Candidates v.s. Noises Estimation for Large Multi-Class Classification Problem

Lei Han and Tong Zhang

Tencent AI Lab, Shenzhen, China

Abstract

This paper proposes a method for multi-class classification problems, where the number of classes K is large. The method, referred to as *Candidates v.s. Noises Estimation* (CANE), selects a small subset of candidate classes and samples the remaining classes. We show that CANE is always consistent and computationally efficient. Moreover, the resulting estimator has low statistical variance approaching that of the maximum likelihood estimator, when the observed label belongs to the selected candidates with high probability. In practice, we use a tree structure with leaves as classes to promote fast beam search for candidate selection. We also apply the CANE method to estimate word probabilities in neural language models. Experiments show that CANE achieves better prediction accuracy over the Noise-Contrastive Estimation (NCE), its variants and a number of the state-of-the-art tree classifiers, while it gains significant speedup compared to the standard $\mathcal{O}(K)$ methods.

1 Introduction

In practice one often encounters multi-class classification problem with a large number of classes. For example, applications in image classification [33] and language modeling [18] usually have tens to hundreds of thousands of classes. In this case, training the standard softmax logistic or one-against-all (OAA) models become impractical.

One promising way to handle the large class size is to use sampling. In language models, a commonly adopted technique is *Noise-Contrastive Estimation* (NCE) [13]. This method is originally proposed for estimating probability densities and has been applied to various language modeling situations, such as learning word embedding, context generation and neural machine translation [27, 26, 38, 36]. NCE reduces the problem of multi-class classification to binary classification problem, which discriminates between a target class distribution and a noise distribution, where a few noise classes are sampled as a representation of the entire noise distribution. In general, the noise distribution is given a priori. For example, a power-raised unigram distribution has been shown to be effective in language models [24, 16, 27]. Recently, some variants of NCE have been proposed. The Negative Sampling [24] is a simplified version of NCE that ignores the numerical probabilities in the distributions and discriminates between only the target class and noise samples; the One v.s. Each [37] solves a very similar problem motivated by bounding the softmax logistic log-likelihood. Two other variants, *BlackOut* [16] and *complementary sum sampling* [4], employ parametric forms of the noise distribution and use sampled noises to approximate the normalization factor. NCE and its variants use (only) the observed class versus the noises. By sampling the noises, these methods avoid the costly computation of the normalization factor to achieve fast training speed. In this paper, we will generalize the idea by using a subset of classes (which can be automatically learned)

called candidate classes against the remaining noise classes. Compared to NCE, this approach can significantly improve the statistical efficiency when the true class belongs to the candidate classes with high probability.

Another type of popular methods for large class space is the tree structured classifier [3, 2, 9, 6, 7, 15]. In these methods, a tree structure is defined over the classes which are treated as leaves. Each internal node of the tree is assigned with a local classifier, routing the examples to one of its descendants. Decisions are made from the root until reaching a leaf. Therefore, the multi-class classification problem is reduced to solving a number of small local models defined by a tree, which typically admits a logarithmic complexity on the total number of classes. Generally speaking, tree classifiers gain training and prediction speed while suffering a loss of accuracy. The performance of tree classifier may rely heavily on the quality of the tree structure [25]. Earlier approaches use fixed tree, such as the Filter Tree [3] and the Hierarchical Softmax (HSM) [28]. Recent tree classifiers are able to adjust the tree and learn the local classifiers simultaneously, such as the LOMTree [6] and Recall Tree [7], etc. Our approach is complementary to these tree classifiers, because we study the orthogonal issue of consistent class sampling, which in principle can be combined with many of these tree methods. In fact, a tree structure will be used in our approach to select a small subset of candidate classes. Since we focus on the class sampling aspect, we do not necessarily employ the best tree construction method in our experiments.

In this paper, we propose a method to efficiently deal with the large class problem by paying attention to a small subset of candidate classes instead of the entire class space. Given a data point $\boldsymbol{x}$ (without observing y), we select a small number of competitive candidates as a set $\mathcal{C}_{\boldsymbol{x}}$. Then, we sample the remaining classes, which are treated as noises, to represent the entire noises in the large normalization factor. The estimation is referred to as *Candidates v.s. Noises Estimation* (CANE). We show that CANE is consistent and its computation using stochastic gradient method is independent of the class size K. Moreover, the statistical variance of the CANE estimator can approach that of the maximum likelihood estimator (MLE) of the softmax logistic regression when $\mathcal{C}_{\boldsymbol{x}}$ can cover the target class y with high probability. This statistical efficiency is a key advantage of CANE over NCE, and its effect can be observed in practice.

We then describe two concrete algorithms: the first one is a generic stochastic optimization procedure for CANE; and the second algorithm employs a tree structure with leaves as classes to enable fast beam search for candidate selection. We also apply the CANE method to solve the word probability estimation problem in neural language modeling. Experimental results conducted on both classification problems and neural language modeling problems show that CANE achieves significant speedup compared to the standard softmax logistic regression. Moreover, it achieves superior performance over NCE, its variants, and a number of the state-of-the-art tree classifiers.

2 Candidates v.s. Noises Estimation

Consider a K-class classification problem (K is large) with n training examples $(\boldsymbol{x}_i, y_i)|_{i=1}^n$, where $\boldsymbol{x}_i$ is from an input space $\mathcal{X} \in \mathbb{R}^d$ and $y_i \in \{1, \cdots, K\}$. The softmax logistic regression solves

$$\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{I}(y_i = k) \log \frac{e^{s_k(\boldsymbol{x}_i, \boldsymbol{\theta})}}{\sum_{k'=1}^{K} e^{s_{k'}(\boldsymbol{x}_i, \boldsymbol{\theta})}}, \tag{1}$$

where $s_k(\boldsymbol{x}, \boldsymbol{\theta})$ for $k = 1, \cdots, K$ is a model parameterized by $\boldsymbol{\theta}$. Solving Eq. (1) requires computing a score for every class and the summation in the normalization factor, which is very expensive when K is large.

Generally speaking, given $\boldsymbol{x}$, only a small number of classes in the entire class space might be competitive to the true class. Therefore, we propose to find a small subset of classes as a candidate set $\mathcal{C}_{\boldsymbol{x}} \subset \{1, \cdots, K\}$ and treat the classes outside $\mathcal{C}_{\boldsymbol{x}}$ as noises, so that we can focus on the small set $\mathcal{C}_{\boldsymbol{x}}$ instead of the entire K classes. We will discuss one way to choose $\mathcal{C}_{\boldsymbol{x}}$ in Section 4. Denote the remaining $K - |\mathcal{C}_{\boldsymbol{x}}|$ noises as a set $\mathcal{N}_{\boldsymbol{x}}$, so $\mathcal{N}_{\boldsymbol{x}}$ is the complementary set of $\mathcal{C}_{\boldsymbol{x}}$. We propose to sample some noise class $j \in \mathcal{N}_{\boldsymbol{x}}$ to represent the entire $\mathcal{N}_{\boldsymbol{x}}$. That is, we replace the partial summation $\sum_{j \in \mathcal{N}_{\boldsymbol{x}}} e^{s_j(\boldsymbol{x},\boldsymbol{\theta})}$ in the denominator of Eq. (1) by $e^{s_j(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(j)$ using some sampled class j with *an arbitrary sampling probability* $q_{\boldsymbol{x}}(j)$, where $q_{\boldsymbol{x}}(j) \in (0, 1)$ and $\sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_{\boldsymbol{x}}(j) = 1$. Thus, the denominator will be approximated as

$$\sum_{k'=1}^{K} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} \approx \sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} + e^{s_j(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(j).$$

Given example $(\boldsymbol{x}, y)$ and its candidate set $\mathcal{C}_{\boldsymbol{x}}$, if $y \in \mathcal{C}_{\boldsymbol{x}}$, then for some sampled noise class j, the probability will be defined as

$$\tilde{\mathbb{P}}(y|\boldsymbol{x}, \mathcal{C}_{\boldsymbol{x}}) = \frac{e^{s_y(\boldsymbol{x},\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} + e^{s_j(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(j)}; \tag{2}$$

otherwise, if $y \notin \mathcal{C}_{\boldsymbol{x}}$, we define the probability as

$$\tilde{\mathbb{P}}(y|\boldsymbol{x}, \mathcal{C}_{\boldsymbol{x}}) = \frac{e^{s_y(\boldsymbol{x},\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} + e^{s_y(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(y)}. \tag{3}$$

Now, with Eqs. (2) and (3), in expectation, we will need to solve the following objective:

$$\begin{aligned}\max_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}} \Bigg[& \sum_{k \in \mathcal{C}_{\boldsymbol{x}}} p(y = k|\boldsymbol{x}) \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_{\boldsymbol{x}}(j) \log \frac{e^{s_k(\boldsymbol{x},\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} + e^{s_j(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(j)} \\ & + \sum_{k \in \mathcal{N}_{\boldsymbol{x}}} p(y = k|\boldsymbol{x}) \log \frac{e^{s_k(\boldsymbol{x},\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x},\boldsymbol{\theta})} + e^{s_k(\boldsymbol{x},\boldsymbol{\theta})}/q_{\boldsymbol{x}}(k)} \Bigg].\end{aligned} \tag{4}$$

Empirically, we will need to solve

$$\begin{aligned}\max_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \Bigg[& \mathbb{I}(y_i \in \mathcal{C}_{\boldsymbol{x}_i}) \sum_{j \in \mathcal{N}_{\boldsymbol{x}_i}} q_{\boldsymbol{x}_i}(j) \log \frac{e^{s_{y_i}(\boldsymbol{x}_i,\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}_i}} e^{s_{k'}(\boldsymbol{x}_i,\boldsymbol{\theta})} + e^{s_j(\boldsymbol{x}_i,\boldsymbol{\theta})}/q_{\boldsymbol{x}_i}(j)} \\ & \mathbb{I}(y_i \notin \mathcal{C}_{\boldsymbol{x}_i}) \log \frac{e^{s_{y_i}(\boldsymbol{x}_i,\boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}_i}} e^{s_{k'}(\boldsymbol{x}_i,\boldsymbol{\theta})} + e^{s_{y_i}(\boldsymbol{x}_i,\boldsymbol{\theta})}/q_{\boldsymbol{x}_i}(y_i)} \Bigg].\end{aligned} \tag{5}$$

Eq. (5) consists of two summations over both the data points and the classes in the noise set $\mathcal{N}_{\boldsymbol{x}}$. Therefore, we can employ a 'doubly' stochastic gradient optimization method by sampling both data points $i \in \{1, \ldots, n\}$ and noise classes $j \in \mathcal{N}_{\boldsymbol{x}_i}$. It is not difficult to check that each stochastic gradient is bounded under reasonable conditions, which means that the computational cost for solving (5) using stochastic gradient is independent of the class number K. Since we only choose a small number of candidates in $\mathcal{C}_{\boldsymbol{x}}$, the computation for each stochastic gradient in Eq. (5) is efficient. The above method is referred to as Candidates v.s. Noises Estimation (CANE).

