix can be reproduced:

$$Z_{[nxp]} = U D V^T$$, How does $Z$ compare to original $X$?

To reduce features in new dataset:

$$A_{[m \times p]} VD = A_{small[m \times 3]}$$
Dimensionality Reduction - PCA

- Goal: Minimize the sum of reconstruction errors:
  \[
  \sum_{i=1}^{N} \sum_{j=1}^{D} \|x_{ij} - z_{ij}\|^2
  \]
  - where \(x_{ij}\) are the “old” and \(z_{ij}\) are the “new” coordinates

X: original matrix, \(U\): “left singular vectors”, \(D\): “singular values” (diagonal), \(V\): “right singular vectors”

To check how well the original matrix can be reproduced:
\[Z_{[n \times p]} = U D V^T\], How does Z compare to original X?

To reduce features in new dataset:
\[A_{[m \times p]} \times D = A_{\text{small}[m \times 3]}\]
Dimensionality Reduction - PCA

- **Goal:** Minimize the sum of reconstruction errors:

\[
\sum_{i=1}^{N} \sum_{j=1}^{D} \| x_{ij} - z_{ij} \|^2
\]

- where \( x_{ij} \) are the “old” and \( z_{ij} \) are the “new” coordinates.

- **X:** original matrix, **U:** “left singular vectors”, **D:** “singular values” (diagonal), **V:** “right singular vectors”.

- To check how well the original matrix can be reproduced:

\[
Z_{[n \times p]} = U D V^T
\]

- How does \( Z \) compare to original \( X \)?

- To reduce features in new dataset:

\[
A_{[m \times p]} VD = A_{small[ m \times 3 ]}
\]

This is the objective that SVD Solves.
Dimensionality Reduction - PCA

Linear approximates of data in $r$ dimensions.

Found via *Singular Value Decomposition*:

$$X_{[n \times p]} \approx U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

U, D, and V are unique

D: always positive
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert
Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

\[ p(\text{context} \mid \text{word}) \]
Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

\[ p(\text{context} \mid \text{word}) \]

To learn, maximize
Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

\[ p(\text{context} \mid \text{word}) \]

To learn, maximize.
In practice, minimize

\[ J = 1 - p(\text{context} \mid \text{word}) \]
Word2Vec: Context

2 Versions of Context:
1. Continuous bag of words (CBOW): Predict word from context
2. Skip-Grams (SG): predict context words from target
Word2Vec: Context

2 Versions of Context:
1. Continuous bag of words (CBOW): Predict word from context
2. Skip-Grams (SG): predict context words from target

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

2. Skip-Grams (SG): predict context words from target

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

*The nail hit the beam behind the wall.*

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

\[
\begin{align*}
  x &= (\text{hit, beam}), \ y = 1 \\
  x &= (\text{the, beam}), \ y = 1 \\
  x &= (\text{behind, beam}), \ y = 1 \\
  &\ldots
\end{align*}
\]

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

*The nail hit the beam behind the wall.*

(Jurafsky, 2017)
**Word2Vec: Context**

\[ p(\text{context} \mid \text{word}) \]

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 to train a classifier to distinguish those two cases.

4. Use the weights as the embeddings

*The nail hit the beam behind the wall.*

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

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 to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

\[ \text{The nail hit the beam behind the wall.} \]

\[ \text{The nail hit the beam behind the wall.} \]

\[ \begin{array}{cccc}
  c_1 & c_2 & c_3 & c_4 \\
\end{array} \]

\[ x = (\text{hit, beam}), y = 1 \\
x = (\text{the, beam}), y = 1 \\
x = (\text{behind, beam}), y = 1 \\
\ldots \\
x = (\text{happy, beam}), y = 0 \\
x = (\text{think, beam}), y = 0 \\
\ldots \\
\]

\[ k \text{ negative example (y=0) for every positive. How?} \]

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

- Continuous bag of words (CBOW): Predict word from context
- Skip-Grams (SG): predict context words from target

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 to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

The nail hit the beam behind the wall.

