Word Vectors

CS114B Labs 6-7

Kenneth Lai

March 10-17, 2022

The Distributional Hypothesis

- "You shall know a word by the company it keeps." (Firth 1957)
- "It may be presumed that any two morphemes A and B having different meanings, also differ somewhere in distribution: there are some environments in which one occurs and the other does not." (Harris 1951)
- "The similarity of the contextual representations of two words contributes to the semantic similarity of those words." (Miller and Charles 1991) (emphasis mine)

because "meaning"of word is
difficult to mathematically model

The Distributional Hypothesis

- “You shall know a word by the company it keeps.” (Firth 1957)
- “It may be presumed that any two morphemes A and B having different meanings, also differ somewhere in distribution: there are some environments in which one occurs and the other does not.” (Harris 1951)
- “The similarity of the contextual representations of two words contributes to the semantic similarity of those words.” (Miller and Charles 1991) (emphasis mine)

- Words can be represented by (abstractions over) their contexts

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space

```
What is similarity?
Current best way to measure similarity
is to "embed" the word into some kind of
space, making them "vectors".
```

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space `Euclidean space`
- Two approaches to distributed, distributional representations (Baroni et al. 2014):

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based
    - Count occurrences of words in contexts, optionally followed by some mathematical transformation (e.g. tf-idf, PPMI, SVD)

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based
    - Count occurrences of words in contexts, optionally followed by some mathematical transformation (e.g. tf-idf, PPMI, SVD)
  - Prediction-based

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based
    - Count occurrences of words in contexts, optionally followed by some mathematical transformation (e.g. tf-idf, PPMI, SVD)
  - Prediction-based
    - Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
    - a.k.a. language modeling-based

Count-Based Word Vectors

- What are contexts?

Count-Based Word Vectors

- What are contexts?
    - Contexts are documents
        - Term-document matrices

Count-Based Word Vectors

- What are contexts?
  - Contexts are documents
    - Term-document matrices
  - Contexts are words (within some window)
    - Term-term matrices

```
Genre of document – type
Interword relations – context
```

Term-Document Matrices

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.3 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

- Document vectors: coordinates are counts of each word in the document

```
As documents can be classified into
limited amount of genre, so are words.
```

Term-Document Matrices

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.5 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.

- Word vectors: coordinates are counts of the word in each document

Term-Term Matrices

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

- Word vectors: coordinates are counts of times the row (target) word and the column (context) word co-occur in some context in some training corpus

Term-Term Matrices

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

- Word vectors: coordinates are counts of times the row (target) word and the column (context) word co-occur in some context in some training corpus
  - e.g., in a 4 word window (4 words to the left and 4 words to the right)

Primary, direct way to vector

Weighting Terms in Vectors

- Words that occur frequently in some contexts are important

```
Different weights for each context words (dimension)
```

Weighting Terms in Vectors

- Words that occur frequently in some contexts are important
  - But words that occur frequently in every context are not!

Weighting Terms in Vectors

`Some words are frequent in every context!`

- Words that occur frequently in some contexts are important
  - But words that occur frequently in every context are not!
- Term frequency-inverse document frequency (tf-idf)
  - Words that occur in most/all documents are less important
  - More useful for document vectors

Weighting Terms in Vectors

- Words that occur frequently in some contexts are important
  - But words that occur frequently in every context are not!
- Term frequency-inverse document frequency (tf-idf)
  - Words that occur in most/all documents are less important
  - More useful for document vectors
- Positive pointwise mutual information (PPMI)
  - How often do two words co-occur in some context, compared with what we would expect by chance?

```
PPMI(w,c) = log{P(w,c),0}{P(w)P(c)}
```

The Distributional Hypothesis

- "You shall know a word by the company it keeps." (Firth 1957)
- "It may be presumed that any two morphemes A and B having different meanings, also differ somewhere in distribution: there are some environments in which one occurs and the other does not." (Harris 1951)
- "The similarity of the contextual representations of two words contributes to the semantic similarity of those words." (Miller and Charles 1991) (emphasis mine)

