
Sampled Estimators For Softmax Must Be Biased

Li-Chung Lin

National Taiwan University
r08922141@ntu.edu.tw

Yaxu Liu*

National Taiwan University
d08944045@ntu.edu.tw

Mohamed bin Zayed University of Artificial Intelligence
yaxu.liu@mbzuai.ac.ae

Chih-Jen Lin

National Taiwan University
cjlin@csie.ntu.edu.tw
Mohamed bin Zayed University of Artificial Intelligence
chihjen.lin@mbzuai.ac.ae

Abstract

Models requiring probabilistic outputs are ubiquitous and used in fields such as natural language processing, contrastive learning, and recommendation systems. The standard method of designing such a model is to output unconstrained logits, which are normalized into probabilities with the softmax function. The normalization involves computing a summation across all classes, which becomes prohibitively expensive for problems with a large number of classes. An important strategy to reduce the cost is to sum over a sampled subset of classes in the softmax function, known as the sampled softmax. It was known that the sampled softmax is biased; the expectation taken over the sampled classes is not equal to the softmax function. Many works focused on reducing the bias by using a better way of sampling the subset. However, while sampled softmax is biased, it is unclear whether an unbiased function different from sampled softmax exists. In this paper, we show that all functions that only access a sampled subset of classes must be biased. With this result, we prevent efforts in finding unbiased loss functions and validate that past efforts devoted to reducing bias are the best we can do.

1 Introduction

Training losses based on the softmax function are extensively used across various fields, e.g., natural language processing [12, 13, 17, 1], contrastive learning [5, 4], and recommendation systems [6, 22]. The most common loss is to take the log of the softmax function, known as the log-softmax or cross-entropy loss,

$$L(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) = \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-).$$

In this loss, the softmax function gives the probability of a class by

$$\text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) = \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}}, \quad (1)$$

where $\hat{y}^+ \in \mathbb{R}$ is the logit of the target class, $\hat{y}_i^- \in \mathbb{R}$ for every $1 \leq i \leq n$ are the logits of all negative classes, and there are $n + 1$ total classes. By summing this loss over all training data, we obtain the overall learning objective.

*Work mainly done while visiting Mohamed bin Zayed University of Artificial Intelligence.

One typically employs gradient methods to train machine learning models. At each training step, the gradient of log-softmax must be calculated, leading to the following partial derivatives²

$$\frac{\partial}{\partial \hat{y}^+} \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) = 1 - \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) \quad (2)$$

and for every $1 \leq i \leq n$,

$$\frac{\partial}{\partial \hat{y}_i^-} \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) = -\text{Softmax}(\hat{y}_i^-, \hat{y}_1^-, \dots, \hat{y}_{i-1}^-, \hat{y}^+, \hat{y}_{i+1}^-, \dots, \hat{y}_n^-). \quad (3)$$

Both the loss and gradient computations inevitably rely on the softmax function. The denominator in (1) sums over logits for all n negative classes, leading to a computational cost of $\mathcal{O}(n)$ for each calculation. This cost can become prohibitively expensive when training models on datasets with a large number of classes. A typical example involves recommendation systems, like [22], in which the size of n can reach millions. In addition to the $\mathcal{O}(n)$ computational complexity of the denominator, evaluating (1) requires computing all n logits, $\hat{y}_i^-$. When using large neural networks such as BERT [9], computing all n logits can become prohibitively expensive.

As shown in [22], an intuitive idea to reduce the computation cost is to sample a subset of k classes for loss calculation, and $k \ll n$ typically. An important scenario is the stochastic gradient descent method, where each training step involves a mini-batch of data, and within each batch only the sampled k classes are used to compute the stochastic gradient. That is, we use only k sampled classes in calculating (2) and (3), where a widely used setting is to consider the following sampled softmax:

$$\frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{j=1}^k w(i_j) e^{\hat{y}_{i_j}^-}}, \quad (4)$$

where k classes $(i_1, \dots, i_k)$ are randomly sampled from the n classes and $w(i_j)$ is a weighting term inversely proportional to the probability of sampling the i_j th sampled class [2]. In contrast, (1) represents the *full softmax*, which involves all $n + 1$ classes.

To ensure equivalence with training a model using the full softmax, it is desirable for the sampled softmax to satisfy the condition of unbiasedness. Specifically, this condition is defined as follows:

$$\mathbb{E} [\text{Stochastic Gradient}] = \text{Full Gradient}, \quad (5)$$

or, equivalently,

$$\mathbb{E} [(4)] = (1), \quad (6)$$

where $\mathbb{E} [\cdot]$ denotes the expectation over the sampled classes $(i_1, \dots, i_k)$. If this condition does not hold, we consider that (4) is biased with respect to (1). The different gradients are potentially harmful to the optimization process and solutions. Unfortunately, previous works like [2, 3] have already demonstrated that (6) does not hold, highlighting the inherent bias of the sampled softmax. Moreover, experiments in [3, 16] have demonstrated that bias degrades model performance in tasks such as multi-label classification and content recommendation.

The bias of sampled softmax leads us to wonder whether, beyond (4), there could be other forms of approximation that are unbiased. In this paper, we show that any general function $s : \mathbb{R}^{k+1} \rightarrow \mathbb{R}$ with access to $k < n$ logits must be biased, that is,

$$\mathbb{E} [s(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-)] \neq (1), \quad (7)$$

where k classes $(i_1, \dots, i_k)$ are randomly sampled from n classes. By extending from the sampled softmax to any general function, this conclusion generalizes the finding of [3] and forms our main contribution. Our negative answer to the existence of unbiased approximations establishes a theoretical boundary on what is and is not achievable. Our result not only fills a notable gap in the literature but also sets clear limits for future research directions.

