# Some mathematics of Word2Vec algorithm and approximated Softmax

A/Prof Richard Yi Da Xu, Xuan Liang
richardxu.com

University of Technology Sydney (UTS)

August 16, 2022

# Negative sampling (1)

- negative sampling based on Skip-Gram model, it is optimizing **different objective**, let $\theta = [\mathbf{u}, \mathbf{v}]$:
- we let $\bar{w}$ to indicate **negative samples**, and come from a negative population $\bar{D}$

$$
\begin{aligned}
\theta &= \arg\max_{\theta} \prod_{(w,c)\in D} \Pr(D=1|w, c, \theta) \prod_{(\bar{w},c)\in \bar{D}} \Pr(D=0|\bar{w}, c, \theta) \\
&= \arg\max_{\theta} \prod_{(w,c)\in D} \Pr(D=1|w, c, \theta) \prod_{(\bar{w},c)\in \bar{D}} (1 - \Pr(D=1|\bar{w}, c, \theta)) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \log\left(\Pr(D=1|w, c, \theta)\right) + \sum_{(\bar{w},c)\in \bar{D}} \log\left(1 - \Pr(D=1|\bar{w}, c, \theta)\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+\exp\left[-\mathbf{u}_w^\top \mathbf{v}_c\right]} + \sum_{(\bar{w},c)\in \bar{D}} \log\left(1 - \frac{1}{1+\exp\left[-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c\right]}\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \sigma(-\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \log\left(\frac{1}{1+\exp\left[\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c\right]}\right) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)
\end{aligned}
$$

# Negative sampling (2)

- negative sampling based on Skip-Gram model, it is optimizing **different objective**, let $\theta = [\mathbf{u}, \mathbf{v}]$:

$$\theta = \arg\max_{\theta} \sum_{(w,c)\in D} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{(\bar{w},c)\in \bar{D}} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)$$

- it still has a huge sum term $\sum_{(\bar{w},c)\in\bar{D}}(.)$, so we change to:

$$\theta = \arg\max_{\theta} \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + \sum_{\bar{w}=1}^{k} \mathbb{E}_{\bar{w}\sim P(w)} \log \sigma(-\mathbf{u}_{\bar{w}}^\top \mathbf{v}_c)$$

- sample a fraction of negative samples in second terms: $\{\bar{w}\}$ instead of going for $\forall(\bar{w} \neq w) \in \mathcal{V}$
- $\bar{w} \sim \Pr_{\bar{D}}(w)$, where $\Pr_{\bar{D}}(.)$ is probability of negative sample:

  can use Unigram Model raised to the power of $\frac{3}{4}$
- doing so, we can:

  increase probability of popular words marginally

  increase probability of rarer words dramatically

  making "rarer" words also have chance to be sampled
- in unigram model, probability of each word only depends on that word's own probability

# Noise Contrastive Estimation (NCE) (1)

- let $u_\theta(w, c)$ be un-normalized score function, i.e., $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$

$$P_\theta(w|c) = \frac{u_\theta(w, c)}{\sum_{w' \in \mathcal{V}} u_\theta(w', c)} = \frac{u_\theta(w, c)}{Z_c}$$

- $\tilde{p}(w|c)$ and $\tilde{p}(c)$ are empirical distributions
  we **know** them from data, so we can sample $(w, c)$ from it!
- a "noise" distribution $q(w)$ is used - uniform or uniform unigram
  we also **know** them, again, we can sample $\bar{w} \sim q(.)$
- **task** is to use sample from both distributions, then to assist us find $\theta$ making $P_\theta(w|c)$ to approximate empirical distribution as closely as possible (by minimal cross entropy)

# Noise Contrastive Estimation (NCE) (2)

- **training data generation**: $(w, c, D) \sim \mathcal{D}$
- of course, to utilize $\tilde{p}(w|c)$, $\tilde{p}(c)$ and $q(w)$, which we already have knowledge of:
  1. sample a $c \sim \tilde{p}(c)$, $w \sim \tilde{p}(w|c)$ and label it as $D = 1$
  2. $k$ "noise" samples from $q(.)$, and label it as $D = 0$
- NCE transforms:
  *"problem of model estimation"* (computationally expensive) to
  *"problem of estimating parameters of probabilistic binary classifier uses* **same parameters** *to distinguish between samples"*:(computationally acceptable)
  - from empirical distribution
  - from noise distribution

