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: \( \approx 171k \) words in the Oxford English Dictionary \( \rightarrow \) 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: \( \approx 171k \) words in the Oxford English Dictionary \( \rightarrow \) 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:
  
  \[ O(n) \quad O(\log n) \]
  
  softmax  
  one-vs-all  
  \[ \cdots \]  
  CART  
  LOMTree  
  \[ \cdots \]

- 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:

\[ O(n) \quad \text{softmax} \quad \text{one-vs-all} \quad O(\log n) \]

\[ \text{CART} \quad \text{LOMTree} \]
Proposed model: Softmax Tree (ST)

- Family of functions with decreasing prediction time:

\[ \mathcal{O}(n) \quad \mathcal{O}(\log n) \]

- softmax
- one-vs-all
- ST
- CART
- LOMTree

- This provides speedup of \( \mathcal{O}\left(\frac{K}{\Delta+k}\right) \approx \mathcal{O}\left(\frac{K}{k}\right) \) compared to one-vs-all while still being accurate!

- Related models have 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 $T$ is given (say, binary complete of depth $\Delta$), consider the following regularized objective:

$$E(\Theta) = \sum_{n=1}^{N} L(y_n, T(x_n; \Theta)) + \alpha \sum_{i \in \mathcal{N}} \|\theta_i\|_1$$

given a training set $\{(x_n, y_n)\}_{n=1}^{N}$. $\Theta = \{\theta_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes. The loss function $L(y, z)$ 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 reduced set $\mathcal{R}_i$;
- Solve weighted binary classification problem (linear):

$$x \in \mathcal{R}_i$$

$$L_{left}(x)$$

$$L_{right}(x)$$
Reduced problem over a leaf

- Actual model prediction is given by leaves:

\[
\min_{\theta_i} E_i(\theta_i) = \sum_{n \in \mathcal{R}_i} L(y_n, g_i(x_n; \theta_i)) + \alpha \|\theta\|_i
\]

where \( g_i \) is a predictor function at each leaf: \( g_i(x; \theta_i): \mathbb{R}^D \rightarrow \mathbb{R}^k \) 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 \( \{(x_n, y_n)\}_{n=1}^N \);
initial tree \( T(\cdot; \Theta) \) of depth \( \Delta \) with parameters \( \Theta = \{\theta_i\} \);
\( N_0, \ldots, N_\Delta \leftarrow \) nodes at depth 0, \ldots, \( \Delta \), respectively;
generate \( R_i \) (instances that reach node \( i \)) using an initial tree;
repeat
   for \( d = \Delta \) down to 0
      parfor \( i \in N_d \)
      if \( i \) is a leaf then
         \( \bar{R}_i \leftarrow \) instances of the most populous \( k \) classes in \( R_i \)
         \( \theta_i \leftarrow \) fit a linear classifier on \( \bar{R}_i \)
      else
         generate pseudolabels \( \bar{y}_n \) for each point \( n \in R_i \)
         \( \theta_i \leftarrow \) fit a weighted binary classifier on \( R_i \)
      update \( R_i \) for each node
   until stop
return \( T \)
Practicalities: dealing with zero probabilities

\[ x \in \mathcal{R}_i \]

\[ P_i(y|x) = 0 \]

\[ P_r(y|x) \]
Practicalities: dealing with zero probabilities

This is quite possible given $k \ll K$. But $\log P_l(y|x) = \log 0 = -\infty$.

Possible ways to resolve:

- Remove from the reduced problem $\rightarrow$ poor performance.
- Replace loss=$\infty$ by loss=$\beta$ (e.g. 100, $10^7$) $\rightarrow$ performs well but requires tuning $\beta$.
- Use 0/1 loss to compute pseudolabels $\rightarrow$ slightly worse than previous option but requires no hyperparameter. **Default choice.**
Practicalities: obtaining an initial tree

Default option:

- Complete binary tree of depth $\Delta$ (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 $\Delta$ (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

<table>
<thead>
<tr>
<th>Method</th>
<th>top-1</th>
<th>Δ</th>
<th>inf.(ms)</th>
<th>size(GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecallTree [3]</td>
<td>92.64</td>
<td>15</td>
<td>0.97</td>
<td>0.8</td>
</tr>
<tr>
<td>one-vs-all</td>
<td>85.71</td>
<td>0</td>
<td>10.70</td>
<td>53.5</td>
</tr>
<tr>
<td>MACH [6]</td>
<td>84.80</td>
<td>–</td>
<td>252.64</td>
<td>1.3</td>
</tr>
<tr>
<td>$(\pi, \kappa)$-DS [5]</td>
<td>78.02</td>
<td>–</td>
<td>10.33</td>
<td>0.01</td>
</tr>
<tr>
<td>ST($k = 100$)</td>
<td>77.26</td>
<td>7</td>
<td>0.33</td>
<td>0.03</td>
</tr>
<tr>
<td>ST($k = 150$)</td>
<td>76.33</td>
<td>8</td>
<td>0.57</td>
<td>0.05</td>
</tr>
<tr>
<td>ST$^+(k = 150)$</td>
<td>75.65</td>
<td>8</td>
<td>0.52</td>
<td>0.05</td>
</tr>
<tr>
<td>RecallTree [3]</td>
<td>94.64</td>
<td>6</td>
<td>8.42</td>
<td>3.4</td>
</tr>
<tr>
<td>LOMTree [2]</td>
<td>(93.46)</td>
<td>(17)</td>
<td>(0.26)</td>
<td>–</td>
</tr>
<tr>
<td>one-vs-all</td>
<td>89.22</td>
<td>0</td>
<td>1317.58</td>
<td>155.7</td>
</tr>
<tr>
<td>$(\pi, \kappa)$-DS [5]</td>
<td>86.31</td>
<td>–</td>
<td>36.41</td>
<td>1.0</td>
</tr>
<tr>
<td>MACH [6]</td>
<td>84.55</td>
<td>–</td>
<td>684.04</td>
<td>1.2</td>
</tr>
<tr>
<td>ST($k = 300$)</td>
<td>83.78</td>
<td>9</td>
<td>9.59</td>
<td>0.1</td>
</tr>
<tr>
<td>ST$^+(k = 300)$</td>
<td>81.84</td>
<td>9</td>
<td>9.87</td>
<td>0.1</td>
</tr>
</tbody>
</table>

+ means $\infty$ loss was replaced with $\beta$. 


### Experiments: language modeling

#### Results on Penn Treebank:

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

* 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


