1 Logistic function

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \tag{1}$$

$$\sigma(-x) = 1 - \sigma(x) \tag{2}$$

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x)\sigma(-x) \tag{3}$$

2 Softmax formulation

Let (a, b) a pair of items, where $a \in A$ is the source and $b \in B$ the target. The actual meaning depends on the use case.

The conditional probability of observing b given a is defined by a softmax on all possibilities, as it is a regular multi-class task:

$$P(b \mid a; \mathbf{u}, \mathbf{v}) = \frac{e^{\mathbf{u}_a^T \mathbf{v}_b}}{\sum_{b'} e^{\mathbf{u}_a^T \mathbf{v}_{b'}}} \tag{4}$$

Negative log-likelihood:

$$\mathcal{L}(a, b; \mathbf{u}, \mathbf{v}) = -\log P(b \mid a; \mathbf{u}, \mathbf{v}) = -\mathbf{u}_a^T \mathbf{v}_b + \log \sum_{b'} e^{\mathbf{u}_a^T \mathbf{v}_{b'}} \tag{5}$$

$$\frac{\partial}{\partial \mathbf{u}_a}\mathcal{L}(a, b; \mathbf{u}, \mathbf{v}) = -\mathbf{v}_b + \sum_{b'} P(b' \mid a; \mathbf{u}, \mathbf{v})\mathbf{v}_{b'} \tag{6}$$

3 Noise contrastive estimation formulation

Noise Contrastive Estimation (Gutmann and Hyvärinen [4]) is proposed by Mnih and Teh [6] as a stable sampling method, to reduce the cost induced by softmax computation. In a nutshell, the model is trained to distinguish observed (positive) samples from random noise. Logistic regression is applied to minimize the negative log-likelihood, i.e. cross-entropy of our training example against the k noise samples:

$$\mathcal{L}(a, b) = -\log P(y = 1 \mid a, b) + k\mathbb{E}_{b' \sim Q}\left[-\log P(y = 0 \mid a, b)\right] \tag{7}$$

To avoid computating the expectation on the whole vocabulary, a Monte Carlo approximation is applied. $B^* \subseteq B$, with $|B^*| = k$, is therefore the set of random samples used to estimate it:

$$\mathcal{L}(a, b) = -\log P(y = 1 \mid a, b) - k \sum_{b' \in B^* \subseteq B} \log P(y = 0 \mid a, b') \tag{8}$$

We are effectively generating samples from two different distributions: positive pairs are sampled from the empirical training set, while negative pairs come from the noise distribution Q.

$$P(y, b \mid a) = \frac{1}{k+1} P(b \mid a) + \frac{k}{k+1} Q(b) \tag{9}$$

Hence, the probability that a sample came from the training distribution:

$$P(y = 1 \mid a, b) = \frac{P(b \mid a)}{P(b \mid a) + kQ(b)} \tag{10}$$

$$P(y = 0 \mid a, b) = 1 - P(y = 1 \mid a, b) \tag{11}$$

However, $P(b \mid a)$ is still defined as a softmax:

$$P(b \mid a; \mathbf{u}, \mathbf{v}) = \frac{e^{\mathbf{u}_a^T \mathbf{v}_b}}{\sum_{b'} e^{\mathbf{u}_a^T \mathbf{v}_{b'}}} \tag{12}$$

Both Mnih and Teh [6] and Vaswani et al. [8] arbitrarily set the denominator to 1. The underlying idea is that, instead of explicitly computing this value, it could be defined as a trainable parameter. Zoph et al. [9] actually report that even when trained, it stays close to 1 with a low variance.

Hence:

$$P(b \mid a; \mathbf{u}, \mathbf{v}) = e^{\mathbf{u}_a^T \mathbf{v}_b} \tag{13}$$

The negative log-likelihood can then be computed as usual:

$$\mathcal{L}(a, b; \mathbf{u}, \mathbf{v}) = -\log P(a, b; \mathbf{u}, \mathbf{v}) \tag{14}$$

Mnih and Teh [6] report that using $k = 25$ is sufficient to match the performance of the regular softmax.

