Complementary Sum Sampling for Likelihood Approximation in Large Scale Classification

Aleksandar Botev[1] **Bowen Zheng**[1] **David Barber**[1,2]
University College London[1] Alan Turing Institute[2]

Abstract

We consider training probabilistic classifiers in the case that the number of classes is too large to perform exact normalisation over all classes. We show that the source of high variance in standard sampling approximations is due to simply not including the correct class of the datapoint into the approximation. To account for this we explicitly sum over a subset of classes and sample the remaining. We show that this simple approach is competitive with recently introduced non likelihood-based approximations.

1 Probabilistic Classifier

Given an input x, we define a distribution over class labels $c \in \{1, \dots, C\}$ as

$$p_\theta(c|x) = \frac{u_\theta(c,x)}{Z_\theta(x)}, \qquad u_\theta \geq 0 \tag{1}$$

with normalisation

$$Z_\theta(x) \equiv \sum_{c=1}^{C} u_\theta(c,x) \tag{2}$$

Here θ represents the parameters of the model. A well known example is the softmax model in which $u_\theta(c,x) = \exp(s_\theta(c,x))$, with a typical setting for the score function $s_\theta(c,x) = \mathbf{w}_c^\mathsf{T}\mathbf{x}$ for input vector $\mathbf{x}$ and parameters $\theta = \{\mathbf{w}_1, \dots, \mathbf{w}_C\}$.

Computing the exact probability requires the normalisation to be computed over all C classes and we are interested in the situation in which the number of classes is large. For example, in language models, it is not unusual to have of the order of $C = 100,000$ classes, each class corresponding to a specific word. In maximum likelihood parameter estimation, this causes a bottleneck in the computation. Particularly in language modelling several attempts have been considered to alleviate this difficulty. Early attempts were to approximate the normalisation term by Importance Sampling [1, 2]. Alternative, non-likelihood based training approaches such as Noise Contrastive Estimation [7, 14], Negative Sampling [12] and BlackOut [10] have been recently introduced. Whilst each approach has its merits, we introduce a very simple modification of a standard sampling approximation and find that this gives excellent comparative performance.

1.1 Maximum Likelihood

Given data $\mathcal{D} \equiv \{(x_n, c_n) \,|\, n = 1, \dots, N\}$ a natural[1] way to train the model is to maximise the log likelihood $L(\theta) \equiv \sum_{n=1}^{N} L_n(\theta)$ where the log likelihood of an individual datapoint is

$$L_n(\theta) = \log p_\theta(c_n|x_n) = \log u_\theta(c_n, x_n) - \log Z_\theta(x_n)$$

The gradient is given by $g(\theta) = \sum_{n=1}^{N} g_n(\theta)$ where the gradient associated with an individual datapoint n is given by

$$g_n(\theta) = \partial_\theta \log u_\theta(c_n, x_n) - \frac{1}{Z_\theta(x_n)} \sum_{c=1}^{C} \partial_\theta u_\theta(c, x_n)$$

The gradient can be expressed as a weighted combination of vectors,

$$g_n(\theta) = \sum_{c=1}^{C} \gamma_n(c) \partial_\theta \log u_\theta(c, x_n) \tag{3}$$

with weights given by[2]

$$\gamma_n(c) \equiv \delta(c, c_n) - p_\theta(c|x_n) \tag{4}$$

where the Kronecker delta $\delta(c, c_n)$ is 1 if $c = c_n$ and zero otherwise. This has the natural property that

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

[1] Maximum Likelihood has the well-known property that is is asymptotically efficient.

[2] We drop the dependence of γ on θ for convenience.

when the model $p_\theta(c|x_n)$ predicts the correct class label c_n for each datapoint n, then the gradient is zero. Since $p \in [0, 1]$, we note that the weights are bounded $\gamma_n(c) \in [-1, 1]$.

In practice, rather than calculating the gradient on the full batch of data, we use a much smaller randomly sampled minibatch $\mathcal{M}$ of datapoints, $|\mathcal{M}| \ll N$ (typically of the order of 100 examples) and use the minibatch gradient $g(\theta) = \sum_{n\in\mathcal{M}} g_n(\theta)$ to update the parameters at each iteration (for example by Stochastic Gradient Ascent).

2 Approximating a Sum by Sampling

The exact log-likelihood requires the calculation of a sum over all classes, which we assume to be prohibitively expensive. We consider therefore the general problem of summing over a collection of elements $Z = \sum_{c=1}^{C} z_c$.

2.1 Importance Sampling

A standard approach to estimating a sum is to use Importance Sampling (IS). Based on the identity $Z = \sum_c q(c)z_c/q(c)$ we draw a bag of indices $\mathcal{S} \equiv \{i_1, \ldots, i_S\}$ from q, where $i_s \in \{1, \ldots, C\}$, and form the approximation

$$\tilde{Z} = \frac{1}{S}\sum_{s\in\mathcal{S}} \frac{z_s}{q(s)} \tag{5}$$

where the sum is over all elements of $\mathcal{S}$, including repetitions. $\tilde{Z}$ is an unbiased estimator of Z with variance

$$\frac{1}{S}\left(\sum_{c=1}^{C} \frac{z_c^2}{q(c)} - Z^2\right) \tag{6}$$

Even as the number of samples S is increased beyond the number of elements C, $\tilde{Z}$ generally remains an approximation (unless $q(c) \propto z_c$). That is, even if the method uses *more* computation than the exact calculation would require, it remains an approximation. For this reason, we consider an alternative sampling approach with bounded computation.