# Noise Contrastive Estimation (NCE) (3)

- let $u_\theta(w, c)$ be un-normalized score function, i.e., $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$

$$
\begin{aligned}
P(D=0|c,w) &= \frac{P(D=0, w|c)}{P(w|c)} = \frac{p(w|D=0,c)P(D=0)}{\sum_{d\in\{0,1\}} p(w|D=d,c)P(D=d)} \\
&= \frac{q(w) \times \frac{k}{1+k}}{\tilde{P}(w|c) \times \frac{1}{k+1} + q(w) \times \frac{k}{1+k}} \\
&= \frac{kq(w)}{\tilde{P}(w|c) + kq(w)} \\
P(D=1|c,w) &= 1 - P(D=0|c,w) \\
&= \frac{\tilde{P}(w|c)}{\tilde{P}(w|c) + kq(w)}
\end{aligned}
$$

# Noise Contrastive Estimation (NCE) (4)

- NCE replaces empirical distribution $\tilde{p}(w|c)$ with model distribution $p_\theta(w|c)$

$$P(D=0|c,w) = \frac{kq(w)}{\tilde{P}(w|c) + kq(w)} = \frac{kq(w)}{\frac{u_\theta(w|c)}{Z_c} + kq(w)}$$

$$P(D=1|c,w) = \frac{\tilde{P}(w|c)}{\tilde{P}(w|c) + kq(w)} = \frac{\frac{u_\theta(w|c)}{Z_c}}{\frac{u_\theta(w|c)}{Z_c} + kq(w)}$$

- $\theta$ is then chosen to maximize likelihood of "proxy corpus" created from **training data generation**:

$$\mathcal{L}^{\text{NCE}} = \log p(D=1|c,w) + k \sum_{(w,c)\in\mathcal{D}} \mathbb{E}_{w'\sim q} \log p(D=0|c,w')$$

- for neural networks: $Z_c$ can also be trained or set to some fixed number, e.g., $Z_c = 1$
- **negative sampling** is its special case $k = |\mathcal{V}|$ and $q(.)$ is uniform, and $Z_c = 1$:

$$P(D=0|c,w) = \frac{|\mathcal{V}|\frac{1}{|\mathcal{V}|}}{u_\theta(w|c) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{1}{u_\theta(w|c) + 1}$$

$$P(D=1|c,w) = \frac{u_\theta(w|c)}{u_\theta(w|c) + |\mathcal{V}|\frac{1}{|\mathcal{V}|}} = \frac{u_\theta(w|c)}{u_\theta(w|c) + 1}$$

# Self Normalization

- in previous slide, we want to normalize, s.t., $Z_c = 1$
- start with $u_\theta(w, c) = \exp(\mathbf{u}_w^\top \mathbf{v}_c)$:

$$
\begin{aligned}
P_\theta(w|c) &= \prod_w \frac{\exp(\mathbf{u}_w^\top \mathbf{v}_c)}{Z_c} \\
\implies \quad J_\theta = -\prod_w \log(P_\theta(w|c) &= -\sum_w \log\left(\frac{\exp(\mathbf{u}_w^\top \mathbf{v}_c)}{Z_c}\right) \\
&= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c - \log(Z_c)
\end{aligned}
$$

- to constrain model and sets $Z(c) = 1 \implies \log Z(c) = 0$:

$$
\begin{aligned}
J_\theta &= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c + \log Z(c) - \alpha\left(\log(Z(c)) - 0\right)^2 \\
&= -\sum_w \mathbf{u}_w^\top \mathbf{v}_c + \log Z(c) - \alpha \log^2 Z(c)
\end{aligned}
$$

# FastText

A library created by Facebook research team for

1. efficient learning of word representations(Enriching Word Vectors with Subword Information)
2. sentence classification(Bag of Tricks for Efficient Text Classification)

# FastText

- So how is it different from Word2Vec?
- Instead of words, we now have **ngrams** of subwords, what is its **advantage**?
  1. Helpful for finding representations for rare words
  2. Give vector representations for words not present in dictionary
- for example, $n = 3$ , i.e., 3-grams:
  - **word**: *"where"*,
  - **sub-words**: *"wh"*, *"whe"*, *"her"*, *"ere"*, *"re"*
- we then represent a word by the **sum of the vector representations** of all its n-grams
- to compute an un-normalised score with center word $\mathbf{v}_c$, given a word $w$, $g_w$ is the set of $n$-grams appearing in $w$, $z_g$ is the representation to each individual $n$-gram

$$u(w, c) = \exp\left[\sum_{g \in g_w} z_g^\top \mathbf{v}_c\right]$$

# Global Vectors for Word Representation(GloVe)

- **co-occurrence probabilities** are useful
- GloVe learns word vectors through *word co-occurrences*
- co-occurrence matrix $P$ where $P_{ij}$ is how often word $i$ appears in the context of word $j$
- Fast training and scalable to huge corpora
- loss function:

$$\theta^* = \arg\min_{\theta} \left( J(\theta) \equiv \frac{1}{2} \sum_{\mathbf{u}_i \mathbf{v}_j \in \mathcal{V}} f(P_{ij})(\mathbf{u}_i^\top \mathbf{v}_j - \log P_{ij})^2 \right)$$

