HYPER PARAMETERS EXPLORED

Batch_size

It is the total number of (input,output) pairs that is pass to the neural network in one step (epoch). Processing data in batches is speed efficient and prevents overfitting on single elements. This improves learning and also doesn't need much memory since we pass data in small chunks. As the batch_size increases, the accuracy of the model increases but is constrained by memory usage since larger batch_size might consume a lot of resources.
It also might require considerable amount of time to complete every iteration.

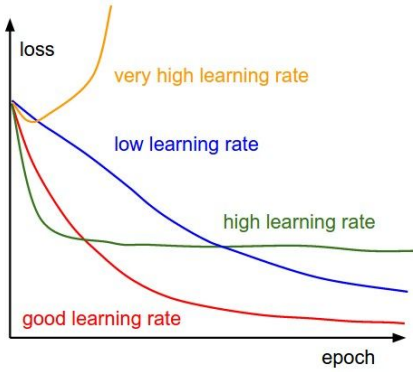
Number of Training Steps - Epoch Value

This represents the number of steps a model is trained for before getting results. Increasing the number of epochs improves the quality of the word representations but drastically increases the training time as it requires more time for computation. After a particular number of steps, the model does not train significantly as it is quite sufficiently trained till that point.

Learning Rate

Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower it moves along the decreasing slope and the more it learns thus giving a higher accuracy. Learning rate affects how quickly the model converges to a local minima.

Image Credits : http://cs231n.github.io/neural-networks-3/



Window Size

The size of the context window determines how many words before and after a given word would be included as context words of the given word. A smaller window tends to predict words that are more related. If the corpus is huge, the window size has a slight effect on accuracy.

Word Vector Size

This represents the size of the vector used to represent a word. The accuracy decreases for very low as well as for extremely high dimensional word vectors as low dimensiion vectors cannot accurately represent a word and high dimensional vectors tend to represent noise as well.(High Variance Problem)

EXPERIMENTAL ANALYSIS

1.Epochs

Parameters: batch_size = 128, skip_window = 4, learning rate = 1.0, embedding_size=128, num_skips=8

Learning Steps	NCE Loss Value	Accuracy
100001	1.2365	32.8%
200001	1.2278	33.5%
300001	1.2158	33.6%
400001	1.1952	33.7%
500001	1.1298	33.8%

From my observations, the accuracy of the model increases with the number of steps as increase in the number of training steps trains the model with more words. Als0, increasing the number of epochs takes more time for computation. After a particular number of epochs, the accuracy is not expected to grow much as the model gets very well trained till that point.

2.Batch_Size

Parameters: max_num_steps = 200001, skip_window=4, learning rate = 1.0, embedding_Size=128, num_skips=8

batch_size	NCE Loss Value	Word Analogy Accuracy of the NCE Model
128	1.604	33.5%
256	1.59	33.9%
512	1.53	34.8%
1024	1.49	34.9%

As per my model, as I increase the batch_size, the accuracy of the model increases. But, I learnt that gains in accuracy are significant up to a particular batch_size. After a size, there are only marginal gains in accuracy. The batch_size depends on the data set, hardware, and the libraries used for mathematical computations. Also, feeding data to the model in batch requires less memory. I tried increasing the batch_size to a large number but that gave me an OutOfMemory Error. Also, with a batch_size of 32 I got really low computational speed as vectorization is not used in the best possible way.

3.Skip_Window

Parameters: max_num_steps = 200001, batch_size = 128, learning rate = 1.0, embedding_Size=128, num_skips=8

Skip_Window	NCE Loss Value	Accuracy
2	1.140	33.3%
4	1.348	33.5%
8	1.829	33.6%
16	1.817	33.8%
32	1.531	33.9%

The accuracy of the model does increase slightly as the size of the skip_window increases. As we have a huge vocabulary size, increasing skip_window size has little improvements on accuracy.

4.Word Vector Dimension

Parameters: max_num_steps = 200001, batch_size = 128, learning rate = 1.0, skip_window = 4,num_skips=8

embedding_size	NCE Loss Value	Accuracy
64	1.623	32.2%

128	1.691	33.5%
512	1.782	30.0%
1024	1.892	29.8%

I learnt that lower dimensional word vectors are not able to capture and represent the different meanings of different words in the vocabulary. Also, larger dimension vector represents noise in the vocabulary and hence the accuracy drops which is also known as high variance problem. Thus, it is best to use a medium sized word vector for best representation of a word.