2.2 Bernoulli Sampling

An alternative to IS is to consider the identity

$$Z = \sum_{c=1}^{C} z_c = \mathbb{E}_{\mathbf{s}\sim\mathbf{b}}\left(\sum_{c=1}^{C} \frac{s_c}{b_c} z_c\right) \tag{7}$$

where each independent Bernoulli variable $s_c \in \{0, 1\}$ and $p(s_c = 1) = b_c$. Unlike IS, no samples can be repeated. We propose to take a single joint sample $\mathbf{s}$ to form the Bernoulli sample approximation

$$Z \approx \sum_{c:s_c=1} \frac{z_c}{b_c} \tag{8}$$

where the sum is over the components corresponding to the non-zero elements of the vector $\mathbf{s}$. This Bernoulli sampler of Z is unbiased with variance

$$\sum_{c=1}^{C}\left(\frac{1}{b_c} - 1\right) z_c^2 \tag{9}$$

For $b_c \to 1$, s_c is sampled in state 1 with probability 1 and the approximation recovers the exact summation.

3 Likelihood Approximation by IS

For the classification model the log likelihood contribution for an individual datapoint n is given by

$$L_n(\theta) = \log u_\theta(c_n, x_n) - \log Z_\theta(x_n) \tag{10}$$

A standard approximation [6, 1] of Z uses IS[3]. By drawing a bag of samples $\mathcal{S}$ from an importance distribution q over the C classes, a sampling approximation is then given by

$$Z_\theta(x_n) \approx \frac{1}{S}\sum_{s\in\mathcal{S}} \frac{u_\theta(s, x_n)}{q(s)} \tag{11}$$

This approximate Z is used to replace the exact Z in eq(10). Gradients are then taken based on this approximate objective, which will now only contain a summation over the sampled classes, not all classes. We call this the ‘standard’ IS approach.

Whilst this is an unbiased estimator of Z (but not $\log Z$), the variance of this estimator is high [2] and in practice can cause divergent parameter updates.

3.1 IS Gives Unstable Updates

This standard IS approach has the undesirable property that the approximated gradient scalar (from eq(4))

$$\gamma_n(c) \approx \delta(c, c_n) - \frac{u_\theta(c, x_n)}{\frac{1}{S}\sum_{s\in\mathcal{S}} \frac{u_\theta(s,x_n)}{q(s)}} \tag{12}$$

is not bounded between −1 and 1. This can create highly inaccurate gradient updates; even the sign of the gradient direction $g_n(\theta)$ is not guaranteed to be correct. Whilst there have been attempts to adapt q

[3] Note that this approximation is an unbiased estimator of Z, but a *biased* estimator of $\log Z$. The resulting approximate gradient is thus also biased.

to reduce the variance these are typically more complex and may require significant computation to correct wild gradient estimates [2].

Note that the standard IS gradient approximation being not bound between 0 and 1 is likely to happen in practice. As training progresses, the term $p_\theta(c_n|x_n)$ should ideally approach 1 as the model learns to classify the training points accurately. As the probability of the correct class approaches 1, $u_\theta(c_n, x_n)$ dominates all $u_\theta(d, x_n)$, for $d \neq c_n$. For this scenario, and considering the IS approximation:

$$p_\theta(c_n|x_n) \approx \frac{u_\theta(c_n, x_n)}{\frac{1}{S}\sum_{s\in\mathcal{S}} \frac{u_\theta(s,x_n)}{q(s)}} \tag{13}$$

if the term $u_\theta(c_n, x_n)$ is not included in the sample bag $\mathcal{S}$, the ratio will be very large and the gradient estimate highly inaccurate. Thus, unless q accounts for the importance of including the sample class c_n, it is almost guaranteed that learning will become unstable.

We show this phenomenon in fig(1) in which we consider evaluating an expression of the form

$$p(c = 1|x) = \frac{u(1)}{\sum_{c=1}^{C} u(c)} \tag{14}$$

for different randomly chosen u. Using IS with a uniform q to approximate the denominator

$$p(c = 1|x) \approx \frac{u(1)}{\frac{1}{S}\sum_{s\in\mathcal{S}} \frac{u(s)}{q(s)}} \tag{15}$$

gives rise to highly inaccurate estimates, see fig(1). However, there is a simple 'fix' for this, as we describe in the following section.