3 Properties

In this section, we investigate the statistical properties of CANE. The parameter space of the softmax logistic model in Eq. (1) has redundancy, observing that adding any function $h(\boldsymbol{x})$ to $s_k(\boldsymbol{x}, \boldsymbol{\theta})$ will not change the objective. Similar situation happens for Eqs. (4) and (5). To avoid this redundancy, one can add some constraints on the K scores or simply fix one of them as zero, e.g., let $s_K(\boldsymbol{x}, \boldsymbol{\theta}) = 0$. To facilitate the analysis, we will fix $s_K(\boldsymbol{x}, \boldsymbol{\theta}) = 0$ and consider $\mathcal{C}_{\boldsymbol{x}} \cup \mathcal{N}_{\boldsymbol{x}} = \{1, \cdots, K-1\}$ within this section. First, we have the following result.

Theorem 1 (*Infinity-Sample Consistency*)**.** *By viewing the objective R as a function of $\{s_1, \cdots, s_{K-1}\}$, R achieves its maximum if and only if $s_k = \log \frac{p(y=k|\boldsymbol{x})}{p(y=K|\boldsymbol{x})}$ for $k = 1, \cdots, K-1$.*

In Theorem 1, the global optima is exactly the log-odds function with class K as the reference class. Now, considering the parametric form $s_k(\boldsymbol{x}, \boldsymbol{\theta})$, there exists a true parameter $\boldsymbol{\theta}^*$ so that $s_k(\boldsymbol{x}, \boldsymbol{\theta}^*) = \log \frac{p(y=k|\boldsymbol{x})}{p(y=K|\boldsymbol{x})}$ if the model $s_k(\boldsymbol{x}, \boldsymbol{\theta})$ is correctly specified. The following theorem shows that the CANE estimator $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta})$ is consistent with $\boldsymbol{\theta}^*$.

Theorem 2 (*Finite-Sample Asymptotic Consistency*)**.** *Given $\boldsymbol{x}$, denote $\mathcal{C}_{\boldsymbol{x}}$ as $\{i_1, \cdots, i_{|\mathcal{C}_{\boldsymbol{x}}|}\}$ and $\mathcal{N}_{\boldsymbol{x}}$ as $\{j_1, \cdots, j_{|\mathcal{N}_{\boldsymbol{x}}|}\}$. Assume $\|\nabla_{\boldsymbol{\theta}} s_k(\boldsymbol{x}, \boldsymbol{\theta})\|$, $\|\nabla^2_{\boldsymbol{\theta}} s_k(\boldsymbol{x}, \boldsymbol{\theta})\|$ and $\|\nabla^3_{\boldsymbol{\theta}} s_k(\boldsymbol{x}, \boldsymbol{\theta})\|$ for $k = 1, \cdots, K-1$ are bounded under some norm $\|\cdot\|$ defined on the parameter space of $\boldsymbol{\theta}$. Furthermore, assume the matrix $\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top$ is positive definite, where*

$$\boldsymbol{M} = \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_{\boldsymbol{x}}(j) \left[diag\left(\mathbf{u}_j\right) - \frac{1}{p(K, \boldsymbol{x}) + \sum_{k \in \mathcal{C}_{\boldsymbol{x}}} p(k, \boldsymbol{x}) + p(j, \boldsymbol{x})/q_{\boldsymbol{x}}(j)} \mathbf{u}_j \mathbf{u}_j^\top \right] \in \mathbb{R}^{K-1 \times K-1},$$

$$\mathbf{u}_j = \big(\underbrace{p(i_1, \boldsymbol{x}), \cdots, p(i_{|\mathcal{C}_{\boldsymbol{x}}|}, \boldsymbol{x})}_{\textit{The candidate part}}, \underbrace{0, \cdots, p(j, \boldsymbol{x})/q_{\boldsymbol{x}}(j), \cdots, 0}_{\textit{The noise part}} \big)^\top \in \mathbb{R}^{K-1}, \quad \textit{for } j = j_1, \cdots, j_{|\mathcal{N}_{\boldsymbol{x}}|},$$

$$\boldsymbol{\nabla} = diag\left(\left(\nabla_{\boldsymbol{\theta}} s_{i_1}(\boldsymbol{x}, \boldsymbol{\theta}), \cdots, \nabla_{\boldsymbol{\theta}} s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}(\boldsymbol{x}, \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} s_{j_1}(\boldsymbol{x}, \boldsymbol{\theta}), \cdots, \nabla_{\boldsymbol{\theta}} s_{j_{|\mathcal{N}_{\boldsymbol{x}}|}}(\boldsymbol{x}, \boldsymbol{\theta}) \right)^\top \right).$$

Then, as $n \to \infty$, the estimator $\hat{\boldsymbol{\theta}}$ converges to $\boldsymbol{\theta}^*$.

The above theorem shows that similar to the maximum likelihood estimator of Eq. (1), the CANE estimator in Eq. (5) is also consistent. Next, we have the asymptotic normality for $\hat{\boldsymbol{\theta}}$ as follows.

Theorem 3 (*Asymptotic Normality*)**.** *Under the same assumption used in Theorem 2, as $n \to \infty$, $\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)$ follows the asymptotic normal distribution:*

$$\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) \xrightarrow{d} \mathbb{N}(\mathbf{0}, [\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top]^{-1}). \tag{6}$$

Theorem 3 shows that the CANE method has a statistical variance of $[\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top]^{-1}$. As we will see in the next corollary, if one can successfully choose the candidate set $\mathcal{C}_{\boldsymbol{x}}$ so that it covers the observed label y with high probability, then the difference between the statistical variance of CANE and that of Eq. (1) is small. Therefore, choosing a good candidate set can be important for practical applications. Moreover, under standard conditions, the computation of CANE using stochastic gradient is independent of the class size K because the variance of stochastic gradient is bounded.

Corollary 1 (*Low Statistical Variance*)**.** *The variance of the maximum likelihood estimator for the softmax logistic regression in Eq. (1) has the form of $[\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M}^{mle} \boldsymbol{\nabla}^\top]^{-1}$. If $\sum_{k \in \mathcal{C}_{\boldsymbol{x}} \cup \{K\}} p(k, \boldsymbol{x}) \to 1$, i.e., the probability that $\mathcal{C}_{\boldsymbol{x}} \cup \{K\}$ covers the observed class label y approches 1, then we have*

$$[\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top]^{-1} \to [\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M}^{mle} \boldsymbol{\nabla}^\top]^{-1}.$$

Algorithm 1 A general optimization procedure for CANE.

1: **Input:** K, $(\boldsymbol{x}_i, y_i)|_{i=1}^n$, number of candidates $N_c = |\mathcal{C}_{\boldsymbol{x}}|$, number of sampled noises $N_n = |T_{\boldsymbol{x}}|$, sampling strategy $\boldsymbol{q}$ and learning rate η.
2: **Output:** $\hat{\boldsymbol{\theta}}$.

3: Initialize $\boldsymbol{\theta}$;
4: **for** every sampled example **do**
5: Receive example $(\boldsymbol{x}, y)$;
6: Find the candidate set $\mathcal{C}_{\boldsymbol{x}}$;
7: **if** $y \in \mathcal{C}_{\boldsymbol{x}}$ **then**
8: Sample N_n noises outside $\mathcal{C}_{\boldsymbol{x}}$ according to $\boldsymbol{q}$ and denote the selected noise set as $T_{\boldsymbol{x}}$;
9: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \nabla_{\boldsymbol{\theta}} \hat{R}$ with $\nabla_{\boldsymbol{\theta}} \hat{R}$ given by

$$\nabla_{\boldsymbol{\theta}} s_y(\boldsymbol{x}, \boldsymbol{\theta}) - \frac{1}{|T_{\boldsymbol{x}}|} \sum_{j \in T_{\boldsymbol{x}}} \left[\frac{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} s_{k'}(\boldsymbol{x}, \boldsymbol{\theta}) + e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(j) \nabla_{\boldsymbol{\theta}} s_j(\boldsymbol{x}, \boldsymbol{\theta})}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(j)} \right]; \tag{7}$$

10: **else**
11: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \nabla_{\boldsymbol{\theta}} \hat{R}$ with $\nabla_{\boldsymbol{\theta}} \hat{R}$ given by

$$\nabla_{\boldsymbol{\theta}} s_y(\boldsymbol{x}, \boldsymbol{\theta}) - \frac{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} s_{k'}(\boldsymbol{x}, \boldsymbol{\theta}) + e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(y) \nabla_{\boldsymbol{\theta}} s_y(\boldsymbol{x}, \boldsymbol{\theta})}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(y)}; \tag{8}$$

12: **end if**
13: **end for**

4 Algorithm

In this section, we propose two algorithms. The first one is a general optimization procedure for CANE. The second implementation provides an efficient way to select a competitive set $\mathcal{C}_{\boldsymbol{x}}$ using a tree structure defined on the classes.

4.1 A General Optimization Algorithm

Eq. (5) suggests an efficient algorithm using a 'doubly' stochastic gradient descend (SGD) method by sampling both the data points and classes. That is, by sampling a data point $(\boldsymbol{x}, y)$, we find the candidate set $\mathcal{C}_{\boldsymbol{x}} \subset \{1, \cdots, K\}$. If $y \in \mathcal{C}_{\boldsymbol{x}}$, we sample noise $j \in \mathcal{N}_{\boldsymbol{x}}$ according to probability $q_{\boldsymbol{x}}(j)$ N_n times and denote the selected noises as a set $T_{\boldsymbol{x}}$ ($|T_{\boldsymbol{x}}| = N_n$). We then optimize

$$\frac{1}{|T_{\boldsymbol{x}}|} \sum_{j \in T_{\boldsymbol{x}}} \log \frac{e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(j)},$$

with gradient $\nabla_{\boldsymbol{\theta}} \hat{R}$ given by Eq. (7). Otherwise, if $y \notin \mathcal{C}_{\boldsymbol{x}}$, we optimize

$$\log \frac{e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(y)},$$

with gradient $\nabla_{\boldsymbol{\theta}} \hat{R}$ given by Eq. (8). This general procedure is provided in Algorithm 1. Algorithm 1 has a complexity of $\mathcal{O}(N_c + N_n)$ ($N_c = |\mathcal{C}_{\boldsymbol{x}}|$), which is independent of the class size K. In step 6, any method can be used to select $\mathcal{C}_{\boldsymbol{x}}$.

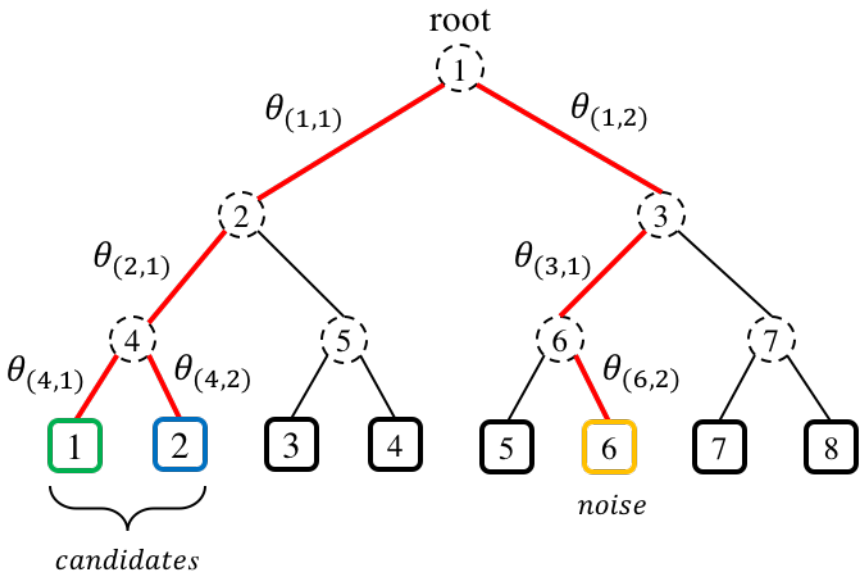


Figure 1: Illustration of the tree model. Suppose an example $(\boldsymbol{x}, 2)$ is arriving, and two candidate classes 1 and 2 are selected by beam search. The class 6 is sampled as noise.

