Doubly Sparse: Sparse Mixture of Sparse Experts for Efficient Softmax Inference

Shun Liao *
University of Toronto
sliao3@cs.toronto.edu

Ting Chen *
Google Brain
iamtingchen@google.com

Tian Lin
Google
tinglin@google.com

Denny Zhou
Google Brain
dennyzhou@google.com

Chong Wang
ByteDance
chongwang@bytedance.com

Abstract

Computations for the softmax function are significantly expensive when the number of output classes is large. In this paper, we present a novel softmax *inference* speedup method, Doubly Sparse Softmax (DS-Softmax), that leverages a sparse mixture of sparse experts to efficiently retrieve top-k classes. Most existing methods convert softmax inference into a Maximum Inner Product Search (MIPS) problem, which requires and approximates a fixed softmax embedding. However, most approximations cannot achieve significant speedup without sacrificing accuracy. Our method can achieve a better trade-off between efficiency and accuracy by encoding the inference speedup objective into training and adapt the softmax embedding for a better trade-off, making this method learning-based. In particular, DS-Softmax is forced to learn a two-level hierarchy which divides entire output class space into several partially overlapping experts. Each expert is sparse and only contains a subset of output classes. To find top-k classes, a sparse mixture is to find the most probable expert, and the sparse expert can search within a small-scale softmax. We empirically conduct an evaluation on several real-world tasks, including neural machine translation, language modeling and image classification, and achieve significant computation reductions at no performance loss.

1 Introduction

Deep learning models have demonstrated impressive performance in many classification problems [1]. In many models, the softmax function is commonly used to produce categorical distributions over the output space. Due to its linear complexity, the computation for the softmax layer becomes a bottleneck with large output dimensions [2]. In language modelling task, softmax contributes to more than 95% computation of the small model. This becomes a significant bottleneck with limited computational resource, such as mobile devices [3].

Many methods are proposed to reduce softmax complexity, in both training and inference phases. For training, the goal is to approximate the normalization term quickly. Unlike training, the goal in inference is to search for top-k classes efficiently. Most existing methods formulate this as a maximum inner product search (MIPS) problem, post-approximation methods here: given an already learned/fixed softmax, how to search the top-k classes without linear complexity [4, 5, 2]. However, the fixed softmax may not be structured in a (hierarchical) way such that locating top-k can be easily achieved, leading to sub-optimal trade-offs between efficiency and accuracy [2].

In this work, we propose a novel Doubly Sparse Softmax (DS-Softmax), which can search top-k efficiently in inference. DS-Softmax is a learning-based method and change softmax embedding

EMC^2: 5th Edition Co-located with Neural Information Processing Systems 2019, Vancouver, Canada.

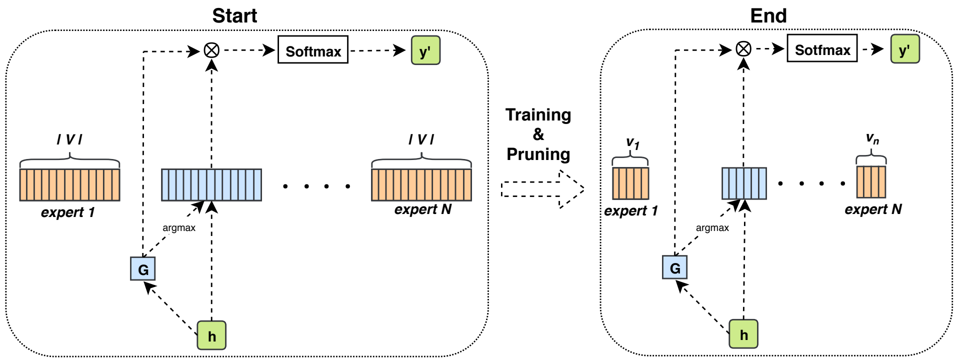


Figure 1: Overview of Doubly Sparse Softmax (DS-Softmax). Each expert is initialized with the full output space and only the expert with the highest gating value is selected feed-forward. During training, each expert is pruned iteratively so that it finally contains a subset of classes, $|v_n|$.

to be hierarchically structured during training, aiming for a better trade-off. During training, the model learns a two-level overlapping hierarchy using *sparse* mixture of *sparse* experts structure. Each expert is sparse and only contains a small subset of entire output class space, while each class is permitted to belong to more than one expert. In inference, given an input vector and a set of experts, DS-Softmax first selects the top expert that is most related to the input (in contrast to a dense mixture of experts). Then, the single selected expert can return the categorical distribution on a small subset of classes. Therefore, complexity reduction is achieved as the model does not need to consider the whole vocabulary. Due to our design, DS-Softmax is also orthogonal with post-approximation methods, so they can be applied to DS-Softmax by treating each expert as another softmax to approximate.