4 Complementary Sum Sampling

For each datapoint n we define a small set of classes $\mathcal{C}_n$ that are explicitly summed over in forming the approximation. This defines for each datapoint n a complementary set of classes $\mathcal{C}_n^c$, (all classes except for those in $\mathcal{C}_n$). We can then write

$$Z_\theta(x_n) = \sum_{c\in\mathcal{C}_n} u_\theta(c, x_n) + \sum_{d\in\mathcal{C}_n^c} u_\theta(d, x_n) \tag{16}$$

We propose to simply approximate the sum over the complementary classes by sampling. To ensure that this results in an approximate $-1 \leq \gamma_n(c) \leq 1$, we require that $\mathcal{C}_n$ contains the correct class c_n. This setting significantly reduces the variance in the sampling estimate of the gradient.

In our experiments, we simply set $\mathcal{C}_n = \{c_n\}$. That is, the normalisation approximation explicitly includes the correct class with the remaining sum over the 'negative' classes approximated by sampling.

In using sampling to approximate the complementary class summation in eq(16), both Importance and Bernoulli Sampling give approximations in the form

$$\tilde{Z}_\theta(x_n) = \sum_{c\in\mathcal{C}_n} u_\theta(c, x_n) + \sum_{d\in\mathcal{N}_n} \kappa_{d,n} u_\theta(d, x_n) \tag{17}$$

where $\mathcal{N}_n$ is a bag of S negative sampled classes from the complementary set. In the Importance case[4], $\kappa_{d,n} = 1/(Sq_n(d))$; in the Bernoulli case $\kappa_{d,n} = 1/b_{d,n}$ is the probability that $s_d = 1$.

The approximate log likelihood contribution for an individual datapoint n is then given by

$$\tilde{L}_n(\theta) = \log u_\theta(c_n, x_n) - \log \tilde{Z}_\theta(x_n) \tag{18}$$

with derivative

$$\partial_\theta \tilde{L}_n(\theta) = \sum_{c\in\mathcal{C}'_n} \left(\delta(c, c_n) - \tilde{p}(c|x_n)\right) \partial_\theta \log u_\theta(c, x_n)$$

where

$$\tilde{p}(c|x_n) = \begin{cases} u_\theta(c, x_n)/\tilde{Z}_\theta(x_n) & c \in \mathcal{C}_n \\ \kappa_{c,n} u_\theta(c, x_n)/\tilde{Z}_\theta(x_n) & c \in \mathcal{N}_n \end{cases} \tag{19}$$

Note that $\tilde{p}$ is a distribution[5] over the classes $\mathcal{C}'_n = \mathcal{C}_n \cup \mathcal{N}_n$. Hence the approximation

$$\tilde{\gamma}_n(c) \equiv \delta(c_n, c) - \tilde{p}(c|x_n) \tag{20}$$

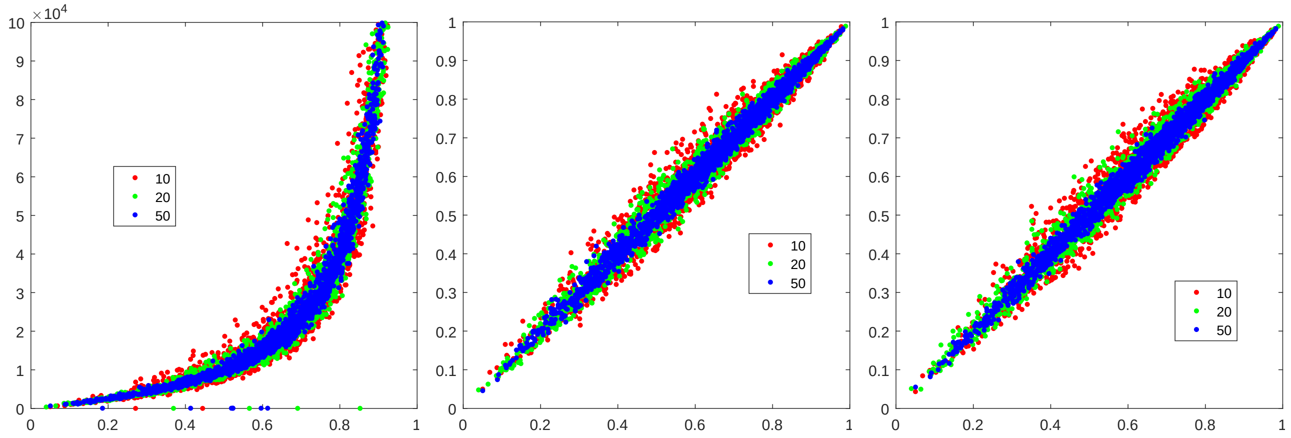
has the property $\tilde{\gamma}_n(c) \in [-1, 1]$.

The estimator of $p(c|x_n)$ is biased since the estimator of the inverse normalisation $1/Z_\theta$ is biased[6]. In the limit of a large number of Importance samples, $\kappa \to 1$; similarly for Bernoulli Sampling, as S tends to the size of the complementary set, $\kappa \to 1$. In this limit $\tilde{Z}_\theta(x_n)$ approaches the exact value $Z_\theta(x_n)$. The resulting estimators of $p(c|x_n)$ are therefore consistent.