4.2 Beam Tree Algorithm

In the second algorithm, we provide an efficient way to find a competitive $\mathcal{C}_{\boldsymbol{x}}$. An attractive strategy is to use a tree defined on the classes, because one can perform fast heuristic search algorithms based on a tree structure to prune the uncompetitive classes. Indeed, any structure, e.g., a graph, can be used alternatively as long as the structure allows to efficiently prune low quality branches. We will use tree structure for candidate selection in this paper.

Given a tree structure defined on the K classes, the model $s_k(\boldsymbol{x}, \boldsymbol{\theta})$ is then interpreted as a tree model illustrated in Fig. 1. For simplicity, Fig. 1 uses a binary tree over $K = 8$ labels as example while any tree structure can be used for selecting $\mathcal{C}_{\boldsymbol{x}}$. In the example, circles denote internal nodes and squares indicate classes. The parameters are kept in the edges and denoted as $\boldsymbol{\theta}_{(o,c)}$, where o indicates an internal node and c is the index of the c-th child of node o. Therefore, a pair (o, c) represents an edge from node o to its c-th child. The dashed circles indicate that we do not keep any parameters in the internal nodes. Now, we define $s_k(\boldsymbol{x}, \boldsymbol{\theta})$ as

$$s_k(\boldsymbol{x}, \boldsymbol{\theta}) = g_{\boldsymbol{\psi}}(\boldsymbol{x}) \cdot \sum_{(o,c) \in \mathcal{P}_k} \boldsymbol{\theta}_{(o,c)}, \tag{9}$$

where $g_{\boldsymbol{\psi}}(\boldsymbol{x})$ is a function parameterized by $\boldsymbol{\psi}$ and it maps the input $\boldsymbol{x} \in \mathbb{R}^d$ to a representation $g_{\boldsymbol{\psi}}(\boldsymbol{x}) \in \mathbb{R}^{d_r}$ for some d_r. For example, in image classification, a good choice of the representation $g_{\boldsymbol{\psi}}(\boldsymbol{x})$ of the raw pixels $\boldsymbol{x}$ is usually a deep neural network. $\mathcal{P}_k$ denotes the path from the root to the class k. Eq. (9) implies that the score of an example belonging to a class is calculated by summing up the scores along the corresponding path. Now, in Fig. 1, suppose that we are given an example $(\boldsymbol{x}, y)$ with class $y = 2$ (blue color). Using beam search, we find two candidates with high scores, i.e., class 1 (green color) and class 2. Then, we let $\mathcal{C}_{\boldsymbol{x}} = \{1, 2\}$. In this case, we have $y \in \mathcal{C}_{\boldsymbol{x}}$, so we need to sample noises. Suppose we sample one class 6 (orange color). According to Eq. (7), the parameters along the corresponding paths (red color) will be updated.

Formally, given example $(\boldsymbol{x}, y)$, if $y \in \mathcal{C}_{\boldsymbol{x}}$, we sample the noises as a set $T_{\boldsymbol{x}}$. Then for $(o, c) \in$

Algorithm 2 The Beam Tree Algorithm.

1: **Input:** K, $(\boldsymbol{x}_i, y_i)|_{i=1}^n$, representation function $g_{\boldsymbol{\psi}}(\boldsymbol{x})$, number of candidates $N_c = |\mathcal{C}_{\boldsymbol{x}}|$, number of sampled noises $N_n = |T_{\boldsymbol{x}}|$, sampling strategy $\boldsymbol{q}$ and learning rate η.
2: **Output:** $\hat{\boldsymbol{\theta}}$.

3: Construct a tree on the K classes;
4: Initialize $\boldsymbol{\theta}$;
5: **for** every sampled example **do**
6: Receive example $(\boldsymbol{x}, y)$;
7: Given $\boldsymbol{x}$, use beam search to find the N_c classes with high scores to compose $\mathcal{C}_{\boldsymbol{x}}$;
8: **if** $y \in \mathcal{C}_{\boldsymbol{x}}$ **then**
9: Sample N_n noises outside $\mathcal{C}_{\boldsymbol{x}}$ according to $\boldsymbol{q}$ and denote the selected noise set as $T_{\boldsymbol{x}}$;
10: Find the paths with respect to the classes in $\mathcal{C}_{\boldsymbol{x}} \cup T_{\boldsymbol{x}}$;
11: **else**
12: Find the paths with respect to the classes in $\mathcal{C}_{\boldsymbol{x}} \cup \{y\}$;
13: **end if**
14: Sum up the scores along each selected path for the corresponding class;
15: *// Update parameters in the tree model*
16: $\boldsymbol{\theta}_{(o,c)} \leftarrow \boldsymbol{\theta}_{(o,c)} + \eta \frac{\partial \hat{R}}{\partial \boldsymbol{\theta}_{(o,c)}}$ for each (o, c) included in the selected paths according to Eqs. (10) and (11);
17: *// Update parameters in the representation function if it is parameterized*
18: $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \eta \frac{\partial \hat{R}}{\partial g} \frac{\partial g}{\partial \boldsymbol{\psi}}$;
19: **end for**

$\mathcal{P}_{\mathcal{C}_{\boldsymbol{x}} \cup T_{\boldsymbol{x}}}$, where $\mathcal{P}_{\mathcal{C}_{\boldsymbol{x}} \cup T_{\boldsymbol{x}}} = \cup_{k \in \mathcal{C}_{\boldsymbol{x}} \cup T_{\boldsymbol{x}}} \mathcal{P}_k$, the gradient with respect to $\boldsymbol{\theta}_{(o,c)}$ is thus

$$\frac{\partial \hat{R}}{\partial \boldsymbol{\theta}_{(o,c)}} = \frac{1}{|T_{\boldsymbol{x}}|} \sum_{j \in T_{\boldsymbol{x}}} \left[\mathbb{I}\left((o,c) \in \mathcal{P}_y\right) - \frac{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} \mathbb{I}((o,c) \in \mathcal{P}_{k'}) e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + \mathbb{I}((o,c) \in \mathcal{P}_j) \frac{e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})}}{q_{\boldsymbol{x}}(j)}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + \frac{e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})}}{q_{\boldsymbol{x}}(j)}} \right] g_{\boldsymbol{\psi}}(\boldsymbol{x}). \quad (10)$$

Note that an edge may be included in multiple selected paths. For example, $\mathcal{P}_1$ and $\mathcal{P}_2$ share edges $(1,1)$ and $(2,1)$ in Fig. 1. The case of $y \notin \mathcal{C}_{\boldsymbol{x}}$ can be illustrated similarly. The gradient with respect to $\boldsymbol{\theta}_{(o,c)}$ when $y \notin \mathcal{C}_{\boldsymbol{x}}$ is

$$\frac{\partial \hat{R}}{\partial \boldsymbol{\theta}_{(o,c)}} = \left[\mathbb{I}\left((o,c) \in \mathcal{P}_y\right) - \frac{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} \mathbb{I}((o,c) \in \mathcal{P}_{k'}) e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + \mathbb{I}((o,c) \in \mathcal{P}_y) \frac{e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})}}{q_{\boldsymbol{x}}(y)}}{\sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + \frac{e^{s_y(\boldsymbol{x}, \boldsymbol{\theta})}}{q_{\boldsymbol{x}}(y)}} \right] g_{\boldsymbol{\psi}}(\boldsymbol{x}). \quad (11)$$

The gradients in Eqs. (10) and (11) enjoy the following property.

Proposition 1. *If an edge (o, c) is included in every selected path, $\boldsymbol{\theta}_{(o,c)}$ does not need to be updated.*

The proof of Proposition 1 is straightforward that if (o, c) belongs to every selected path, then the gradients in Eqs. (10) and (11) are 0. The above property allows a fast detection of those parameters which do not need to be updated in SGD and hence can save more computations.

Since we use beam search to choose the candidates in a tree structure, the proposed algorithm is referred to as *Beam Tree*, which is depicted in Algorithm 2.[1] For the tree construction method in step 3, we can use some hierarchical clustering based methods which will be detailed in the experiments and supplementary material. In the algorithm, the beam search needs $\mathcal{O}(N_c \log_b K)$ operations, where b is a constant related to the tree structure, e.g., binary tree for $b = 2$. The parameter updating needs $\mathcal{O}((N_c + N_n) \log_b K)$ operations, where $N_n = |T_{\boldsymbol{x}}|$. Therefore, Algorithm 2 has a complexity of $\mathcal{O}((2N_c + N_n) \log_b K)$ which is logarithmic with respect to K. The term $\log_b K$

[1]The beam search procedure in step 7 is provided in the appendix.

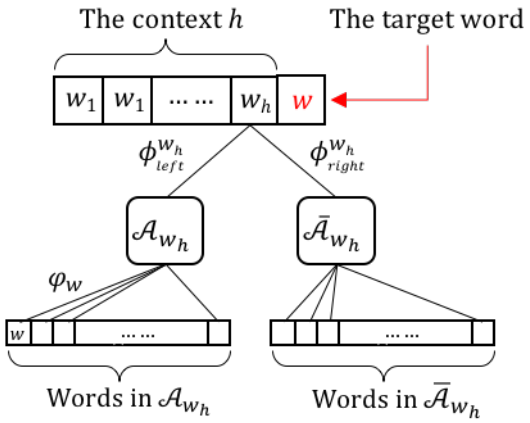


Figure 2: Illustration of the CANE model used in neural language modeling.

is from the tree structure used in this specific candidate selection method, so it does not conflict with the complexity of the general Algorithm 1, which is independent of K. Another advantage of the Beam Tree algorithm is that it allows fast predictions and can naturally output the top-J predictions using beam search. The prediction time has an order of $\mathcal{O}(J \log_b K)$ for the top-J predictions.

