Adaptive Softmax Trees for Many-Class Classification

Rasul Kairgeldin
Dept. CSE, University of California, Merced
rkairgeldin@ucmerced.edu

Magzhan Gabidolla
Dept. CSE, University of California, Merced
mgabidolla@ucmerced.edu

Miguel Á. Carreira-Perpiñán
Dept. CSE, University of California, Merced
mcarreira-perpinan@ucmerced.edu

Classification problems involving thousands to millions of classes occur naturally in many real-world applications. Examples include predicting the next word in a sentence where the vocabulary size can be in the order of hundreds of thousands, and categorizing products for e-commerce systems where the number of distinct labels can be in the order of millions. A linear softmax model, either standalone or as the last layer in a neural network, is widely used for general classification problems. Its inference time, however, is proportional to the number of classes K, as it *needs to evaluate the score for every class no matter the input*, which makes it very slow for large-K classification problems. A natural way to speed it up would be through conditional computation during inference, so that only a small subset of classes needs consideration [2, 5, 4, 3]. Recently, Zharmagambetov et al. [6] proposed a novel *Softmax Tree (ST)* model that strikes a good balance between linear methods and decision trees: the model takes the form of a (hard) decision tree with sparse oblique (linear) decision nodes and small softmaxes at the leaves. However, *a significant drawback is that it assumes a complete tree structure, whose size grows exponentially with depth, and this limits their power in both accuracy and inference time. The key to achieve fast inference time is to decrease the size of the leaf softmaxes by increasing the depth of the leaf path.* Thus, we propose a new model, *Adaptive Softmax Trees (ASTs)*, where we learn jointly the structure and parameters of the tree, by interleaving steps that grow the structure optimally with steps that optimize the parameters of the current structure. This makes it possible to learn ASTs that can grow much deeper but in an irregular way, adapting to the data distribution. As we show experimentally, the resulting ASTs improve considerably the predictive accuracy while reducing the number of parameters and inference time even further.

Softmax tree Let $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \subset \mathbb{R}^D \times \{1, \dots, K\}$ be our training set of size N of D-dimensional input features and K classes. Write the *Softmax Tree* as $\boldsymbol{\tau}(\mathbf{x}; \boldsymbol{\Theta})$, a rooted binary tree with a set of decision (internal) nodes $\mathcal{N}_{\text{dec}}$ and a set of leaf nodes $\mathcal{N}_{\text{leaf}}$. Each decision node $i \in \mathcal{N}_{\text{dec}}$ has a decision function $g_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \{\mathcal{N}_{\text{dec}} \cup \mathcal{N}_{\text{leaf}}\}$ that sends an instance $\mathbf{x}$ to its left or to its right child. We use oblique (linear) decision nodes: "if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$ then $g_i(\mathbf{x}) = \texttt{right}_i$, otherwise $g_i(\mathbf{x}) = \texttt{left}_i$" where the learnable parameters are $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$. Note how the decision function makes hard decisions, unlike in soft trees, where an instance $\mathbf{x}$ is propagated to both children with a positive probability. Each leaf $j \in \mathcal{N}_{\text{leaf}}$ contains a predictive function $\mathbf{f}_j(\mathbf{x}; \boldsymbol{\theta}_j)$: $\mathbb{R}^D \rightarrow \mathbb{S}^K$ that produces the actual output of the tree $\boldsymbol{\tau}(\mathbf{x}; \boldsymbol{\Theta})$ for an instance $\mathbf{x}$, where $\mathbb{S}^K = \{\mathbf{x} \in [0,1]^K \colon \mathbf{1}^T\mathbf{x} = 1\}$. In Softmax Trees, $\mathbf{f}_j(\mathbf{x}; \boldsymbol{\theta}_j)$ takes the form of a small softmax linear classifier: $\mathbf{f}_j(\mathbf{x}; \boldsymbol{\theta}_j) = \sigma(\mathbf{W}_j\mathbf{x} + \mathbf{w}_{j0})$ where $\boldsymbol{\theta}_j = \{\mathbf{W}_j \in \mathbb{R}^{k \times D},\ \mathbf{w}_{j0} \in \mathbb{R}^k\}$ are the learnable parameters, and $\sigma(\cdot)$ is the softmax function. The leaf predictor function $\mathbf{f}_j(\mathbf{x}; \boldsymbol{\theta}_j)$ can output only k nonzero probabilities, with $k \leq K$, for a set of k classes (this set is learned); for all the other $K - k$ classes $\mathbf{f}_j(\mathbf{x}; \boldsymbol{\theta}_j)$ assigns exactly zero probability. For problems with a large number of classes we want $k \ll K$ to allow for fast inference. The predictive function of the whole Softmax Tree $\boldsymbol{\tau}(\mathbf{x}; \boldsymbol{\Theta})$ then works by routing an instance $\mathbf{x}$ to exactly one leaf through a root-leaf path of (oblique) decision nodes and applying that leaf's small softmax predictor function. Overall, a ST can be seen as a hierarchical collection of local softmax classifiers each operating on a small subset of classes.