We conduct experiments in one synthetic dataset and three different real tasks, including language modeling, neural machine translation, and image classification. We demonstrate our method can reduce softmax computation dramatically without loss of prediction performance. For example, we achieved more than 23.8x speedup in language modeling and 15.0x speedup in translation without loss of performance. By combing SVD-Softmax, DS-Softmax achieves 32.7x in language modelling.

To our knowledge, the proposed method is novel that adapts softmax weights for top-k retrieval instead of simply approximating the softmax. Through comprehensive experiments, we show the proposed method provides significant inference speedup for softmax without performance loss.

2 Related Work

Post-approximation based. Most existing works for reducing the softmax inference complexity are based on *post-approximation of a fixed softmax*. Some classical techniques have been demonstrated as powerful in this problem [6, 5]. Factorization and clustering are also shown as effective by recent works [4, 2]. However, as an approximation to a fixed softmax, the main drawback is that it always suffers high cost when high precision is required [2], suggesting a worse trade-off between efficiency and accuracy. In contrast, the proposed DS-softmax is able to adapt the softmax and learn a hierarchical structure to find top-k classes adaptively. Furthermore, it is possible that those methods can also be applied upon our method, where each expert can be viewed as a single softmax.

Hierarchy based. The most related ones under this category are D-softmax [7, 8]. They construct hierarchy manually through unbalanced word/class distribution (Zipf's law). There are two major issues. Firstly, their hierarchy is pre-defined by heuristics that could be sub-optimal. Secondly, the skewness of class distribution in some tasks, e.g. image classification, is not as significant as in language. DS-softmax overcomes those limitations by learning the two-level overlapping hierarchy.

Mixture of Experts. Our method is inspired by sparsely-gated mixture-of-experts (MoE) [9]. MoE aims to achieve better expressiveness through larger but only sparsely activated model. However, MoE cannot speedup the softmax inference by definition.

3 Method

Softmax Inference Problem. Given a context vector $h \in \mathbb{R}^d$, a softmax layer is used in order to compute a categorical distribution over a set of classes, defined as $P(class = c|h) = \exp(W_c h)/Z$ where $Z = \sum_i \exp(W_i h)$ is the normalization term and $W \in \mathbb{R}^{N \times d}$ is the softmax embedding. For inference, our goal is to find the top-k classes, i.e. $\{c|P(class = c|h) \geq p_k\}$ where p_k is the k-th largest value of $P(class|h)$. Conventionally, top-k searching has $\mathcal{O}(N)$ complexity but it can become a computational bottleneck if the output space is huge (i.e. given a large N).

Motivation. Many natural discrete objects/classes, such as natural language, exhibit some hierarchical structure where objects are organized in a tree-like fashion. A two-level hierarchy is studied for language modeling, where each word belongs to a unique cluster while the hierarchy is constructed with different approaches (A "cluster" here refers to a cluster of words). However, the construction of the hierarchy is very challenging and heuristics. Moreover, exclusiveness limits the expressiveness because exactly assigning a word to a single cluster is often difficult. For example, one possible next word of "I want to eat ___" can be "cookie". We can dramatically increase the efficiency if only search inside words with the eatable property. However, "cookie" might also appear under some non-edible context such as "a piece of data" in computer science literature. Thus, a two-level overlapping hierarchy can naturally accommodate word homonyms like this by allowing each word to belong to more than one cluster. We believe this observation is valid in other applications.