\[ \begin{align*}
    x &= (\text{hit, beam}), y = 1 \\
    x &= (\text{the, beam}), y = 1 \\
    x &= (\text{behind, beam}), y = 1 \\
    \ldots \\
    x &= (\text{happy, beam}), y = 0 \\
    x &= (\text{think, beam}), y = 0 \\
    \ldots
\end{align*} \]

\[ P(w) = \frac{\text{count}(w)}{\sum_w \text{count}(w)} \]

How? Randomly draw from unigram distribution

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

\[ x = (\text{hit}, \text{beam}), y = 1 \]
\[ x = (\text{the}, \text{beam}), y = 1 \]
\[ x = (\text{behind}, \text{beam}), y = 1 \]

...\[ x = (\text{happy}, \text{beam}), y = 0 \]
\[ x = (\text{think}, \text{beam}), y = 0 \]

...\[ x = (\text{hit}, \text{beam}), y = 1 \]
\[ x = (\text{the}, \text{beam}), y = 1 \]
\[ x = (\text{behind}, \text{beam}), y = 1 \]

\[ x = (\text{happy}, \text{beam}), y = 0 \]
\[ x = (\text{think}, \text{beam}), y = 0 \]

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 to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

\[ \alpha = 0.75 \]

\[ P_\alpha(w) = \frac{\text{count}(w)\alpha}{\sum_w \text{count}(w)\alpha} \]

\[ k \] negative example \((y=0)\) for every positive.

**How?** Randomly draw from unigram distribution adjusted:

\[ \text{The nail hit the beam behind the wall.} \]

(Jurafsky, 2017)
Word2Vec: Context

\[ p(\text{context} \mid \text{word}) \]

\[
x = (\text{hit, beam}), y = 1 \\
x = (\text{the, beam}), y = 1 \\
x = (\text{behind, beam}), y = 1 \\
\ldots \\
x = (\text{happy, beam}), y = 0 \\
x = (\text{think, beam}), y = 0 \\
\ldots
\]

\( k \) negative example (\( y=0 \)) for every positive.

How? Randomly draw from unigram distribution adjusted:

\[
P_\alpha(w) = \frac{\text{count}(w)^\alpha}{\sum_w \text{count}(w)^\alpha}
\]

\( \alpha = 0.75 \)

1. Treat the target word as positive.
2. Randomly draw from unigram distribution adjusted.
3. Use logistic regression.
4. Use the weight from regression.

(Jurafsky, 2017)
Word2Vec: Context

$ x = (\text{hit}, \text{beam}), y = 1$
$ x = (\text{the}, \text{beam}), y = 1$
$ x = (\text{behind}, \text{beam}), y = 1$

...$ x = (\text{happy}, \text{beam}), y = 0$
$ x = (\text{think}, \text{beam}), y = 0$

...$n$.

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

The nail hit the beam behind the wall.

(Jurafsky, 2017)
**Word2Vec: Context**

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

The nail hit the beam behind the wall.

---

**Logistic:** \[ \sigma(z) = \frac{1}{1 + e^{-z}} \]

- Single context:
  \[ P(y=1 \mid c, t) = \frac{1}{1 + e^{-t \cdot c}} \]
- All Contexts:
  \[ P(y=1 \mid c, t) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}} \]

---

(Jurafsky, 2017)
Word2Vec: Context

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

\[
P(y=1|\ c,\ t) = \frac{1}{1+e^{-t\cdot c}}
\]

Intuition: \(t\cdot c\) is a measure of similarity:
\[
a \cdot b = ||a|| \cdot ||b|| \cdot \cos \theta
\]
But, it is not a probability! To make it one, apply logistic activation:
\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]

The nail hit the beam behind the wall. (Jurafsky, 2017)
Word2Vec: Context

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
...
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

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 to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

The nail hit the beam behind the wall.

(Jurafsky, 2017)
Word2Vec: How to Learn?

P(y=1| c, t)

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

The nail hit the beam behind the wall.

(Jurafsky, 2017)
Word2Vec: How to Learn?