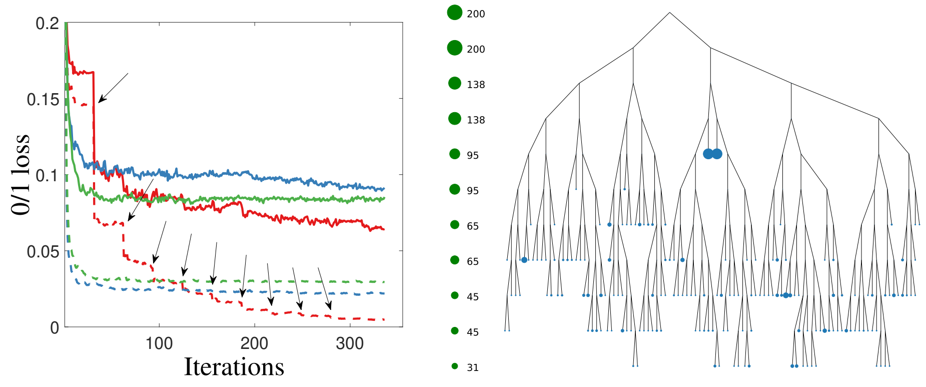

Figure 1: (Left) 0/1 loss of the final AST model for training (dashed line) and test (solid line), compared with the complete Softmax Tree. The arrows point to where expansions of the AST happened. The line colors indicate the performance of the ST (blue), ST(AST) (green) and AST (red). This shows that the adaptive growth gradually enhances the performance of the model on both training and test tests (red solid and dashed lines). On the other hand, a ST initialized randomly (blue line) or on the final structure of AST (green line) is unable to improve after a certain number of iterations. (Right) AST for the Wiki-Small subs. dataset. Size of the blue nodes (on the tree) shows the actual number of classes in the leaves after pruning. Green (left column) shows theoretical max. values at each aligned depth.

