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 typically handles such problems, but this is very slow at inference time, because every class score must be calculated to find the top class. Although learning accurate tree-based models that exploit the structure at the decision nodes (which make hard, not soft, decisions) and small softmax classifiers at the leaves is proven difficult in the past, we are able to overcome this by using a variety of recent algorithms, tree alternating optimization (TAO). Compared to a softmax and other classifiers, the resulting softmax trees are both more accurate 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

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

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

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 typically handles such problems, but this is very slow at inference time, because every class score must be calculated to find the top class. Although learning accurate tree-based models that exploit the structure at the decision nodes (which make hard, not soft, decisions) and small softmax classifiers at the leaves is proven difficult in the past, we are able to overcome this by using a variety of recent algorithms, tree alternating optimization (TAO). Compared to a softmax and other classifiers, the resulting softmax trees are both more accurate 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: ≤17k words in the English Oxford Dictionary → 171k classes and grows as we include all forms of a word, names, acronyms, etc.

Website categorization given its content: ODP contains over 171k classes and grows as we include all forms of a word, names, acronyms, etc.

"website categorization given its content: ODP contains over 171k classes and grows as we include all forms of a word, names, acronyms, etc.

A common way to speed up → use decision trees, e.g., CART: typically perform poorly.

Proposed model: Softmax Tree

- Sparse oblique decision nodes: $x_i$ → $w_i x_i b_i$ in the above figure.
- Sparse linear softmax leaves where each leaf focuses only on $k ≤ K$ classes (K total number of classes).

3 Softmax Tree: learning

- The proposed model provides speedup of $O(\frac{K}{n})$ 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 proven successful in various training tree-based models.

- Assuming a tree structure $T$ is given (say, binary complete of depth $\Delta$), consider the following regularized objective:

$$E(\theta) = \sum_{i,j} \ell(y_{ij}, T(x_i \theta)) + \frac{1}{2} \sum_{i,j} \left| \theta_{ij} \right|^2$$

given a training set $(x_i, y_{ij})$ and $(\theta_{ij})$ is a set of parameters at all tree nodes.

The loss function $\ell(y_{ij}, \theta)$ is cross-entropy.

TAO optimizes a tree $T$ based on two heuristics. First, eq. (1) separates over any subset of non-descendant nodes; this follows from the fact that the tree makes hard decisions. At all such nodes may be optimized in parallel. Second, optimizing over the parameters of a single node $i_j$ simplifies to a well-defined reduced problem over the instances that currently reach node $j$ (the reduced set $\{x_i | \pi_j(i) \}$). The form of the reduced problem depends on the type of node, and can be summarized in the pseudocode:

Result trained tree $T(\cdot, \theta)$

Input training set $\{(x_i, y_{ij})\}$, initial tree $T(\cdot, \theta)$, depth $\Delta$

repeat

for depth $d ≤ \Delta$ downto 0

for $i$ from nodes at depth $d$

if $i$ is a leaf then

$N_i$ ← instances of the most populous $k$ classes in $T_i$

$\theta_i$ ← fit a linear classifier (softmax) on $T_i$ and

else

generate pseudolabels $\pi_j(i)$ for each node $j \in N_i$,

done by evaluating loss from left/right subtree and picking the best

$\theta_i$ ← fit a weighted binary classifier

update $N_i$ for each node $j$

until max number of iterations

postprocessing: remove dead or pure subtrees

Practicalities:

- Dealing with zero probabilities → problematic during decision node optimization: $P(x_i | y_{ij}) \rightarrow 0$. Interfering loss (quite possible given $k < K$). Possible ways to resolve: select $\alpha$-0 for $\gamma_{ij}$ or $\alpha_j = (\text{e}^{0.1 K}, 0.1)$ and $\pi_j(i)$'s, compute pseudolabels.

- Obtaining an initial tree: a complete tree of depth $\Delta$ with random parameters (default option); b) clustering-based initialization.

4 Experiments: Document Classification

Method top-1 top-5 PPL($\%$ covered) $\Delta$ inf.(ms) size(GB)

Table 1: Results on text classification datasets. We report the top-1 test error, maximum depth ($\Delta$), 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.

5 Experiments: Language Modeling

Method top-top-5 PPL($\%$ covered) $\Delta$ inf.(ms)

Table 3: Like Table 2 but on PTB-language modeling task. We also report the test Perplexity (with percentage of the correct points) and top-5 error. "c" indicates that smoothing was applied to replace 0 probabilities with some small epsilon and renormalize the output.