The remainder of this paper is organized as follows: In Section 2, we summarize related works. In Section 3, we formally define the problem we aim to address and then state the main results of this work and our assumption. Section 4 starts with preliminaries in Section 4.1 that provide the necessary background and lemmas, followed by a sketch of the proof in Section 4.2. The detailed proof is presented in Sections 4.3 and Appendix E. Finally, Section 5 concludes the paper. The main notations used in this work are in Table 1, and the appendix is available at https://www.csie.ntu.edu.tw/~cjlin/papers/softmax_biased/.

²The detailed proof is given in Appendix A.

Table 1: Notation

Notation	Description
n	Number of classes excluding the target
$\hat{y}^+ \in \mathbb{R}$	Logit of the target class
$\hat{y}_1^-, \dots, \hat{y}_n^- \in \mathbb{R}$	Logits of the remaining classes
k	Number of sampled classes
$[r] = \{1, \dots, r\}$	Set of positive integers up to r
S_r	Set of permutations on $[r]$
$\binom{q}{r} = \frac{q!}{r!(q-r)!}$	Number of combinations of r items chosen from q items
$\mathbb{R}_{>0} = \{a \in \mathbb{R} \mid a > 0\}$	Set of positive real numbers
$h_k : \mathbb{R}^k \rightarrow \mathbb{R}$	Estimator taking k arguments
$f_k : \mathbb{R}^k \rightarrow \mathbb{R}$	The symmetrization of h_k

2 Related Works

Prior efforts all implicitly assume that an unbiased solution does not exist without attempting to justify or challenge it. They can be roughly categorized into three directions.

- **Focusing on the sampled softmax (4) and attempting to minimize its bias.** While [3] demonstrated that the inherent bias in the sampled softmax cannot be completely eliminated, they proposed a strategy to minimize this bias. Specifically, the closer the sampling distribution used for selecting $(i_1, \dots, i_k)$ approximates the full softmax, the less biased the sampled softmax becomes. To this end, [3] and its follow-up study [16] developed computationally efficient approximations for sampling. Additionally, numerous hard negative mining strategies have been widely adopted alongside the sampled softmax, such as those used in representation learning [21, 20, 7]. Hard negatives refer to negative classes that produce large gradients when used in the sampled log-softmax loss:

$$-\log \left(\frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{j=1}^k e^{\hat{y}_{i_j}^-}} \right).$$

This setting effectively involves sampling negatives based on their corresponding logits. Furthermore, studies like [24] have argued that an effective hard negative mining method should be based on a good approximation of the full softmax distribution, consistent with the ideas proposed in [3]. In addition to the aforementioned approach of selecting an appropriate sampling distribution, [3] mentioned another idea: using a larger sample size k to reduce bias. However, increasing k naturally leads to higher computational cost. To address this issue, subsequent works [19, 23] introduced the concept of applying moving averages over multiple mini-batches of gradients, thereby approximating the effect of a large sample size k while maintaining low cost.

- **Exploring alternative functions that are computationally more efficient than the full softmax (1).** Works in this category generally try to avoid sampling and explore a new loss with all n logits in their calculation. Well-known examples include hierarchical softmax [14] and spherical softmax [8]. However, as reported in [8], such alternative functions may perform worse than the full softmax (1). Since these methods do not involve sampling over the n classes, we do not extend our discussion to them.
- **Designing a different learning problem with the same optimal solution.** In [11, 15, 10], they transform the original optimization problem involving a log-softmax term in the objective function into a new optimization problem. In the transformed problem, there is no log-softmax term in the objective function. They then show that for special types of models, the transformed problem has the same optimal solution as the original problem. The equivalence of the optimal solution is what they refer to as “unbiased,” which is unrelated to the problem of bias discussed in this paper. While these reformulation approaches are still under development, our negative answer to directly sampling softmax indicates that they are directions worth investigation.

By proving the non-existence of an unbiased solution, our work provides a theoretical boundary and serves as a meaningful complement to these studies.

3 Problem Definition and Main Result

Let $(i_1, \dots, i_k) \in [n]^k$ be random variables denoting a subset of k indices sampled from $[n] = \{1, \dots, n\}$ without replacement. We wish to find an estimator $s(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-)$ such that

$$\mathbb{E} [s(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-)] = \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}}, \quad (8)$$

where the expectation is over $(i_1, \dots, i_k)$.³

Our main results are Theorem 3.1 and Corollary 3.3.

Theorem 3.1. *For every $k < n$, there is no estimator s that satisfies (8) when the k logits are sampled uniformly without replacement.*

Theorem 3.1 states that there is no estimator that can be unbiased to the softmax function when the k logits are uniformly sampled. In this work, we focus on the existence of an unbiased estimator of the softmax function for two main reasons. First, during training, what matters most is the gradient of log-softmax, as model parameters are updated through gradient-based optimization. Second, during inference, the output of interest is softmax itself, which represents the predicted probability distribution over classes. In both situations, the exact value of the log-softmax is not directly required. Regardless, with Theorem 3.1, we show that an unbiased estimator of log-softmax does not exist in Appendix F.

After we establish that Theorem 3.1 applies for uniform sampling, we next examine whether Theorem 3.1 still holds for other ways of sampling the k classes $(i_1, \dots, i_k)$. An example of non-uniform sampling is [13], where the samples are the union of the target class, in-batch negative samples, and samples based on BM25 [18]. In-batch negative samples refer to using the target classes of other instances in a mini-batch as the negative classes of the current instance.

