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

A. Zharmagambetov, M. Gabidolla, M. Á. Carreira-Perpiñán

Dept. of Computer Science & Engineering
University of California, Merced

EMNLP 2021

Problem motivation

- The goal in extreme (or extra) classification is to train classifiers on datasets with large number of label set (i.e., large number of classes).
- **Some examples:**
  - 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. Open Directory Project contains >1M website categories. So, automatically tagging a website will require identifying a subset of categories relevant to it.
  - Recommending a shopping item in e-commerce where each of the selling item (e.g. on Amazon) is a separate class label.

Problem motivation

- The goal in extreme (or extra) classification is to train classifiers on datasets with large number of label set (i.e., large number of classes).
- **Some examples:**
  - 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. Open Directory Project contains >1M website categories. So, automatically tagging a website will require identifying a subset of categories relevant to it.
  - Recommending a shopping item in e-commerce where each of the selling item (e.g. on Amazon) is a separate class label.
- **Research question:** how to efficiently predict one (or several) of K classes in sub-linear time and how to efficiently train such models?

Why sub-linear time?

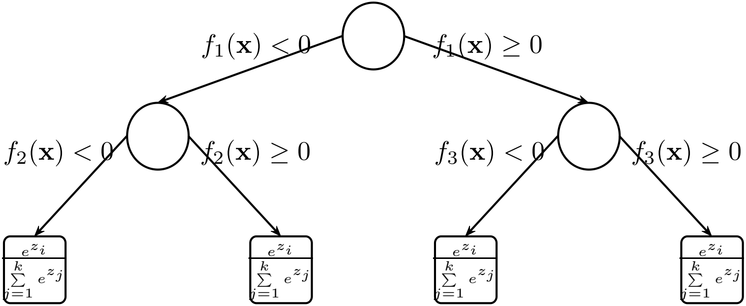
- Family of functions with decreasing prediction time:

$\mathcal{O}(n)$ —————————— $\mathcal{O}(\log n)$

softmax, one-vs-all, ... | CART, LOMTree, ...

- Obvious way to speed-up – use hierarchical models, e.g. CART [1], LOMTree [2], Nested dichotomies [4], etc.
- Other approaches have been studied as well: using hashing techniques [6], class/data subsampling [5], etc.

Proposed model: Softmax Tree (ST)



- 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).

Proposed model: Softmax Tree (ST)

- Family of functions with decreasing prediction time:

Proposed model: Softmax Tree (ST)

- Family of functions with decreasing prediction time:

- This provides speedup of $\mathcal{O}(\frac{K}{\Delta+k}) \approx \mathcal{O}(\frac{K}{k})$ compared to one-vs-all while still being accurate!
- Similar model has been proposed in Daumé III et al. 2017 [3], Sun et al. 2019 [7], etc. However, no sparsity and different training methods.

Model optimization

- STs are hard to train: nonconvex, nondifferentiable, discontinuous.
- Traditional tree learning algorithms are greedy: CART, C4.5, etc.

Model optimization

- STs are hard to train: nonconvex, nondifferentiable, discontinuous.
- Traditional tree learning algorithms are greedy: CART, C4.5, etc.
- We use Tree Alternating Optimization (TAO): non-greedy, generally finds better optima, has shown a huge success in training various tree-based models [8, 9].

Model optimization

- STs are hard to train: nonconvex, nondifferentiable, discontinuous.
- Traditional tree learning algorithms are greedy: CART, C4.5, etc.
- We use Tree Alternating Optimization (TAO): non-greedy, generally finds better optima, has shown a huge success in training various tree-based models [8, 9].

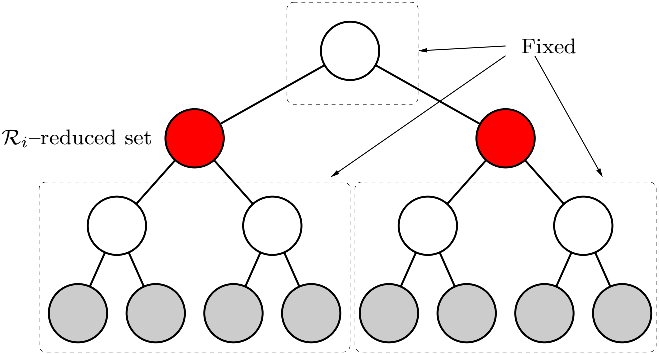
  Assuming a tree structure $\mathbf{T}$ is given (say, binary complete of depth Δ), consider the following regularized objective:

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

  given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ 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).

