Natural Language Processing (NLP) and Large Language Models (LLMs)
Lecture 2-2: Word2Vec

Chendi Wang (王晨笛)
chendi.wang@xmu.edu.cn

WISE @ XMU

2025 年 3 月 4 日

Slides adapted in part from Stanford CS224n: Natural Language Processing and 《动手学深度学习》

Tutorials

- https://www.bilibili.com/video/BV1eHPjeqE3c/

The idea of Word2Vec (Mikolov et al. 2013)

- For each position t in the text, identify a center word (c) and its context words (o).
- Use the similarity between word vectors of c and o to calculate the probability of o given c (or vice versa).
- Updating the vectors to maximize the probability.

An example of a window

- Fix a window size of 2 and consider the sentence

  I have an adorable cat named Yingjun because he is incredibly handsome.

- For the center word cat, the corresponding context words (with window size 2) are "an adorable" and "named Yingjun".

Two Word2Vec models (Mikolov et al., 2013)

- The CBOW (Continuous bag of words) architecture predicts the center word based on the context
- The Skip-gram predicts context words given the center word.

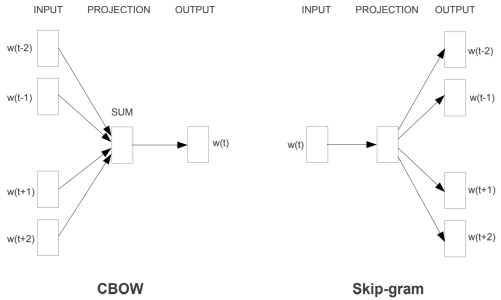

Figure: An illustration of the CBOW model and the skip-gram model..

The figure is from Efficient Estimation of Word Representations in Vector Space

Code for 2 models

- Recall the previous Word2Vec model

```
from gensim.models import Word2Vec
model = Word2Vec(sentences=tokenized_paragraphs,
vector_size=100, window=5, min_count=1, workers=4)
```

- This is for the CBOW model.

Code for 2 models

- Recall the previous Word2Vec model

```
from gensim.models import Word2Vec
model = Word2Vec(sentences=tokenized_paragraphs,
vector_size=100, window=5, min_count=1, workers=4)
```

- This is for the CBOW model.
- For skip-gram, we use

```
model = Word2Vec(sentences=tokenized_paragraphs,
vector_size=100, window=5, min_count=1, workers=4, sg=2)
```

Embeddings trained by different models are different

```
Embedding for 'professor': [-0.00520926 -0.00628066 -0.00812961  0.00816755 -0.00170414 -0.00753981
 -0.00364565  0.0059722  -0.00326475 -0.00431022  0.00131458 -0.00371828
 -0.00164022  0.00143599 -0.00274969  0.00795382  0.00385163 -0.01032443
  0.00596917 -0.00780076  0.00120682  0.00466636 -0.00487019 -0.00246929
  0.00861683 -0.00448501 -0.0080286   0.00890752 -0.00256834 -0.00496423
  0.00885705  0.00460832  0.004871    0.00848894 -0.008644    0.00571229
  0.00208124  0.00408743  0.00173661  0.0039902   0.00462527  0.00590951
 -0.00350244 -0.0047707  -0.00023781  0.00228568 -0.00353596  0.00572596
  0.0045715   0.00833482  0.00282669  0.0076673  -0.00139048  0.00815134
  0.00394601 -0.00773727 -0.00372471 -0.00229705  0.00468275 -0.00039886
 -0.00242086  0.00745917  0.00980032 -0.00154442 -0.00528267 -0.00422345
 -0.00487744 -0.0092963   0.00042248 -0.00352166  0.00222845  0.00596103
 -0.00348641 -0.00959224  0.00182645 -0.0068563   0.00256692 -0.00379669
  0.00702295  0.00078641  0.00316962 -0.00252646 -0.00248253  0.00808141
  0.00108319 -0.00624198 -0.00676784  0.00146691  0.00706652  0.00579118
 -0.00848786  0.00909958  0.00412425  0.00776276  0.0105122  -0.00688115
 -0.00902423  0.00585236  0.00939178  0.00338649]
Shape of embedding: (100,)
```

The embedding of the word “professor” for the skip-gram model

A probabilistic model

- For a center word x_t and its context word x_{t+j}, predicting the context word using the center word is to compute the conditional probability $p(x_{t+j} \mid x_t)$.