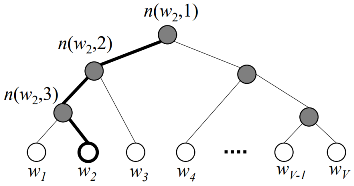
- it tries to minimize difference:

$$\left(\mathbf{u}_i^\top \mathbf{v}_j - \log P_{ij}\right)$$

- more frequently two words appear together,more similar their vector representation should be
- $f(.)$ is weighting function to "prevent" certain scenarios, for example:

$$P_{ij} = 0 \implies \log P_{ij} = -\infty \implies f(0) = 0$$

# Hierarchical Softmax (1)




*Xin Rong, word2vec Parameter Learning Explained*

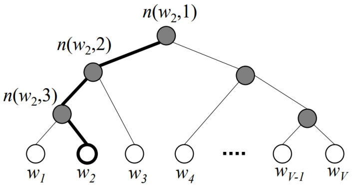
- **super advantage**: $\Pr(w|c)$ is already a probability by multiplying all probabilities of path, no need to normalize!

- each word $w_i$ has a unique (pre-defined) path (not a random path!), which performs a left or right turn from nodes: $n(w_i, 1), n(w_i, 2), n(w_i, 3), \dots$
- the route is defined in such a way that, each child node is from a (LEFT/RIGHT) "channel" of its parent: i.e., $n(w, j+1) = \text{ch}(n(w, j))$
- for example:

$$\underbrace{\begin{aligned} &1.\ n(w_2, 2) = \text{LEFT}\,(n(w_2, 1)) \\ &2.\ n(w_2, 3) = \text{LEFT}\,(n(w_2, 2)) \\ &3.\ n(w_2, 4) = \text{RIGHT}(n(w_2, 3)) \end{aligned}}_{w_2}$$

- there are $V$ words in leaf (white node)
- there are $V - 1$ inner (non-leaf) nodes (grey node) each associate with a value of $\mathbf{v}$ which is shared among all words going through this node

# Hierarchical Softmax (2)




*Xin Rong, word2vec Parameter Learning Explained*

- **super advantage**: $\Pr(w|c)$ is already a probability by multiplying all probabilities of path, no need to normalize!

we define: $\xi[.] = \begin{cases} 1 : & \text{true} \\ -1 : & \text{false} \end{cases}$

$$\Pr(w|c) = \prod_{j=1}^{L(w)-1} \sigma\left( \underbrace{\xi\left[n(w, j+1) = \text{ch}(n(w,j))\right]}_{\text{control its sign}} \mathbf{v}_{n(w,j)}^{\top} \mathbf{u}_c \right)$$

- looking at $\Pr(w_2|c)$ and $\Pr(w_3|c)$:
- $n(w_2, 1) = n(w_3, 1)$ in fact $\{n(w_i, 1)\}_{i=1}^{|\mathcal{V}|}$ all equal
- $n(w_2, 2) = n(w_3, 2)$

$$\begin{aligned}
\Pr(w_2|c) &= p(n(w_2, 1), \text{LEFT})\, p(n(w_2, 2), \text{LEFT})\, p(n(w_2, 3), \text{RIGHT}) \\
&= \sigma\left(\mathbf{v}_{n(w_2,1)}{}^{\top} \mathbf{u}_c\right) \sigma\left(\mathbf{v}_{n(w_2,2)}{}^{\top} \mathbf{u}_c\right) \sigma\left(-\mathbf{v}_{n(w_2,3)}^{\top} \mathbf{u}_c\right) \\
\Pr(w_3|c) &= p(n(w_3, 1), \text{LEFT})\, p(n(w_3, 2), \text{RIGHT}) \\
&= \sigma\left(\mathbf{v}_{n(w_3,1)}{}^{\top} \mathbf{u}_c\right) \sigma\left(-\mathbf{v}_{n(w_3,2)}{}^{\top} \mathbf{u}_c\right)
\end{aligned}$$