[4] The IS distribution $q_n(d)$ depends on the data index n, since the IS distribution must not include the classes in the set $\mathcal{C}_n$.

[5] In [9] IS is used to motivate an approximation that results in a distribution over a predefined subset of the classes. However the approximation is based on a biased estimator of $Z(x_n)$ and as such is not an Importance sampler in the standard sense.

[6] One can form an effectively unbiased estimator of $1/Z_\theta$ by a suitable truncated Taylor expansion of $1/Z$, see [3]. However, each term in the expansion requires a separate independent joint sample from $p(\mathbf{s})$ (for the BS) and as such is sampling intensive, reducing the effectiveness of the approach.



(a) (b) (c)

Figure 1: Approximating the probability $p(c = 1|x)$, eq(14) for $C = 10000$ classes. Here the $u(i)$, $i > 1$ elements are sampled uniformly between $[0, 1]$ and $u(1) = \exp(Cy)$, where y is sampled from a normal distribution. On the horizontal axis we plot the exact value $p(c = 1|x)$ and on the vertical axis the approximation. The colours correspond to the number of samples used in the approximation, $S \in \{10, 20, 50\}$. (a) Using standard Importance Sampling with a uniform importance distribution to approximate the normalisation $\sum_{c=1}^{C} u(c)$. This gives a wildly inaccurate estimate of the probability since the term $u(1)$ is unlikely to be included in the importance samples. This is already a significant issue for small $p(c = 1|x)$ and becomes increasingly problematic as $p(c = 1|x)$ increases towards 1 and the classifier fits the training data more accurately. (b) Complementary Sum Sampling with Importance Sampling (uniform importance distribution) to approximate the term $\sum_{c=2}^{C} u(c)$. (c) Complementary Sum Sampling with Bernoulli Sampling (uniform) to approximate $\sum_{c=2}^{C} u(c)$.

4.1 CSS Gives Stable Updates

We return to estimating eq(14). Using CSS with a single member of $\mathcal{C}_n$ is equivalent to the approximation (for Importance Sampling)

$$p(c = 1|x) \approx \frac{u(1)}{u(1) + \frac{1}{S}\sum_{s\in\mathcal{S}} \frac{u(s)}{q(s)}} \tag{21}$$

where q is a distribution over the negative indices $\{2, \dots, C\}$ (hence the sample bag $\mathcal{S}$ does not contain index 1), and for Bernoulli Sampling:

$$p(c = 1|x) \approx \frac{u(1)}{u(1) + \sum_{c:s_c=1}^{C} \frac{u(c)}{b_c}} \tag{22}$$

where the negative indices are such that $c \neq 1$.

As we see in fig(1) this simple modification dramatically reduces the error in the approximation compared to using the standard IS approach eq(15). In this case there is no significant difference between using CSS with Bernoulli or Importance Sampling to approximate the probability.

5 Relation to Other Approaches

The closest approaches to ours are taken in [1, 2] which use the maximum likelihood objective, approximated by 'standard' IS, eq(11). Since this has high variance, alternatives have been considered by several authors.

5.1 Hierarchical Softmax

Hierarchical softmax [15] defines a binary tree such that the probability of a leaf class is the product of edges from the root to the leaf. Each (left) edge child of a node i is associated with a probability $\sigma\left(\mathbf{w}_i^\mathsf{T}\mathbf{x}\right)$, with a corresponding weight $\mathbf{w}_i$ for each node. This implicitly defines a distribution over all C classes, removing the requirement to explicitly normalise over all classes. The computational cost of calculating the probability of an observed class then scales with $\log C$, rather than with C in the standard softmax approach. However, this formally defines a new model and as such we will not consider it further here.

5.2 Noise Contrastive Estimation

NCE [7, 8] is a general approach that can be used to perform estimation in unnormalised probability models and has been successfully applied in the context of language modelling in [14, 13]. The method generates data from the 'noise' classes[7] (which range over

[7] In our experiments we tried different noise distributions $p_\nu(c) \propto f(c)^\alpha$ where $f(c)$ is the unigram distribution

all classes, not just the negative classes) for each datapoint in the minibatch.

The objective is related to a supervised learning problem to distinguish whether a datapoint is drawn from the data or noise distribution. The method forms a consistent estimator of θ in the limit of an infinite number of samples $i_1, \dots, i_S$ $(S \to \infty)$ independently drawn from a 'noise' distribution $p_\nu(c)$, $c \in \{1, \dots, C\}$. For minibatch datapoint (c_n, x_n) the method has gradient

$$\frac{Sp_\nu(c_n)}{p'_\theta(c_n|x_n) + Sp_\nu(c_n)} \partial_\theta \log p'_\theta(c_n|x_n) - \sum_{s=1}^{S} \frac{p'_\theta(i_s)}{p'_\theta(i_s|x_n) + Sp_\nu(i_s)} \partial_\theta \log p'_\theta(i_s|x_n)$$

where

$$p'_\theta(i_s|x_n) = u_\theta(i_s, x_n)/z_n \qquad (23)$$

and the total gradient sums the gradients over the minibatch. The method requires that each datapoint in the minibatch has a corresponding scalar parameter z_n (part of the full parameter set θ) which approximates the normalisation $Z_n(\theta)$. Formally, the objective is optimised when $z_n = Z_n(\theta)$ which would require an expensive inner optimisation loop for each minibatch over these parameters. For this reason, in practice, these normalisation parameters are set to $z_n = 1$ [14]. Formally speaking this invalidates the consistency of the approach unless the model is rich enough that it can implicitly approximate the normalisation constant[8].