A probabilistic model

- For a center word x_t and its context word x_{t+j}, predicting the context word using the center word is to compute the conditional probability $p(x_{t+j} \mid x_t)$.

A probabilistic model

- For a center word x_t and its context word x_{t+j}, predicting the context word using the center word is to compute the conditional probability $p(x_{t+j} \mid x_t)$.
- Objective: Find p that maximizes the likelihood.

A probabilistic model

- For a center word x_t and its context word x_{t+j}, predicting the context word using the center word is to compute the conditional probability $p(x_{t+j} \mid x_t)$.
- Objective: Find p that maximizes the likelihood.
- What if our objective is to predict the center word using the context word?

Two Questions

- How to parameterize p?
- How to find p that maximizes the likelihood?

Different word vectors for different roles of a word

- Each word x_t can be represented by two distinct vectors.

Different word vectors for different roles of a word

- Each word x_t can be represented by two distinct vectors.
- A center word vector $v_t \in \mathbb{R}^p$ represents x_t when it acts as the center word.

Different word vectors for different roles of a word

- Each word x_t can be represented by two distinct vectors.
- A center word vector $v_t \in \mathbb{R}^p$ represents x_t when it acts as the center word.
- A context word vector $u_t \in \mathbb{R}^p$ represents x_t when it appears as a context word.

Different word vectors for different roles of a word

- Each word x_t can be represented by two distinct vectors.
- A center word vector $v_t \in \mathbb{R}^p$ represents x_t when it acts as the center word.
- A context word vector $u_t \in \mathbb{R}^p$ represents x_t when it appears as a context word.
- The final embedding of x_t is usually obtained by averaging u_t and v_t.

Conditional probability

- Consider a center word x_c (with word vector v_c) and a context ("outside") word x_o (with word vector u_o).
- The skip-gram model aims to estimate the conditional probability $p(x_o \mid x_c)$.

Conditional probability

- Consider a center word x_c (with word vector v_c) and a context ("outside") word x_o (with word vector u_o).
- The skip-gram model aims to estimate the conditional probability $p(x_o \mid x_c)$.
- Recall that if the word vectors are normalized (i.e., have unit length), the similarity between x_c and x_o can be measured by the dot product $u_o^T v_c$.

Conditional probability

- Consider a center word x_c (with word vector v_c) and a context ("outside") word x_o (with word vector u_o).
- The skip-gram model aims to estimate the conditional probability $p(x_o \mid x_c)$.
- Recall that if the word vectors are normalized (i.e., have unit length), the similarity between x_c and x_o can be measured by the dot product $u_o^T v_c$.
- We define the conditional probability based on this similarity as

$$p(x_o \mid x_c) = \frac{\exp(u_o^T v_c)}{\sum_{i \in \mathcal{V}} \exp(u_i^T v_c)},$$

where $\mathcal{V}$ is the vocabulary (the set of all possible words).

SoftMax Function

- The $\mathrm{SoftMax}$ function is used to convert a sequence of real numbers into a probability distribution.
- Mathematically, the SoftMax function $\mathrm{SoftMax} : \mathbb{R}^n \to (0,1)^n$ maps any sequence $(a_i)_{i=1}^n \in \mathbb{R}^n$ to a probability distribution $(b_i)_{i=1}^n$ given by

$$b_i = \frac{\exp(a_i)}{\sum_{j=1}^n \exp(a_j)}.$$

SoftMax Function

- The $\mathrm{SoftMax}$ function is used to convert a sequence of real numbers into a probability distribution.
- Mathematically, the SoftMax function $\mathrm{SoftMax} : \mathbb{R}^n \to (0,1)^n$ maps any sequence $(a_i)_{i=1}^n \in \mathbb{R}^n$ to a probability distribution $(b_i)_{i=1}^n$ given by

$$b_i = \frac{\exp(a_i)}{\sum_{j=1}^n \exp(a_j)}.$$

- Soft: The output is a probability distribution.

