Improving Language Modelling with Noise Contrastive Estimation

Farhana Ferdousi Liza, Marek Grzes
School of Computing, University of Kent
Canterbury, CT2 7NF, UK
{fl207, m.grzes}@kent.ac.uk

Abstract

Neural language models do not scale well when the vocabulary is large. Noise contrastive estimation (NCE) is a sampling-based method that allows for fast learning with large vocabularies. Although NCE has shown promising performance in neural machine translation, its full potential has not been demonstrated in the language modelling literature. A sufficient investigation of the hyperparameters in the NCE-based neural language models was clearly missing. In this paper, we showed that NCE can be a very successful approach in neural language modelling when the hyperparameters of a neural network are tuned appropriately. We introduced the 'search-then-converge' learning rate schedule for NCE and designed a heuristic that specifies how to use this schedule. The impact of the other important hyperparameters, such as the dropout rate and the weight initialisation range, was also demonstrated. Using a popular benchmark, we showed that appropriate tuning of NCE in neural language models outperforms the state-of-the-art single-model methods based on standard dropout and the standard LSTM recurrent neural networks.

1 Introduction

Language models (LMs), which predict the probability of a next word given its context, play an important role in many downstream applications such as machine translation, question answering and text summarisation. Neural language models that apply various neural architectures (Bengio et al. 2003; Mikolov et al. 2010; Józefowicz et al. 2016) have recently demonstrated significant achievements.

In real applications, the vocabulary size is large and the language models have to estimate a probability distribution over many words. The need for a normalised probability distribution becomes a computational bottleneck because the normalisation constant (i.e. the partition function) has to be computed for the output layer.

Many solutions have been proposed to address the computational complexity of the partition function. Several approaches try to make it more efficient, e.g., hierarchical softmax (Mnih and Hinton 2009), a shortlisting method (Schwenk 2007), or self-normalisation techniques (Chen, Grangier, and Auli 2015). The other methods, such as importance sampling (Bengio and Sénécal 2003; Jean et al. 2014) or noise contrastive estimation (Mnih and Teh 2012), aim at computing an unnormalised statistical model.

In this paper, we investigate noise contrastive estimation (NCE) because of its statistical consistency, and the fact that its potential has not been sufficiently explored in the literature on language models (LMs). NCE has also achieved promising results in machine translation (Vaswani et al. 2013; Baltescu and Blunsom 2015) which indicates that its performance on language models could be better than what is known in current research.

NCE was first proposed in (Gutmann and Hyvärinen 2010) as an estimation principle for unnormalised statistical models. Unnormalised statistical models compute values which, in contrast to formal probabilities, do not add up to one. In order to normalise those values so that they become valid probabilities, they can be divided by the partition function. However, the partition function is computationally expensive to compute when the number of outcomes is large. Therefore, instead of calculating the partition function, NCE converts the original estimation problem into a nonlinear logistic regression problem which discriminates the noise samples generated from a known (noise) distribution from the original data samples. NCE is statistically consistent and more stable than other Monte Carlo methods such as importance sampling (Mnih and Teh 2012). In (Gutmann and Hyvärinen 2010), NCE achieved the best trade-off between computational and statistical efficiency when compared against importance sampling, contrastive divergence (Hinton 2002), and score matching (Hyvärinen 2005). This method has also been applied in language modelling and machine translation (Mnih and Teh 2012; Vaswani et al. 2013; Baltescu and Blunsom 2015; Zoph et al. 2016).

Many features of NCE are not understood, especially the hyperparameters in deep learning when NCE is used at the output layer. Also, comparisons against standard LSTM-based single-model softmax have never shown that NCE can compete with softmax on those tasks on which softmax is feasible.

Our results are clearly surprising in the face of the existing literature which generally indicates that NCE is an inferior method. For example, studying language modelling in (Józefowicz et al. 2016), the authors argued that importance sampling (IS) may be better than NCE as IS optimises a multiclass classification task whereas NCE is solving a binary task. In (Józefowicz et al. 2016), the authors managed to improve the results on language modelling using IS, whereas similar improvements for NCE were not found. Overall, the current literature does not have substantial, empirical evidence that NCE is a powerful method for neural network language modelling.

Another example that demonstrates weak performance of NCE on language modelling is (Chen, Grangier, and Auli 2015). The authors explain that a limited number (50) of noise samples in NCE does not allow for frequent sampling of every word in a large vocabulary. The number of noise samples has to be relatively small to make the method feasible. In their experiments, NCE performed better than softmax only on billionW, a dataset on which softmax is very slow due to a very large vocabulary. So NCE was better only because softmax was not feasible on a large vocabulary. In this paper, we show, for the first time, that NCE can outperform softmax in a situation when softmax is feasible and it is known to perform very well. To demonstrate that, we used the Penn Tree Bank (PTB) dataset[1] (Marcus, Marcinkiewicz, and Santorini 1993), which is a popular language modelling benchmark with a vocabulary size of 10k words. Softmax is known for competitive performance on this data, and it is feasible to apply it to this data using GPUs.