While we want to consider sampling distributions that are as general as possible, we must be careful not to include impractical sampling distributions. Recall that the purpose of a sampled estimator is to avoid computing all n logits; using the logits as part of the sampling process defeats the entire point of sampling. Consequently, we make the following assumption.

Assumption 3.2. The sampling distribution is selected before computing the logits.

An example of violating Assumption 3.2 is to use a distribution where the probability of sampling each class i is proportional to the exponential of its logit e^{y_i} . In such a case, it was proven [3] that the sampled softmax is an unbiased estimator. However, constructing the distribution is equivalent to computing the softmax function for each class. Clearly, if we already have the softmax function, we would not need to compute a sampled estimator. We obviously want to exclude such distributions from consideration.

In the previous example of [13], in-batch sampling and negative sampling are both selected before computing the logits, so they satisfy Assumption 3.2. Similarly, we find that most works on sampled softmax satisfy Assumption 3.2. For the minority of exceptions [3, 16], we give a brief discussion in Appendix D.

Given Assumption 3.2 and setting $s(\cdot)$ to (4), [3] concluded that the sampled softmax is biased with respect to the full softmax,

$$\mathbb{E} \left[\frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{j=1}^k w(i_j) e^{\hat{y}_{i_j}^-}} \right] \neq \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}}.$$

In the following sections, we further extend their conclusion to any general $s(\cdot)$ under Assumption 3.2.

Corollary 3.3. *Theorem 3.1 also holds if the k logits are sampled from any distribution satisfying Assumption 3.2.*

³Note that (4) with $w(i_j)$ can be also expressed by (8). Details are in Appendix B.

The proof of Corollary 3.3 is given in Appendix E.

In the following proof of (7), due to a simplification outlined in Section 4.2, we will work with a function $h_k : \mathbb{R}^k \rightarrow \mathbb{R}$ instead of $s : \mathbb{R}^{k+1} \rightarrow \mathbb{R}$, but they refer to the same problem of estimating the full softmax.

4 Main Proof

In this section, we prove Theorem 3.1. We begin with necessary preliminaries and lemmas, followed by a sketch that highlights the main ideas of the proof before presenting the complete derivation.

4.1 Preliminaries

This section introduces lemmas used in the proof of the main theorem. The proofs are provided in Appendix C.

Definition 4.1. Let $h_k : \mathbb{R}^k \rightarrow \mathbb{R}$ be a function and S_k be the set of permutations on $[k]$. The symmetrization f_k of h_k is defined by

$$f_k(x_1, \dots, x_k) \equiv \frac{1}{k!} \sum_{\tau \in S_k} h_k(x_{\tau(1)}, \dots, x_{\tau(k)}).$$

The symmetrization is so named because it is symmetric in its arguments, stated as follows.

Lemma 4.2. For every $\sigma \in S_k$,

$$f_k(x_1, \dots, x_k) = f_k(x_{\sigma(1)}, \dots, x_{\sigma(k)}).$$

Furthermore, the symmetrization has the same expectation under a uniform sample, stated as follows.

Lemma 4.3. Let $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{R}^n$ be n real numbers and $(i_1, \dots, i_k) \in [n]^k$ be random variables denoting k indices sampled uniformly from $[n]$ without replacement. Then

$$\mathbb{E}[f_k(a_{i_1}, \dots, a_{i_k})] = \mathbb{E}[h_k(a_{i_1}, \dots, a_{i_k})],$$

where the expectation is over $(i_1, \dots, i_k)$.

4.2 Sketch of Proof for Theorem 3.1

The core idea of our proof is that unbiasedness imposes a strong requirement, demanding the estimator to generalize across arbitrary cases. To illustrate this, we constructed two of the simplest yet representative cases: one where all logits are identical and another where the logits are divided into two groups, each with a distinct value. It turns out that these two minimal cases alone have already revealed a set of contradictory constraints. The proof is non-trivial, as it involves identifying subtle examples that clearly demonstrate the inherent contradictions.

The proof of Theorem 3.1 proceeds by contradiction, as shown below:

1. We begin by reformulating the problem of estimating the full softmax into an equivalent problem of estimating:

$$F(\mathbf{a}) = \frac{1}{1 + \sum_{i=1}^n a_i},$$

where $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{R}_{>0}^n$ and $\mathbb{R}_{>0} = \{a \in \mathbb{R} \mid a > 0\}$.

2. We hypothesize the existence of an unbiased estimator $h_k : \mathbb{R}^k \rightarrow \mathbb{R}$ for F , which accepts k arguments, and uses the symmetrization f_k of h_k , which has the same expectation, to simplify the derivations.
3. We analyze the expectation of f_k across various possible values of $\mathbf{a}$ and derive a necessary equation for each vector $\mathbf{a}$ that f_k must satisfy to be unbiased. These equations are derived using mathematical induction.

4. Finally, we complete our proof by demonstrating that it is impossible for f_k to simultaneously satisfy all equations, leading to a contradiction.

Specifically, for Step 3 above, we check different $\mathbf{a}$ of the form,

$$(b_1, \dots, b_1, \underbrace{b_2, \dots, b_2}_m),$$

where $b_1, b_2 \in \mathbb{R}_{>0}$ and $m = 0, \dots, k$.

4.3 Proof of Theorem 3.1

Proof. First, we note that

$$\text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) = \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}} = \frac{1}{1 + \sum_{i=1}^n e^{\hat{y}_i^- - \hat{y}^+}}.$$

It follows that there is an unbiased estimator for the full softmax for every possible model output if and only if there is an unbiased estimator for F

$$F(\mathbf{a}) = \frac{1}{1 + \sum_{i=1}^n a_i} \quad (9)$$

for every $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{R}_{>0}^n$.