In the limit of the number of noise samples tending to infinity, the optimum of the NCE objective coincides with maximum likelihood optimum. A disadvantage of this approach is that (in addition to the formal requirement of optimising over the z_n) it requires unbounded computation to exactly match the maximum likelihood objective. Whilst this method has been shown to be effective for complex 'self normalising' models, in our experiments with softmax regression this approach (setting $z_n = 1$) did not lead to a practically usable algorithm.

5.3 Ranking Approaches

An alternative to learning the parameters of the model by maximum likelihood is to argue that, when the correct class is c_n, we need $u_\theta(c_n, x_n)$ to be greater than $u_\theta(d, x_n)$ for all classes $d \neq c_n$. For example in the softmax regression setting $u_\theta(c, x) = \exp(s_\theta(c, x))$ we may stipulate that the score of the correct class $s_\theta(c, x)$ is greater than the scores of the incorrect classes, namely

$$s_\theta(c_n, x_n) - s_\theta(d, x_n) > \Delta, \qquad d \neq c_n \qquad (24)$$

for some positive constant Δ. This is the hinge loss ranking approach taken in [5] in which, without loss of generality, $\Delta = 1$ is used. A minor modification that results in a differentiable objective is to maximise the log ranking

$$\log \sigma(s_\theta(c_n, x_n) - s_\theta(d, x_n) - \Delta) \qquad (25)$$

where $\sigma(x) = 1/(1 + e^{-x})$.

There is an interesting connection between the ranking objective and CSS. Following section(4) we write

$$p(c_n|x_n) = \frac{u_\theta(c_n, x_n)}{u_\theta(c_n, x_n) + \sum_{d \neq c_n} u_\theta(d, x_n)} \qquad (26)$$

and use IS with a distribution $q(d)$ over the negative classes (*i.e.* all classes not equal to c_n) to approximate the term $\sum_{d \neq c_n} u_\theta(d, x_n)$. Using a uniform distribution $q(d) = 1/(C-1)$ over the $C - 1$ negative classes, and drawing only a single negative sample $d_n \neq c_n$ then

$$\sum_{d \neq c_n} u_\theta(d, x_n) \approx (C-1) u_\theta(d_n, x_n) \qquad (27)$$

and the approximation to $\log p(c_n|x_n)$ becomes

$$\log \sigma\left(s_\theta(c_n, x_n) - s_\theta(d_n, x_n) - \log(C-1)\right)$$

This matches the ranking objective using the setting $\Delta = \log(C - 1)$.

One can therefore view the ranking approach as a single-sample estimate of the CSS maximum likelihood approach. As such, we would generally expect the ranking approach to be inferior to alternative, more accurate approximations to the likelihood. The more general ranking objective is

$$\sum_{n \in \mathcal{M}} \sum_{d \neq c_n} \log \sigma\left(s_\theta(c_n, x_n) - s_\theta(d, x_n) - \Delta\right)$$

for subsets of classes $d \neq c_n$ (one subset for each minibatch member) of randomly selected negative classes $d \neq c_n$.

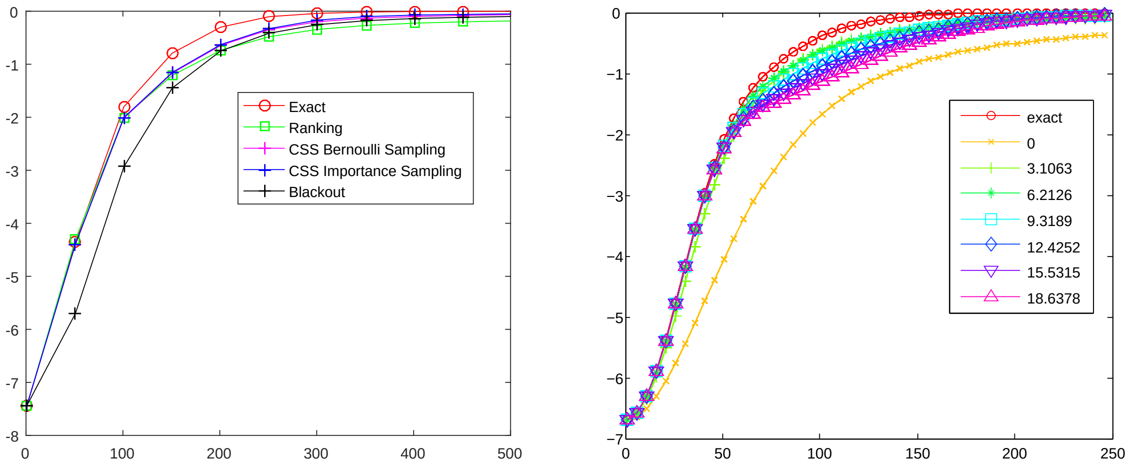
5.3.1 Negative Sampling

A similar approach to ranking is to maximise $\log \sigma(s_\theta(c_n, x_n))$ whilst minimising $\log \sigma(s_\theta(d, x_n))$, for a randomly chosen subset of negative classes $d \neq c_n$. This is motivated in [12] as an approximation of the NCE method and has the objective