There exist papers in which the researchers tried to create conditions which make softmax feasible to be executed on large datasets. For example, the experiments in (Baltescu and Blunsom 2015) are based on a few billions of training examples and a vocabulary with over 100k tokens. To manage the softmax computation, the authors partitioned the vocabulary into K classes. Under those conditions, the authors showed that NCE performed almost as well as softmax. Softmax in their comparisons was approximate, however, due to partitioning. In our paper, the goal is to compete with the original softmax without any approximations. Our aims are justified by the following reasoning. In theory, NCE, being a statistically consistent method, converges to the maximum likelihood estimation method when the number of noise samples is increased. However, the fact that NCE solves a different optimisation problem means that stochastic gradient descent applied to neural networks with NCE may find a different, better local optimum than when it is applied to networks with softmax. Therefore, when the objective function is highly non-convex, NCE can beat softmax even though it is only an approximation to softmax.

Tuning hyperparameters has been an important element of neural networks research (Bengio 2012). The main contribution of our paper is based on a carefully designed hyperparameter tuning strategy for NCE. The separate 'search' and 'convergence' phases for controlling the learning rate (Darken and Moody 1991) have never been applied to NCE-based neural networks. Also, the importance of the search phase and better convergence thanks to the randomness attributed to NCE was never observed in the literature. They appeared to be the key components in this research, and they allowed NCE to outperform softmax on a problem on which softmax is known to have competitive performance and to be computationally feasible.

[1]http://www.fit.vutbr.cz/ imikolov/rnnlm/simple-examples.tgz

Some researchers, e.g., (Chen, Grangier, and Auli 2015), concluded that NCE *'is not among the best techniques for minimising perplexity'*, and this was probably the reason why more sophisticated mechanisms, such as the 'search' and 'convergence' phases for controlling the learning rate, were not used with NCE. This seems to be a common pattern in deep learning research. For example, in 2006, the community used unsupervised learning to initialise supervised learning for neural networks, whereas today, the appropriate resources and engineering practices allow feedforward networks to perform very well without unsupervised initialisation (Goodfellow, Bengio, and Courville 2016, Ch. 6). Analogously, our paper shows that appropriate techniques exist to turn NCE into a very successful method for language modelling.

The paper is organised as follows. Section 2 introduces the NCE model with a deep neural architecture, and Section 3 describes our approach to NCE-based neural language modelling (NCENLM). Sections 4 and 5 describe the experimental design and the results showing that the proposed method improves the state-of-the-art results on the Penn Tree Bank dataset using language modelling based on a standard LSTM (Hochreiter and Schmidhuber 1997; Gers 2001).

2 Background

We study language models where given a sequence of words $W = (w_1, w_2, \ldots, w_T)$ over the vocabulary V, we model sequence probability

$$p(W) = \prod_{i=0}^{T-1} p(w_{i+1}|w_1, \ldots, w_i) = \prod_{i=0}^{T-1} p(w_{i+1}|c_i). \quad (1)$$

Here, for a given word w_{i+1}, $c_i =< w_1, \ldots, w_i >$ represents its full, non-truncated context. In many applications, one is interested in $p(w_{i+1}|c_i)$. Recurrent neural networks try to model such probabilities that depend on a sequence of words c_i. The recurrent connections introduce a notion of 'memory' which can remember a substantial part of word's context c_i. However, due to the gradient vanishing and exploding problems (Pascanu, Mikolov, and Bengio 2013), it is challenging to optimise standard recurrent neural networks even though their expressive power is sufficient in many situations. For this reason, long short term memory (LSTM) was introduced to improve learning with a long context, c_i (Hochreiter and Schmidhuber 1997; Gers 2001). LSTM introduces the concept of memory cells that are used to create layers. Several layers can be stacked into larger blocks (similar to layers of neurons in the multilayer perceptron). The blocks of those layers are then unrolled for several time steps during learning. When n is the last hidden layer, and i is the last unrolled time step, v_i^n is the activation vector that results after c_i has been presented to the network. Then, the final output layer has one vector θ_j for every word w_j in the vocabulary, and the probability of the next word can be computed using the softmax function:

$$P_\theta^{SOFT}(w_{i+1}|c_i) = \frac{\exp\big(\theta_{i+1}^\top v_i^n\big)}{\sum_{j=1}^{|V|}\exp\big(\theta_j^\top v_i^n\big)} = \frac{\exp\big(\theta_{i+1}^\top v_i^n\big)}{Z}. \tag{2}$$

Here, $P_\theta^{SOFT}(w_{i+1}|c_i)$ is the probability of word w_{i+1} given context c_i, θ_{i+1} is the weight vector corresponding to the word w_{i+1} at the output layer, θ_j is the weight vector for the word w_j in vocabulary, and $|V|$ is the vocabulary size. The normalising term Z is known as the **partition function**. Note that unnormalised products $\theta_{i+1}^\top v_i^n$ are not sufficient to evaluate the words.