5 Application to Neural Language Modeling

In this section, we apply the CANE method to neural language modeling which solves a probability density estimation problem. In neural language models, the conditional probability distribution of the target word w given context h is defined as

$$P_h(w) = \frac{e^{s_w(h,\boldsymbol{\theta})}}{\sum_{w' \in \mathcal{V}} e^{s_{w'}(h,\boldsymbol{\theta})}},$$

where $s_w(h, \boldsymbol{\theta})$ is the scoring function with parameter $\boldsymbol{\theta}$. A word w in the context h will be represented with an embedding vector $\boldsymbol{u}_w \in \mathbb{R}^d$ with embedding size d. Given context h, the model computes the score for the target word w as

$$s_w(h, \boldsymbol{\theta}) = \bar{\boldsymbol{u}}_h^\top \boldsymbol{v}_w,$$

where $\boldsymbol{\theta} = \{\boldsymbol{u}, \boldsymbol{v}\}$, $\bar{\boldsymbol{u}}_h$ is the average embedding vector of the words in context h and $\boldsymbol{v}_w$ is the weight parameter for the target word w. Both the word embedding $\boldsymbol{u}$ and weight parameter $\boldsymbol{v}$ need to be estimated. In language models, the vocabulary size $|\mathcal{V}|$ is usually very large and the computation of the normalization factor is extremely expensive. Therefore, instead of estimating the exact probability distribution $P_h(w)$, sampling methods [26, 16] such as NCE and its variants are typically adopted to approximate $P_h(w)$.

In order to apply the CANE method, we will need to select the candidates given any context h. For multi-class classification problem, we have devised a Beam Tree algorithm in Algorithm 2 that uses a tree structure to select candidates, and the tree can be obtained by some hierarchical clustering methods over the observations $\boldsymbol{x}$ before learning. However, different from the classification problem, the word embeddings in the language model are not known before training, and a hierarchical clustering based tree is not easy to obtain. Therefore we propose another candidate set slection

method for CANE. Suppose we are using N-gram model and predicting the next word. Given a context h, there exists a 'next word' set for the last word w_h in h such that the set contains all the words which appear just after w_h at least once in the training texts. Denote the set as $\mathcal{A}_{w_h}$, and its complementary set according to the entire vocabulary as $\bar{\mathcal{A}}_{w_h}$, then we may construct a light tree for every last word in the contexts as shown in Fig. 2, and we may re-parameterize $\boldsymbol{v}_w$ for the target word w as

$$\boldsymbol{v}_w = \boldsymbol{\varphi}_w + \mathbb{I}(w \in \mathcal{A}_{w_h})\boldsymbol{\phi}_{left}^{w_h} + \mathbb{I}(w \in \bar{\mathcal{A}}_{w_h})\boldsymbol{\phi}_{right}^{w_h},$$

where $\boldsymbol{\varphi}_w$ is the parameter of the target word w, and it is shared by every last word w_h. The two parameters $\boldsymbol{\phi}_{left}^{w_h}$ and $\boldsymbol{\phi}_{right}^{w_h}$ associated with w_h determine the chance that the candidates will be selected from which subset. Given the context h, suppose the model chooses the set $\mathcal{A}_{w_h}$, then we sample all the candidates from $\mathcal{A}_{w_h}$ (i.e., $\mathcal{C}_h \subseteq \mathcal{A}_{w_h}$) according to some distribution such as the power-raised unigram distribution. For the noises in CANE, we directly sample words out of $\mathcal{C}_h$ according to the distribution $\boldsymbol{q}$, which can be chosen as the power-raised unigram distribution as well.

6 Related Algorithms

We provide a discussion comparing CANE with the existing techniques for solving the large class problem. Given $(\boldsymbol{x}, y)$, NCE and its variants [13, 26, 24, 16, 37, 4] use the observed class y as the only 'candidate', while CANE chooses a subset of candidates $\mathcal{C}_{\boldsymbol{x}}$ according to $\boldsymbol{x}$ without observing y. NCE assumes the entire noise distribution $P_{noise}(y)$ is known. For example, a power-raised unigram distribution is shown to be effective in language models because the word frequency can represent the noise (word) distribution well. However, in general multi-class classification problems, when the knowledge of the noise distribution is absent, NCE may have unstable estimations using an inaccurate noise distribution. CANE works for general multi-class classification problems and does not rely on a known noise distribution. Instead, it only focuses on a small candidate set $\mathcal{C}_{\boldsymbol{x}}$. Once the true class label is contained in $\mathcal{C}_{\boldsymbol{x}}$ with high probability, CANE will have robust estimations as supported by our theoretical results. The variants of NCE [24, 16, 37, 4] also sample one or multiple noises to replace the large normalization factor; however, theoretical guarantees on the consistency and variance of their estimations are not provided. In addition, NCE and its variants can not speed up prediction while the Beam Tree algorithm for CANE can reduce the prediction complexity to $O(\log K)$.

The Beam Tree algorithm is related to the tree classifiers, while it is clear that the use of tree structure is only for selecting candidates in CANE. The Beam Tree method itself is also different from existing tree classifiers. Most of the state-of-the-art tree classifiers, e.g., LOMTree [6] and Recall Tree [7], store local classifiers in their internal nodes. Then, examples are pushed through the root until reaching the leaf. The Bean Tree algorithm shown in Fig. 1 does not maintain local classifiers, and it only uses the tree structure to perform global heuristic search for candidate selection. We will also compare our approach to some other state-of-the-art tree classifiers in our experiments.

7 Experiments

We evaluate the CANE method in various applications in this section, including both multi-class classification problems and neural language modeling.

7.1 Classification Problems

In this section, we consider multi-class classification problem with a large number of classes. We compare CANE with NCE, its variants and a number of the state-of-the-art tree classifiers that have been used for large class space problems. The evaluated methods include:

- Softmax: the standard softmax logistic regression used as a baseline.
- NCE: the Noise-Contrastive Estimation method [26, 27].
- BlackOut: a variant of the NCE [16]. In [4], the results show that the complementary sum sampling [4] is comparable to BlackOut, and both of them outperform Negative Sampling [24]. So we compare BlackOut in our experiments.
- Filter Tree: the Filter Tree classifier [3] with implementation in Vowpal-Wabbit (VW), which is a public fast learning system.[2]
- LOMTree: the LOMTree classifier proposed in [6] with implementation in VW.
- Recall Tree: the Recall Tree classifier proposed in [7] with implementation in VW.
- CANE: our CANE method with Beam Tree algorithm.

For simplicity, we use b-nary tree for CANE and set $b = 10$ in all the experiments. We trade off between the number of candidates in $\mathcal{C}_{\boldsymbol{x}}$ and the number of selected noises in $T_{\boldsymbol{x}}$ to see how these parameters affect the learning performance. Different estimations will be referred to as 'CANE-($|\mathcal{C}_{\boldsymbol{x}}|$ v.s. $|T_{\boldsymbol{x}}|$)'. We always let $|\mathcal{C}_{\boldsymbol{x}}| + |T_{\boldsymbol{x}}|$ equal the number of noises used in NCE and BlackOut, so these methods will have the same number of considered classes. We use 'NCE-k' and 'BlackOut-k' to represent the corresponding method with k noises. We uniformly sample noises in CANE. For NCE and BlackOut, by following [27, 26, 16, 4], we use the power-raised unigram distribution with the power factor selected from $\{0, 0.1, 0.3, 0.5, 0.75, 1\}$ to sample the noises. However, when the classes are balanced as in many cases of the classification datasets, this distribution reduces to the uniform distribution.

All the methods use SGD with learning rate selected from $\{0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1.0\}$. The Beam Tree algorithm requires a tree structure and we use some tree generated by a simple hierarchical clustering method on the centers of the individual classes.[3] All the evaluations are performed on a single machine with 3.3GHz quad-core Intel Core i5 processor.

For the compared tree classifiers, the Filter Tree generates a fixed tree itself in VW, the LOMTree and Recall Tree methods use binary trees and they are able to dynamically learn the tree structure.[4]

We test all the compared methods on 6 large multi-class classification datasets:

- Sector[5]: a multi-class classification dataset with 105 classes [5]. Data are split into 90% training and 10% testing.
- ALOI[6]: a color image classification problem with 1000 classes [12]. Data are split into 90% training and 10% testing.

[2] `https://github.com/JohnLangford/vowpal_wabbit/wiki`

[3] This only needs to scan the data once and performs hierarchical clustering on K centers. The time cost can be neglected compared to the training phrase. The tree construction method is provided in the appendix.

[4] More details of the experimental setting are provided in the appendix.

[5] `http://www.cs.utexas.edu/~xrhuang/PDSparse/`

[6] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

Table 1: Statistics of the number of classes, number of features, training size and testing size in different datasets.

Data	# classes	# features	# train	# test
Sector	105	55K	8K	1K
ALOI	1K	128	100K	10K
LSHTC1	12K	347K	84K	5K
Dmoz	12K	833K	335K	38K
ImageNet 2010	1K	4K	1.3M	50K
ImageNet-10K	10K	4K	4.5M	4.5M

- LSHTC1 and Dmoz[5]: two multilingual datasets from [31]. Both of them have around 12K classes. Training and testing sets are from [41].
- ImageNet 2010[7]: the image classification task in ImageNet 2010 competition [33]. There are 1000 classes and 1.3M images for training. We use the validation set which contains 50,000 images as the test set.
- ImageNet-10K[7]: the large image classification task in ImageNet Fall 2009 release [8]. There are approximately 10K classes and 9M images. By following the protocols in [8, 34, 19], we randomly split the data into two halves for training and testing.

Table 1 shows the statistics of the datasets. For the Sector, ALOI, LSHTC1 and Dmoz datasets, we use the original features downloaded from the provided sources and use linear models. For the ImageNet 2010 and ImageNet-10K datasets, similar to [30], we transfer the mid-level representations from the pre-trained VGG-16 net [35] on ImageNet 2012 data [33] to our case. Then, we concatenate CANE or other compared methods above the partial VGG-16 net as the top layer (the network structure is provided in the supplementary material). Similar to [30], the parameters of the partial VGG-16 net are pre-trained and kept fixed[8]. Only the parameters in the top layer are trained on the target datasets, i.e., ImageNet 2010 and ImageNet-10K. We run all the methods 50 epochs on Sector, ALOI, LSHTC1, Dmoz and ImageNet 2010 datasets and 20 epochs on ImageNet-10K, and report the accuracy v.s. epoch relationship.

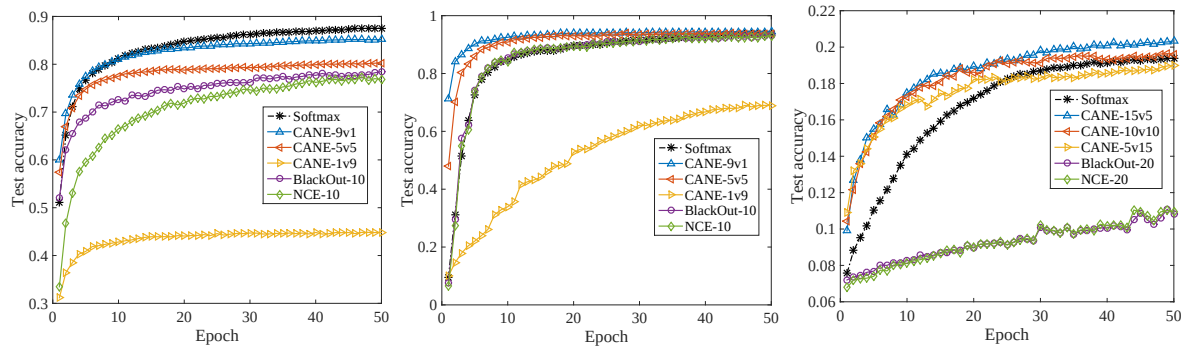
Table 2: Training / testing time of the sampling methods. Running Softmax and testing NCE and BlackOut on large datasets are time consuming. We use multi-thread implementation for these methods and estimate the running time. '∼' indicates the estimated time.