SoftMax Function

- The $\mathrm{SoftMax}$ function is used to convert a sequence of real numbers into a probability distribution.
- Mathematically, the SoftMax function $\mathrm{SoftMax} : \mathbb{R}^n \to (0,1)^n$ maps any sequence $(a_i)_{i=1}^n \in \mathbb{R}^n$ to a probability distribution $(b_i)_{i=1}^n$ given by

$$b_i = \frac{\exp(a_i)}{\sum_{j=1}^n \exp(a_j)}.$$

- Soft: The output is a probability distribution.
- Max: It amplifies the probability of the largest a_i relative to the others. (e.g., $exp(6)/(exp(6) + exp(4)) \approx 0.88 > 0.6$)

A probabilistic model for the the whole sequence

- For a sequence $\{x_t\}_{t=1}^T$ of length T, for each center word x_t, we aim to estimate the conditional probabilities

$$p(x_{t+j}|x_t) = \frac{\exp(u_{t+j}^T v_t)}{\sum_{i \in \mathcal{V}} \exp(u_i^T v_t)}$$

  for all integer $j \neq 0$ in the range $[-M, M]$.
- Here M is the window size.

A probabilistic model for the the whole sequence

- For a sequence $\{x_t\}_{t=1}^T$ of length T, for each center word x_t, we aim to estimate the conditional probabilities

$$p(x_{t+j}|x_t) = \frac{\exp(u_{t+j}^T v_t)}{\sum_{i\in\mathcal{V}} \exp(u_i^T v_t)}$$

  for all integer $j \neq 0$ in the range $[-M, M]$.
- Here M is the window size.
- In machine learning, to model the probability p, we usually introduce a parameterization of p_θ for some parameter $\theta \in \mathbb{R}^d$.
- In this model, the parameter θ consists all vectors u_t and v_t, that is

$$\theta = (u_t, v_t)_{t\in T} \in \mathbb{R}^{2Tp}.$$

Maximum likelihood estimate for Word2Vec

- To train the parameter θ, we consider a maximum likelihood estimate (MLE) method.
- Consider a sequence of $\{x_t\}_{t=1}^{T}$ of length T. The likelihood function is given by

$$L(\theta) = \prod_{t=1}^{T} \prod_{-M \le j \le M, j \neq 0} p_\theta(x_{t+j} \mid x_t).$$

Maximum likelihood estimate for Word2Vec

- To train the parameter θ, we consider a maximum likelihood estimate (MLE) method.
- Consider a sequence of $\{x_t\}_{t=1}^{T}$ of length T. The likelihood function is given by

$$L(\theta) = \prod_{t=1}^{T} \prod_{-M \le j \le M, j \neq 0} p_\theta(x_{t+j} \mid x_t).$$

- The goal of maximum likelihood estimation is to find a parameter $\widehat{\theta}$ that maximize the likelihood function, that is,

$$L(\widehat{\theta}) = \max_{\theta \in \mathbb{R}^d} L(\theta)$$

Maximum likelihood estimate for Word2Vec

- To train the parameter θ, we consider a maximum likelihood estimate (MLE) method.
- Consider a sequence of $\{x_t\}_{t=1}^{T}$ of length T. The likelihood function is given by

$$L(\theta) = \prod_{t=1}^{T} \prod_{-M \le j \le M, j \neq 0} p_\theta(x_{t+j} \mid x_t).$$

- The goal of maximum likelihood estimation is to find a parameter $\widehat{\theta}$ that maximize the likelihood function, that is,

$$L(\widehat{\theta}) = \max_{\theta \in \mathbb{R}^d} L(\theta)$$

- The minimizer $\widehat{\theta}$ is not necessarily unique. (Under what conditions is it unique?)

Log-likelihood function

- The product in the likelihood function is computationally challenging to handle directly.
- Converting the product into a summation simplifies the problem significantly. (Why? Because it becomes linear in the gradient, making optimization more efficient.)