- Similar words should have similar contexts

The Distributional Hypothesis

- "You shall know a word by the company it keeps." (Firth 1957)
- "It may be presumed that any two morphemes A and B having different meanings, also differ somewhere in distribution: there are some environments in which one occurs and the other does not." (Harris 1951)
- "The similarity of the contextual representations of two words contributes to the semantic similarity of those words." (Miller and Charles 1991) (emphasis mine)

- Similar words should have similar vectors

The Distributional Hypothesis

- "You shall know a word by the company it keeps." (Firth 1957)
- "It may be presumed that any two morphemes A and B having different meanings, also differ somewhere in distribution: there are some environments in which one occurs and the other does not." (Harris 1951)
- "The similarity of the contextual representations of two words contributes to the semantic similarity of those words." (Miller and Charles 1991) (emphasis mine)

- Similar words should have similar vectors
  - How to measure similarity (or distance) between vectors?

Euclidean Distance

- A normed vector space is a vector space with a norm

Euclidean Distance

- A normed vector space is a vector space with a norm
  - A norm generalizes the notion of “length”

Euclidean Distance

- A normed vector space is a vector space with a norm
  - A norm generalizes the notion of "length"
- On a real coordinate space, we can define the Euclidean norm

$$||\mathbf{x}|| = \sqrt{\sum_{j=1}^{n} x_j^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

Euclidean Distance

- A normed vector space is a vector space with a norm
  - A norm generalizes the notion of "length"
- On a real coordinate space, we can define the Euclidean norm

$$||\mathbf{x}|| = \sqrt{\sum_{j=1}^{n} x_j^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

- A norm induces a distance (induced metric)

Euclidean Distance

- A normed vector space is a vector space with a norm
  - A norm generalizes the notion of “length”
- On a real coordinate space, we can define the Euclidean norm

$$||\mathbf{x}|| = \sqrt{\sum_{j=1}^{n} x_j^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

- A norm induces a distance (induced metric)
- Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{y} - \mathbf{x}|| = \sqrt{\sum_{j=1}^{n} (y_j - x_j)^2}$$

Cosine Similarity

- Euclidean distance favors long vectors

Cosine Similarity

- Euclidean distance favors frequent words

Cosine Similarity

- Euclidean distance favors frequent words
- Consider the "angle" between two vectors, rather than distance

```
Vectors do not have the same lengths
But their directions are more decisive of meaning
```

Cosine Similarity

- Euclidean distance favors frequent words
- Consider the "angle" between two vectors, rather than distance
- Cosine similarity

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||}$$

Sparse and Dense Vectors

- tf-idf and PPMI vectors are long and sparse

Sparse and Dense Vectors

- tf-idf and PPMI vectors are long and sparse
- We want to learn vectors that are short and dense

```
We need to reduce the dimensions!
```

Principal Component Analysis

- Idea: Given a matrix $\mathbf{X}$ (where rows are vectors), apply a linear map $\mathbf{V}$
$$\mathbf{T} = \mathbf{X} \cdot \mathbf{V}$$

Principal Component Analysis

- Idea: Given a matrix $\mathbf{X}$ (where rows are vectors), apply a linear map $\mathbf{V}$
$$\mathbf{T} = \mathbf{X} \cdot \mathbf{V}$$
- If we choose $\mathbf{V}$ correctly, then $\mathbf{T}$ is a projection of $\mathbf{X}$ onto a new coordinate space, where the first coordinate (called the first principal component) has the greatest variance, the second coordinate has the second greatest variance, etc.