Data	NCE-10	BlackOut-10	CANE-1v9	CANE-5v5	CANE-9v1	Softmax
Sector	24s / 0.75s	20s / 0.76s	54s / 0.05s	1m45s / 0.14s	2m15s / 0.19s	6m7s / 0.88s
ALOI	3m / 5.9s	3m / 5.8s	4m / 0.1s	7m / 0.3s	8m / 0.5s	27m35s / 6.5s
Data	NCE-20	BlackOut-20	CANE-5v15	CANE-10v10	CANE-15v5	Softmax
LSHTC1	26m / 23m57s	26m / 24m14s	31m / 3s	44m / 6s	52m / 9s	5d11h / 13m7s
Dmoz	1h20m / 32m4s	1h20m / 31m48s	1h58m / 30s	3h36m / 1m4s	3h53m / 1m27s	∼25d / 33m
ImageNet 2010	3h27m / 7m53s	3h23m / 7m50s	4h3m / 22s	5h48m / 41s	6h25m / 51s	4d / 8m42s
ImageNet-10K	13h / ∼5d	12h / ∼5d	20h / 51m	1d9h / 1h33m	1d15h / 2h	∼140d / ∼5d

Fig. 3 and Table 2 show the accuracy v.s. epoch curves and the training / testing time for NCE, BlackOut, CANE and Softmax. The tree classifiers in VW can not directly output test accuracy after each epoch and we report the final results of the tree classifiers in Table 3. For ImageNet-10K

[7] `http://image-net.org`

[8] The pre-trained parameters can be downloaded from the link: `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`.



(a) Sector (b) ALOI (c) LSHTC1

(d) Dmoz (e) ImageNet 2010 (f) ImageNet-10K

Figure 3: Results of test accuracy v.s. epoch on different classification datasets.

Table 3: Test accuracy and training / testing time of the tree classifiers.

Data		Filter Tree	LOMTree	Recall Tree
Sector	Acc.	84.67	84.91	86.89
	Time	21s / 0.4s	32s / 0.2s	42s / 0.2s
ALOI	Acc.	20.07	82.70	83.03
	Time	58s / 0.2s	3m20s / 1.0s	2m30s / 0.2s
LSHTC1	Acc.	12.38	11.80	11.58
	Time	1h16m / 12s	1h36m / 13s	3h20m / 23s
Dmoz	Acc.	24.79	24.13	18.11
	Time	2h42m / 19s	3h35m / 23s	17h30m / 39s
ImageNet 2010	Acc.	48.29	49.87	61.28
	Time	6h50m / 7s	17h50m / 20s	1d8h / 30s
ImageNet-10K	Acc.	4.49	9.72	22.74
	Time	2d7h / 15m	2d10h / 20m	7d2h / 1h14m

data, the Softmax method is very time consuming (even with multi-thread implementation) and we do not report this result. As we can observe, by fixing $|\mathcal{C}_{\boldsymbol{x}}| + |T_{\boldsymbol{x}}|$, using more candidates than noises in CANE will achieve better performance, because a larger $\mathcal{C}_{\boldsymbol{x}}$ will increase the chance to cover the target class y. The probability that the target class is included in the selected candidate set on the test data is reported in Table 4, and the probability is indeed the top-$|\mathcal{C}_{\boldsymbol{x}}|$ accuracy. On all the datasets, CANE with larger candidate set achieves considerable improvement compared to other method in terms of accuracy, and sometimes CANE even surpasses the Softmax. The per epoch training time of CANE is slightly slower than the training of NCE and BlackOut because of the beam search, however, CANE converges much faster according to the training epochs. Moreover,

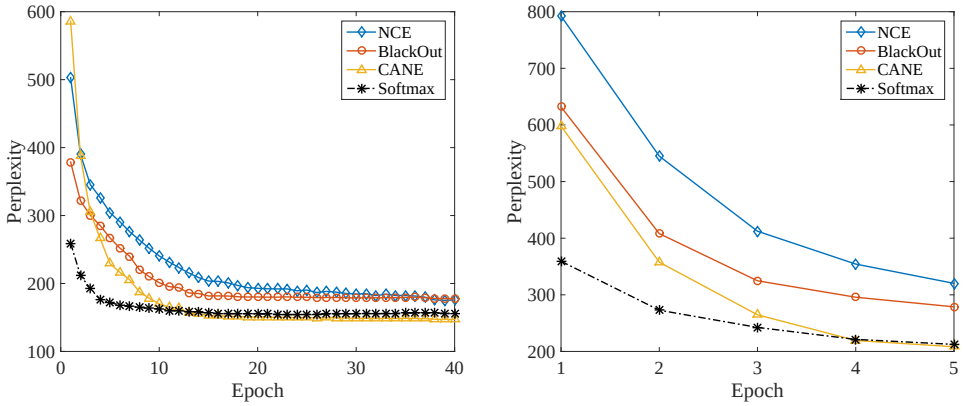
Table 4: The probability (%) that the true label is included in the selected candidate set on the test set, i.e., the top-$|\mathcal{C}_{\boldsymbol{x}}|$ accuracy.

# candidates	Sector	ALOI	# candidates	LSHTC1	Dmoz	ImageNet 2010	ImageNet-10K
1 (CANE-1v9)	68.89	44.84	5 (CANE-5v15)	28.28	41.26	76.59	39.59
5 (CANE-5v5)	96.57	86.47	10 (CANE-10v10)	33.44	52.65	87.29	53.28
9 (CANE-9v1)	97.92	93.59	15 (CANE-15v5)	39.16	58.02	91.17	60.22

the prediction time of CANE is much faster than those of NCE and BlackOut. It is worth noting that CANE also exceeds some state-of-the-art results on the ImageNet-10K data, e.g., 19.2% top-1 accuracy reported in [19] and 21.9% top-1 accuracy reported in [22] which are conducted from $\mathcal{O}(K)$ methods, while it still underperforms the recent result 28.4% in [14]. This is probably because the VGG-16 net works better than the neural net structure used in [19] and the distance-based method in [22] on this dataset, while [14] adopts a deep neural network combined with metric learning to achieve better feature embedding, leading to better prediction performance on this dataset.

7.2 Neural Language Modeling

In this experiment, we apply the CANE method to neural language modeling. We test on the Penn TreeBank corpus and Gutenberg corpus. The Penn TreeBank dataset contains 1M words and we choose 12K words appearing at least 5 times as the vocabulary. The Gutenberg dataset contains 50M words and we choose 116K words appearing at least 10 times as the vocabulary. For both datasets, the embedding vector has a dimension of 200. We use a 4-gram model and set the learning rate as 0.025. We sample 20 noises for NCE and Blackout, and sample 10 candidates and 10 noises for CANE. The tree classifiers evaluated in the previous classification problems can not be directly applied in language modeling. Therefore we compare CANE to NCE and BlackOut methods. For all the compared methods, we use power-raised unigram distribution to sample the noises with the power factor selected from $\{0, 0.1, 0.3, 0.5, 0.75, 1\}$. For the CANE method, given context h, when the word-wise tree (introduced in Section 5) directs to one set ($\mathcal{A}_{w_h}$ or $\bar{\mathcal{A}}_{w_h}$), then we use the power-raised unigram distribution to sample the candidates in that set.



(a) Penn TreeBank (b) Gutenberg

Figure 4: Test perplexity v.s. training epoch on Penn TreeBank and Gutenberg datasets.

The test perplexity is shown in Fig. 4. As we can observe, the CANE method shows faster convergence compared to NCE and Blackout and achieves the same test perplexity as Softmax in a few epochs. Similar to some cases observed in [16], the perplexity performances of both NCE and

Table 5: Training speed (words/sec) of different methods.

Data	NCE	BlackOut	CANE	Softmax
Penn TreeBank	6.3k	6.2k	5.5k	63
Gutenberg	778	657	420	12

BlackOut converge quite slowly if we compare them to that of Softmax (and CANE). Table 5 shows the training speed of different methods. From the table, CANE shows comparable speed with NCE and BlackOut for processing each word, while its test performance converges significantly faster than NCE and BlackOut, showing better statistical efficiency as expected by our theory. Moreover, CANE achieves 35$\sim$90 times speedup compared to Softmax. For testing, all the methods have the same complexity because we evaluate the perplexity over the entire distribution.

8 Conclusion

We proposed Candidates v.s. Noises Estimation (CANE) for fast learning in multi-class classification problems with many labels, and applied this method to the word probability estimation problem in neural language models. We showed CANE is consistent and the computation using SGD is always efficient (that is, independent of the class size K). Moreover, the new estimator has low statistical variance approaching that of the softmax logistic regression, if the observed class label belongs to the candidate set with high probability. Empirical results demonstrated that CANE is effective for speeding up both training and prediction in large multi-class classification problems and CANE is effective in neural language modeling. We note that this work employs a fixed distribution, such as the power-raised unigram distribution, to sample noises in CANE. However it can be very useful in practice to estimate the noise distribution, i.e., $\boldsymbol{q}$, during training, and select noise classes according to this distribution.

References

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of The International Conference on World Wide Web (WWW)*, pages 13–24, 2013.

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 163–171, 2010.

[3] A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 247–262. Springer, 2009.

[4] A. Botev, B. Zheng, and D. Barber. Complementary sum sampling for likelihood approximation in large scale classification. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1030–1038, 2017.

[5] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[6] A. E. Choromanska and J. Langford. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 55–63, 2015.

[7] H. Daume III, N. Karampatziakis, J. Langford, and P. Mineiro. Logarithmic time one-against-some. *arXiv preprint arXiv:1606.04988*, 2016.

[8] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision (ECCV)*, pages 71–84, 2010.

[9] J. Deng, S. Satheesh, A. C. Berg, and F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 567–575, 2011.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[11] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[12] J.-M. Geusebroek, G. J. Burghouts, and A. W. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision (IJCV)*, 61(1):103–112, 2005.

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

[14] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016.

[15] Y. Jernite, A. Choromanska, D. Sontag, and Y. LeCun. Simultaneous learning of trees and representations for extreme classification with application to language modeling. *arXiv preprint arXiv:1610.04658*, 2016.

[16] S. Ji, S. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *arXiv preprint arXiv:1511.06909*, 2015.

[17] A. Z. Kouzani and G. Nasireding. Multilabel classification by bch code and random forests. *International Journal of Recent Trends in Engineering*, 2(1):113–116, 2009.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[19] Q. V. Le. Building high-level features using large scale unsupervised learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8595–8598, 2013.

[20] B. Liu, F. Sadeghi, M. Tappen, O. Shamir, and C. Liu. Probabilistic label trees for efficient large scale image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 843–850, 2013.

[21] L. Liu, P. M. Comar, S. Saha, P.-N. Tan, and A. Nucci. Recursive nmf: Efficient label tree learning for large multi-class problems. In *The International Conference on Pattern Recognition (ICPR)*, pages 2148–2151, 2012.

[22] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(11):2624–2637, 2013.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[25] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1081–1088, 2009.

