Sparse Oblique Decision Trees: A Tool to Interpret Natural Language Processing Datasets

Suryabhan Singh Hada Miguel Á. Carreira-Perpiñán
{shada,mcarreira-perpinan}@ucmerced.edu

Dept. of Computer Science and Engineering,
University of California, Merced
http://eecs.ucmerced.edu

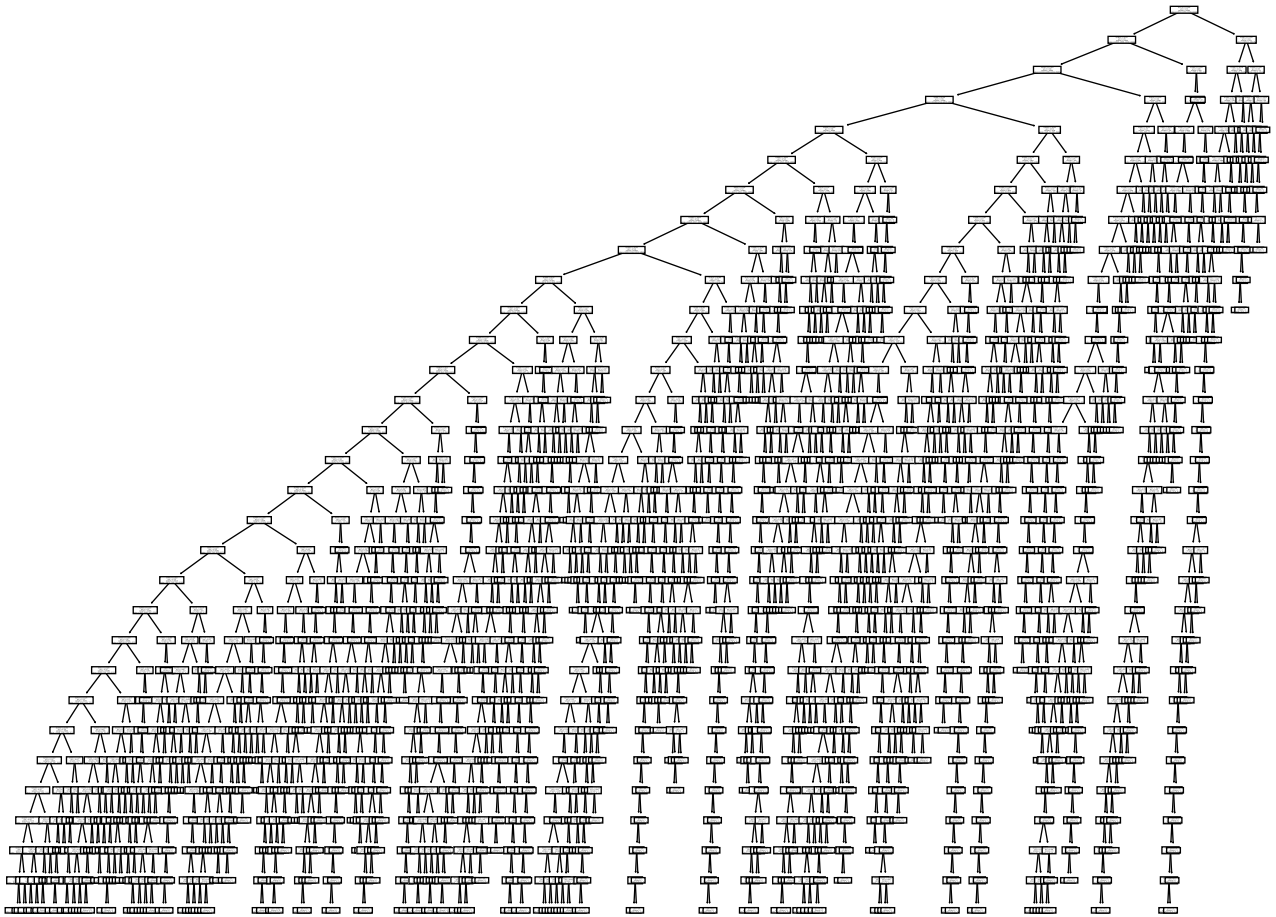
July 10, 2022

Introduction

- Text classification on raw text data is difficult for many reasons. Different text instances have different lengths, and text can contain numbers or other symbols, making data handling difficult.
- Several techniques are used to convert raw text data into meaningful features, such as bag-of-words, bag-of-phrases, and word-embedding. However, these features do not convey much information about the dataset. This is because either the features are of large dimension (bag-of-words), or they are already non-interpretable (word-embedding).
- We do not know how the classes are distributed in the input space; how two classes or groups of classes differentiate with each other; or if there is any sub-groups exist in a given class; or the selected features are optimal for a class, or a sub-group of a class, or even a single instance?
- We address these issues by using sparse oblique trees as a tool to understand the given NLP dataset.

Axis-aligned trees are not interpretable

- Axis-aligned trees are interpretable only for a small dataset, but as the dataset size increases, their size grows by a large margin, making them challenging to interpret.

