Softmax Tree: An Accurate, Fast Classifier When the Number of Classes Is Large

Arman Zharmagambetov, **Magzhan Gabidolla** and **Miguel Á. Carreira-Perpiñán**
Dept. Computer Science & Engineering, UC Merced

EMNLP 2021
Online & in the Dominican Republic
Conference on Empirical Methods in Natural Language Processing

1 Abstract

Classification problems having thousands or more classes naturally occur in NLP, for example language models or document classification. A softmax or one-vs-all classifier naturally handles many classes, but it is very slow at inference time, because every class score must be calculated to find the top class. We propose the "softmax tree", consisting of a binary tree having sparse hyperplanes at the decision nodes (which make hard, not soft, decisions) and small softmax classifiers at the leaves. This is much faster at inference because the input instance follows a single path to a leaf (whose length is logarithmic on the number of leaves) and the softmax classifier at each leaf operates on a small subset of the classes. Although learning accurate tree-based models has proven difficult in the past, we are able to overcome this by using a variation of a recent algorithm, tree alternating optimization (TAO). Compared to a softmax and other classifiers, the resulting softmax trees are both more accurate in prediction and faster in inference, as shown in NLP problems having from one thousand to one hundred thousand classes.

Work supported by NSF award IIS–2007147

2 Softmax Tree: motivation

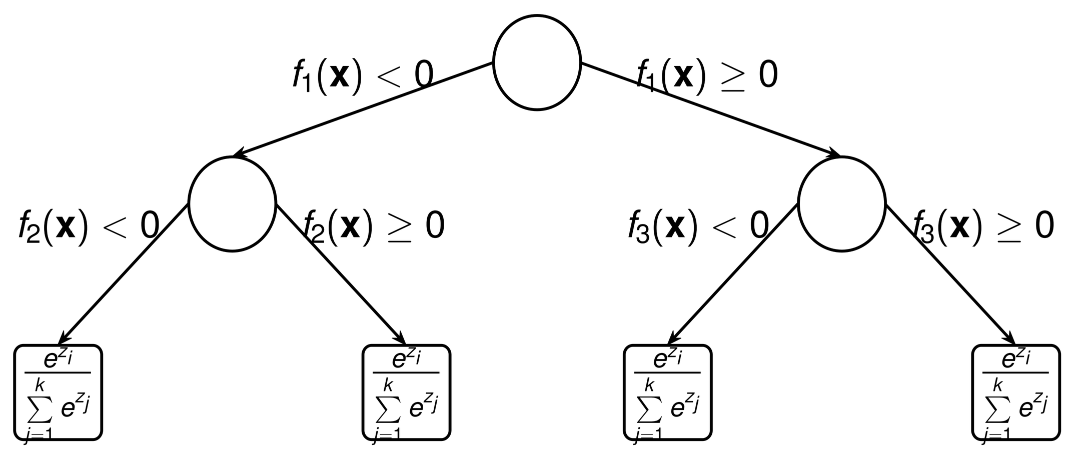
A large number of classes ($= K$) are quite common in NLP problems:

- Language modeling: ≈171k words in the Oxford English Dictionary → 171k classes and grows as we include all forms of a word, names, acronyms, etc.
- Website categorization given its content: ODP contains >1M website categories. So, automatically tagging a web page will require identifying a subset of categories relevant to it.

Traditional linear models (softmax or one-vs-all) do not scale well for these problems:

- During inference one must compute the score or probability of (nearly) all classes, conditional on the input instance, in order to determine the (top-n) predicted class.
- Obvious way to speed-up – use decision trees, e.g. CART: typically perform poorly.

Proposed model: Softmax Tree



- Sparse oblique decision nodes: $f_i(x) = \mathbf{w}_i^T \mathbf{x} + b_i$ in the above figure.
- Sparse linear softmax leaves where each leaf focuses only on $k \ll K$ classes (K total number of classes).

3 Softmax Tree: learning

- The proposed model provides speedup of $\mathcal{O}(\frac{K}{\Delta+k}) \approx \mathcal{O}(\frac{K}{k})$ compared to one-vs-all while still being accurate!
- However, STs are hard to train: nonconvex, nondifferentiable, discontinuous.
- We use Tree Alternating Optimization (TAO): non-greedy, generally finds better optima, has shown a huge success in training various tree-based models.