[26] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2265–2273, 2013.

[27] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1751–1758, 2012.

[28] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *The International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 246–252, 2005.

[29] P. Norvig. *Paradigms of artificial intelligence programming: case studies in Common LISP*. Morgan Kaufmann, 1992.

[30] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[31] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.

[32] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 263–272, 2014.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[34] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1665–1672, 2011.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015.

[37] M. K. Titsias. One-vs-each approximation to softmax for scalable estimation of probabilities. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4161–4169, 2016.

[38] A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. Decoding with large-scale neural language models improves translation. In *The Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1387–1392. Citeseer, 2013.

[39] P. Vincent, A. de Brébisson, and X. Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1108–1116, 2015.

[40] J. Weston, A. Makadia, and H. Yee. Label partitioning for sublinear ranking. In *Proceedings of The International Conference on Machine Learning (ICML)*, pages 181–189, 2013.

[41] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of The International Conference on Machine Learning (ICML)*, pages 3069–3077, 2016.

Supplementary Material

A Proofs

In the theorectical analysis, we fix $s_K(\boldsymbol{x}, \boldsymbol{\theta}) = 0$. Then, we only need to consider $\mathcal{C}_{\boldsymbol{x}} \cup \mathcal{N}_{\boldsymbol{x}} = \{1, \cdots, K-1\}$. Now, the normalization factor becomes

$$E(\boldsymbol{x}, j) = 1 + \sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}(\boldsymbol{x}, \boldsymbol{\theta})} + e^{s_j(\boldsymbol{x}, \boldsymbol{\theta})} / q_{\boldsymbol{x}}(j),$$

with some sampled class $j \in \mathcal{N}_{\boldsymbol{x}}$. Now, we can rewrite R and $\hat{R}$ as

$$\begin{aligned}\max_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}} \sum_{k \in \mathcal{C}_{\boldsymbol{x}}} p(y=k|\boldsymbol{x}) \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_{\boldsymbol{x}}(j) \log \frac{e^{s_k(\boldsymbol{x}, \boldsymbol{\theta})}}{E(\boldsymbol{x}, j)} + \sum_{k \in \mathcal{N}_{\boldsymbol{x}}} p(y=k|\boldsymbol{x}) \log \frac{e^{s_k(\boldsymbol{x}, \boldsymbol{\theta})}}{E(\boldsymbol{x}, k)} \\ + p(y=K|\boldsymbol{x}) \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_{\boldsymbol{x}}(j) \log \frac{1}{E(\boldsymbol{x}, j)}.\end{aligned}$$

$$\begin{aligned}\max_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \Bigg[\sum_{k \in \mathcal{C}_{\boldsymbol{x}_i}} \mathbb{I}(y_i = k) \sum_{j \in \mathcal{C}_{\boldsymbol{x}_i}} q_{\boldsymbol{x}_i}(j) \log \frac{e^{s_k(\boldsymbol{x}_i, \boldsymbol{\theta})}}{E(\boldsymbol{x}_i, j)} + \sum_{k \in \mathcal{N}_{\boldsymbol{x}_i}} \mathbb{I}(y_i = k) \log \frac{e^{s_k(\boldsymbol{x}_i, \boldsymbol{\theta})}}{E(\boldsymbol{x}_i, k)} \\ + \mathbb{I}(y_i = K) \sum_{j \in \mathcal{C}_{\boldsymbol{x}_i}} q_{\boldsymbol{x}_i}(j) \log \frac{1}{E(\boldsymbol{x}_i, j)} \Bigg].\end{aligned}$$

In the proofs, we will use point-wise notations p_k, s_k, q_k and E_k to represent $p(y=k|\boldsymbol{x})$, $s_k(\boldsymbol{x}, \boldsymbol{\theta})$, $q_{\boldsymbol{x}}(k)$ and $E(\boldsymbol{x}, k)$ for simplicity.

A.1 Useful Lemma

We will need the following lemma in our analysis.

Lemma 1. *For any norm $\|\cdot\|$ defined on the parameter space of $\boldsymbol{\theta}$, assume the quantities $\|\nabla_{\boldsymbol{\theta}} s_k\|$, $\|\nabla^2_{\boldsymbol{\theta}} s_k\|$ and $\|\nabla^3_{\boldsymbol{\theta}} s_k\|$ for $k = 1, \cdots, K-1$ are bounded. Then, for any compact set $\mathbb{S}$ defined on the parameter space, we have*

$$\sup_{\boldsymbol{\theta}\in\mathbb{S}} |\hat{R}_n(\boldsymbol{\theta}) - R(\boldsymbol{\theta})| \xrightarrow{p} 0,$$
$$\sup_{\boldsymbol{\theta}\in\mathbb{S}} \|\nabla \hat{R}_n(\boldsymbol{\theta}) - \nabla R(\boldsymbol{\theta})\| \xrightarrow{p} 0,$$
$$\sup_{\boldsymbol{\theta}\in\mathbb{S}} \|\nabla^2 \hat{R}_n(\boldsymbol{\theta}) - \nabla^2 R(\boldsymbol{\theta})\| \xrightarrow{p} 0.$$

Proof. For fixed $\boldsymbol{\theta}$, let

$$\psi(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_{k\in\mathcal{C}_{\boldsymbol{x}}} \mathbb{I}(y=k) \sum_{j\in\mathcal{N}_{\boldsymbol{x}}} q_j \log \frac{e^{s_k}}{1+\sum_{k'\in C_{\boldsymbol{x}_i}} e^{s_{k'}} + \frac{e^{s_j}}{q_j}} +$$
$$\mathbb{I}(y=K) \sum_{j\in\mathcal{N}_{\boldsymbol{x}}} q_j \log \frac{1}{1+\sum_{k'\in C_{\boldsymbol{x}}} e^{s_{k'}} + \frac{e^{s_j}}{q_j}} + \sum_{k\in\mathcal{N}_{\boldsymbol{x}}} \mathbb{I}(y=k) \log \frac{e^{s_k}}{1+\sum_{k'\in C_{\boldsymbol{x}}} e^{s_{k'}} + \frac{e^{s_k}}{q_k}}.$$

Then we have $\hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \psi(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{\theta})$ and $R(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},y}\ \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$. By the Law of Large Numbers, we know that $\hat{R}_n(\boldsymbol{\theta})$ converges point-wisely to $R(\boldsymbol{\theta})$ in probability.

According to the assumption, there exists a constant $M > 0$ such that

$$\|\nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{x}, y, \boldsymbol{\theta})\| \le \sum_{k=1}^{K-1} \|\nabla_{\boldsymbol{\theta}} s_k\| \le M.$$

Given any $\epsilon > 0$, we may find a finite cover $\mathbb{S}_\epsilon \subset \mathbb{S}$ so that for any $\boldsymbol{\theta} \in \mathbb{S}$, there exists $\boldsymbol{\theta'} \in \mathbb{S}_\epsilon$ such that $|\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) - \psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta'})| \le M\|\boldsymbol{\theta} - \boldsymbol{\theta'}\| < \epsilon$. Since $\mathbb{S}_\epsilon$ is finite, as $n \to \infty$, $\sup_{\boldsymbol{\theta}\in\mathbb{S}_\epsilon} |\hat{R}_n(\boldsymbol{\theta}) - R(\boldsymbol{\theta})|$ converges to 0 in probability. Therefore, as $n \to \infty$, with probability 1, we have

$$\sup_{\boldsymbol{\theta}\in\mathbb{S}} |\hat{R}_n(\boldsymbol{\theta}) - R(\boldsymbol{\theta})| < 2\epsilon + \sup_{\boldsymbol{\theta}\in\mathbb{S}_\epsilon} |\hat{R}_n(\boldsymbol{\theta}) - R(\boldsymbol{\theta})| \to 2\epsilon.$$

Let $\epsilon \to 0$, we obtain the first bound. The second and the third bounds can be similarly obtained. □

A.2 Proof of Theorem 1

Proof. R can be re-written as

$$R = \mathbb{E}_{\boldsymbol{x}} \sum_{j\in\mathcal{N}_{\boldsymbol{x}}} q_j \left(\sum_{k\in\mathcal{C}_{\boldsymbol{x}}} p_k \log \frac{e^{s_k}}{1+\sum_{k'\in\mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}} + e^{s_j}/q_j} + p_K \log \frac{1}{1+\sum_{k'\in\mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}} + e^{s_j}/q_j} \right.$$
$$\left. + p_j/q_j \log \frac{e^{s_j}}{1+\sum_{k'\in\mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}} + e^{s_j}/q_j} \right).$$

For $i \in \mathcal{C}_{\boldsymbol{x}}$, we have

$$\begin{aligned}\nabla_{s_i} R &= \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \left[p_i \left(1 - \frac{e^{s_i}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} \right) - \sum_{k \neq i \in \mathcal{C}_{\boldsymbol{x}}} p_k \frac{e^{s_i}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} \right. \\ &\qquad \left. - p_K \frac{e^{s_i}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} - p_j/q_j \frac{e^{s_i}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} \right] \\ &= \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \left[p_i - \left(p_K + \sum_{k \in \mathcal{C}_x} p_k + p_j/q_j \right) \frac{e^{s_i}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} \right].\end{aligned}$$

Similarly, for $j \in \mathcal{N}_{\boldsymbol{x}}$, we have

$$\begin{aligned}\nabla_{s_j} R &= \mathbb{E}_{\boldsymbol{x}}\ q_j \left[- \left(p_K + \sum_{k \in \mathcal{C}_{\boldsymbol{x}}} p_k \right) \frac{e^{s_j}/q_j}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} + p_j/q_j \left(1 - \frac{e^{s_j}/q_j}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j} \right) \right] \\ &= \mathbb{E}_{\boldsymbol{x}}\ p_j - \left(p_K + \sum_{k \in \mathcal{C}_{\boldsymbol{x}}} p_k + p_j/q_j \right) \frac{e^{s_j}}{1 + \sum_{k' \in \mathcal{C}_x} e^{s_{k'}} + e^{s_j}/q_j}.\end{aligned}$$

By measuring $s_k = \log \frac{p_k}{p_K}$, we see that $\nabla_{s_k} R = 0$ for $k = 1, \cdots, K-1$. Therefore, $s_k = \log \frac{p_k}{p_K}$ is an extrema of R. Now, for $i, i' \in C_{\boldsymbol{x}}$ and $j, j' \in \mathcal{N}_{\boldsymbol{x}}$, we have

$$\begin{aligned}\mathbb{H}_{ii} &= \nabla^2_{s_i s_i} R = -\mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j D_j \frac{e^{s_i}(E_j - e^{s_i})}{E_j^2}, \\ \mathbb{H}_{ii'} &= \nabla^2_{s_i s_{i'}} R = \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j D_j \frac{e^{s_i} e^{s_{i'}}}{E_j^2}, \\ \mathbb{H}_{ij} &= \mathbb{H}_{ji} = \nabla^2_{s_i s_j} R = \nabla^2_{s_j s_i} R = \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} D_j \frac{e^{s_i} e^{s_j}}{E_j^2}, \\ \mathbb{H}_{jj} &= \nabla^2_{s_j s_j} R = -\mathbb{E}_{\boldsymbol{x}}\ D_j \frac{e^{s_j}(E_j - e^{s_j}/q_j)}{E_j^2}, \\ \mathbb{H}_{jj'} &= \nabla^2_{s_j s_{j'}} R = 0,\end{aligned}$$

where

$$D_j = p_K + \sum_{k' \in \mathcal{C}_x} p_{k'} + p_j/q_j.$$

Now, we can write

$$\nabla_s^2 R = \left[\begin{array}{ccc|ccccc} \mathbb{H}_{i_1 i_1} & \cdots & \mathbb{H}_{i_1 i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{H}_{i_1 j} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbb{H}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} i_1} & \cdots & \mathbb{H}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{H}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} j} & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbb{H}_{j i_1} & \cdots & \mathbb{H}_{j i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{H}_{jj} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \end{array}\right]$$

$$= -\mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \frac{D_j}{E_j} \left\{ diag\left(\begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix} \right) - \frac{1}{E_j} \begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix}^\top \right\}.$$