Adaptive Softmax Tree We first train a shallow (e.g. depth $\Delta = 2$) complete Softmax Tree $\boldsymbol{\tau}(\cdot;\boldsymbol{\Theta})$ with relatively large k_0-class softmaxes in the leaves. The number of classes k_0 is set such that the total number of predictable classes by the model is at least the total number of classes K in the dataset: $k_0 2^{\Delta} \geq K$. We then attempt to replace each leaf $j \in \mathcal{N}_{\text{leaf}}$ softmax predictor function $\mathbf{f}_j(\cdot;\boldsymbol{\theta}_j)$ by yet another shallow Softmax Tree $\hat{\boldsymbol{\tau}}_j(\cdot;\hat{\boldsymbol{\Theta}}_j)$ of depth $\hat{\Delta} = 1$ or 2, whose leaves contain smaller $\hat{k}_j$-class softmaxes, $\hat{k}_j < k_0$. To control by how much these large softmaxes are reduced we use the following simple heuristic: $\hat{k}_j = \alpha\, k_0$, where $\alpha \in (0,1)$ is the *softmax contraction coefficient hyperparameter*. We obtain this small tree $\hat{\boldsymbol{\tau}}_j(\cdot;\hat{\boldsymbol{\Theta}}_j)$ by fitting it using the TAO [1] algorithm on the training instances that reach the leaf j, i.e., on the reduced set $\mathcal{R}_j$. This step can be considered as a recursive application of the Softmax Tree method with the goal of replacing large, flat softmaxes with faster "softmax subtrees". But instead of directly substituting the leaf softmax $\mathbf{f}_j(\cdot;\boldsymbol{\theta}_j)$ with the tree $\hat{\boldsymbol{\tau}}_j(\cdot;\hat{\boldsymbol{\Theta}}_j)$, we first ensure that the accuracy of $\hat{\boldsymbol{\tau}}_j(\cdot;\hat{\boldsymbol{\Theta}}_j)$ is at least as good as the original softmax $\mathbf{f}_j(\cdot;\boldsymbol{\theta}_j)$ or within a reasonable *tolerance ratio hyperparameter* $\rho > 1$. If this is not the case, the leaf predictor function $\mathbf{f}_j(\cdot;\boldsymbol{\theta}_j)$ remains unchanged. Otherwise, the substitution happens, and this results in the structure change of the original tree model $\boldsymbol{\tau}(\cdot;\boldsymbol{\Theta})$ where it is expanded through the leaf j (the *expansion step*). In this way, after attempting to expand all the leaves $j \in \mathcal{N}_{\text{leaf}}$, and assuming some or all of them are expanded, we obtain a deeper, irregular Softmax Tree $\boldsymbol{\tau}_{\text{exp}}(\cdot;\boldsymbol{\Theta}_{\text{exp}})$ with smaller leaf softmaxes which has comparable or better training accuracy and faster inference. Now, importantly, we retrain the whole model $\boldsymbol{\tau}_{\text{exp}}(\cdot;\boldsymbol{\Theta}_{\text{exp}})$ globally using TAO (the *regular step*), which will further improve the model accuracy and possibly sparsify nodes. We repeat these local expansion and global optimization steps until the model converges or some predetermined stopping criterion is reached. Table shows results on the text classification dataset WIKI-Small. We report the test error, depth Δ of the tree, and the average inference time per test sample in milliseconds.

Method	$E_{\text{test}}(\%)$	Δ	inf.(ms)	Train time
RecallTree	92.64	15	0.97	53m
one-vs-all	85.71	0	10.70	> 7d
MACH	84.80	–	252.64	1445m
ST(k = 200)	84.70	8	0.18	≈1000m
(π, κ)-DS	78.50	–	10.33	–
ST(k = 150)	77.26	8	0.57	≈1000m
AST(α=0.6, ρ=1.0)	77.30	12	**0.03**	≈2000m
AST(α=0.60, ρ=1.1)	**76.21**	12	0.04	≈2000m

Acknowledgments: Work funded in part by NSF award IIS–2007147.

References

[1] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.

[2] J. Goodman. Classes for fast maximum entropy training. In *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'01)*, pages 561–564, Salt Lake City, Utah, USA, May 7–11 2001.

[3] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. arXiv:1309.4168 [cs.CL], Sept. 13 2013.

[4] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In D. Koller, Y. Bengio, D. Schuurmans, L. Bottou, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1081–1088. MIT Press, Cambridge, MA, 2009.

[5] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In R. G. Cowell and Z. Ghahramani, editors, *Proc. of the 10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 246–252, Barbados, Jan. 6–8 2005.

[6] A. Zharmagambetov, M. Gabidolla, and M. Á. Carreira-Perpiñán. Softmax tree: An accurate, fast classifier when the number of classes is large. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP 2021)*, pages 10730–10745, Online, 2021.