The softmax-based training of recurrent neural networks that uses stochastic gradient descent (SGD) and backpropagation (BP) maximises the log likelihood or equivalently minimises the cross-entropy of the training sequence containing N words. This objective can be formally expressed as

$$J_{SOFT}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\ln P_\theta^{SOFT}(w_{i+1}|c_i). \tag{3}$$

The gradient used for updating the parameters θ is

$$\begin{aligned}\frac{\partial J_{SOFT}(\theta)}{\partial\theta} = &-\frac{1}{N}\sum_{i=1}^{N}\Big[\frac{\partial(\theta_{i+1}^\top v_i^n)}{\partial\theta}\\ &-\sum_{j=1}^{|V|}P_\theta^{SOFT}(w_j|c_i)\frac{\partial(\theta_j^\top v_i^n)}{\partial\theta}\Big].\end{aligned} \tag{4}$$

Gradient computation is usually time-consuming because when the vocabulary is large, the partition function in P_θ^{SOFT} creates the performance bottleneck for the training and testing phases. It is advantageous to avoid this expensive normalisation term. Noise contrastive estimation (NCE) bypasses this calculation by converting the original optimisation problem to a binary classification problem.

In NCE, we see the corpus as a new dataset of n words of the following format:

$$((c_1, w_2), D_1)), \ldots, ((c_n, w_{n+1}), D_n)$$

where c_i represents the context, w_{i+1} represents the next word after c_i, and a random variable D is set to one when w_{i+1} is from the training corpus (true data distribution) and D is set to zero when w_{i+1} is from a known chosen noise distribution, P_n. For a given context c_i, the NCE-based neural language model (NCENLM) models data samples (from the corpus) as if they were generated from a mixture of two distributions (P_θ^{NCE} and P_n). The mixture is normalised; hence, the requirement for the normalisation term is satisfied implicitly as shown in Eq. (5).

The posterior probability of a sample word w_{i+1} generated from the mixture of the P_θ^{NCE} and the noise distribution P_n are as follows:

$$\begin{aligned}P(D=1|w_{i+1},c_i) &= \frac{P_\theta^{NCE}(w_{i+1}|c_i)}{P_\theta^{NCE}(w_{i+1}|c_i)+kP_n(w_{i+1}|c_i)}\\ P(D=0|\widetilde{w_{i+1}},c_i) &= \frac{kP_n(\widetilde{w_{i+1}}|c_i)}{P_\theta^{NCE}(\widetilde{w_{i+1}}|c_i)+kP_n(\widetilde{w_{i+1}}|c_i)}\end{aligned} \tag{5}$$

where $\widetilde{w_{i+1}}$ is a word sampled from a known noise distribution P_n (e.g., a uniform distribution) and k is the ratio of the number of noise samples to the number of the data samples. A general assumption is that noise samples are k times more frequent than data samples. Based on this posterior distribution, NCE minimises the following objective function:

$$\begin{aligned}J_{NCE}(\theta) = &-\frac{1}{N}\sum_{i=1}^{N}\Big[\ln P(D=1|w_{i+1},c_i)\\ &+\sum_{j=1}^{k}\ln P(D=0|\widetilde{w_{i+1,j}},c_i)\Big].\end{aligned} \tag{6}$$

which is the same objective function (up to a factor of $\frac{1}{2}$) that is minimised by the traditional logistic regression. Here, for every word w_{i+1} that comes from a true data distribution, k noise samples $\widetilde{w_{i+1,j}}$ are generated from a known noise distribution, P_n. (Mnih and Teh 2012) show that for large k NCE-based parameter estimation is a close approximation of the maximum likelihood estimation.

The gradient of the objective function is as follows:

$$\begin{aligned}\frac{\partial J_{NCE}(\theta)}{\partial\theta} = -\frac{1}{N}\sum_{i=1}^{N}\Bigg[&\frac{kP_n(w_{i+1}|c_i)}{P_\theta^{NCE}(w_{i+1}|c_i)+kP_n(w_{i+1}|c_i)}\\ &\times\frac{\partial}{\partial\theta}\ln P_\theta^{NCE}(w_{i+1}|c_i)\\ &-\sum_{j=1}^{k}\frac{P_\theta^{NCE}(\widetilde{w_{i+1,j}}|c_i)}{P_\theta^{NCE}(\widetilde{w_{i+1,j}}|c_i)+kP_n(\widetilde{w_{i+1,j}}|c_i)}\\ &\times\frac{\partial}{\partial\theta}\ln P_\theta^{NCE}(\widetilde{w_{i+1,j}}|c_i)\Bigg]\end{aligned} \tag{7}$$

where,

$$\begin{aligned}P_\theta^{NCE}(w_{i+1}|c_i) &= \frac{\exp\big(\theta_{i+1}^\top v_i^n\big)}{Z}\\ P_\theta^{NCE}(\widetilde{w_{i+1,j}}|c_i) &= \frac{\exp\big(\theta_j^\top v_i^n\big)}{Z}.\end{aligned} \tag{8}$$

