# CSE538 Natural Language Processing Fall 18


## Prof. Niranjan Balasubramanian
## Stony Brook University


## Assignment 1
## 3rd October, 2018


## Name - Rushikesh Nalla
## ID - 111971011


# 1 Hyper parameters explored -

### Solution -

The hyper parameters that I explored are batch size, skip window, num skips, max num steps and learning rate. This is the general behaviour that is expected if each of the parameters are changed-

- **Batch size -** Number of samples (context, predicting) word pairs drawn in each batch.
  - **Increasing -** More samples are considered in every training step, it would result in a better model (if it doesn't cause over fitting).
  - **Decreasing -** If less samples are considered in every training step then we might need to run the code for more training steps to avoid under fitting.
- **Skip window -** Number of predicting words that can be considered to the left and right of context word.
  - **Increasing -** More predicting words are drawn in so more information about each sample is considered while training. Thinking logically, it should improve the model.
  - **Decreasing -** Less information about the predicting words is known to the model. It makes the predicting word options limited i.e, more specific.
- **Num skips -** How many predicting words to consider for each center word.
  - **Increasing -** More representation for each center word in the training phase. Helps in understanding the context better.
  - **Decreasing -** Lesser representation for each center word. Need more training steps for better results.
- **Max num steps -** Number of times you generate batches for training.
  - **Increasing -** Training for more steps increasing the performance if the model doesn't over fit the data. More time is required for training.
  - **Decreasing -** Training for lesser steps can hinder the results if the model is not completely trained with the data. It can also improve the results if there is nothing much to learn from the data as it prevents over fitting.
- **Learning rate -** It is used to determine how much each updating step should effect the weights.
  - **Increasing -** Each step has greater influence on the weights. Can help in finding optimal weights much faster.

– **Decreasing -** Each step has lesser influence on the weights. Might take more number of training steps to reach the optimum.

## 2 Results on analogy task -

### Solution -

I am using Using Python 3.5.3, Baseline Accuracy is 29%. Below is the table containing some of the configurations that I have tried -

| Configuration Number | Batch Size | Skip Window | Num Skips | Max Num Steps | Learning Rate | Cross Entropy Loss | Cross Entropy Accuracy | NCE Loss | NCE Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 128 | 4 | 8 | 200001 | 1 | 4.827 | 29.0% | 1.319 | 31.5% |
| 2 | 64 | 2 | 4 | 200001 | 1 | 4.056 | 30.6% | 1.267 | 32.0% |
| 3 | 256 | 8 | 16 | 200001 | 1 | 5.535 | 29.3% | 1.557 | 31.1% |
| 4 | 32 | 4 | 4 | 400001 | 1 | 3.392 | 31.7% | 1.397 | 33.1% |
| 5 | 64 | 4 | 8 | 400001 | 0.1 | 4.120 | 30.0% | 1.339 | 30.2% |

Some insights and observations -

- In general the results obtained with NCE loss method is better than Cross Entropy loss method for most configurations.
- On increasing the batch size, skip window and num skips together we would usually expect the model to do better but it didn't happen because the vocabulary has less words and training with the same words may lead to over fitting.
- As a proof for the above point we see that we get the best model when the batch size and skip window is less (seen in configuration 4 in the table).
- Decreasing the learning rate and increasing max num steps (learning slowly for more time) only slightly improved the baseline accuracy.
- Surprisingly both the loss methods produced the best models with same hyper parameter configuration.

**Best Configuration setting for both cross entropy and nce loss -**

- Batch Size = 32
- Skip Window = 4
- Num Skips = 4
- Max Num Steps = 400001
- Learning Rate = 1

## 3 Top 20 similar words -

### Solution -

- We can see from the table that overall NCE loss model did better than Cross Entropy loss method.
- The difference can be clearly seen by looking at the starting few words of "first".
- For the word "american", NCE did slightly better than cross entropy.
- The top 20 words for "would" are almost the same in both models.