Doubly Sparse Softmax The framework is illustrated in Figure 1, which contains two major components: (1) the ***sparse mixture*** indicates the gating and enables the selection of a top-1 expert, where the utilization of different expert is balanced by a loading balance term ($\mathcal{L}_{load}$), and (2) the ***sparse experts*** that contains a subset of classes, which are created by iterative pruning with group lasso regularization ($\mathcal{L}_{lasso}$ and $\mathcal{L}_{expert}$). The learning objective $\mathcal{L}_{all}$ is to minimize those three losses in addition to task-specific loss $\mathcal{L}_{task}$, shown in Eq. 1 with weight hyper-parameters γ. The training is end-to-end but pre-training other layers except softmax can achieve faster converge in practice. After training, to generate the top-k classes, the sparse mixture selects the right expert according to context vector h, and the selected sparse expert can only search over a small subset.

$$\mathcal{L}_{all} = \mathcal{L}_{task} + \gamma_{load}\mathcal{L}_{load} + \gamma_{lasso}\mathcal{L}_{lasso} + \gamma_{expert}\mathcal{L}_{expert} \tag{1}$$

Sparse mixture. To facilitate faster inference, only a *single* most suitable expert is selected. To be more specific, suppose we have K experts. Given the context vector h and gating network weight U, the gating values $G_k(h)$, $k = 1, ..., K$, are calculated and normalized prior to the selection as shown in Eq. 2. We set all gates to be zero except the largest one. More specifically,

$$G_k(h) = \begin{cases} \frac{\exp(U_k h)}{\sum_{k'} \exp(U_{k'} h)}, & \text{if } k = \arg\max_i \frac{\exp(U_k h)}{\sum_{k'} \exp(U_{k'} h)}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

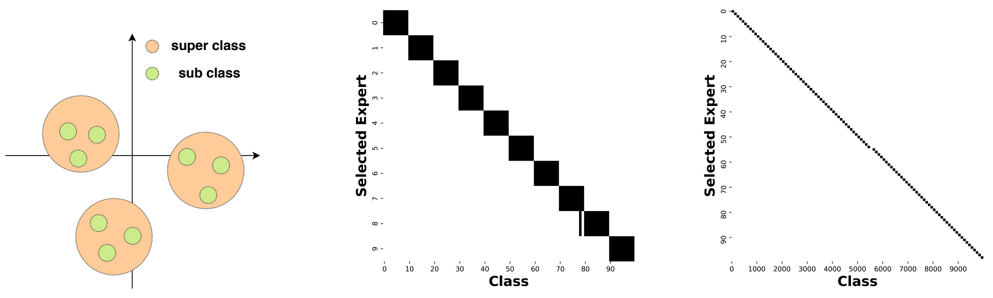
Where $U \in \mathbb{R}^{K \times d}$ is the weighting matrix for expert selection, and only the top-1 expert is selected. It is worth noting that our formulation has a valid gradient due to the normalization term while MoE cannot [9]. Given the sparse gate, we can compute the probability of class c under the context h as:

$$O(h) = p(class = c|h) = \frac{\exp(\sum_k G_k(h) W_c^{(k)} h)}{\sum_{c'} \exp(\sum_k G_k(h) W_{c'}^{(k)} h)}, \tag{3}$$

where $W^{(k)} \in \mathbb{R}^{N \times d}$ is softmax embedding weight matrix for the k-th expert. To balance the utilization of each expert, the gating network is regularized through $\mathcal{L}_{load}$ (refer to [9]).

Sparse expert. After training, each expert only contains a small subset of whole classes. To obtain a sparse expert, we start by initializing an expert as a full softmax, and then train with group lasso regularization, $\mathcal{L}_{lasso}$ (Eq. 4). In addition, expert level group lasso loss, $\mathcal{L}_{expert}$ (Eq. 4), is included to encourage each class exist in only one or a few experts. In each training epoch, pruning is conducted to remove embedding vectors with norm smaller than threshold λ.

$$\begin{aligned} \mathcal{L}_{lasso} &= \sum_k \sum_c \|\hat{W}_c^{(k)}\|_2 \\ \mathcal{L}_{expert} &= \sum_k \sqrt{\sum_c \|W_c^{(k)}\|_2^2} \end{aligned} \tag{4}$$



(a) Synthetic Data Generation (b) Results on 10 x 10 (c) Results on 100 x 100

Figure 2: (a) Illustration of synthetic data. (b) and (c) Results on tasks with 10x10 and 100x100 sizes. The x-axis indicates sub class (ordered by super class information for visualization purpose) and y-axis shows the expert. Black means this expert is handling this sub cluster.

Table 1: Comparison with SVD-softmax and D-Softmax on real device latency. The "ms" indicates the latency in microseconds. "FLOPs" indicates FLOPs speedup. The value is the task performance.