Principal Component Analysis

- Idea: Given a matrix $\mathbf{X}$ (where rows are vectors), apply a linear map $\mathbf{V}$
$$\mathbf{T} = \mathbf{X} \cdot \mathbf{V}$$
- If we choose $\mathbf{V}$ correctly, then $\mathbf{T}$ is a projection of $\mathbf{X}$ onto a new coordinate space, where the first coordinate (called the first principal component) has the greatest variance, the second coordinate has the second greatest variance, etc.
  - (Specifically, the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{X}^T \cdot \mathbf{X}$)

```
First dimension varies the most, second
dimension the second…
```

Principal Component Analysis

- Then we can truncate our new vectors (rows of $\mathbf{T}$) to dimension L by keeping only the first L principal components

$$\mathbf{T}_L = \mathbf{X} \cdot \mathbf{V}_L$$

X: (1000,10) VL(10, 5) -> TL (1000, 5)

Principal Component Analysis

- Then we can truncate our new vectors (rows of $\mathbf{T}$) to dimension L by keeping only the first L principal components

$$\mathbf{T}_L = \mathbf{X} \cdot \mathbf{V}_L$$

- Our new vectors (rows of $\mathbf{T}_L$) are short and dense, but retain much of the variance in the original vectors

Singular Value Decomposition

- As it turns out, we can get $\mathbf{V}$ (technically $\mathbf{V}^T$) from the singular value decomposition (SVD) of $\mathbf{X}$

$$\mathbf{X} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T$$

$Am\times n=Um\times m\Sigma m\times nVTn\times n\approx Um\times k\Sigma k\times kVTk\times n$

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based
    - Count occurrences of words in contexts, optionally followed by some mathematical transformation (e.g. tf-idf, PPMI, SVD)
  - Prediction-based
    - Given some context vector(s) **c**, predict some word **x** (or vice versa)
    - a.k.a. language modeling-based

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)

```
Usually some neighboring words
```

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$
    - Examples: n-gram language models

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$
    - Examples: n-gram language models
      - Unigram: predict $P(\mathbf{x}_i)$
      - Bigram: predict $P(\mathbf{x}_i|\mathbf{x}_{i-1})$
      - Trigram: predict $P(\mathbf{x}_i|\mathbf{x}_{i-2}, \mathbf{x}_{i-1})$

```
We have information before the task and the
information can be used to perform the task
```

Language Models

- Given some context vector(s) **c**, predict some word **x** (or vice versa)
- Two approaches to language models:
  - Discriminative models

```
We have to learn from a certain set of data
```

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly
    - Examples: neural network language models

Language Models

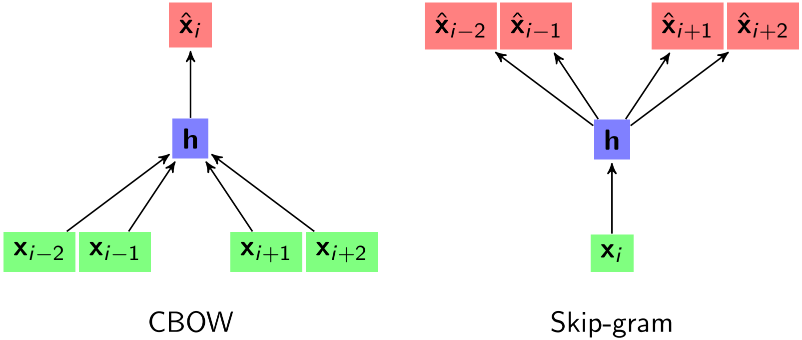
- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly
    - Examples: neural network language models
      - Feedforward: word2vec (Mikolov et al. 2013a, 2013b)
      - Recurrent:  (Peters et al. 2018)
      - Transformer:  (Devlin et al. 2019)