$$\log \sigma\left(s_\theta(c_n, x_n)\right) + \frac{1}{S} \sum_d \log\left(1 - \sigma\left(s_\theta(d, x_n)\right)\right) \qquad (28)$$

and α was set to either 0.75 or 1.

[8] This is the assumption in [14] in which the model is assumed to be powerful enough to be 'self normalising'.



(a) (b)

Figure 2: Softmax Regression. (a) Competing training approaches (CSS Bernoulli and CSS IS are virtually indistinguishable). The exact log likelihood (y-axis) is plotted against iteration number (x-axis) for a set of $N = 2000$ datapoints, each $D = 100$ dimensional. There are $C = 1000$ classes and the training data was generated from the model to make a realisable problem. Each minibatch contains $|\mathcal{M}| = 50$ datapoints. In all approximations, $S = 20$ additional 'negative' classes were randomly sampled in addition to the classes in each minibatch. All approximations used roughly 1050 calculations of the form $\exp(\mathbf{w}^\mathsf{T}\mathbf{x})$ per minibatch, compared to 50,000 calculations for the exact approach, leading to a roughly 50 fold decrease in computation cost. Learning rates for the exact and normalisation approximations were all set the same; the BlackOut and Ranking learning rates were set to the largest values that ensured convergence. All methods used gradient ascent with momentum 0.99. The Noise Contrastive Estimation and Negative Sampling approaches are not shown since the 'self normalisation' setting $z_n = 1$ here results in very poor performance. (b) Plotted is the value of the exact log likelihood based on gradient ascent of the ranking objective for different Δ values. In this case $\log(C - 1) = 6.2126$ is the suggested optimal setting.

We included the $1/S$ scale factor so that the negative terms contribute similarly to the positive term. As pointed out in [12] this objective will not, in general, have its optimum at the same point as the log likelihood. The main motivation for the method is that it is a fast procedure which has previously been succesfully used in learing wordvecs [12].

5.4 BlackOut

The recently introduced BlackOut [10] is a discriminative approach based on an approximation to the true discrimination probability. This forms the approximation

$$\tilde{p}(c|x,\theta) = \frac{u_\theta(c,x)/q(c)}{u_\theta(c,x)/q(c) + \sum_{d\in\mathcal{N}_c} u_\theta(d,x)/q(d)} \quad (29)$$

Training maximises the discriminative objective

$$\log \tilde{p}(c_n|x_n,\theta) + \sum_{d\in\mathcal{N}_{c_n}} \log\left(1 - \tilde{p}(d|x_n,\theta)\right) \quad (30)$$

where c_n is the correct class for input x_n and $\mathcal{N}_{c_n}$ is a bag of negative classes for c_n. The objective is summed over all points in the minibatch. BlackOut shares similarities with NCE but avoids the difficulty of the unknown normalisation constant. For the IS distribution the authors propose to use $q(c) \propto f(c)^\alpha$ where $f(c)$ is the empirically observed class distribution $f(c) \propto \sum_n \mathbb{I}[c_n = c]$ and $0 \leq \alpha < 1$ is found by validation. BlackOut also shares similarities with our approach and includes an explicit summation over the class c_n thus avoiding instability. However, the training objective is different – BlackOut uses a discriminative criterion rather than the likelihood.

6 Experiments

6.1 Softmax Regression

We consider the simple softmax regression model $u(c,x) = \exp(\mathbf{w}_c^\mathsf{T}\mathbf{x})$. The exact log-likelihood in this case is concave, as are our approximate objectives. This convexity means that our results do not depend on the difficulty of optimisation and focus on the quality of the objective in terms of mimicking the true log likelihood. The training data is formed by randomly sampling a (fixed) set of parameters θ_0 and inputs $\mathbf{x}_n$

from which we then sample training classes c_n from this softmax regression model. Neither standard IS, Noise Contrastive Estimation (with $z_n = 1$) nor Negative Sampling are given in the results since these perform significantly worse than the other approaches.

In fig(2a) we plot the exact log likelihood against iterations of stochastic gradient ascent, comparing the exact minibatch gradient to our normalisation approximations, ranking and BlackOut. The empirical class frequency $f(c)$ was used to form the IS distribution, with $S = 20$ samples drawn. For Bernoulli Sampling, using $b(c) = f(c)^{0.54}$ results in an expected number of $S = 20$ samples. For this simple problem all displayed methods work well.