In the softmax gradient in Eq. 4, the normalisation term Z is required to compute P_θ^{SOFT}, which is a problem during training because Z has to be computed for every gradient calculation. Studying the NCE gradient in Eq. 7, one can see a subtraction of two large products. The first terms of those products are normalised implicitly regardless P_θ^{NCE} is normalised or not. Thus, the only terms in which normalisation can matter are the gradients in Eq. 7. It has been argued in the literature, however, that as far as the gradient is concerned, Z can be learnt as a parameter (Gutmann and Hyvärinen 2010) or it can be seen as a constant, for instance, $Z = 1$ was used for all contexts in (Mnih and Teh 2012; Vaswani et al. 2013; Zoph et al. 2016). This is the precise reason why the partition function Z does not have to be computed in every iteration in NCE.

3 Our Approach

In this section, we present our training procedure with special hyperparameter tuning for NCE-based neural language models (NCENLM). For training the model, stochastic gradient descent (SGD) is used because SGD yields significantly better generalisation than batch methods (Bousquet and Bottou 2008), specially when an appropriate learning rate schedule is utilised. The following hyperparameters turned out to be important for the NCENLM: the learning rate schedule, the dropout rate, and the weight initialisation strategy. In our paper, the words are represented as the word vectors (Mikolov et al. 2013) trained using the standard continuous skip-gram model with negative sampling.

3.1 Learning rate

The learning rate is one of the most prominent hyperparameters in deep network training (Bengio 2012). The 'search-then-converge' learning rate schedule for SGD usually has the form of $\eta(t) = \eta_0(1 + \frac{t}{\tau})^{-1}$ (Darken and Moody 1991), where t is the epoch number, η_0 is the initial learning rate, τ is a parameter, and $\eta(t)$ is the learning rate for epoch t. This allows the learning rate to stay high during the 'search period' $t \leq \tau$. It is expected that during this period the parameters will hover around a good minimum. Then, for $t > \tau$, the learning rate decreases as $\frac{1}{t}$, and the parameters converge to a local optimum because this schedule agrees with the stochastic approximation theory (Robbins and Monro 1951).

In our implementation, we used the 'search-then-converge' learning rate schedule of the form:

$$\eta(t) = \eta_0 \times \left(\frac{1}{\psi}\right)^{\max(t+1-\tau, 0.0)}, \tag{9}$$

which previously appeared in other studies that involve neural networks (Zaremba, Sutskever, and Vinyals 2014). The hyperparameter ψ is kept constant in our experiments and its value was set according to (Zaremba, Sutskever, and Vinyals 2014). During the search period ($t \leq \tau$ epochs), the learning rate is constant and equal to η_0, and during the convergence period the learning rate is decreased by a factor of $\frac{1}{\psi}$. The initial learning rate η_0 is one in (Zaremba, Sutskever, and Vinyals 2014), and we use the same value in our experiments.

Our investigation has shown that learning with NCE is more sensitive to the length of the search period ($t \leq \tau$) when comparing with softmax. Choosing appropriate τ is, therefore, crucial for convergence of NCE-based learning with SGD. Our research suggests that, when NCE is used, τ should be between 1 and two-thirds of the total number of training epochs. For instance, if we need 40 training epochs then τ could be between 1 and 26. This was one of the most important insights that allowed us to improve the performance of NCE.

3.2 Weight Initialisation

In neural networks, the initial weights are usually drawn from a uniform distribution (Glorot and Bengio 2010), unit Gaussian (Sutskever et al. 2013) or a general Gaussian distribution (He et al. 2015). For our NCE training, we used a uniform distribution for the weight initialisation. All three distributions described above were compared, but the uniform distribution led to slightly better results. *However, regardless which distribution is used, we found that NCE works much better when the initial weights are within a smaller range, i.e. when the variance of the initial weights is smaller than what is suggested in (Glorot and Bengio 2010). This was another insight that led to significant improvements in NCE performance.*

4 Experimental Methodology and Implementation

We aim at showing that NCE can outperform alternative methods for language modelling. In particular, we investigate its performance in the context of softmax because NCE approximates softmax being consistent with softmax in the limit. Our implementation of NCE follows our approach presented in Sec. 3.

In our experiments, we focus on the popular perplexity measure (PPL) using the Penn Tree Bank (PTB) dataset. It is feasible to run 'exact' softmax on this dataset, and the large literature that uses it allows for comparisons with other approaches (see Tab. 1). The PTB dataset consists of 929k training words, 73k validation words, and 82k test words. The vocabulary size is 10k. Softmax usually becomes inefficient when the vocabulary size exceeds 10k words.