We prove by contradiction, so we assume there is an unbiased estimator h_k . That is, there exists an h_k such that for every $\mathbf{a}$

$$F(\mathbf{a}) = \mathbb{E} [h_k(a_{i_1}, \dots, a_{i_k})], \quad (10)$$

where $(i_1, \dots, i_k) \in [n]^k$ are random variables denoting k indices sampled uniformly from $[n]$ without replacement.

Next, we calculate the expectation in (10) in detail. For the selection of $(i_1, \dots, i_k)$, the number of all possible samples of indices is

$$\binom{n}{k} \cdot k! = \frac{n!}{(n-k)!},$$

which is the number of combinations of k choices from n items multiplied by the number of permutations on k items. Therefore, for every function $G : \mathbb{R}^k \rightarrow \mathbb{R}$ we have

$$\mathbb{E} [G(a_{i_1}, \dots, a_{i_k})] = \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} G(x_{\sigma(1)}, \dots, x_{\sigma(k)}), \quad (11)$$

where $C(n, k; \mathbf{a})$ denotes the set of $\binom{n}{k}$ choices of k elements from $\mathbf{a}$.

Let f_k be the symmetrization of h_k . Then

$$F(\mathbf{a}) = \mathbb{E} [h_k(a_{i_1}, \dots, a_{i_k})] = \mathbb{E} [f_k(a_{i_1}, \dots, a_{i_k})] \quad (12)$$

$$\begin{aligned} &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} f_k(x_{\sigma(1)}, \dots, x_{\sigma(k)}) \\ &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} f_k(x_1, \dots, x_k) \end{aligned} \quad (13)$$

$$\begin{aligned} &= \frac{(n-k)!k!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} f_k(x_1, \dots, x_k) \\ &= \frac{1}{\binom{n}{k}} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} f_k(x_1, \dots, x_k), \end{aligned} \quad (14)$$

where (12) follows from Lemma 4.3 and (13) follows from Lemma 4.2.

We claim that for every $b_1 \in \mathbb{R}_{>0}$, we have

$$f_k(b_1, \dots, b_1) = \frac{1}{1 + nb_1}. \quad (15)$$

This is because f_k is unbiased for every $\mathbf{a}$, including for $\mathbf{a} = (b_1, \dots, b_1)$, implying

$$F(\mathbf{a}) = \frac{1}{1 + nb_1} \quad (16)$$

$$= \mathbb{E} [f_k(a_{i_1}, \dots, a_{i_k})] \quad (17)$$

$$= \frac{1}{\binom{n}{k}} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} f_k(b_1, \dots, b_1) \quad (18)$$

$$= f_k(b_1, \dots, b_1),$$

where (16) is the definition of F in (9), (17) is from (12) and (18) is from (14).

Similarly, suppose $\mathbf{a} = (b_1, \dots, b_1, b_2)$ for some $b_1, b_2 \in \mathbb{R}_{>0}$. Then we have

$$\begin{aligned} F(\mathbf{a}) &= \frac{1}{1 + (n-1)b_1 + b_2} \\ &= \mathbb{E} [f_k(a_{i_1}, \dots, a_{i_k})] \\ &= \frac{1}{\binom{n}{k}} \left(\binom{n-1}{k} f_k(b_1, \dots, b_1) + \binom{n-1}{k-1} f_k(b_1, \dots, b_1, b_2) \right) \quad (19) \\ &= \frac{(n-k)!k!}{n!} \frac{(n-1)!}{(n-k-1)!k!} \frac{1}{1 + nb_1} + \frac{(n-k)!k!}{n!} \frac{(n-1)!}{(n-k)!(k-1)!} f_k(b_1, \dots, b_1, b_2) \quad (20) \end{aligned}$$

$$= \frac{n-k}{n} \frac{1}{1 + nb_1} + \frac{k}{n} f_k(b_1, \dots, b_1, b_2).$$

The term $\binom{n-1}{k}$ in (19) is the number of ways of choosing k counts of b_1 and the term $\binom{n-1}{k-1}$ is the number of ways of choosing $k-1$ counts of b_1 and 1 count of b_2 . Also, in (20), we use (15). Rearranging, we have

$$f_k(b_1, \dots, b_1, b_2) = \frac{n}{k} \frac{1}{1 + (n-1)b_1 + b_2} - \frac{n-k}{k} \frac{1}{1 + nb_1}.$$

More generally, for each $m \leq k$, if we consider

$$\mathbf{a} = (b_1, \dots, b_1, \underbrace{b_2, \dots, b_2}_m),$$

we find that

$$F(\mathbf{a}) = \frac{1}{1 + (n-m)b_1 + mb_2} = \mathbb{E} [f_k(a_{i_1}, \dots, a_{i_k})] = \frac{1}{\binom{n}{k}} \sum_{j=0}^m \binom{n-m}{k-j} \binom{m}{j} g(j), \quad (21)$$

where

$$g(j) \equiv f_k(b_1, \dots, b_1, \underbrace{b_2, \dots, b_2}_j) \quad j = 0, \dots, k, \quad (22)$$

and the factor $\binom{n-m}{k-j} \binom{m}{j}$ is the number of ways of choosing $k-j$ counts of b_1 and j counts of b_2 . The dependence of g on n, k, b_1 and b_2 are suppressed in the notation for brevity. We prove by strong induction on m that

$$g(m) = \sum_{l=0}^m \frac{c(m, l)}{1 + (n-l)b_1 + lb_2} \quad m = 0, \dots, k, \quad (23)$$

where

$$c(m, l) = \begin{cases} \frac{\binom{n}{k}}{\binom{n-m}{k-m}} & \text{if } m = l, \\ -\frac{1}{\binom{n-m}{k-m}} \sum_{j=l}^{m-1} \binom{n-m}{k-j} \binom{m}{j} c(j, l) & \text{if } m > l, \\ 0 & \text{otherwise.} \end{cases} \quad (24)$$

Likewise, the dependence of c on n and k are suppressed in the notation for brevity.