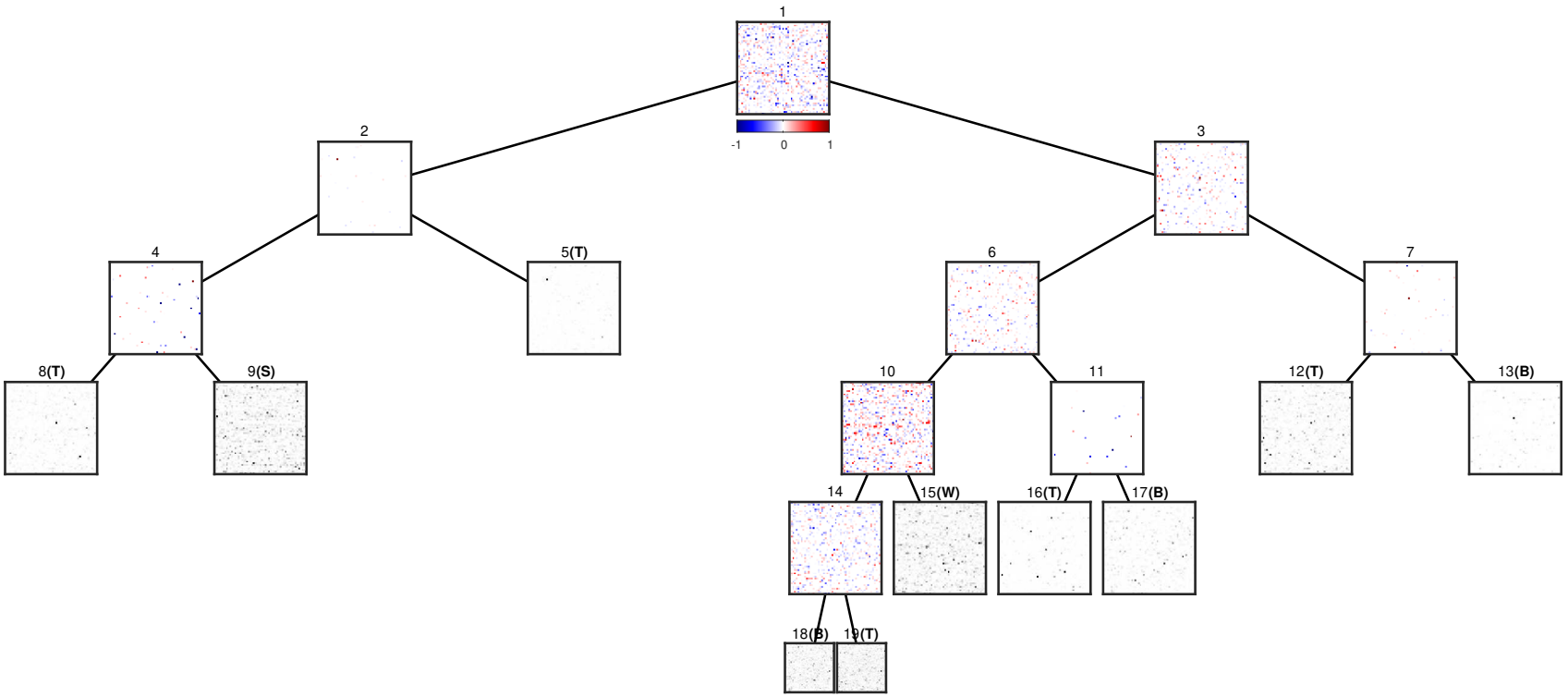


Axis-aligned tree trained on Ag news dataset.

Sparse oblique trees

- Axis-aligned trees are interpretable only for a small dataset, but as the dataset size increases, their size grows by a large margin, making them challenging to interpret.
- Unlike axis-aligned trees that operate only on a single feature at each node, the sparse oblique tree operates on a small, learnable subset of features.
- Sparse oblique trees are not only accurate but also very interpretable.

Sparse oblique trees



Axis-aligned tree trained on Ag news dataset.

Tree alternating optimization (TAO)

- Traditionally, decision trees have been trained with a recursive partition procedure, such as CART and C4.5. However, this produces sub-optimal trees and does not work well with oblique trees.
- Tree alternating optimization (TAO) [2] is a recently proposed algorithm that can achieve highly accurate oblique or axis-aligned trees.
- TAO can learn far more accurate oblique trees that remain small and very interpretable.
- TAO has been shown to outperform existing tree algorithms by a large margin [5], and to improve forests [3, 4].

Related work

The roots of DTs are in the 1950s, although they became really popular in the early 1980s. Since then, many approaches have been proposed to train them. Some common approaches:

- Greedy recursive splitting: start from the root and recursively split into two or more children based on solving a "purity" optimization problem (e.g. CART [1]). Simple and fast, but generates suboptimal trees.
- Approximate brute force search: attempts to find an optimal decision tree via mixed-integer programming. Do not scale beyond small or toy datasets.