Log-likelihood function

- The product in the likelihood function is computationally challenging to handle directly.
- Converting the product into a summation simplifies the problem significantly. (Why? Because it becomes linear in the gradient, making optimization more efficient.)
- To achieve this, we consider the log-likelihood function:

$$\log L(\theta) = \sum_{t=1}^{T} \sum_{-M \leq j \leq M} \log p_\theta(x_{t+j} \mid x_t)$$

Objective (Loss) Function

- In optimization, we commonly convert a maximization problem into a minimization problem.
- The objective function (or loss function in machine learning) is defined as:

$$\ell(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M}\log p_\theta(x_{t+j} \mid x_t)$$

- $L(\widehat{\theta}) = \max L(\theta) \Leftrightarrow \ell(\widehat{\theta}) = \min \ell(\theta)$.

Summary

- The skip-gram model estimates the conditional probability of a context word given the center word using a softmax function based on word similarity.
- The objective is to maximize the likelihood of the entire sentence, expressed as the product of all conditional probabilities.
- This maximization is reformulated as minimizing a corresponding loss function.

Gradient descent (idea)

- We aim to minimize a loss function $\ell(\theta)$.
- Gradient descent (GD) is one of the most widely used optimization methods for minimizing $\ell(\theta)$.
- From a random initialization $\theta = \theta_0$, we update θ by moving in the direction of the negative gradient $-\nabla_\theta \ell(\theta)$. (The idea is from 1st-order Taylor's expansion)

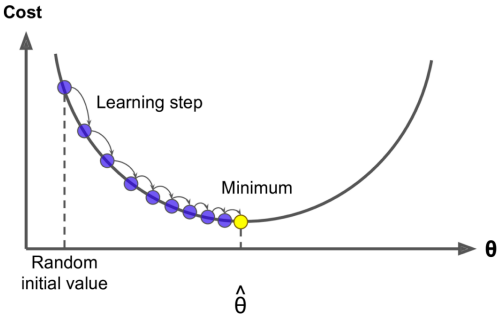

Figure: Using GD to minimize a quadratic.

Figure is from Stanford CS224n.

Gradient Descent (Math)

- Starting from a random initialization $\theta^{(0)}$, iterate as follows:

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla \ell(\theta^{(k)}),$$

where η_k is the stepsize (or learning rate).

Gradient Descent (Math)

- Starting from a random initialization $\theta^{(0)}$, iterate as follows:

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla \ell(\theta^{(k)}),$$

  where η_k is the stepsize (or learning rate).
- When the loss function is (strongly) convex, we have

$$\ell(\theta^{(k+1)}) \leq \ell(\theta^{(k)}),$$

  and as k increases, the algorithm converges to $\min_\theta \ell(\theta)$.
- Life is not always convex, but gradient descent may still work for non-convex loss functions (although we still don't fully understand why).

Learning rate matters

- The learning rate is (one of) the most critical hyperparameter(s) for ensuring fast convergence in gradient descent.

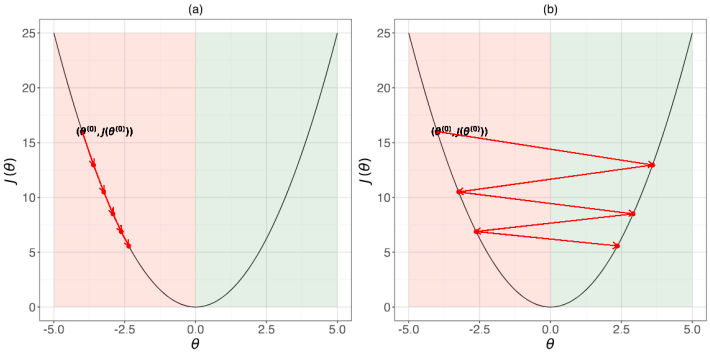

Figure: An example of using GD to minimize $\ell(\theta) = \theta^2$. (a) uses a learning rate of $\eta_k \equiv 0.05$, while (b) uses a learning rate of $\eta_k \equiv 0.95$.

Figure is from Zhonglei Wang.

Train the skip-gram model using GD

- Recall the loss function

$$\ell(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0} \log p_\theta(x_{t+j} \mid x_t)$$

  for the skip gram-model.