$P(y=1| c, t)$

Assume $300 \times |\text{vocab}|$ weights (parameters) for each of $c$ and $t$

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

The nail hit the beam behind the wall.

(Jurafsky, 2017)
Word2Vec: How to Learn?

\[ P(y=1|c, t) \]

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

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 to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

The nail hit the beam behind the wall.

(Jurafsky, 2017)
The nail hit the beam behind the wall.

\[ P(y=1| c, t) \]

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

Goal:
- Maximize similarity of (c, t) in positive data (y = 1)
- Minimize similarity of (c, t) in negative data (y = 0)
**Word2Vec: How to Learn?**

\[ P(y=1|c, t) \]

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

**Goal:**
- Maximize similarity of (c, t) in positive data (y = 1)
- Minimize similarity of (c, t) in negative data (y = 0)

\[
\sum_{(c,t)} (y) \log P(y = 1|c, t) + (y - 1) \log P(y = 0|c, t)
\]
Word2Vec: How to Learn?

\[ P(y=1|c, t) \]

Assume 300 * |vocab| weights (parameters) for each of c and t. Start with random vectors (or all 0s).

Goal:
- Maximize similarity of (c, t) in positive data (y = 1)
- Minimize similarity of (c, t) in negative data (y = 0)

\[
\sum_{(c,t)} (y) \log P(y = 1|c, t) + (y - 1) \log P(y = 0|c, t)
\]

\[
1 - P(y = 1|c, t) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}
\]
Word2Vec: How to Learn?

Assume $300 * |\text{vocab}|$ weights (parameters) for each of $c$ and $t$

Start with random vectors (or all 0s)

Goal:
- Maximize similarity of $(c, t)$ in positive data ($y = 1$)
- Minimize similarity of $(c, t)$ in negative data ($y = 0$)

$$\sum_{(c, t)} (y) \log P(y = 1|c, t) + (y - 1) \log P(y = 0|c, t)$$

$$1 - P(y = 1|c, t) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

Optimized using gradient descent type methods.
\[ \sum_{(c,t)} (y)\log P(y = 1|c, t) + (y - 1)\log P(y = 0|c, t) \]  

(Jurafsky, 2017)
Word2Vec captures analogies (kind of)

(Jurafsky, 2017)
Word2Vec: Quantitative Evaluations

Compare to manually annotated pairs of words: WordSim-353 (Finkelstein et al., 2002)

Compare to words in context (Huang et al., 2012)

Answer TOEFL synonym questions.
Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):
   
   *The nail hit the beam behind the wall.*
   *They reflected a beam off the moon.*

   ![Diagram showing different contextual uses of the word 'beam']
Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):
   - The nail hit the beam behind the wall.
   - They reflected a beam off the moon.

2. Embeddings can capture changes in word meaning.

(Kulkarni et al., 2015)
Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):
   - The nail hit the beam behind the wall.
   - They reflected a beam off the moon.

2. Embeddings can capture changes in word meaning.

3. Embeddings capture demographic biases in data. (Garg et al., 2018)
1. Contextual word embeddings (a different embedding depending on context):

   The nail hit the beam behind the wall.
   They reflected a beam off the moon.

2. Embeddings can capture changes in word meaning.

3. Embeddings capture demographic biases in data.
   a. Efforts to debias
      (Garg et al., 2018)
   b. Useful for tracking bias over time.
Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):
   - The nail hit the beam behind the wall.
   - They reflected a beam off the moon.

2. Embeddings can capture changes in word meaning.

3. Embeddings capture demographic biases in data.
   a. Efforts to debias
   b. Useful for tracking bias over time. (Garg et al., 2018)
Vector Semantics and Embeddings

Take-Aways

- Dense representation of meaning is desirable.
- Approach 1: Dimensionality reduction techniques
- Approach 2: Learning representations by trying to predict held-out words.
- Word2Vec skipgram model attempts to solve by predicting target word from context word:
  maximize similarity between true pairs; minimize similarity between random pairs.
- Embeddings do in fact seem to capture meaning in applications
- Dimensionality reduction techniques just as good by some evaluations.
- Current Trends: Integrating context, Tracking changes in meaning.