In fig(2b) we examine the effect of different separations Δ in the ranking method; we see that Δ can significantly affect the effectiveness of the approach, with $\Delta = \log(C-1)$ being a reasonable setting. This is in line with the discussion in section(5.3) which related ranking to a single sample estimate of the CSS maximum likelihood criterion.

6.2 Neural Word Prediction Model

Our interest is not to find a state-of-the-art word prediction model, but rather to demonstrate how a large model can be trained efficiently, as measured by the true likelihood score. We therefore trained a standard Recurrent Neural Network Language Model to predict the next word in a sentence based on all preceding words. The training data is the English part of the European Parliament Proceedings Parallel Corpus [11].

We converted all words to lower case and tokenized them. We also removed stop words, punctuation and tokens that occurred less than 4 times. Tokens were defined to denote the start and end of each sentence and we discarded any sentence longer than 70 tokens. This processed corpus contains 2 million sentences with 28 million words and $C = 47170$ distinct words.

The prediction model is the Gated Recurrent Unit (GRU) architecture, as described in [4] with 128 hidden units[9]. The GRU update equations are[10]:

$$\begin{aligned} r_t &= \sigma(W_{r,x}x_t + W_{r,h}h_{t-1} + b_r) \\ u_t &= \sigma(W_{u,x}x_t + W_{u,h}h_{t-1} + b_u) \\ c_t &= \sigma(W_{c,x}x_t + r_t \odot (W_{u,h}h_{t-1}) + b_c) \\ h_t &= (1-u_t) \odot h_{t-1} + u_t \odot c_t \end{aligned}$$

At time t the input x_t to the network is the word embedding of the corresponding word. The output of the network is the set of scores $s_t(c) = w_c^T P h_t$ for the classes required by the approximation. Here w_c is the word embedding for class c, h_t is the hidden state of the RNN and P is a projection matrix. The word embeddings were chosen to be 100 dimensional. The predicted probability of the next word is then proportional to the exponentiated score $u_t(c) = \exp(s_t(c))$. The initial hidden states of the GRU are set to zero.

For each training objective we learned the network parameters, word embeddings and projection P by Stochastic Gradient Descent (SGD) optimisation, with parameters uniformly initialized from the interval $[-0.005, 0.005]$. The word embedding matrix was randomly initialized from the uniform distribution over the interval $[-0.1, 0.1]$.

We compare the different approximations by maximising the corresponding approximate objective instead of the true log-likelihood. We evaluate the performance of each training method by the exact log likelihood (on the full training set) of the resulting parameters. For the IS-based methods we chose the Importance Sampling distribution to be proportional to $f(w)^\alpha$, where $f(w)$ is the empirical distribution of words in the training dataset (*i.e.* the unigram). We experimented with different values $\alpha \in \{0, 0.75, 1\}$. Generally $\alpha = 0$, which corresponds to using a uniform importance distribution works less well, and for that reason we plot only the results for $\alpha \in \{0.75, 1\}$.

We compared a set of different approaches[11], see fig(3). As shown in [10], BlackOut and NCE share the same objective function but differ slightly in their details, with BlackOut being found to be a superior method. For this reason, we conducted experiments only with the more competitive BlackOut approach.

Each method requires a bag of S samples, which are generated according to the processes described below. The samples are generated identically for every method in order to aid comparison of the underlying approximations.

Negative Sampling: As in [12] we generated the negative samples according to $f(w)^\alpha$.

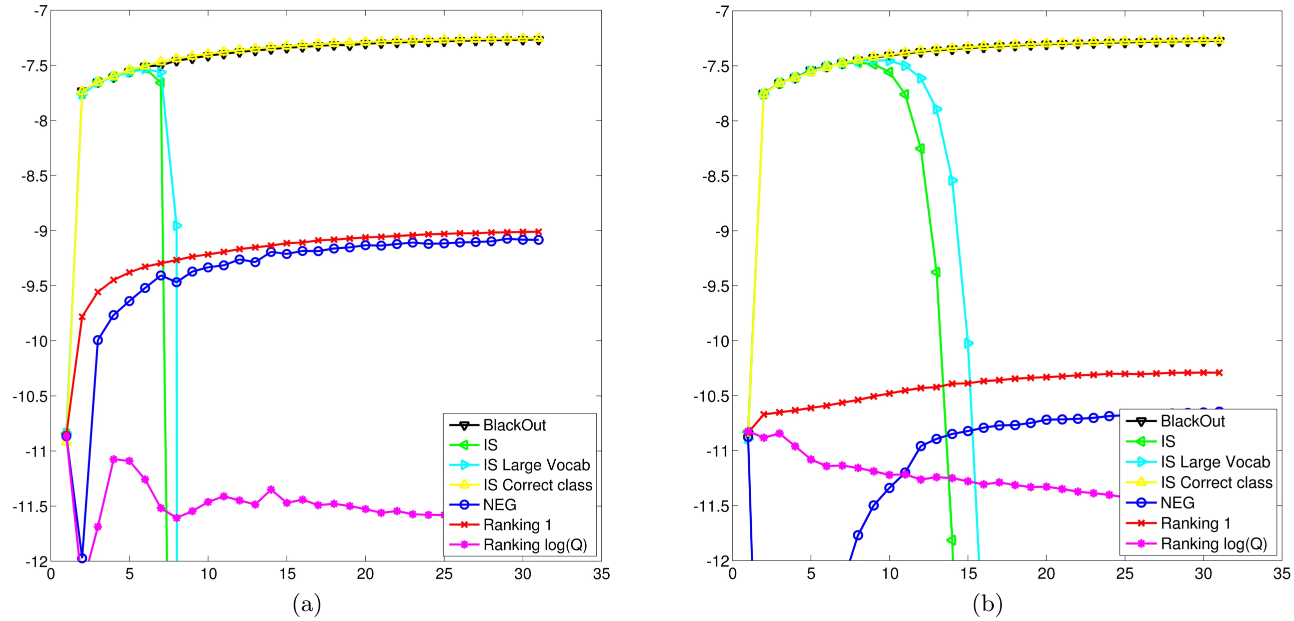
IS Large Vocab: This is the method described in [9]. The method is inspired by IS and uses a softmax style objective over a subset of classes, with sampling proportional to $f(w)^\alpha$.