- Since the gradient operator is linear, we have

$$\nabla\ell(\theta) = \nabla_\theta\left[-\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0} \log p_\theta(x_{t+j} \mid x_t)\right] = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0} \nabla \log p_\theta(x_{t+j} \mid x_t)$$

Train the skip-gram model using GD

- Recall the loss function

$$\ell(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0}\log p_\theta(x_{t+j} \mid x_t)$$

for the skip gram-model.

- Since the gradient operator is linear, we have

$$\nabla\ell(\theta) = \nabla_\theta\left[-\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0}\log p_\theta(x_{t+j} \mid x_t)\right] = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-M\le j\le M, j\neq 0}\nabla\log p_\theta(x_{t+j} \mid x_t)$$

- Since $\theta = (u_t, v_t)_{t=1}^{T}$, we have

$$\nabla\log p_\theta(x_{t+j}|x_t) = \left(\frac{\partial\log p_\theta(x_{t+j}|x_t)}{\partial u_t}, \frac{\partial\log p_\theta(x_{t+j}|x_t)}{\partial v_t}\right)_{t=1}^{T}.$$

Differentiation (calculate it yourself)

- Note that

$$\log p_\theta(x_o|x_c) = u_o^T v_c - \log \sum_{i \in \mathcal{V}} \exp(u_i^T v_c).$$

- Calculate

$$\frac{\partial \log p_\theta(x_o|x_c)}{\partial v_c}.$$

Differentiation (calculate it yourself)

- Note that

$$\log p_\theta(x_o|x_c) = u_o^T v_c - \log \sum_{i \in \mathcal{V}} \exp(u_i^T v_c).$$

- Calculate

$$\frac{\partial \log p_\theta(x_o|x_c)}{\partial v_c}.$$

- Result:

$$\frac{\partial \log p_\theta(x_o|x_c)}{\partial v_c} = u_o - \sum_{j \in \mathcal{V}} p_\theta(x_j|x_c) u_j.$$

A more efficient way minimize the loss function

- GD requires calculating the gradient of all windows in each iteration, which is time consuming.
- In each iteration, can we just calculate a small part of all windows?

Stochastic gradient descent (SGD)

- Note that the loss function has a special form

$$\ell(\theta) = \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t), \qquad \text{for some function } f.$$

  Here z_t represents a window centered at x_t.

- Rather than using the full gradient $\nabla \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t)$, we consider the gradient of a small subset of windows $\frac{1}{b}\sum_{z_t \in \mathcal{B}} f(\theta, z_t)$, where $\mathcal{B} \subseteq \{z_t\}_{t=1}^{T}$ of size $b < n$.

Stochastic gradient descent (SGD)

- Note that the loss function has a special form

$$\ell(\theta) = \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t), \qquad \text{for some function } f.$$

  Here z_t represents a window centered at x_t.

- Rather than using the full gradient $\nabla \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t)$, we consider the gradient of a small subset of windows $\frac{1}{b}\sum_{z_t \in \mathcal{B}} f(\theta, z_t)$, where $\mathcal{B} \subseteq \{z_t\}_{t=1}^{T}$ of size $b < n$.
- In each iteration, we update

$$\theta^{(k+1)} = \theta^{(k)} - \frac{\eta_k}{b}\sum_{z_t \in \mathcal{B}_k} \nabla f(\theta^{(k)}, z_t),$$

  with $\mathcal{B}_k$ drawn uniformly at random from $\{z_t\}_{t=1}^{T}$.

Stochastic gradient descent (SGD)

- Note that the loss function has a special form

$$\ell(\theta) = \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t), \qquad \text{for some function } f.$$

  Here z_t represents a window centered at x_t.
- Rather than using the full gradient $\nabla \frac{1}{T}\sum_{t=1}^{T} f(\theta, z_t)$, we consider the gradient of a small subset of windows $\frac{1}{b}\sum_{z_t \in \mathcal{B}} f(\theta, z_t)$, where $\mathcal{B} \subseteq \{z_t\}_{t=1}^{T}$ of size $b < n$.
- In each iteration, we update

$$\theta^{(k+1)} = \theta^{(k)} - \frac{\eta_k}{b}\sum_{z_t \in \mathcal{B}_k} \nabla f(\theta^{(k)}, z_t),$$

  with $\mathcal{B}_k$ drawn uniformly at random from $\{z_t\}_{t=1}^{T}$.
- It converges to the minimizer (when the loss is convex) since $\frac{1}{b}\sum_{z_t \in \mathcal{B}_k} \nabla f(\theta, z_t)$ is an unbiased estimate of $\nabla \ell(\theta)$.