All models were implemented in Tensorflow[2] and executed on NVIDIA K80 GPUs. The standard components of our models follow (Zaremba, Sutskever, and Vinyals 2014) where excellent results on this dataset were reported. The words were represented with dense vectors trained using the skip-gram model with negative sampling (Mikolov et al. 2013) on the Wikipedia corpus downloaded on December 2016. This word representation was used across all our experiments.

In order to perform more experiments, we designed models of three sizes: small (S), medium (M) and large (L). The small model is non-regularised whereas the medium and large models are dropout regularised with 50% and 60% dropout rate on the non-recurrent connections in the medium and larger models correspondingly. This led to the best empirical results after investigating different dropout rates in the suggested ranges (Srivastava et al. 2014). All the models have two LSTM layers with the hidden layer size of 200 (S), 650 (M), and 1500 (L). The LSTM was unrolled for 20 time steps for the small model and 35 time steps for the medium and large models. We used mini-batch SGD for training where the mini batch size was 20.

For sampling the initial weights, a smaller range than the one suggested in (Glorot and Bengio 2010) turned out to be beneficial for NCE. We tested several initialisation heuristics which are described in the corresponding column in Tab. 3. Row number 1 shows the formula suggested in (Glorot and Bengio 2010). Note that U denotes a uniform distribution with its minimum and maximum values, and n_i denotes the number of hidden nodes in layer i. Row number 2 shows our updated formula that reduces the first range by a factor of 4, and row number 3 shows the range values that led to the best results in our experiments.

[2]`https://www.tensorflow.org/`

Table 1: Comparison with the state-of-the-art results of different models on the PTB dataset

Classic RNN and LSTM

Model Description	Validation PPL	Test PPL
Deep RNN (Pascanu et al. 2013)	-	107.5
Sum-Prod Net (Cheng et al. 2014)	-	100.0
RNN-LDA + KN-5 + cache (Mikolov and Zweig 2012)	-	92.0
Conv.+Highway+ regularized LSTM (Kim et al. 2016)	-	78.9
Non regularised LSTM with Softmax (Zaremba, Sutskever, and Vinyals 2014)	120.7	114.5
Medium regularised LSTM with Softmax (Zaremba, Sutskever, and Vinyals 2014)	86.2	82.7
Large regularised LSTM with Softmax (Zaremba, Sutskever, and Vinyals 2014)	82.2	78.4
Non regularised LSTM with NCE (our method)	106.196	102.245
Medium regularised LSTM with NCE (our method)	78.762	75.286
Large regularised LSTM with NCE (our method)	72.726	**69.995**

Extended or Improved LSTM

Model Description	Validation PPL	Test PPL
Variational LSTM (Gal and Ghahramani 2016)	77.3	75.0
Variational LSTM + Weight Tying(Press and Wolf 2016)	75.8	73.2
Pointer Sentinel LSTM (Merity et al. 2016)	72.4	70.9
Variational LSTM + Weight Tying + augmented loss (Inan, Khosravi, and Socher 2016)	71.1	68.5
Variational RHN (Zilly et al. 2016)	71.2	68.5
Variational RHN + Weight Tying (Zilly et al. 2016)	67.9	65.4
Neural Architecture Search with base 8 and shared embeddings Utilises a novel recurrent cell and reinforcement learning (Zoph and Le 2016)	-	62.4

Model Averaging/ Ensembles

Model Description	Validation PPL	Test PPL
38 large regularized LSTMs (Zaremba, Sutskever, and Vinyals 2014)	71.9	68.7
Model averaging with dynamic RNNs and n-gram models (Mikolov and Zweig 2012)	-	72.9

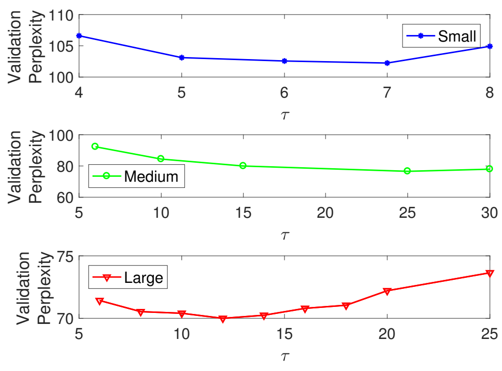

Figure 1: Selection of the learning rate parameter τ

The learning rate was scheduled using Eq. 9. The search time limit τ was chosen empirically using Fig. 1. As a result, τ was set to 7, 25 and 12 for the small, medium and larger models correspondingly. During the convergence period, the parameter ψ was set to 2, 1.2 and 1.15 for the small, medium and larger models as suggested by (Zaremba, Sutskever, and Vinyals 2014). We trained the models for 20, 39 and 55 epochs respectively.