For the base case,

$$g(0) = f_k(b_1, \dots, b_1) = \frac{1}{1 + nb_1} = \frac{c(0, 0)}{1 + nb_1},$$

where we respectively use (22), (15) and

$$c(0, 0) = \binom{n}{k} / \binom{n}{k} = 1 \quad (25)$$

for the derivation above.

For the induction step, we split the summation in (21) to single out $g(m)$:

$$\frac{1}{\binom{n}{k}} \binom{n-m}{k-m} \binom{m}{m} g(m) + \frac{1}{\binom{n}{k}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} g(j). \quad (26)$$

Rearranging (21), we have

$$\begin{aligned} g(m) &= \frac{\binom{n}{k}}{\binom{n-m}{k-m} \binom{m}{m}} \left(\frac{1}{1 + (n-m)b_1 + mb_2} - \frac{1}{\binom{n}{k}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} g(j) \right) \\ &= \frac{\binom{n}{k}}{\binom{n-m}{k-m}} \frac{1}{1 + (n-m)b_1 + mb_2} - \frac{1}{\binom{n-m}{k-m}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} g(j). \end{aligned} \quad (27)$$

For the first term in (27), by the definition in (24), we have

$$c(m, m) = \frac{\binom{n}{k}}{\binom{n-m}{k-m}}. \quad (28)$$

For the second term, assuming the induction hypothesis (23) is true for all $j = 0, \dots, m-1$, we have

$$\begin{aligned} & -\frac{1}{\binom{n-m}{k-m}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} g(j) \\ &= \frac{-1}{\binom{n-m}{k-m}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} \sum_{l=0}^j \frac{c(j, l)}{1 + (n-l)b_1 + lb_2} \end{aligned} \quad (29)$$

$$\begin{aligned} &= \frac{-1}{\binom{n-m}{k-m}} \sum_{j=0}^{m-1} \sum_{l=0}^j \binom{n-m}{k-j} \binom{m}{j} \frac{c(j, l)}{1 + (n-l)b_1 + lb_2} \\ &= \frac{-1}{\binom{n-m}{k-m}} \sum_{l=0}^{m-1} \sum_{j=l}^{m-1} \binom{n-m}{k-j} \binom{m}{j} \frac{c(j, l)}{1 + (n-l)b_1 + lb_2} \end{aligned} \quad (30)$$

$$\begin{aligned} &= \sum_{l=0}^{m-1} \frac{\frac{-1}{\binom{n-m}{k-m}} \sum_{j=l}^{m-1} \binom{n-m}{k-j} \binom{m}{j} c(j, l)}{1 + (n-l)b_1 + lb_2} \\ &= \sum_{l=0}^{m-1} \frac{c(m, l)}{1 + (n-l)b_1 + lb_2}. \end{aligned} \quad (31)$$

In (29), $g(j)$ is substituted with the induction hypothesis (23); (30) exchanges the order of summation over j and l ; and $c(m, l)$ in (31) follows from the definition in (24). By replacing values in (27) with (28) and (31), we have

$$\begin{aligned} g(m) &= \frac{\binom{n}{k}}{\binom{n-m}{k-m}} \frac{1}{1 + (n-m)b_1 + mb_2} - \frac{1}{\binom{n-m}{k-m}} \sum_{j=0}^{m-1} \binom{n-m}{k-j} \binom{m}{j} g(j) \\ &= \frac{c(m, m)}{1 + (n-m)b_1 + mb_2} + \sum_{l=0}^{m-1} \frac{c(m, l)}{1 + (n-l)b_1 + lb_2} \\ &= \sum_{l=0}^m \frac{c(m, l)}{1 + (n-l)b_1 + lb_2}. \end{aligned}$$

Therefore, we have finished proving (23) by induction.

Let

$$c^* = \max_{j, l \leq k} |c(j, l)|.$$

Recalling that $k < n$, we have for all $j \leq k$

$$\begin{aligned} |g(j)| &= \left| \sum_{l=0}^j \frac{c(j, l)}{1 + (n-l)b_1 + lb_2} \right| \\ &\leq \sum_{l=0}^j \frac{|c(j, l)|}{1 + (n-l)b_1 + lb_2} \end{aligned} \tag{32}$$

$$\begin{aligned} &\leq \sum_{l=0}^j \frac{c^*}{1 + (n-l)b_1 + lb_2} \\ &< \sum_{l=0}^j \frac{c^*}{1 + (n-k)b_1} \end{aligned} \tag{33}$$

$$= \frac{(j+1)c^*}{1 + (n-k)b_1} \quad j = 0, \dots, k, \tag{34}$$

where (32) is due to triangle inequality, and (33) follows from the facts that $b_1, b_2 \in \mathbb{R}_{>0}$ and $k > 0$, since k is the number of samples. However, by the definition of g in (22) and (15),

$$g(k) = f_k(b_2, \dots, b_2) = \frac{1}{1 + nb_2}.$$

So we can use (34) to have

$$\frac{1}{1 + nb_2} = |g(k)| < \frac{(k+1)c^*}{1 + (n-k)b_1},$$

which is a contradiction because if

$$b_1 = \frac{2(k+1)c^* - 1}{n - k}$$

and

$$b_2 = \frac{1}{n},$$

then

$$\frac{1}{1 + nb_2} = \frac{1}{2} = \frac{(k+1)c^*}{1 + (n-k)b_1}.$$

To show that b_1 is well-defined, i.e., $b_1 > 0$, we have

$$\begin{aligned}
b_1 &= \frac{2(k+1)c^* - 1}{n-k} \\
&= \frac{(2(k+1) \max_{j,l \leq k} |c(j,l)|) - 1}{n-k} \\
&\geq \frac{2(k+1)|c(0,0)| - 1}{n-k} \\
&= \frac{2(k+1) - 1}{n-k} \\
&> 0,
\end{aligned} \tag{35}$$