| **Top Word Rank** | **Cross Entropy** | | | **NCE** | | |
|---|---|---|---|---|---|---|
| | **"first"** | **"american"** | **"would"** | **"first"** | **"american"** | **"would"** |
| 1 | same | australian | could | name | old | could |
| 2 | con | italian | should | last | international | had |
| 3 | latter | international | must | original | british | will |
| 4 | last | soviet | did | next | official | can |
| 5 | hugin | security | can | kingdom | end | must |
| 6 | mundo | ancient | will | same | italian | did |
| 7 | main | batavia | does | best | german | do |
| 8 | original | british | may | end | english | have |
| 9 | demographically | european | might | late | ancient | may |
| 10 | offsets | manpower | cannot | country | majors | should |
| 11 | second | eastern | had | title | french | came |
| 12 | shiftrows | software | began | united | author | has |
| 13 | scout | supply | quicktime | war | tversky | took |
| 14 | final | austrian | rtd | greatest | mcloughlin | gave |
| 15 | switched | foreign | iaaf | rest | verein | began |
| 16 | next | mnemosyne | to | following | kitt | are |
| 17 | south | blinds | anodized | second | schistosomiasis | believe |
| 18 | illyrian | galloway | was | battle, | canadian | we |
| 19 | highest | middle | has | church, | abelian | was |
| 20 | bay | multiple | became | world | implacable | became |

## 4 Summary of NCE loss method -

### Solution -

The standard procedure for predicting an outer word given a sample word is to maximize the likelihood of a sample in the training data. This requires taking sum over all the words in the vocabulary as seen in cross entropy loss function (using softmax) which is very expensive.

NCE is a sampling based approach which address this issue of reducing the computation. We try to sample the data from true distribution and some noise distribution which is the unigram probability distribution in our case. We have,

$$Pr(D, w_o|w_c) = \begin{cases} \frac{k}{1+k} \times q(w_o) & \text{D=1} \\ \frac{1}{1+k} \times p(w_o|w_c) & \text{D=0} \end{cases}$$

where D=1 means a single sample is drawn from true distribution and D=0 means k number of samples are drawn from noise distribution.

The basic idea is to train a logistic regression (binary) classifier to discriminate between these samples. Using conditional probability we get, $Pr(D = 0|w_o, w_c)$ which is the probability that sample came from noise distribution and $Pr(D = 1|w_o, w_c)$ which is the probability that sample came from true distribution.

$$Pr(D = 0|w_o, w_c) = \frac{\frac{k}{1+k} \times q(w_o)}{\frac{k}{1+k} \times q(w_o) + \frac{1}{1+k} \times p(w_o|w_c)} = \frac{k \times q(w_o)}{p(w_o|w_c) + k \times q(w_o)}$$

$$Pr(D = 1|w_o, w_c) = \frac{p(w_o|w_c)}{p(w_o|w_c) + k \times q(w_o)}$$

We can replace the term $p(w_o|w_c)$ with $s(w_o, w_c)$ which is the unnormalized probability of the word $w_o$ given $w_c$. $Pr(D = 1|w_o, w_c)$ can be written in terms of sigmoid as given in the assignment. The probability that the sample has come from the true distribution is

$$Pr(D = 1|w_o, w_c) = \sigma(s(w_o, w_c) - \log(kQ(w_o))$$

where Q is the noise distribution. The scaling factor k in front of $Q(w_o)$ implies that noise samples are k times more frequent than data samples.

For maximizing the probability that word comes from true distribution we have to maximize the part inside sigmoid. The loss function can be written as -

$$J(\theta) = \sum_{w_o, w_c \in D} \left( \log(\sigma(s(w_o, w_c) - \log(kQ(w_o)))) + \sum_{x \in V_k} \log(1 - \sigma(s(w_o, w_c) - \log(kQ(w_x)))) \right)$$

where $V_k$ is a set of k negative words and Q is the noise distribution.

We have to maximize the above positive log likelihood. As the first term represents the probability of the sample being drawn from true distribution we have to maximize it. The second term is the probability of drawing the sample from noise distribution so we have to minimize it but we are subtracting it from 1 so that we can maximize that quantity as well.

The key takeaway of using NCE is that we are summing over k negative samples instead of the entire vocabulary which decreases computation time considerably. As k increases we approach towards the gradient of the softmax function.