The norm of the gradients (which was normalised by the mini batch size) was clipped at 5 and 10 for the medium and large models correspondingly. To compute validation and testing perplexity, we used softmax to guarantee accuracy of our comparisons.

In NCE, we used 600 noise samples. The noise samples were generated from the power law distribution. We evaluated different noise sample sizes (50, 100, 150, 300, 600, and 1200), and 600 had the best trade-off between quality and time. We observed that when a GPU implementation is used, it is possible to increase the sample size within a reasonable range without dramatically increasing the computational complexity.

Our softmax-based language model was implemented and parametrised accroding to (Zaremba, Sutskever, and Vinyals 2014) where it achieved the state-of-the-art results using the standard LSTM network.

The two models that we implemented, i.e. softmax- and NCE-based language models, used a standard LSTM network (Hochreiter and Schmidhuber 1997; Gers 2001). Many extensions to the LSTM architecture exist, e.g., Recurrent Highway Networks (RHN) (Zilly et al. 2016), that may improve LSTM's capabilities in capturing long term dependencies. In this paper, we aimed at comparing NCE and softmax on standard LSTM networks, but our results could generalise to other, potentially more advanced types of LSTM cells. It should be noted, however, that our NCE implementation with standard LSTM outperforms some language models which use more advanced versions of LSTM as shown in Tab. 1.

5 Results and discussion

The results in Tab. 1 compare our best NCE-based result with other state-of-the-art methods. Our result is the best

Table 2: Comparison of softmax and NCE

Large Model	Time	Valid. PPL	Test PPL
Softmax (55 epochs)	9 h 11 min	82.588	78.826
Softmax (20 epochs)	3 h 40 min	79.798	76.935
NCE (55 epochs)	7 h 34 min	72.726	69.995
NCE (20 epochs)	2 h 36 min	76.268	74.129

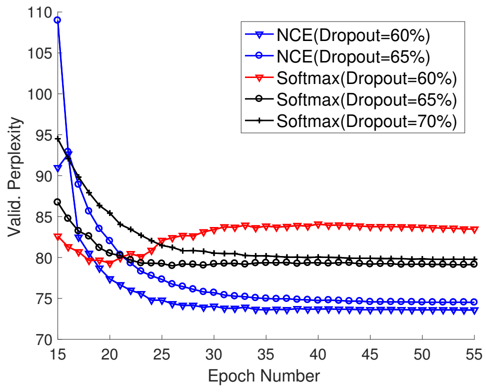

Figure 2: Convergence phase in the large model

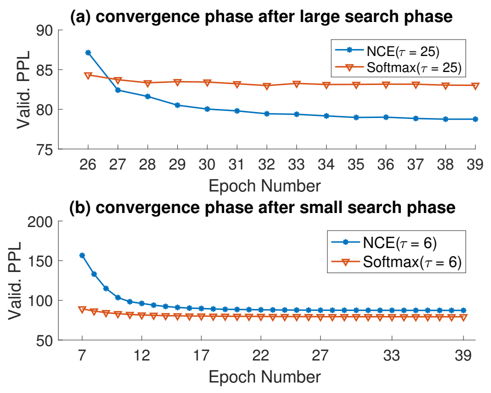

Figure 3: Convergence phase in the medium model

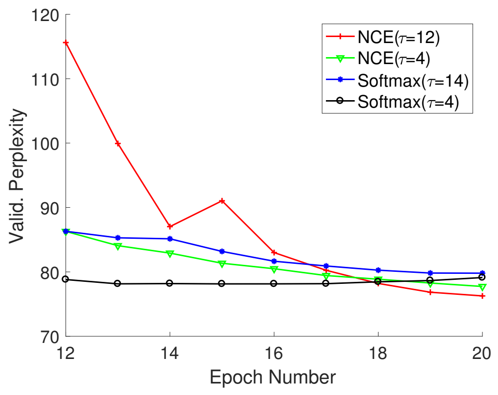

Figure 4: Convergence phase in the large model

in the class of single-model methods that use a standard LSTM; the large model achieved the perplexity of **69.995** after 55 training epochs. This result outperforms all known single-model algorithms that use the same kinds of LSTM cells. The total time for training, validating and testing our large NCE-based model was 7 hours 34 minutes (see Tab. 2). The 55 epochs of softmax took 9 hours 11 minutes, and the testing perplexity was 78.826. Early stopping, which is a common regularisation method (Goodfellow, Bengio, and Courville 2016), allowed softmax to achieve a testing perplexity of 76.935. So, softmax was clearly overfitted after 55 epochs. The same overfitting was not observed in NCE as can be seen in Tab. 2 and Fig. 2.

Below, we present additional results that explain the good performance achieved by NCE and provide further insights into its properties.