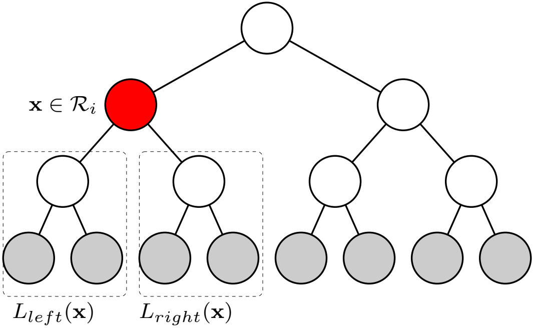
Alternating optimization and separability condition

- Any set of non-descendant nodes of a tree can be optimized independently:



Reduced problem over decision node

- Evaluate loss induced by left/right subtrees;
- Generate pseudolabel for each instance in reducde set $\mathcal{R}_i$;
- Solve weighted binary classification problem (linear):



Reduced problem over a leaf

- Actual model prediction is given by leaves;

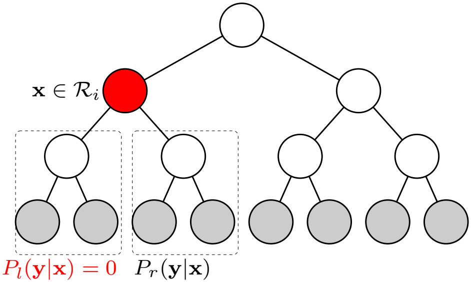
$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} L(\mathbf{y}_n, \mathbf{g}_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha \left\| \boldsymbol{\theta} \right\|_i)$$

  where $\mathbf{g}_i$ is a predictor function at each leaf: $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^k$ and it is restricted to have k classes.
- Solution: first estimate the k classes (out of K possible classes) as the k most populous classes in $\mathcal{R}_i$. Then we train the softmax, which is a convex problem.

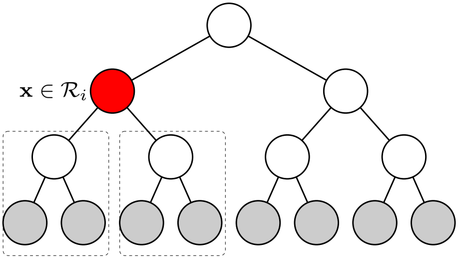
Pseudocode

input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$;
initial tree $\mathbf{T}(\cdot;\boldsymbol{\Theta})$ of depth Δ with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}$;
$\mathcal{N}_0, \dots, \mathcal{N}_\Delta \leftarrow$ nodes at depth $0, \dots, \Delta$, respectively;
generate $\mathcal{R}_i$ (instances that reach node i) using an initial tree;
repeat
  for $d = \Delta$ **down to** 0
    parfor $i \in \mathcal{N}_d$
      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 on $\overline{\mathcal{R}}_i$
      else
        generate pseudolabels $\overline{y}_n$ for each point $n \in \mathcal{R}_i$
        $\boldsymbol{\theta}_i \leftarrow$ fit a weighted binary classifier on $\mathcal{R}_i$
  update $\mathcal{R}_i$ for each node
until stop
return $\mathbf{T}$

Practicalities: dealing with zero probabilities



Practicalities: dealing with zero probabilities



- This is quite possible given $k \ll K$. But $\log P_l(\mathbf{y}|\mathbf{x}) = \log 0 = -\infty$.
- Possible ways to resolve:
  - Remove from the reduced problem → poor performance.
  - Replace loss=∞ by loss=β (e.g. 100, 10^7) → performs well but requires tuning β.
  - Use 0/1 loss to compute pseudolabels → slightly worse than previous option but requires no hyperparameter. **Default choice**.

Practicalities: obtaining an initial tree

- Default option:
  - Complete binary tree of depth Δ (s.t. $k \times L \geq K$, where L is the number of leaves) with random parameters at each node;
  - Generate reduced set $\mathcal{R}$ based on random parameters $\rightarrow$ run TAO;
  - Simple to implement and performs well in practice.