CBOW

- Recall that the CBOW model is to predict the center word based on the context.

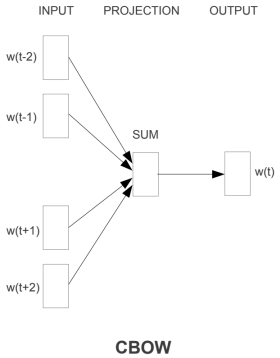

Figure is from Efficient Estimation of Word Representations in Vector Space.

Average the similarity

- Consider a center word x_c (with word vector v_c) and a set of it context words $\mathcal{X}_o = \{x_{o_1}, \cdots, x_{o_{2M}}\}$ (with word vector u_{o_m} for each x_{o_m}, correspondingly).
- The window size is M, so we have $2M$ context words.

Average the similarity

- Consider a center word x_c (with word vector v_c) and a set of it context words $\mathcal{X}_o = \{x_{o_1}, \cdots, x_{o_{2M}}\}$ (with word vector u_{o_m} for each x_{o_m}, correspondingly).
- The window size is M, so we have $2M$ context words.
- The similarity between x_c and each x_{o_m} is then given by $v_c^T u_{o_m}$.
- We then consider the averaged similarity

$$\frac{1}{2M}\sum_{m=1}^{2M} v_c^T u_{o_m} = v_c^T \left(\frac{1}{2M}\sum_{m=1}^{2M} u_{o_m}\right)$$

Probabilistic model

- Recall the averaged similarity $c_c^T \overline{u}_o$ with $\overline{u}_o = \frac{1}{2M} \sum_{m=1}^{2M} u_{o_m}$.
- Similar to the skip-gram model, we aim to estimate $p(x_c|\mathcal{X}_o) = p(x_c|x_{o_1}, \cdots x_{o_{2M}})$.

Probabilistic model

- Recall the averaged similarity $c_c^T \overline{u}_o$ with $\overline{u}_o = \frac{1}{2M} \sum_{m=1}^{2M} u_{o_m}$.
- Similar to the skip-gram model, we aim to estimate $p(x_c|\mathcal{X}_o) = p(x_c|x_{o_1}, \cdots x_{o_{2M}})$.
- Using the SoftMax function, we model the conditional probability as

$$p(x_c|\mathcal{X}_o) = \frac{\exp(v_c^T \overline{u}_o)}{\sum_i \exp(v_i^T \overline{u}_o)}.$$

Loss function

- The likelihood function for the CBOW model is then given by

$$L(\theta) = \prod_{t=1}^{T} p\left(x_t | x_{t+j}, -M \leq j \leq M, j \neq 0\right).$$

- And the loss function is

$$\ell(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \log p\left(x_t | x_{t+j}, -M \leq j \leq M, j \neq 0\right).$$

Gradient of the loss function

- To use the gradient descent, we consider each window

$$\log p(x_c|\mathcal{X}_o) = v_c^T \bar{u}_o - \log\left(\sum_j \exp(v_j^T \bar{u}_o)\right).$$

- Calculate the gradient (for each block)

$$\frac{\partial \log p(x_c|\mathcal{X}_o)}{\partial u_{o_m}}$$

Gradient of the loss function

- To use the gradient descent, we consider each window

$$\log p(x_c|\mathcal{X}_o) = v_c^T \bar{u}_o - \log\left(\sum_j \exp(v_j^T \bar{u}_o)\right).$$

- Calculate the gradient (for each block)

$$\frac{\partial \log p(x_c|\mathcal{X}_o)}{\partial u_{o_m}}$$

- The result is

$$\frac{\partial \log p(x_c|\mathcal{X}_o)}{\partial u_{o_m}} = \frac{1}{2M}\left[v_c - \sum_j p(x_j|\mathcal{X}_o) v_j\right]$$

Motivation

- Both the skip-gram and CBOW models involve a gradient computation that requires summing over the entire vocabulary $\mathcal{V}$:

$$p(x_o \mid x_c) = \frac{\exp(u_o^T v_c)}{\sum_{i \in \mathcal{V}} \exp(u_i^T v_c)}$$

- This summation over a large vocabulary results in significant computational costs during training.