5. Learning Rate

Parameters: max_num_steps = 200001, batch_size = 128, skip_window = 4, embedding_Size=128, num_skips=8

Learning Rate	NCE Loss Value	Accuracy
0.05	1.514	34.3%
0.1	1.192	33.9%
1.0	1.238	33.5%
2.0	5.554	32.7%
3.0	5.982	30.8%

As per my observations, as the learning rate increases the accuracy of the model decreases. Therefore, if the learning rate is low, model is trained better, but optimization takes a lot of time because of smaller steps towards the minimization of the loss function. When the learning rate is high, the model may not converge or even diverge. The optimizer might overshoot the minimum due to the huge weight changes and thus have a bad loss value. Hence, it is better to choose a lower learning rate.

Best accuracies : NCE MODEL - 35.7%, CROSS ENTROPY MODEL - 33.7% (Details in README.md)

TOP 20 SIMILAR WORDS FOR {first, american, would}

NCE Model

first	during, until, early, before, book, at, war, after, most, word, century, when, work, term, th, and, use, law, however, where
american	french, english, war, society, modern, social, including, international, political, ancient, and, against, revolution, movements, called, i, western, law, would, free
would	could, they, will, i, him, called, did, modern, such, also, up, free, what, so, t, political, although, only, and, against

Cross Entropy Model

first	last, same, following, original, second, best, most, end, entire, largest, name, next, during,

	before, latter, main, another, protect, greatest, primary
american	british, german, french, english, italian, russian, canadian, european, autres, reckless, international, irish, feistel, terminal, henna, ranatunga, spanish, dgh, bookings, harassing
would	will, could, must, can, should, did, may, does, do, india, had, we, cannot, might, said, to, appears, families, came, began

SUMMARY OF NCE LOSS

The basic idea of Noise Contrastive Estimation is to convert the problem of predicting the next word to predict whether a pair of words is good or bad (binary classification)
In the probability distribution, D=1 indicates that a single sample is drawn from the true distribution and D=0 indicates the k number of samples are drown from noise distribution (Unigram Distribution). A binary logistic regression (binary classification) is used instead of using softmax to estimate a true probability distribution of the output word. For each training sample, the optimized classifier is given a (center word, another word that appears in its context) and a number of k randomly noise pairs like - (center word, a randomly chosen word from the corpus). To replace the heavy normalized soft max term which is the probability of the correct word given the context, we take the unnormalized probability (score) of the word given the context. To transform this to the model cost function, we express D=1 case as a sigmoid equation as follows:
Equation 1

$$P^h(D=1|w,\theta) = \frac{P_\theta^h(w)}{P_\theta^h(w) + kP_n(w)} = \sigma\left(\Delta s_\theta(w,h)\right),$$

Where,
Equation 2

$$\Delta s_\theta(w,h) = s_\theta(w,h) - \log(kP_n(w))$$

The scaling k in front of Pn(w) accounts for the fact that noise samples are k times more frequent than data samples. Here, first term is the unnormalized score of the word w under the model and the second term is the log probability of word sampling from the noise distribution.Thus, to maximize the probability of a word comes from the true distribution we need to maximize delta. (From equation 1)
Thus by combining equation 1 and 2, we write our new cost function as :

$$\begin{aligned} J^h(\theta) =& E_{P_d^h}\left[\log P^h(D=1|w,\theta)\right] + kE_{P_n}\left[\log P^h(D=0|w,\theta)\right] \\ =& E_{P_d^h}\left[\log \sigma\left(\Delta s_\theta(w,h)\right)\right] + kE_{P_n}\left[\log\left(1-\sigma\left(\Delta s_\theta(w,h)\right)\right)\right], \end{aligned}$$

Where, first term corresponds to D=1 case and second term corresponds to D=0 case. This is like a binary classifier. The classifier thus learns the word vectors by learning to discriminate the true pairs from the noisy ones. So instead of predicting the next word, the optimized classifier simply predicts whether a pair of words is good or bad. Thus, the NCE training time linear in the number of noise samples and independent of the vocabulary size.

References: Notes on Noise Contrastive Estimation and Negative Sampling