Figure 2 presents the validation perplexity (Y axis) of a large model for different dropout rates as a function of an epoch number (X axis) at the convergence stage of learning. One can see that softmax with a dropout rate of 60% overfitted since the 21st epoch. Increasing the dropout rate to 70% allowed softmax to avoid overfitting, but the asymptotic performance was not as good as in NCE. The asymptotic convergence of NCE was superior across a range of dropout rates. In NCE, the gradients (Eq. 7) are different and more noisy than in softmax (Eq. 4). We know that SGD leads to better generalisation than batch gradient descent because of the induced noise by updating the parameters from a single example (Bousquet and Bottou 2008). Similar property of NCE could justify its robust generalisation in Fig. 2.

Figures 3 and 4 show the validation perplexity (Y axis) for selected values of τ as a function of an epoch number (X axis) at the convergence stage of learning. These figures demonstrate the critical impact of the learning rate schedule on NCE. Figure 3 shows that NCE requires a long search period (large $\tau = 25$) to achieve competitive asymptotic convergence on the medium model. Figure 4 for the large model has additional evidence that a long search period is required because larger $\tau = 12$, in addition to having better asymptotic convergence, has poor (i.e. high) perplexity in the initial phase. This poor perplexity indicates that the algorithm explores widely at this stage, but by doing that it can avoid converging to the nearest local optima. High initial perplexity is even more pronounced in Fig. 5 which is for all epochs of the medium model (note that perplexity is on the log scale here). Although difficult to see in the figure, the asymptotic validation perplexity is the best for NCE with $\tau = 25$. There was also a difference in test performance between NCE and softmax: NCE with $\tau = 25$ scored 75.959, NCE with $\tau = 6$ scored 83.858, softmax with $\tau = 25$ scored 79.906, and softmax with $\tau = 6$ achieved 78.567. NCE with high τ was clearly the best, and increasing τ from 6 to 25 reduced perplexity from 83.858 to 75.959, which confirms the significance of our arguments in Sec. 3. Thanks to the noise samples, NCE can explore better than softmax when the exploration phase is long enough which is confirmed through high perplexity in the initial stage of learning. This means

Table 3: Weight initialisation ranges for the uniform distribution (U) and the corresponding test perplexity (PPL)

No.	Initialisation Heuristic	Small Model	Medium Model	Large Model
1	$U\left(-\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}\right)$	U(-0.1225, 0.1225) PPL = 104.449	U(-0.0679, 0.0679) PPL = 75.960	U(-0.04472, 0.04472) PPL = 71.184
2	$U\left(-\frac{\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}}{4}, \frac{\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}}{4}\right)$	U(-0.031, 0.031) PPL = 102.245	U(-0.0169, 0.0169) PPL = 75.959	U(-0.011180, 0.011180) PPL = 70.444
3	Empirically Tuned Ranges	U(-0.0153, 0.0153) PPL = 102.237	$U(-0.00849, 0.00849)$ PPL = 75.286	U(-0.00625, 0.00625) PPL = 69.995

that NCE can find a better solution potentially for the same reasons which make stochastic gradient descent better than batch gradient descent (Bottou 2010).

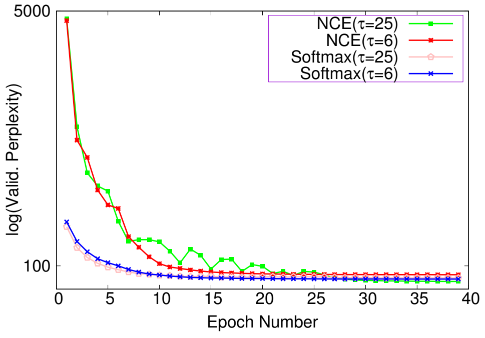

Figure 5: Validation perplexity of the medium model during all epochs of learning

Our results in Fig. 4 and 5 indicated that good NCE results could be attributed to its high error, i.e., high perplexity, in the early stages of learning which may allow for broad exploration. We tried to enforce similar behaviour in softmax using a large learning rate in the search period. Figure 6(a) presents the validation perplexity in the log scale for a large model with softmax (Y axis) as a function of a training epoch (X axis) and the learning rate (LR) which was increased to 1, 2, and 3 during search time. This arrangement increased the validation perplexity for the first few epochs, but the asymptotic convergence of softmax was not improved. When, in Fig. 6(b), we compare the increased initial softmax perplexity with NCE perplexity during the progression of the first epoch, we can see that NCE has much larger perplexity at this stage even though its learning rate is not larger than one. It might be a distinct characteristic of the NCE that helps to converge to a **better local optimum** due to the initial high training error.

The numerical entries in Tab. 3, i.e., in all cells in the bottom right part of the table, contain both the intervals U used to sample initial weights and the resulting perplexity (PPL) on a corresponding model. The results on the large model show that weight initialisation with lower variance led to better results, where the best perplexity of 69.995 was the best result that NCE achieved in our experiments.