where (35) follows from (25). The contradiction completes our proof. $\square$

5 Conclusions

This paper considered the problem of bias for methods that sample classes and examined the possibility of using a different function instead of the sampled softmax. Previous works have taken it as a foundational assumption that such a function cannot be found, treating this claim as if it were true without justification. We proved, for the first time, that no function can be an unbiased estimator, which extends the conclusion in [3] that is only limited to the sampled softmax. Our result now can serve as the theoretical premise for related studies. According to our result, future work should not aim to find an unbiased estimator. By providing a proof of impossibility, our work allows follow-up studies to focus on feasible solutions.

Acknowledgements

This work was supported in part by National Science and Technology Council of Taiwan grant NSTC-113-2222-E-002-005-MY3, and in part by the Featured Area Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (114L900901). The authors thank the reviewers for their insightful and constructive comments.

References

- [1] Yoshua Bengio and Jean-Sébastien Senécal. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 17–24, 2003.
- [2] Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008.
- [3] Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*, pages 590–599, 2018.
- [4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the Thirty-Seventh International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.
- [5] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debaised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 8765–8775, 2020.
- [6] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.
- [7] Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Gururaj K, Prasenjit Dey, Amit Singh, Deepesh Hada, Vidit Jain, Bhawna Paliwal, Anshul Mittal, Sonu Mehta, Ramachandran Ramjee, Sumeet Agarwal, Purushottam Kar, and Manik Varma. NGAME: negative mining-aware mini-batching for extreme classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 258–266, 2023.
- [8] Alexandre de Brébisson and Pascal Vincent. An exploration of softmax alternatives belonging to the spherical loss family. In *International Conference on Learning Representations (ICLR)*, 2016.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186, 2019.
- [10] Francois Fagan and Garud Iyengar. Unbiased scalable softmax optimization. *CoRR*, abs/1803.08577, 2018.
- [11] Siddharth Gopal and Yiming Yang. Distributed training of large-scale logistic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 289–297, 2013.
- [12] Nilesh Gupta, Devvrit, Ankit Singh Rawat, Srinadh Bhojanapalli, Prateek Jain, and Inderjit S. Dhillon. Dual-encoders for extreme multi-label classification. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.
- [13] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

- [14] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252, 2005.
- [15] Parameswaran Raman, Shin Matsushima, Xinhua Zhang, Hyokun Yun, and S. V. N. Vishwanathan. DS-MLR: exploiting double separability for scaling up distributed multinomial logistic regression. *CoRR*, abs/1604.04706, 2016.
- [16] Ankit Singh Rawat, Jiecao Chen, Felix X. Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 13834–13844, 2019.
- [17] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990, 2019.
- [18] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
- [19] Bokun Wang and Tianbao Yang. Finite-sum coupled compositional stochastic optimization: Theory and applications. In *Proceedings of the Thirty-ninth International Conference on Machine Learning (ICML)*, pages 23292–23317, 2022.
- [20] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*, 2021.
- [21] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H. Chi. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference (WWW)*, pages 441–447, 2020.
- [22] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, page 269–277, 2019.
- [23] Zhuoning Yuan, Dixian Zhu, Zi-Hao Qiu, Gang Li, Xuanhui Wang, and Tianbao Yang. LibAUC: A deep learning library for x-risk optimization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 5487–5499, 2023.
- [24] Wenzheng Zhang and Karl Stratos. Understanding hard negatives in noise contrastive estimation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 1090–1101, 2021.

A Using Log-Softmax Loss During Training

The typical loss function for models based on Softmax is the log-softmax function

$$\begin{aligned} L(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) &= \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) \\ &= \hat{y}^+ - \log \left(e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-} \right). \end{aligned}$$

During training, we need to compute the gradient of L at each step. In other words, we need to compute the partial derivatives. The partial derivative of log-softmax with respect to the target class is

$$\begin{aligned} \frac{\partial L(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-)}{\partial \hat{y}^+} &= 1 - \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}} \\ &= 1 - \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-), \end{aligned}$$

and the partial derivatives with respect to the negative classes $j \in \{1, \dots, n\}$ are

$$\begin{aligned} \frac{\partial L(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-)}{\partial \hat{y}_j^-} &= -\frac{e^{\hat{y}_j^-}}{e^{\hat{y}^+} + \sum_{i=1}^n e^{\hat{y}_i^-}} \\ &= -\text{Softmax}(\hat{y}_j^-, \hat{y}_1^-, \dots, \hat{y}_{j-1}^-, \hat{y}^+, \hat{y}_{j+1}^-, \dots, \hat{y}_n^-). \end{aligned}$$

Therefore, computing the gradient of the loss function requires computing Softmax.

B Reducing (4) to (8)

Let $\hat{o}_i^- \in \mathbb{R}$ denote the original logits for the i th negative class. Instead of using $\hat{o}_i^-$ directly, we use

$$\hat{y}_i^- = \hat{o}_i^- + \log(w(i))$$

such that

$$\begin{aligned} e^{\hat{y}_i^-} &= e^{\hat{o}_i^- + \log(w(i))} \\ &= w(i) e^{\hat{o}_i^-}, \end{aligned}$$

where $\forall i, w(i)$ are constants. So we have

$$\begin{aligned} s(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-) &= \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{j=1}^k e^{\hat{y}_{i_j}^-}} \\ &= \frac{e^{\hat{y}^+}}{e^{\hat{y}^+} + \sum_{j=1}^k w(i_j) e^{\hat{o}_{i_j}^-}}, \end{aligned}$$

which is the sampled softmax in (4) on the original logits $\hat{o}_i^-$.