Practicalities: obtaining an initial tree

- Default option:
  - Complete binary tree of depth Δ (s.t. $k \times L \geq K$, where L is the number of leaves) with random parameters at each node;
  - Generate reduced set $\mathcal{R}$ based on random parameters $\rightarrow$ run TAO;
  - Simple to implement and performs well in practice.
- Better option: clustering based initialization.

Experiments: document classification

	Method	top-1	Δ	inf.(ms)	size(GB)
wiki–small(1M,380k,37k)	RecallTree [3]	92.64	15	0.97	0.8
	one-vs-all	85.71	0	10.70	53.5
	MACH [6]	84.80	–	252.64	1.3
	(π, κ)-DS [5]	78.02	–	10.33	0.01
	ST($k = 100$)	77.26	7	0.33	0.03
	ST($k = 150$)	76.33	8	0.57	0.05
	ST^+($k = 150$)	75.65	8	0.52	0.05
ODP(1.6M,423k,105k)	RecallTree [3]	94.64	6	8.42	3.4
	LOMTree [2]	(93.46)	(17)	(0.26)	–
	one-vs-all	89.22	0	1317.58	155.7
	(π, κ)-DS [5]	86.31	–	36.41	1.0
	MACH [6]	84.55	–	684.04	1.2
	ST($k = 300$)	83.78	9	9.59	0.1
	ST^+($k = 300$)	81.84	9	9.87	0.1

$+$ means ∞ loss was replaced with β.

Experiments: language modeling

- Results on Penn Treebank:

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

* means that smoothing was applied to replace 0 probabilities with some small epsilon and renormalize the output.

Conclusion

- We have proposed Softmax Tree (ST) – a sparse oblique decision tree with small linear softmax classifier at each leaf.
- It uses modified TAO algorithm combined with special initialization.
- STs strike a balance between having a single softmax (or one-vs-all) classifier and a decision tree with a single class at each leaf.
- The best performance is achieved by tuning the depth of the tree and the number of classes per leaf softmax.
- It results in classifiers that are both more accurate and much faster than a regular softmax or other hierarchical softmax approaches in many-class problems.
- Future works: forests of STs, growing the tree structure adaptively, etc.
- Work supported by NSF award IIS–2007147

References

[1] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

[2] A. E. Choromanska and J. Langford. Logarithmic time online multiclass prediction. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 28. MIT Press, Cambridge, MA, 2015.

[3] H. Daumé III, N. Karampatziakis, J. Langford, and P. Mineiro. Logarithmic time one-against-some. In D. Precup and Y. W. Teh, editors, *Proc. of the 34th Int. Conf. Machine Learning (ICML 2017)*, pages 923–932, Sydney, Australia, Aug. 6–11 2017.

[4] E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *Proc. of the 21st Int. Conf. Machine Learning (ICML'04)*, pages 305–312, Banff, Canada, July 4–8 2004.

[5] B. Joshi, M. R. Amini, I. Partalas, F. Iutzeler, and Y. Maximov. Aggressive sampling for multi-class to binary reduction with applications to text classification. In I. Guyon, U. v. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 30. MIT Press, Cambridge, MA, 2017.

[6] T. K. R. Medini, Q. Huang, Y. Wang, V. Mohan, and A. Shrivastava. Extreme classification in log memory using count-min sketch: A case study of Amazon search with 50M products. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 32, pages 13265–13275. MIT Press, Cambridge, MA, 2019.

[7] W. Sun, A. Beygelzimer, H. Daum'e III, J. Langford, and P. Mineiro. Contextual memory trees. In K. Chaudhuri and R. Salakhutdinov, editors, *Proc. of the 36th Int. Conf. Machine Learning (ICML 2019)*, pages 6026–6035, Long Beach, CA, June 9–15 2019.

[8] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Smaller, more accurate regression forests using tree alternating optimization. In H. Daumé III and A. Singh, editors, *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pages 11398–11408, Online, July 13–18 2020.

[9] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Learning a tree of neural nets. In *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, pages 3140–3144, Toronto, Canada, June 6–11 2021.