4 Negative sampling formulation

Negative Sampling, popularised by Mikolov et al. [5], can be seen as an approximation of NCE. As defined previously, NCE is based on the following:

$$P(y = 1 \mid a, b; \mathbf{u}, \mathbf{v}) = \frac{e^{\mathbf{u}_a^T \mathbf{v}_b}}{e^{\mathbf{u}_a^T \mathbf{v}_b} + kQ(b)} \tag{15}$$

Negative Sampling simplifies this computation by replacing $kQ(b)$ by 1. Note that $Q(b) = 1$ is true when $B^* = B$ and Q is the uniform distribution.

$$P(y = 1 \mid a, b; \mathbf{u}, \mathbf{v}) = \frac{e^{\mathbf{u}_a^T \mathbf{v}_b}}{e^{\mathbf{u}_a^T \mathbf{v}_b} + 1} = \sigma\left(\mathbf{u}_a^T \mathbf{v}_b\right) \tag{16}$$

Hence:

$$P(a, b; \mathbf{u}, \mathbf{v}) = \sigma\left(\mathbf{u}_a^T \mathbf{v}_b\right) \prod_{b' \in B^* \subseteq B} \left(1 - \sigma\left(\mathbf{u}_a^T \mathbf{v}_b\right)\right) \tag{17}$$

$$\mathcal{L}(a, b; \mathbf{u}, \mathbf{v}) = -\log \sigma\left(\mathbf{u}_a^T \mathbf{v}_b\right) - \sum_{b' \in B^* \subseteq B} \log\left(1 - \sigma\left(\mathbf{u}_a^T \mathbf{v}'_b\right)\right) \tag{18}$$

For more details, see Goldberg and Levy's notes [3].

To compute the gradient, let us rewrite the loss as:

$$\mathcal{L}(a, b; \mathbf{u}, \mathbf{v}) = -\ell_{a,b,1} - \sum_{b' \in B^* \subseteq B} \ell_{a,b',0} \tag{19}$$

where

$$\ell_{a,b,y} = \log \sigma\left(y - \mathbf{u}_a^T \mathbf{v}_b\right) \tag{20}$$

Then:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}_a}\ell(a,b,y) &= \tfrac{1}{y-\sigma(\mathbf{u}_a^T\mathbf{v}_b)}\left(-\sigma\left(\mathbf{u}_a^T\mathbf{v}_b\right)\left(1-\sigma\left(\mathbf{u}_a^T\mathbf{v}_b\right)\right)\right)\mathbf{v}_b \\ &= \left(y-\sigma\left(\mathbf{u}_a^T\mathbf{v}_b\right)\right)\mathbf{v}_b \end{aligned} \tag{21}$$

And similarly:

$$\frac{\partial}{\partial \mathbf{v}_b}\ell(a,b,y) = \left(y-\sigma\left(\mathbf{u}_a^T\mathbf{v}_b\right)\right)\mathbf{u}_a \tag{22}$$

5 Normalization

By setting the denominator to 1, as proposed above, the model essentially learns to self-normalize. However, Devlin et al. [2] suggest to add a squared error penalty to enforce the equivalence. Andreas and Klein [1] even suggest that it should even be sufficient to only normalize a fraction of the training examples and still obtain approximate self-normalising behaviour.

6 Item distribution balancing

In word2vec, Mikolov et al. [5] use a subsampling approach to reduce the impact of frequent words. Each word has a probability

$$P(w_i) = 1 - \sqrt{\left(\frac{t}{f(w_i)}\right)} \tag{23}$$

of being discarded, where $f(w_i)$ is its frequency and t a chosen threshold, typically around 10^{-5}.

7 Parallelization

Hogwild [7].

References

[1] Jacob Andreas and Dan Klein. “When and why are log-linear models self-normalizing?” In: Jan. 2015, pp. 244–249. DOI: 10.3115/v1/N15-1027.

[2] Jacob Devlin et al. "Fast and Robust Neural Network Joint Models for Statistical Machine Translation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 1370–1380. DOI: 10.3115/v1/P14-1129. URL: https://www.aclweb.org/anthology/P14-1129.

[3] Yoav Goldberg and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *ArXiv* abs/1402.3722 (2014).

[4] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 297–304. URL: http://proceedings.mlr.press/v9/gutmann10a.html.

[5] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *NIPS*. 2013.

[6] Andriy Mnih and Yee Whye Teh. "A fast and simple algorithm for training neural probabilistic language models". In: *arXiv preprint arXiv:1206.6426* (2012).

[7] Benjamin Recht et al. "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent". In: *NIPS*. 2011.

[8] Ashish Vaswani et al. "Decoding with Large-Scale Neural Language Models Improves Translation". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1387–1392. URL: https://www.aclweb.org/anthology/D13-1140.

[9] Barret Zoph et al. "Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies". In: Jan. 2016, pp. 1217–1222. DOI: 10.18653/v1/N16-1145.