Assuming a tree structure $\mathbf{T}$ is given (say, binary complete of depth Δ), consider the following regularized objective:

$$E(\Theta) = \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \Theta)) + \alpha \sum_{i \in \mathcal{N}} \|\boldsymbol{\theta}_i\|_1$$

given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$. $\Theta = \{\boldsymbol{\theta}_i\}_{i\in\mathcal{N}}$ is a set of parameters of all tree nodes. The loss function $L(\mathbf{y}, \mathbf{z})$ is cross-entropy (TAO was originally proposed for misclassification loss). TAO optimizes eq. (1) based on two theorems. First, eq. (1) **separates over any subset of non-descendant nodes**; this follows from the fact that the tree makes hard decisions. All such nodes may be optimized in parallel. Second, optimizing over the parameters of a single node i simplifies to a well-defined **reduced problem** over the instances that currently reach node i (the **reduced set** $\mathcal{R}_i \subset \{1, \dots, N\}$). The form of the reduced problem depends on the type of node, and can be summarized in the pseudocode:

Result trained tree $\mathbf{T}(\cdot;\Theta)$
input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, initial tree $\mathbf{T}(\cdot;\Theta)$ of depth Δ
repeat
  for depth $d = \Delta$ downto 0
    parfor $i \in$ nodes at depth d
      if i is a leaf **then**
        $\overline{\mathcal{R}}_i \leftarrow$ instances of the most populous k classes in $\mathcal{R}_i$
        $\boldsymbol{\theta}_i \leftarrow$ fit a linear classifier (softmax) on $\overline{\mathcal{R}}_i$
      else
        generate pseudolabels $\overline{y}_n$ for each point $n \in \mathcal{R}_i$
        (done by evaluating loss from left/right subtree and picking the best)
        $\boldsymbol{\theta}_i \leftarrow$ fit a weighted binary classifier
  update $\mathcal{R}_i$ for each node
until max number of iterations
postprocessing: remove dead or pure subtrees

Practicalities:

- Dealing with zero probabilities – problematic during decision node optimization: $P_i(\mathbf{y}|\mathbf{x}) = 0 \rightarrow$ infinity loss (quite possible given $k \ll K$). Possible ways to resolve: replace loss=∞ by loss=β (e.g. 100, 10^7) or use 0/1 loss to compute pseudolabels.
- Obtaining an initial tree: a) complete tree of depth Δ with random parameters (default option); b) clustering-based initialization.

4 Experiments: Document Classification

	Method	top-1	Δ	inf.(ms)	size(GB)
WIKI–Small	RecallTree	92.64	15	0.97	0.8
	one-vs-all	85.71	0	10.70	53.5
	MACH	84.80	–	252.64	1.3
	(π,κ)-DS	78.02	–	10.33	0.01
	ST($k=100$)	77.26	7	0.33	0.03
	ST($k=300$)	76.86	7	0.49	0.04
	ST($k=150$)	75.65	8	0.52	0.05
ODP	RecallTree	94.64	6	8.42	3.4
	LOMTree	(93.46)	(17)	(0.26)	–
	one-vs-all	89.22	0	1317.58	155.7
	(π,κ)-DS	86.31	–	36.41	1.0
	MACH	84.55	–	684.04	1.2
	ST($k=300$)	81.84	9	9.87	0.1

Table 1: Results on text classification datasets. We report the top-1 test error, maximum depth (Δ), avg. inference time per test sample (in ms) and uncompressed model sizes (in GB). ST($k=x$) indicates our method which uses at most k classes at each leaf. The results in brackets are taken from the corresponding papers.