BlackOut: The method is as described in [10]. Note that BlackOut requires that the negative samples

[9] We also experimented with 100 and 256 hidden units but found no appreciable difference in log likelihood scores.

[10] $\odot$ denotes elementwise multiplication.

[11] We didn't include Bernoulli Sampling since, from our softmax regression experiments, the performance benefits over IS were unclear.



(a)

(b)

Figure 3: Neural Language Model for next word prediction. Plotted is the exact training log likelihood on the full training set against training epochs, for a variety of training methods. All methods rely on sampling 'negative' classes, which are taken from the unigram distribution raised to the power $\alpha = 0.75$ (shown in (a)) and $\alpha = 1$ (shown in (b)). The best methods are BlackOut and our simple CSS IS Correct Class approach.

used for each predicted word in the minibatch cannot contain the correct class.

IS Correct Class: This uses our CSS approach based on eq(16) with $\mathcal{C}_n = \{c_n\}$ and uses IS to approximate the sum over the remaining classes. There is a similar requirement to BlackOut that the negative samples used cannot contain the correct class.

Ranking: We use the ranking objective (section(5.3)). The negative samples again cannot contain the correct class and are drawn from $f(w)^\alpha$.

IS: This is the standard IS approach [1] with samples drawn according to $f(w)^\alpha$.

For all of the methods we used $S = 250$ samples which are shared across every word in every sentence in the minibatch. For the approximations where it is required that for each word the noise samples cannot contain the correct class we sample $S + S'$ samples and for each word individually we pick the first S noise samples which do not coincide with the correct one. We found out that setting $S' = 10$ is more than enough to never encounter a sample where the procedure described above cannot be performed. We trained each model using SGD for 30 epochs and minibatch size of 128. The learning rate at each epoch is defined as $\lambda_i = 0.9^i \lambda_0$, with initial learning rate λ_0 is set to 0.05 (for cases where this led to divergent parameters, we used a lower initial learning rate of 0.02).

The best performing methods are BlackOut and our CSS approach in which we explicitly sum over the correct class and use IS to approximate the remaining sum for Z. Some methods were unstable, including the standard IS approach. The ranking based method was significantly inferior to the likelihood approximation approaches, as was Negative Sampling.

7 Conclusion

The high variance in the classical sampling approximation of the log likelihood gradient in probabilistic classification is caused by not including a term corresponding to the correct class in the normalisation. Without explicitly including this term training is likely to be unstable.

We introduced Complementary Sum Sampling to stabilise the calculation of gradients in parameter learning. The method is competitive against recent non-likelihood based approaches, with good performance on a large-scale word prediction problem. The advantage of our approach is its simplicity and direct connection to the standard and asymptotically efficient maximum likelihood objective.

Acknowledgements

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

[1] Y. Bengio and J-S. Senécal. Quick Training of Probabilistic Neural Nets by Importance Sampling. *AISTATS*, 9, 2003.

[2] Y. Bengio and J-S. Senécal. Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008.

[3] T. E. Booth. Unbiased Monte Carlo Estimation of the Reciprocal of an Integral. *Nuclear Science and Engineering*, 156(3):403–407, 2007.

[4] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[5] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[6] C. J. Geyer. On the convergence of Monte Carlo maximum likelihood calculations. *Journal of the Royal Statistical Society, Series B (Methodological)*, 56(1):261–274, 1994.

[7] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *AISTATS*, pages 297–304, 2010.

[8] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361, 2012.

[9] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On Using Very Large Target Vocabulary for Neural Machine Translation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, 2015.

[10] S. Ji, S. V. N. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. *ICLR*, 2016.

[11] P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. *MT Summit*, 2005.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[13] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Neural Information Processing Systems*, pages 2265–2273, 2013.

[14] A. Mnih and Y. W. Teh. A Fast and Simple Algorithm for Training Neural Probabilistic Language Models. *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 1751–1758, 2012.

[15] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246–252. 2005.