Motivation

- Both the skip-gram and CBOW models involve a gradient computation that requires summing over the entire vocabulary $\mathcal{V}$:

$$p(x_o \mid x_c) = \frac{\exp(u_o^T v_c)}{\sum_{i \in \mathcal{V}} \exp(u_i^T v_c)}$$

- This summation over a large vocabulary results in significant computational costs during training.
- To address this, we introduce approximate training methods for the skip-gram model to reduce computational cost (similar methods apply to the CBOW model).

Negative Sampling (idea)

- We regard "any context word x_o is from the context window with center word x_c" as an event.
- Define a binary random variable $D \in \{0, 1\}$ as follows:
  - The probability x_o and x_c are in the same window is $\mathbb{P}[D = 1|x_o, x_c]$.

Negative Sampling (idea)

- We regard "any context word x_o is from the context window with center word x_c" as an event.
- Define a binary random variable $D \in \{0, 1\}$ as follows:
  - The probability x_o and x_c are in the same window is $\mathbb{P}[D = 1|x_o, x_c]$.
- Parameterization: $\mathbb{P}[D = 1|x_o, x_c] = \sigma(u_o^T v_c)$ for some activation function σ.
- In Word2Vec, the activation function is usually the sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

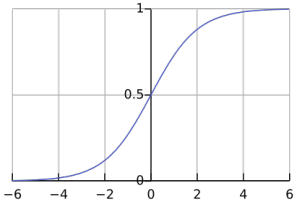

Figure: sigmoid function

Negative Sampling

- Let P be a predefined distribution over the vocabulary.
- We randomly sample K words from P that are not in the context window of x_c.
- These sampled words, denoted as $\{x^{(k)}\}_{k=1}^{K} \sim P$, are called noise words.

Negative Sampling

- Let P be a predefined distribution over the vocabulary.
- We randomly sample K words from P that are not in the context window of x_c.
- These sampled words, denoted as $\{x^{(k)}\}_{k=1}^{K} \sim P$, are called noise words.
- Let $u^{(k)}$ be the word vector (as a context word) corresponding to $x^{(k)}$.
- Since these words are not in the context of x_o, we consider the probability of an exclusive event:

$$\mathbb{P}[D = 0 \mid x_o, x^{(k)}] = 1 - \mathbb{P}[D = 1 \mid x_c, x^{(k)}] = 1 - \sigma((u^{(k)})^T v_c).$$

Probabilistic model

- Using negative sampling, the conditional probability is modeled by

$$p(x_o|x_c) = \mathbb{P}[D = 1|x_o, x_c] \cdot \prod_{k=1}^{K} \mathbb{P}[D = 0|x_c, x^{(k)}]$$

Probabilistic model

- Using negative sampling, the conditional probability is modeled by

$$p(x_o|x_c) = \mathbb{P}[D = 1|x_o, x_c] \cdot \prod_{k=1}^{K} \mathbb{P}[D = 0|x_c, x^{(k)}]$$

- The likelihood function becomes

$$L(\theta) = \prod_{t=1}^{T} \prod_{-M \leq j \leq M, j \neq 0} p(x_{t+j} \mid x_t).$$

Why is negative sampling efficient?

- The loss function can be expressed as (derive it yourself):

$$-\log p(x_{t+j} \mid x_t) = -\log \sigma(u_t^T v_t) - \sum_{k=1}^{K} \log \sigma(-(u^{(k)})^T v_t).$$

- This formulation involves a summation over only K words instead of the entire vocabulary.
- When K is small, the computational cost is low.

Co-occurrence Matrix

- Consider the following example corpus:

I love you tons.
I love you 3000.
I am Iron Man.

- The frequency of co-occurrence (with a window size of 1) can be represented in a (symmetric) co-occurrence matrix W which captures some features of the global corpus.