word2vec

- Based on a feedforward neural network language model

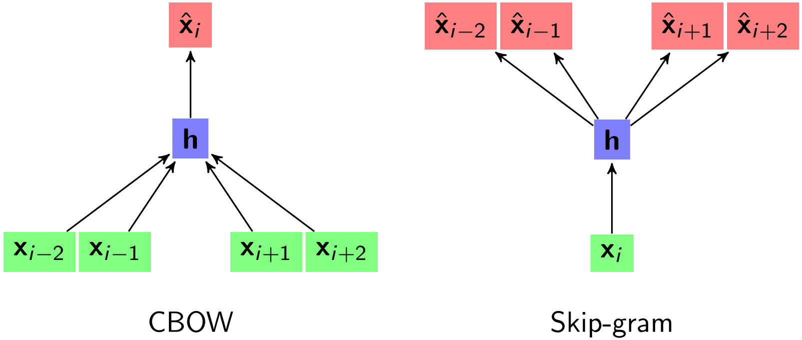
word2vec

- Based on a feedforward neural network language model



word2vec

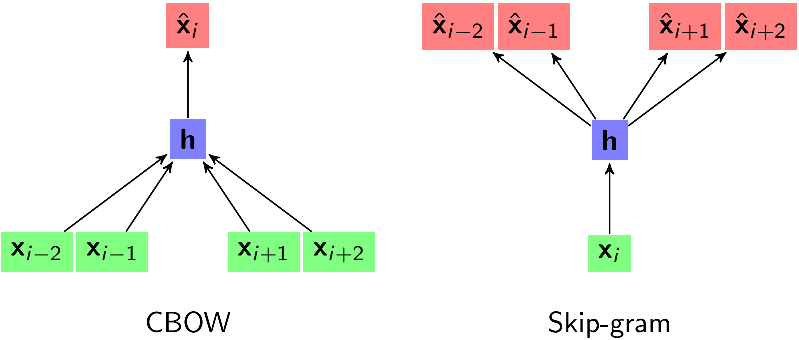
- Based on a feedforward neural network language model



- Continuous bag of words (CBOW): use context to predict current word

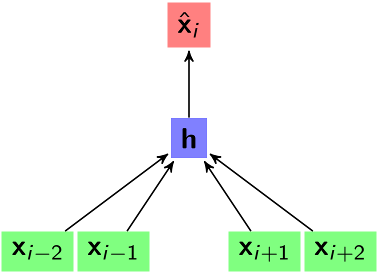
word2vec

- Based on a feedforward neural network language model



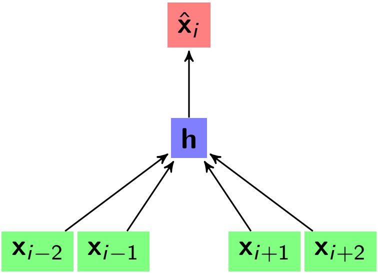
- Continuous bag of words (CBOW): use context to predict current word
- Skip-gram: use current word to predict context

CBOW



- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$

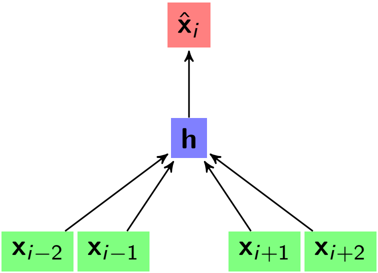
CBOW



- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$
  - Context words within some window

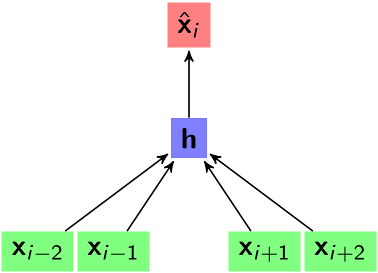
```
or sum of multiple word vectors
```

CBOW



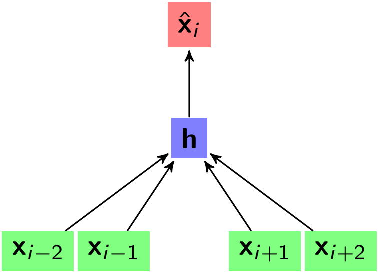
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words

CBOW



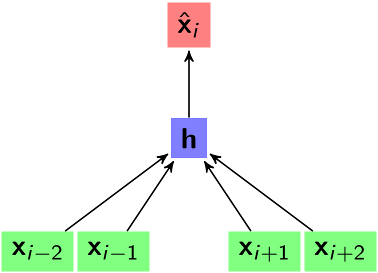
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)