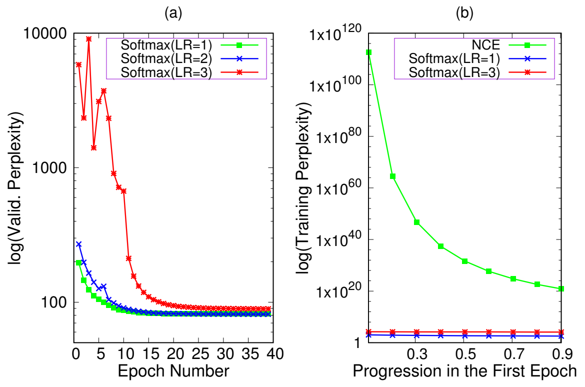

Figure 6: High learning rate (LR) to increase the initial softmax perplexity (a). NCE and softmax initial perplexities in the first epoch; note that only training perplexity is available within one epoch (b).

6 Conclusion

Language modelling techniques can use Noise Contrastive Estimation (NCE) to deal with the partition function problem during learning. Although it was known that NCE can outperform softmax (which computes the exact partition function) on large problems which are too big for softmax, its performance has never been shown to outperform softmax or other methods on tasks on which softmax is feasible and works well. In this paper, we showed that NCE can beat all the previously best results in the class of single-model methods based on a standard LSTM and standard dropout achieving perplexity of 69.995. Our result establishes a new standard on the Penn Tree Bank dataset reducing the perplexity of the best existing method in this class by 8.405.

References

Baltescu, P., and Blunsom, P. 2015. Pragmatic neural language modelling in machine translation. In *Proc. of HLT-NAACL*, 820–829.

Bengio, Y., and Sénécal, J.-S. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proc. of AISTATS*.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003.

A neural probabilistic language model. *JMLR* 3(Feb):1137–1155.

Bengio, Y. 2012. *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Springer Berlin Heidelberg. 437–478.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proc. of COMPSTAT*. Springer. 177–186.

Bousquet, O., and Bottou, L. 2008. The tradeoffs of large scale learning. In *Proc. of NIPS*, 161–168.

Chen, W.; Grangier, D.; and Auli, M. 2015. Strategies for training large vocabulary neural language models. *CoRR* abs/1512.04906.

Cheng, W.-C.; Kok, S.; Pham, H. V.; Chieu, H. L.; and Chai, K. M. A. 2014. Language modeling with sum-product networks. In *Interspeech 2014*.

Darken, C., and Moody, J. E. 1991. Note on learning rate schedules for stochastic optimization. In *Proc. of NIPS*, 832–838.

Gal, Y., and Ghahramani, Z. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. of NIPS*, 1019–1027.

Gers, F. 2001. *Long short-term memory in recurrent neural networks*. Ph.D. Dissertation, Universität Hannover.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, 249–256.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Gutmann, M., and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*, volume 1.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of ICCV*, 1026–1034.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hyvärinen, A. 2005. Estimation of non-normalized statistical models by score matching. *JMLR* 6(Apr):695–709.

Inan, H.; Khosravi, K.; and Socher, R. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR* abs/1611.01462.

Jean, S.; Cho, K.; Memisevic, R.; and Bengio, Y. 2014. On using very large target vocabulary for neural machine translation. *CoRR* abs/1412.2007.

Józefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *Proc. of AAAI*, 2741–2749.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *CoRR* abs/1609.07843.

Mikolov, T., and Zweig, G. 2012. Context dependent recurrent neural network language model. In *Proc. of SLT*, 234–239.

Mikolov, T.; Karafit, M.; Burget, L.; Cernock, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Interspeech*, 1045–1048.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*. 3111–3119.

Mnih, A., and Hinton, G. E. 2009. A scalable hierarchical distributed language model. In *Proc. of NIPS*, 1081–1088.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*.

Pascanu, R.; Gülçehre, Ç.; Cho, K.; and Bengio, Y. 2013. How to construct deep recurrent neural networks. *CoRR* abs/1312.6026.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *Proc. of ICML*, 1310–1318.

Press, O., and Wolf, L. 2016. Using the output embedding to improve language models. *Proc. of EACL* 157–163.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics* 400–407.

Schwenk, H. 2007. Continuous space language models. *Computer Speech & Language* 21(3):492 – 518.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Sutskever, I.; Martens, J.; Dahl, G. E.; and Hinton, G. E. 2013. On the importance of initialization and momentum in deep learning. *ICML (3)* 28:1139–1147.

Vaswani, A.; Zhao, Y.; Fossum, V.; and Chiang, D. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*, 1387–1392.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zilly, J. G.; Srivastava, R. K.; Koutník, J.; and Schmidhuber, J. 2016. Recurrent highway networks. *CoRR* abs/1607.03474.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv:1611.01578*.

Zoph, B.; Vaswani, A.; May, J.; and Knight, K. 2016. Simple, fast noise-contrastive estimation for large rnn vocabularies. In *Proc. of HLT-NAACL*.