Let

$$\boldsymbol{A}_j = diag\left(\begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix} \right) - \frac{1}{E_j} \begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} e^{s_{i_1}} \\ \vdots \\ e^{s_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}} \\ 0 \\ \vdots \\ e^{s_j}/q_j \\ \vdots \\ 0 \end{bmatrix}^\top$$

For any non-zero vector $\boldsymbol{\varphi} = (\varphi_1, \cdots, \varphi_{K-1})^\top \in \mathbb{R}^{K-1}$, we have

$$\begin{aligned} \boldsymbol{\varphi}^\top \boldsymbol{A}_j \boldsymbol{\varphi} &= \sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} \varphi_i^2 + \frac{e^{s_j}}{q_j} \varphi_j^2 - \frac{1}{E_j} \left(\sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} \varphi_i + \frac{e^{s_j}}{q_j} \varphi_j \right)^2 \\ &\geq \frac{\left(\sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} \varphi_i + \frac{e^{s_j}}{q_j} \varphi_j \right)^2}{\sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} + \frac{e^{s_j}}{q_j}} - \frac{1}{E_j} \left(\sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} \varphi_i + \frac{e^{s_j}}{q_j} \varphi_j \right)^2 \\ &> 0, \end{aligned}$$

for every $j \in \mathcal{N}_{\boldsymbol{x}}$, where the first inequality is by the Cauchy-Schwarz inequality and the second inequality is because $0 < \sum_{i \in \mathcal{C}_{\boldsymbol{x}}} e^{s_i} + \frac{e^{s_j}}{q_j} < E_j$. Therefore, $-\nabla_s^2 R = \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \frac{D_j}{E_j} \boldsymbol{A}_j$ is positive-

definite and R is strongly concave with respect to s. Hence, $s_k = \log \frac{p_k}{p_K}$ for $k = 1, \cdots, K-1$ is the only maxima of R. □

A.3 Proof of Theorem 2

Proof. For any $\epsilon > 0$ and non-zero vector $\boldsymbol{\varphi} \in \mathbb{R}^{K-1}$, let $\boldsymbol{\theta} = \boldsymbol{\theta}^* + \epsilon\boldsymbol{\varphi}$. The Taylor expansion of $R(\boldsymbol{\theta})$ can be written as

$$R(\boldsymbol{\theta}) = R(\boldsymbol{\theta}^*) + \epsilon\boldsymbol{\varphi}^\top \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) + \frac{\epsilon^2}{2}\boldsymbol{\varphi}^\top \nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*)\boldsymbol{\varphi} + O(\epsilon^3). \tag{12}$$

From the proof of Theorem 1, we have

$$\nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \left[\sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \frac{D_j}{E_j} \boldsymbol{A}_j \right] \boldsymbol{\nabla}^\top, \tag{13}$$

where

$$\boldsymbol{\nabla} = diag\left(\left(\nabla_{i_1}, \cdots, \nabla_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}, \nabla_{j_1}, \cdots, \nabla_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} \right)^\top \right) \quad \text{and} \quad \nabla_k = \nabla_{\boldsymbol{\theta}} s_k.$$

Measuring $\nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^*$, we have

$$\nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) = -\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top \tag{14}$$

where

$$\boldsymbol{M} = \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \left\{ diag\left(\begin{bmatrix} p_{i_1} \\ \vdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} \\ 0 \\ \vdots \\ p_j/q_j \\ \vdots \\ 0 \end{bmatrix} \right) - \frac{1}{D_j} \begin{bmatrix} p_{i_1} \\ \vdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} \\ 0 \\ \vdots \\ p_j/q_j \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} p_{i_1} \\ \vdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} \\ 0 \\ \vdots \\ p_j/q_j \\ \vdots \\ 0 \end{bmatrix}^\top \right\}.$$

By following the proof of Theorem 1, it is easy to show that $\boldsymbol{M} \succ 0$ is positive definite. According to the assumption, the matrix $\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top$ is positive definite. Moreover, we have $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) = 0$. Now, Eq. (12) implies that

$$R(\boldsymbol{\theta}) - R(\boldsymbol{\theta}^*) = \frac{\epsilon^2}{2}\boldsymbol{\varphi}^\top \nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*)\boldsymbol{\varphi} + O(\epsilon^3) < 0.$$

That is, for any $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$, we have $R(\boldsymbol{\theta}) < R(\boldsymbol{\theta}^*)$. Given any $\varepsilon' > 0$, there exists $\varepsilon > 0$ that $R(\boldsymbol{\theta}^*) - R(\boldsymbol{\theta}) < \varepsilon$ implies $\|\boldsymbol{\theta}^* - \boldsymbol{\theta}\| < \varepsilon'$. Now according to Lemma 1, there exists a $\delta > 0$, when $n \to \infty$, we have

$$\begin{aligned} R(\boldsymbol{\theta}^*) - R(\hat{\boldsymbol{\theta}}) &= R(\boldsymbol{\theta}^*) - \hat{R}_n(\boldsymbol{\theta}^*) + \hat{R}_n(\boldsymbol{\theta}^*) - R(\hat{\boldsymbol{\theta}}) \\ &\leq R(\boldsymbol{\theta}^*) - \hat{R}_n(\boldsymbol{\theta}^*) + \hat{R}_n(\hat{\boldsymbol{\theta}}) - R(\hat{\boldsymbol{\theta}}) \\ &\leq |R(\boldsymbol{\theta}^*) - \hat{R}_n(\boldsymbol{\theta}^*)| + |\hat{R}_n(\hat{\boldsymbol{\theta}}) - R(\hat{\boldsymbol{\theta}})| \\ &< 2\delta. \end{aligned}$$

This implies that $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\| < \delta'$ for any $\delta' > 0$. □

A.4 Proof of Theorem 3

Proof. By the Mean Value Theorem, we have

$$\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) = -\nabla^2 \hat{R}_n(\bar{\boldsymbol{\theta}})^{-1} \sqrt{n} \nabla \hat{R}_n(\boldsymbol{\theta}^*), \tag{15}$$

where $\bar{\boldsymbol{\theta}} = t\boldsymbol{\theta}^* + (1-t)\hat{\boldsymbol{\theta}}$ for some $t \in [0,1]$. Note that Lemma 1 implies that $\nabla^2 \hat{R}_n(\bar{\boldsymbol{\theta}})^{-1}$ converges to $\nabla^2 R(\bar{\boldsymbol{\theta}})^{-1}$ in probability; moreover, $\hat{\boldsymbol{\theta}} \to \boldsymbol{\theta}^*$ in probability and hence $\bar{\boldsymbol{\theta}} \to \boldsymbol{\theta}^*$ in probability. By the Slutsky's Theorem, the limit distribution of $\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)$ is given by

$$-\nabla^2 R(\boldsymbol{\theta}^*)^{-1} \sqrt{n} \nabla \hat{R}_n(\boldsymbol{\theta}^*).$$

Observe that $\sqrt{n}\nabla \hat{R}_n(\boldsymbol{\theta}^*)$ is the sum of n i.i.d. random vectors with mean $\mathbb{E}\sqrt{n}\nabla \hat{R}_n(\boldsymbol{\theta}^*) = \sqrt{n}\mathbb{E}\nabla R(\boldsymbol{\theta}^*) = 0$, and the variance of $\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)$ is

$$Var\left(\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)\right) = \nabla^2 R(\boldsymbol{\theta}^*)^{-1} Var\left(\sqrt{n}\nabla \hat{R}_n(\boldsymbol{\theta}^*)\right) \nabla^2 R(\boldsymbol{\theta}^*)^{-1}.$$

Next, we derive $Var\left(\sqrt{n}\nabla \hat{R}_n(\boldsymbol{\theta}^*)\right)$. Introduce some Bernoulli variables Q_j for $j \in \mathcal{N}_{\boldsymbol{x}}$ with $p(Q_j = 1|\boldsymbol{x}) = q_j$. Now, for $i, i' \in C_{\boldsymbol{x}}$ and $j, j' \in \mathcal{N}_{\boldsymbol{x}}$, we have

$$\begin{aligned}
\mathbb{V}_{ii} =& Var\left(\nabla_i \hat{R}_n(\boldsymbol{\theta}^*), \nabla_i \hat{R}_n(\boldsymbol{\theta}^*)\right) \\
=& \mathbb{E}_{\boldsymbol{x},Q}\, Q\left[p_i\left(1 - \frac{e^{s_i^*}}{1 + \sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}^*} + e^{s_j^*}/q_j}\right)^2 + (D_j - p_i)\left(\frac{e^{s_i^*}}{1 + \sum_{k' \in \mathcal{C}_{\boldsymbol{x}}} e^{s_{k'}^*} + e^{s_j^*}/q_j}\right)^2\right] \cdot \nabla_i \nabla_i^\top \\
=& \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \frac{p_i(D_j - p_i)}{D_j} \cdot \nabla_i \nabla_i^\top,
\end{aligned}$$

$$\begin{aligned}
\mathbb{V}_{ii'} =& Var\left(\nabla_i \hat{R}_n(\boldsymbol{\theta}^*), \nabla_{i'} \hat{R}_n(\boldsymbol{\theta}^*)\right) \\
=& \mathbb{E}_{\boldsymbol{x},Q}\, Q\left[(D_j - p_i - p_{i'})\frac{p_i p_{i'}}{D_j^2} - p_i(1 - \frac{p_i}{D_j})\frac{p_{i'}}{D_j} - p_{i'}(1 - \frac{p_{i'}}{D_j})\frac{p_i}{D_j}\right] \cdot \nabla_i \nabla_{i'}^\top \\
=& -\mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} q_j \frac{p_i p_{i'}}{D_j} \cdot \nabla_i \nabla_{i'}^\top.
\end{aligned}$$

$$\begin{aligned}
\mathbb{V}_{jj} =& Var\left(\nabla_j \hat{R}_n(\boldsymbol{\theta}^*), \nabla_j \hat{R}_n(\boldsymbol{\theta}^*)\right) \\
=& \mathbb{E}_{\boldsymbol{x},Q}\, Q\left[\frac{p_j}{q_j}\left(1 - \frac{p_j/q_j}{D_j}\right)^2 + (D_j - p_j/q_j)\frac{p_j^2/q_j^2}{D_j^2}\right] \cdot \nabla_j \nabla_j^\top \\
=& \mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} \frac{p_j\,(D_j - p_j/q_j)}{D_j} \cdot \nabla_j \nabla_j^\top.
\end{aligned}$$

$$\mathbb{V}_{jj'} = \mathbf{0}.$$

$$\begin{aligned}
\mathbb{V}_{ij} = \mathbb{V}_{ji} =& Var\left(\nabla_i \hat{R}_n(\boldsymbol{\Theta}^*), \nabla_j \hat{R}_n(\boldsymbol{\Theta}^*)\right) \\
=& \mathbb{E}_{\boldsymbol{x},\boldsymbol{Q}}\, Q\left[(D_j - p_i - p_j/q_j)\frac{p_i p_j/q_j}{D_j^2} - p_i\left(1 - \frac{p_i}{D_j}\right)\frac{p_j/q_j}{D_j} - p_j/q_j\left(1 - \frac{p_j/q_j}{D_j}\right)\frac{p_i}{D_j}\right] \cdot \nabla_i \nabla_{i'}^\top \\
=& -\mathbb{E}_{\boldsymbol{x}} \sum_{j \in \mathcal{N}_{\boldsymbol{x}}} \frac{p_i p_j}{D_j} \cdot \nabla_i \nabla_{i'}^\top.
\end{aligned}$$