C Proof of Lemmas

C.1 Proof of Lemma 4.2

Proof.

$$\begin{aligned} f_k(x_1, \dots, x_k) &= \frac{1}{k!} \sum_{\tau \in S_k} h_k(x_{\tau(1)}, \dots, x_{\tau(k)}) \\ &= \frac{1}{k!} \sum_{\tau \in S_k} h_k(x_{\tau(\sigma(1))}, \dots, x_{\tau(\sigma(k))}) \\ &= f_k(x_{\sigma(1)}, \dots, x_{\sigma(k)}), \end{aligned} \tag{36}$$

where (36) follows because the summation over τ sums over the same set of summands. $\square$

C.2 Proof of Lemma 4.3

Proof. We first note that the number of all possible samples of indices is

$$\binom{n}{k} \cdot k! = \frac{n!}{(n-k)!},$$

which is the number of combinations of k choices from n items multiplied by the number of permutations on k items. Let $C(n, k; \mathbf{a})$ denote the set of $\binom{n}{k}$ choices of k elements from $\mathbf{a}$. Then

$$\begin{aligned} \mathbb{E}[f_k(a_{i_1}, \dots, a_{i_k})] &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} f_k(x_{\sigma(1)}, \dots, x_{\sigma(k)}) \\ &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} \frac{1}{k!} \sum_{\tau \in S_k} h_k(x_{\tau(\sigma(1))}, \dots, x_{\tau(\sigma(k))}) \\ &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} \frac{1}{k!} \sum_{\tau \in S_k} h_k(x_{\tau(1)}, \dots, x_{\tau(k)}) \quad (37) \\ &= \frac{(n-k)!}{n!} \sum_{\mathbf{x} \in C(n, k; \mathbf{a})} \sum_{\tau \in S_k} h_k(x_{\tau(1)}, \dots, x_{\tau(k)}) \\ &= \mathbb{E}[h_k(a_{i_1}, \dots, a_{i_k})], \end{aligned}$$

where (37) follows because the summation over τ sums over the same set of summands. $\square$

D Sampling Distributions Depending On All Logits

In this section, we show that [3] and [16] do not meet Assumption 3.2 by giving a brief overview of how they work.

Their methods depend on the following problem structure. Given input space $\mathcal{X}$ and the input $\mathbf{x} \in \mathcal{X}$, logits of class i take the form

$$\hat{y}_i = \langle f(\mathbf{x}), \mathbf{v}_i \rangle,$$

where $f : \mathcal{X} \rightarrow \mathbb{R}^d$ is the model, d is the embedding dimension and $\mathbf{v}_i \in \mathbb{R}^d$ is the embedding of class i . In other words, the probability of class i is given by

$$\begin{aligned} \text{Softmax}(\hat{y}_i, \hat{y}_1, \dots, \hat{y}_{i-1}, \hat{y}_{i+1}, \dots, \hat{y}_{n+1}) &= \frac{e^{\hat{y}_i}}{\sum_{j=1}^{n+1} e^{\hat{y}_j}} \\ &= \frac{e^{\langle f(\mathbf{x}), \mathbf{v}_i \rangle}}{\sum_{j=1}^{n+1} e^{\langle f(\mathbf{x}), \mathbf{v}_j \rangle}}. \end{aligned}$$

In keeping with (1), we let n denote the number of negative classes, so there are a total of $n+1$ classes. They proposed a sampling distribution p that is efficient to compute and has a structure similar to the softmax function. The probability $p(i)$ of sampling class i is given by

$$p(i) = \frac{\langle \phi(f(\mathbf{x})), \phi(\mathbf{v}_i) \rangle}{\sum_{j=1}^{n+1} \langle \phi(f(\mathbf{x})), \phi(\mathbf{v}_j) \rangle}, \quad (38)$$

where $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d^2}$ is a feature map given by

$$\phi(\mathbf{v}) = (v_1 v_1, \dots, v_1 v_d, v_2 v_1, \dots, v_2 v_d, \dots, v_d v_d).$$

Notably, the sampling distribution (38) does not satisfy Assumption 3.2 because (38) effectively involves computing the logits. Expanding out the terms of the inner product, we have

$$\begin{aligned}
\langle \phi(f(\mathbf{x})), \phi(\mathbf{v}_i) \rangle &= \sum_{j=1}^d \sum_{k=1}^d f(\mathbf{x})_j f(\mathbf{x})_k v_{ij} v_{ik} \\
&= \sum_{j=1}^d f(\mathbf{x})_j v_{ij} \sum_{k=1}^d f(\mathbf{x})_k v_{ik} \\
&= \langle f(\mathbf{x}), \mathbf{v}_i \rangle \cdot \langle f(\mathbf{x}), \mathbf{v}_i \rangle \\
&= \hat{y}_i^2.
\end{aligned}$$

E Proof of Corollary 3.3

Proof. The proof is by contradiction. Assume there is an unbiased estimator h_k for F with a sampling distribution p . That is,

$$\Pr(i_1 = j_1, \dots, i_k = j_k) = p(j_1, \dots, j_k).$$

The expectation is given by

$$F(\mathbf{a}) = \mathbb{E}_p [h_k(a_{i_1}, \dots, a_{i_k})] = \frac{1}{(n-k)!} \sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) h_k(a_{\sigma(1)}, \dots, a_{\sigma(k)}). \quad (39)$$

Note that this is different from (11), in which we have k choices from items first due to the uniform sampling. Therefore, here we have S_n instead of S_k in (11). Further, the summation over σ has $n!$ summands but there are only

$$\binom{n}{k} \cdot k! = \frac{n!}{(n-k)!}$$

choices of indices. Therefore, the summation counts each choice of indices $(n-k)!$ times, which is corrected by the first factor $\frac{1}{(n-k)!}$.