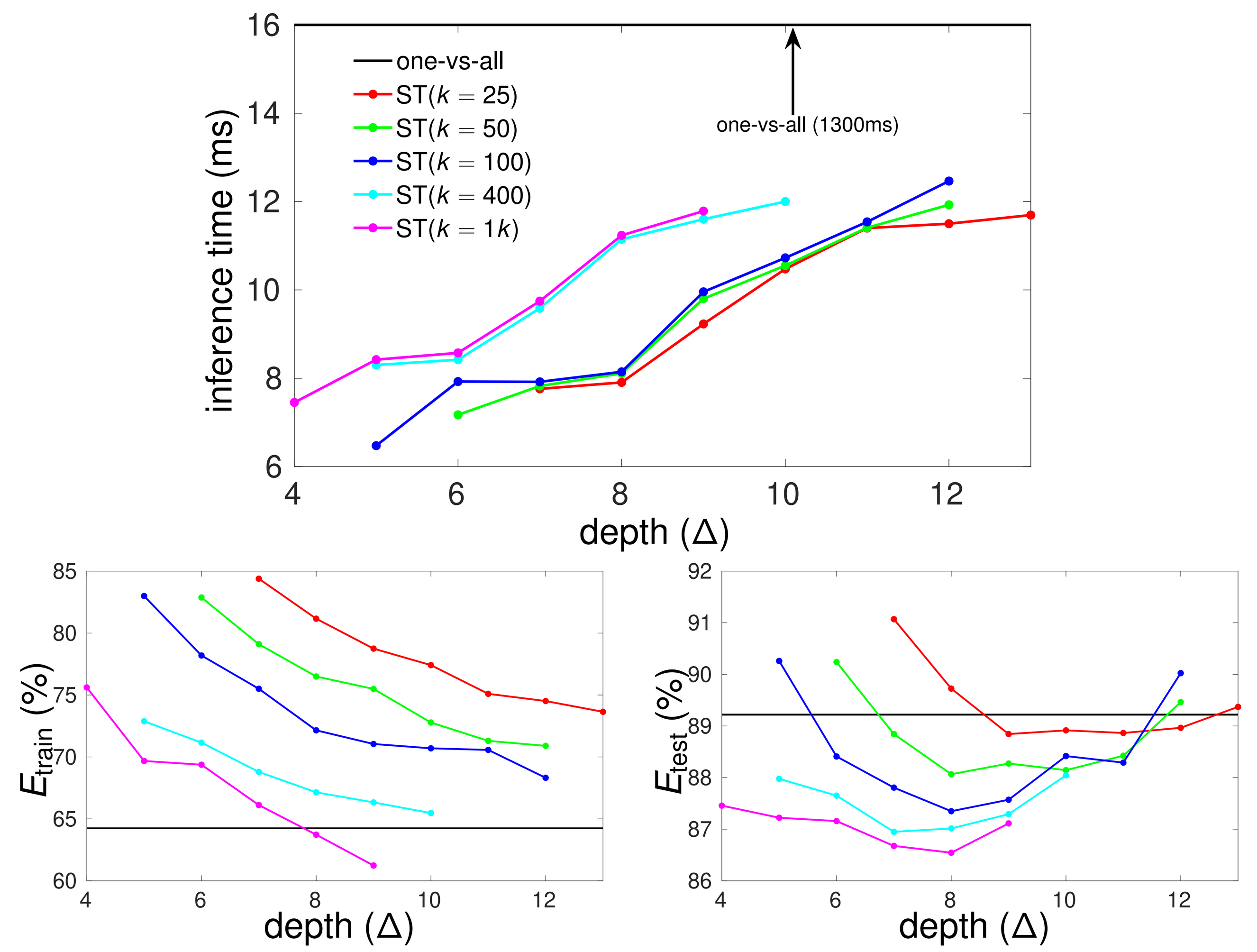


Figure 1: Avg. inf. time tradeoff (top figure) and Top-1 errors (bottom) of the ST for various settings of Δ and k on the ODP dataset.

5 Experiments: Language Modeling

Method	top-1/top-5	PPL(% covered)	Δ	inf.(ms)
HSM-approx	92.2 / 86.5	575 (100%)	18	0.184
HSM	91.1 / 81.1	575 (100%)	18	0.421
one-vs-all	87.5 / 80.2	220 (100%)	0	0.402
softmax	86.9 / 79.6	217 (100%)	0	0.467
ST(k=50)	86.5 / 72.5	17 (44%)	8	0.058
ST(k=100)	86.5 / 71.5	27 (51%)	7	0.058
ST(k=200)	86.4 / 70.6	45 (58%)	6	0.053
ST(k=400)	86.4 / 69.7	71 (67%)	5	0.064
ST(k=800)	86.4 / 68.4	117 (77%)	4	0.066
ST*(k=800)	86.4 / 68.4	427 (100%)	4	0.066

Table 2: Like Table 1 but on PTB–language modeling task. We also report the test Perplexity (with percentage of the covered points) and top-5 error. "*" indicates that smoothing was applied to replace 0 probabilities with some small epsilon and renormalize the output.

Method	top-1/top-5	PPL(% covered)	Δ	inf.(ms)
HSM-appox	78.3 / 64.1	184 (100%)	18	0.097
HSM	77.7 / 63.1	184 (100%)	18	0.372
softmax	74.3 / 54.8	96 (100%)	0	0.346
ST(k=50)	75.2 / 57.3	9 (59%)	8	0.046
ST(k=100)	75.0 / 56.8	13 (64%)	7	0.045
ST(k=200)	74.9 / 56.2	18 (70%)	6	0.067
ST(k=400)	74.7 / 55.9	24 (76%)	5	0.066
ST(k=800)	74.5 / 55.5	33 (81%)	4	0.069
ST*(k=800)	74.5 / 55.5	145 (100%)	4	0.069

Table 3: Like Table 3, but models were trained on the output of the recurrent neural net (LSTM).