Task	Full		SVD-10			D-Softmax			DS-64 (Ours)		
	Value	ms	Value	FLOPs	ms	Value	FLOPs	ms	Value	FLOPs	ms
PTB	0.252	0.73	0.251	5.00×	0.18	0.245	2.00×	0.36	**0.258**	**15.99**×	**0.05**
Wiki-2	0.257	3.07	0.255	5.38×	0.60	0.256	2.00×	1.59	**0.259**	**23.86**×	**0.15**
En-Ve	**25.2**	1.91	25.1	5.06×	0.42	24.8	2.00×	0.98	25.0	**15.08**×	**0.13**
CASIA	**90.6**	1.61	90.2	2.61×	0.68	-	-	-	90.1	**6.91**×	**0.25**

4 Experiments

Empirical evaluations are done on both real and synthetic tasks †. Firstly, one synthetic task with two-level hierarchy is created to test the ability to learn the hierarchical structure. Secondly, we consider three real tasks, natural language modeling, neural machine translation, and Chinese handwritten character recognition. Both theoretical speedup (reduction in FLOPs) and real device latency (on CPU) are reported in Table 1, comparing to full softmax, SVD-Softmax [4] and D-Softmax [7, 8].

4.1 Synthetic Task

As illustrated in Figure 2(a), data points are organized with hierarchical centers, multiple sub classes belong to one super class. We firstly sample the super class c^{super} centroids from a multivariate normal distribution, and then sample the corresponding sub classes around such centroid but smaller variance. Finally, the data points are generated sampled around the sub class centroid with even smaller variance. We treat the coordinates of a data point x_j as features and the sub cluster membership of the data point as the target. During training, the super class information is not available. We construct a two-layer Multi-layer Perception (MLP) with DS-softmax. We investigate the captured hierarchy by examining how the expert learned by DS-softmax is aligned with super class. Two different size of problem are evaluated respectively: 10 super x 10 sub and 100 x 100, shown in Fig. 2(b) and Fig. 2(c). We found DS-Softmax can perfectly capture the hierarchy hidden in dataset.

4.2 Real Task

Language modeling is evaluated by top-1 accuracy and two standard datasets, PennTree Bank (PTB) [10] and WikiText-2 [11], where the output dimensions are 10,000 and 33,278 respectively. In neural machine translation task, we used IWSLT English to Vietnamese dataset [12] (the output vocabulary size is 7,709) and evaluate performance by BLEU score with greedy searching. Regarding image classification, we use CASIA [13], a Chinese handwriting character dataset (around 4,000 output classes). Real device evaluation is done in a machine with two Intel(R) Xeon(R) CPU @ 2.20GHz, and 16G memory. In terms of baseline, SVD-10 uses 10% dimension in preview window and window

†https://github.com/shun1024/ds-softmax

width is 16. D-Softmax is implemented by dividing the embedding size by half for each quarter of classes, order by word frequency. We found DS-Softmax can achieve dramatic inference speedup without loss of performance in all tasks. Moreover, we applied SVD-50 upon of DS-Softmax in WIKI-2, and further boost the speedup to 32.77x with same accuracy.

5 Conclusion

In this paper, we present *doubly sparse: a sparse mixture of sparse experts* for efficient softmax inference. Our method is learning-based and adapts softmax for fast inference. It learns a two-level overlapping class hierarchy. Each expert is learned to be only responsible for a small subset of the output class space. During inference, our method first identifies the responsible expert and then performs a small-scale softmax computation by the expert. Our experiments on several real-world tasks have demonstrated the efficacy of the proposed method.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[2] Patrick H Chen, Si Si, Sanjiv Kumar, Yang Li, and Cho-Jui Hsieh. Learning to screen for fast softmax inference on large vocabulary neural networks. *arXiv preprint arXiv:1810.12406*, 2018.

[3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[4] Kyuhong Shim, Minjae Lee, Iksoo Choi, Yoonho Boo, and Wonyong Sung. Svd-softmax: Fast softmax approximation on large vocabulary neural networks. In *Advances in Neural Information Processing Systems*, pages 5463–5473, 2017.

[5] Minjia Zhang, Xiaodong Liu, Wenhan Wang, Jianfeng Gao, and Yuxiong He. Navigating with graph representations for fast and scalable decoding of neural language models. *arXiv preprint arXiv:1806.04189*, 2018.

[6] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[7] Welin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*, 2015.

[8] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*, 2016.

[9] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[10] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[12] Minh-Thang Luong and Christopher D. Manning. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam, 2015.

[13] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Casia online and offline chinese handwriting databases. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 37–41. IEEE, 2011.