	I	love	you	am	tons	$\cdots$
I	0	2	0	1	0	$\cdots$
love	2	0	2	0	0	$\cdots$
you	0	2	0	0	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Co-occurrence Matrix (Math)

- Consider a word x_i that may occur multiple times in the corpus.
- Let $\mathcal{C}_i$ be a multiset (a set where repeated elements are allowed) that contains all the context words from the window when the center word is x_i.
- For example, $\mathcal{C}_i = \{x_j, x_j, x_k, x_k, x_k\}$, where x_k appears three times.

Co-occurrence Matrix (Math)

- Consider a word x_i that may occur multiple times in the corpus.
- Let $\mathcal{C}_i$ be a multiset (a set where repeated elements are allowed) that contains all the context words from the window when the center word is x_i.
- For example, $\mathcal{C}_i = \{x_j, x_j, x_k, x_k, x_k\}$, where x_k appears three times.
- The co-occurrence matrix $W = (w_{s,t})_{s,t}$ is such that $w_{s,t}$ represents the multiplicity of x_t in the multiset $\mathcal{C}_s$.
- In the example above, we have $w_{ij} = 2$ and $w_{ik} = 3$.

Skip-gram Model with Global Corpus Statistics

- Given the co-occurrence matrix $W = (w_{s,t})_{s,t}$, we can define the loss function for the skip-gram model with global corpus statistics as:

$$-\sum_{i,j \in \mathcal{V}} w_{ij} \log p(x_j \mid x_i)$$

- Recall that the skip-gram model involves a summation over the entire vocabulary $\mathcal{V}$, which is computationally inefficient.

GloVe: Global Vectors for Word Representation

- The GloVe model relates the similarity between words and their co-occurrence by minimizing the least-squares loss:

$$\sum_{i,j} \left(u_j^T v_i - \log w_{ij} \right)^2$$

This paper is available at `https://nlp.stanford.edu/pubs/glove.pdf`

GloVe: Global Vectors for Word Representation

- The GloVe model relates the similarity between words and their co-occurrence by minimizing the least-squares loss:

$$\sum_{i,j} \left(u_j^T v_i - \log w_{ij} \right)^2$$

- Two bias terms (or intercepts) are introduced for the center words (b_i) and the context words (c_j), resulting in the modified loss:

$$\sum_{i,j} \left(u_j^T v_i + b_i + c_j - \log w_{ij} \right)^2 .$$

This paper is available at `https://nlp.stanford.edu/pubs/glove.pdf`

GloVe: Global Vectors for Word Representation

- The GloVe model relates the similarity between words and their co-occurrence by minimizing the least-squares loss:

$$\sum_{i,j} \left(u_j^T v_i - \log w_{ij} \right)^2$$

- Two bias terms (or intercepts) are introduced for the center words (b_i) and the context words (c_j), resulting in the modified loss:

$$\sum_{i,j} \left(u_j^T v_i + b_i + c_j - \log w_{ij} \right)^2 .$$

- A weight function $h(w_{ij})$ is added to the loss, where h is an increasing function, resulting in:

$$\sum_{i,j} h(w_{ij}) \left(u_j^T v_i + b_i + c_j - \log w_{ij} \right)^2 .$$

This paper is available at `https://nlp.stanford.edu/pubs/glove.pdf`

Weight function h

- For the weight function, a suggested choice is

$$h(z) = \min\{(z/c)^{\alpha}, 1\}.$$

- For example, $c = 100$ and $\alpha = 0.75$.
- When $w_{ij} = 0$, we have $h(0) = 0$

Code for Training a Model Yourself Using SGD

- After we cover deep learning concepts, you can try training a Word2Vec model yourself using `PyTorch`.
- This may be included in the main course or the tutorial, depending on the available time.

Summary (and what next?)

- We introduced two probabilistic models based on MLE: the skip-gram model, which predicts context words given a center word, and the CBOW model, which predicts a center word from context words.
- Stochastic gradient descent is widely used to minimize loss functions in machine learning.
- The co-occurrence matrix can be incorporated into the loss function to capture the global information of a corpus. For computational efficiency, the GloVe model is a more efficient alternative.
- How to measure the quality of your embeddings? (Tokenizationation part)