CBOW



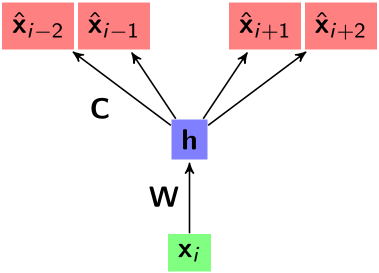
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)
  - Context word vectors are averaged

CBOW



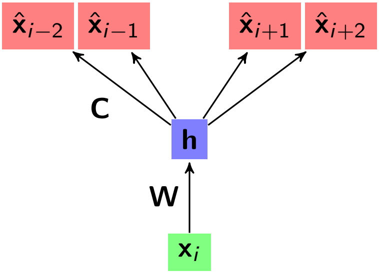
- Output layer: softmax activation function
  - Numbers $\rightarrow$ probabilities

Skip-gram



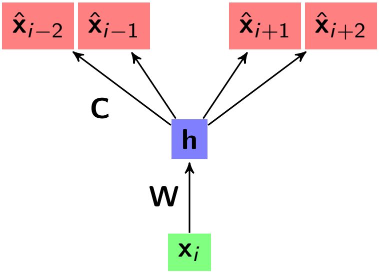
- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$

Skip-gram



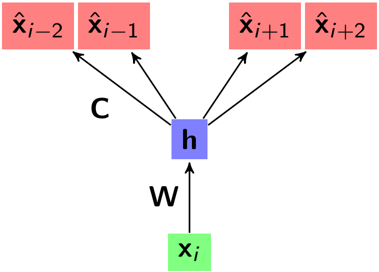
- Hidden (projection) layer: identity activation function, no bias
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)

Skip-gram



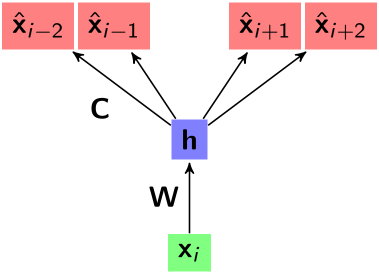
- Output layer: softmax activation function
  - Predict context words within some window

Skip-gram



- Output layer: softmax activation function
  - Predict context words within some window
  - Separate classification for each context word

Skip-gram



- Output layer: softmax activation function
  - Predict context words within some window
  - Separate classification for each context word
  - Closer context words sampled more than distant context words

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word

Negative Sampling

- Skip-gram model: let **w** be a target word vector and **c** be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Regular cross-entropy loss

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Regular cross-entropy loss
  - Maximize the (log-)probability of positive samples (true context words)

Negative Sampling

- Skip-gram model: let **w** be a target word vector and **c** be a context word vector
  - **w** = row of **W** corresponding to input word (= output of hidden layer)
  - **c** = column of **C** corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Minimize the (log-)probability of k negative samples (random non-context words)

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings
  - Target word vector $\mathbf{w}$ (row of $\mathbf{W}$, $=$ output of hidden layer)
  - Context word vector $\mathbf{c}$ (column of $\mathbf{C}$)

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings
  - Target word vector $\mathbf{w}$ (row of $\mathbf{W}$, = output of hidden layer)
  - Context word vector $\mathbf{c}$ (column of $\mathbf{C}$)
- Common final word embeddings
  - Add $\mathbf{w} + \mathbf{c}$
  - Just $\mathbf{w}$ (throw away $\mathbf{c}$)

References

- Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors." Proceedings of ACL. 2014.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT. 2019.
- Firth, John R. "A synopsis of linguistic theory, 1930-1955." Studies in linguistic analysis (1957).
- Harris, Zellig S. Methods in structural linguistics. 1951.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." Proceedings of ICLR. 2013.
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality." Proceedings of NeurIPS. 2013.
- Miller, George A., and Walter G. Charles. "Contextual correlates of semantic similarity." Language and cognitive processes 6.1 (1991): 1-28.
- Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." Proceedings of NAACL-HLT. 2018.