We first observe that F is symmetric in its arguments, i.e., for every permutation $\tau \in S_n$,

$$F(\mathbf{a}) = \frac{1}{1 + \sum_{i=1}^n a_i} = \frac{1}{1 + \sum_{i=1}^n a_{\tau(i)}} = F(a_{\tau(1)}, \dots, a_{\tau(n)}). \quad (40)$$

By Assumption 3.2, p is the same function regardless of the values of $\mathbf{a}$.⁴ This property and (40) imply that, for every permutation $\tau \in S_n$,

$$\begin{aligned}
F(\mathbf{a}) &= F(a_{\tau(1)}, \dots, a_{\tau(n)}) \\
&= \frac{1}{(n-k)!} \sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) h_k(a_{\sigma(\tau(1))}, \dots, a_{\sigma(\tau(k))}).
\end{aligned} \quad (41)$$

⁴Without Assumption 3.2, it is possible to have a different distribution $p_{\mathbf{a}}$ for each $\mathbf{a}$. See Appendix D for more details.

Therefore,

$$\begin{aligned} F(\mathbf{a}) &= \frac{1}{n!} \sum_{\tau \in S_n} F(\mathbf{a}) \\ &= \frac{1}{n!} \sum_{\tau \in S_n} \frac{1}{(n-k)!} \sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) \times h_k(a_{\sigma(\tau(1))}, \dots, a_{\sigma(\tau(k))}) \end{aligned} \quad (42)$$

$$\begin{aligned} &= \frac{1}{n!(n-k)!} \sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) \times \sum_{\tau \in S_n} h_k(a_{\sigma(\tau(1))}, \dots, a_{\sigma(\tau(k))}) \\ &= \frac{1}{n!(n-k)!} \sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) \times \sum_{\tau \in S_n} h_k(a_{\tau(1)}, \dots, a_{\tau(k)}) \end{aligned} \quad (43)$$

$$= \frac{1}{n!(n-k)!} (n-k)! \times \sum_{\tau \in S_n} h_k(a_{\tau(1)}, \dots, a_{\tau(k)}), \quad (44)$$

$$\begin{aligned} &= \frac{1}{n!} \sum_{\tau \in S_n} h_k(a_{\tau(1)}, \dots, a_{\tau(k)}) \\ &= \frac{(n-k)!}{n!} \sum_{x \in C(n, k; \mathbf{a})} \sum_{\sigma \in S_k} h_k(x_{\sigma(1)}, \dots, x_{\sigma(k)}), \end{aligned} \quad (45)$$

where (42) follows from (41), and (43) follows because the summation over τ sums over the same set of summands. From (43) to (44), we use

$$\sum_{\sigma \in S_n} p(\sigma(1), \dots, \sigma(k)) = (n-k)!$$

because, as we mentioned earlier, each element is counted $(n-k)!$ times. Finally, (45) is just (11), the expectation under the uniform distribution. This implies that h_k is also an unbiased estimator under the uniform distribution. Such an h_k does not exist by Theorem 3.1, leading to a contradiction. $\square$

F Impossibility of an Unbiased Log-Softmax Estimator

In this section, we prove that an unbiased and differentiable log-softmax estimator ℓ does not exist. We prove by contradiction. Assume that

$$\mathbb{E} [\ell(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-)] = \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-).$$

Then

$$\begin{aligned} \frac{\partial}{\partial \hat{y}^+} \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) &= \frac{\partial}{\partial \hat{y}^+} \mathbb{E} [\ell(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-)] \\ &= \frac{\partial}{\partial \hat{y}^+} \frac{1}{n!} \sum_{\sigma \in S_n} \ell(\hat{y}^+, \hat{y}_{\sigma(1)}^-, \dots, \hat{y}_{\sigma(k)}^-) \\ &= \frac{1}{n!} \sum_{\sigma \in S_n} \frac{\partial}{\partial \hat{y}^+} \ell(\hat{y}^+, \hat{y}_{\sigma(1)}^-, \dots, \hat{y}_{\sigma(k)}^-) \\ &= \mathbb{E} \left[\frac{\partial}{\partial \hat{y}^+} \ell(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-) \right]. \end{aligned} \quad (46)$$

So

$$\begin{aligned} \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) &= -\frac{\partial}{\partial \hat{y}^+} \log \text{Softmax}(\hat{y}^+, \hat{y}_1^-, \dots, \hat{y}_n^-) - 1 \\ &= \mathbb{E} \left[-\frac{\partial}{\partial \hat{y}^+} \ell(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-) - 1 \right], \end{aligned} \quad (47)$$

where the first equality follows from (2) and the second equality follows from (46). The expression

$$-\frac{\partial}{\partial \hat{y}^+} \ell(\hat{y}^+, \hat{y}_{i_1}^-, \dots, \hat{y}_{i_k}^-) - 1$$

in (47) is therefore an unbiased estimator for softmax, which contradicts Theorem 3.1.

Note that the proof above assumes a differentiable estimator ℓ , but the proof of Theorem 3.1 does not. Since we take the gradient of the loss estimator during training, we can reasonably assume that practical loss estimators are differentiable.