Now, the variance can be written as

$$V(\boldsymbol{\theta}^*) = Var\left(\sqrt{n}\nabla\hat{R}_n(\boldsymbol{\theta}^*)\right) = \left[\begin{array}{ccc|ccccc} \mathbb{V}_{i_1 i_1} & \cdots & \mathbb{V}_{i_1 i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{V}_{i_1 j} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbb{V}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} i_1} & \cdots & \mathbb{V}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{V}_{i_{|\mathcal{C}_{\boldsymbol{x}}|} j} & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbb{V}_{j i_1} & \cdots & \mathbb{V}_{j i_{|\mathcal{C}_{\boldsymbol{x}}|}} & 0 & \cdots & \mathbb{V}_{jj} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \end{array}\right]. \tag{16}$$

By comparing $\nabla^2 R(\boldsymbol{\theta}^*)$ and $V(\boldsymbol{\theta}^*)$, we immediately have $-\nabla^2 R(\boldsymbol{\theta}^*) = V(\boldsymbol{\theta}^*)$ and hence

$$Var\left(\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)\right) = \left[\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M} \boldsymbol{\nabla}^\top\right]^{-1}.$$

□

A.5 Proof of Corollary 1

Proof. By following the proof of Theorem 3, it is easy to show that the statistical variance of the softmax logistic regression in Eq. (1) is $[\mathbb{E}_{\boldsymbol{x}} \boldsymbol{\nabla} \boldsymbol{M}^{mle} \boldsymbol{\nabla}^\top]^{-1}$ (with $s_K = 0$ fixed), where

$$\boldsymbol{M}^{mle} = diag\left(\begin{bmatrix} p_1 \\ \vdots \\ p_{K-1} \end{bmatrix}\right) - \begin{bmatrix} p_1 \\ \vdots \\ p_{K-1} \end{bmatrix}\begin{bmatrix} p_1 \\ \vdots \\ p_{K-1} \end{bmatrix}^\top.$$

When $\sum_{k \in \mathcal{C}_{\boldsymbol{x}} \cup \{K\}} p(k, \boldsymbol{x}) \to 1$, we have $\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} \to 0$ and $D_j \to 1$. Then,

$$\boldsymbol{M} = diag\left(\begin{bmatrix} p_{i_1} \\ \vdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} \\ p_{j_1} \\ \vdots \\ p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} \end{bmatrix}\right) - \left[\begin{array}{ccc|ccc} p_{i_1}p_{i_1} & \cdots & p_{i_1}p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} & p_{i_1}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & \cdots & p_{i_1}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}p_{i_1} & \cdots & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}} & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & \cdots & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} \\ \hline p_{i_1}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & \cdots & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & p_{j_1}^2/q_{j_1} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{i_1}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & \cdots & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}\sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'} & 0 & \cdots & p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}}^2/q_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} \end{array}\right].$$

If we arrange the index order in $\boldsymbol{M}^{mle}$ according to the index order in $\boldsymbol{M}$ and denote $\boldsymbol{\Delta} = \boldsymbol{M} - \boldsymbol{M}^{mle}$, we have

$$\boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\Delta}_1 & \boldsymbol{\Delta}_2 \\ \boldsymbol{\Delta}_2^\top & \boldsymbol{\Delta}_3 \end{bmatrix} \to \mathbf{0},$$

because

$$\boldsymbol{\Delta}_1 = \mathbf{0},$$

$$\boldsymbol{\Delta}_2 = \begin{bmatrix} p_{i_1}(p_{j_1} - \sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'}) & \cdots & p_{i_1}(p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} - \sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'}) \\ \cdots & \cdots & \cdots \\ p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}(p_{j_1} - \sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'}) & \cdots & p_{i_{|\mathcal{C}_{\boldsymbol{x}}|}}(p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} - \sum_{j' \in \mathcal{N}_{\boldsymbol{x}}} p_{j'}) \end{bmatrix} \to \mathbf{0},$$

$$\boldsymbol{\Delta}_3 = \begin{bmatrix} p_{j_1}^2(1 - 1/q_{j_1}) & \cdots & p_{j_1} p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} \\ \cdots & \cdots & \cdots \\ p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}} p_{j_1} & \cdots & p_{j_{|\mathcal{N}_{\boldsymbol{x}}|}}^2(1 - 1/q_{j_{|\mathcal{N}_{\boldsymbol{x}}|}}) \end{bmatrix} \to \mathbf{0}.$$

This completes the proof. □

B The Beam Search Algorithm

The beam search algorithm used in both training and testing is depicted in Algorithm 3.

C A Hierarchical Clustering Method for Generating the Tree Structure

Given the data points of a dataset, we can obtain the center, i.e., the average data point, of each class by scanning the data once and get $\bar{\boldsymbol{X}} \in \mathbb{R}^{K \times d}$, where K is the number of classes and d is the feature dimension. Then, a hierarchical clustering algorithm in Algorithm 4 is performed by viewing each row of $\bar{\boldsymbol{X}}$ as a separate data point. In Algorithm 4, the function 'Split(root)' in step 16 has already constructed a b-nary tree, which can be used by the Beam Tree. However, the clustering algorithm, e.g., the k-means algorithm used in this paper, may generate imbalanced clusters in step 9, and the resulting b-nary tree in step 16 may be imbalanced and affect the efficiency of Beam Tree. A simple way to fix this problem is to fetch the labels (leaves) in the tree in step 16 from left to right, where the obtained label order contains a rough similarity relationship among the classes. We then assign the ordered labels to the leaves of a new balanced b-nary tree from left to right. In the LSHTC1 and Dmoz datasets, the feature dimension d is large while the feature space is sparse. In order to obtain a rough tree structure, we only keep the top 1000 features with the largest summation values over the class centers. As shown in the experiments, the above tree structure is sufficient for CANE to achieve good performance.

D Experimental Details

Hyper-parameter tuning is computationally expensive. In order to efficiently select a good setting of the hyper-parameters, we let each method process half epoch of the training data and use another 10% held-out subset of the training set to tune hyper-parameters. For every classifier, the learning rate η needs to be tuned. For the LOMTree method, by following [6], we choose the number of the

Algorithm 3 The Beam Search Algorithm.

Input: The root of the tree, input data point $\boldsymbol{x}$ and Beam width J.
Output: The J candidate classes.

Initialize stack $\mathcal{S} \leftarrow root$ and stack $\mathcal{S}' \leftarrow \emptyset$;
Initialize the candidate class set $\mathcal{E} \leftarrow \emptyset$;
while true **do**
  if $\mathcal{S}$ is empty **then**
    Break;
  end if
  for $i = 1$ to $\mathcal{S}.size()$ **do**
    if $\mathcal{S}_i$ is a leaf **then**
      $\mathcal{E}.pushback(\mathcal{S}_i)$;
    else
      for $c = 1$ to $\mathcal{S}_i.Child.size()$ **do**
        Accumulate the score to $\mathcal{S}_i.Child(c)$;
        $\mathcal{S}'.pushback(\mathcal{S}_i.Child(c))$;
      end for
    end if
  end for
  $\mathcal{S}.clear()$;
  if $\mathcal{S}'.size() > J$ **then**
    // Using the max heap.
    Find the top-J nodes with the highest accumulated scores in $\mathcal{S}'$ and push them into $\mathcal{S}$;
  else
    $\mathcal{S} \leftarrow \mathcal{S}'$;
  end if
  $\mathcal{S}'.clear()$;
end while
// Using the max heap.
Return the top-J classes with the highest scores in $\mathcal{E}$;

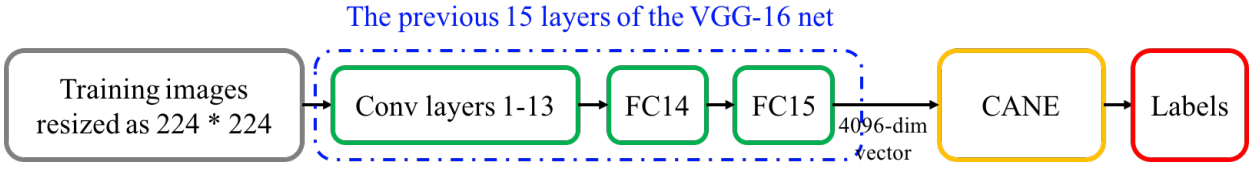


Figure 5: The neural network structure used for the ImageNet datasets. 'FC' indicates fully-connected layer.

internal nodes in its binary tree from a set $\{K-1, 4K-1, 16K-1, 64K-1\}$, and tune the swap resistance from $\{4, 16, 64, 256\}$. The Recall Tree method has a default setting for large class problem in [7], which is also adopted in the experiments.

The VGG-16 network structure used in ImageNet 2010 and ImageNet-10K datasets is provided in Fig. 5. Parameters of Conv layers 1-13, FC14 and FC15 are pre-trained on the ImageNet 2012 dataset.

Algorithm 4 A Hierarchical Clustering Algorithm for Generating the Tree over Class Labels.

Input: K, b and $\bar{\boldsymbol{X}}$.
Output: a b-nary tree.

Function Split(node o)
while true **do**
  if o is assigned with only one label **then**
    $o.isleaf = true$;
    Return;
  end if
  Perform any clustering algorithm, e.g., k-means, on the labels associated with the node o and obtain b clusters $\{\mathcal{L}_1, \cdots, \mathcal{L}_b\}$;
  Split o into b children $\{o_1, \cdots, o_b\}$ and assign the label clusters $\{\mathcal{L}_1, \cdots, \mathcal{L}_b\}$ to them respectively;
  for $i = 1$ to b **do**
    Split(o_i);
  end for
end while

Assign root with all labels $\{1, 2, \cdots, K\}$;
Split(root);
Get the label order in the leaves from left to right;
Assign the labels to the leaves of a new balanced b-nary tree from left to right;
Return the balanced b-nary tree;