- Non-greedy, global optimization algorithms: neither of the above. Tries to find approximate solution by optimizing over the entire tree. Well-known example: "soft decision trees". Our proposed algorithm, **Tree Alternating Optimization (TAO)** [2], is in this category but does not use soft trees.

Related work (cont.)

The literature of DTs is not restricted by training a single tree, Various methods have been proposed to ensemble them:

- **Bagging** train each tree independently on a different data sample: Random Forests, Extra Randomized Trees, etc...
- **Boosted Trees** sequentially train trees on reweighted versions of the data: AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost, etc...

Moreover, there are attempts to combine decision trees and other models (say neural nets) and train trees with more complex models at each node (e.g. SVMs, LDA). Some examples: neural decision trees, hierarchical mixture of experts (HME), Naive Bayes trees, etc.

TAO: general formulation

We consider trees whose nodes make hard decisions (not soft trees). Optimizing such trees is difficult because they are not differentiable. Assuming a tree structure $\mathbf{T}$ is given (say, binary complete of depth Δ), consider the following optimization problem over its parameters:

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

given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. $\mathbf{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{N}}$ is a set of parameters of all tree nodes. The loss function $L(\mathbf{y}, \mathbf{z})$ can be any loss which separates over training instances (e.g. squared error, cross-entropy, etc.). The regularization term ϕ_i (e.g. ℓ_1 norm) penalizes the parameters $\boldsymbol{\theta}_i$ of each node (to prevent overfitting).

TAO: separability of nodes

The TAO algorithm is based on 3 theorems: separability condition, reduced problem over a leaf, reduced problem over a decision node.

1. Separability condition

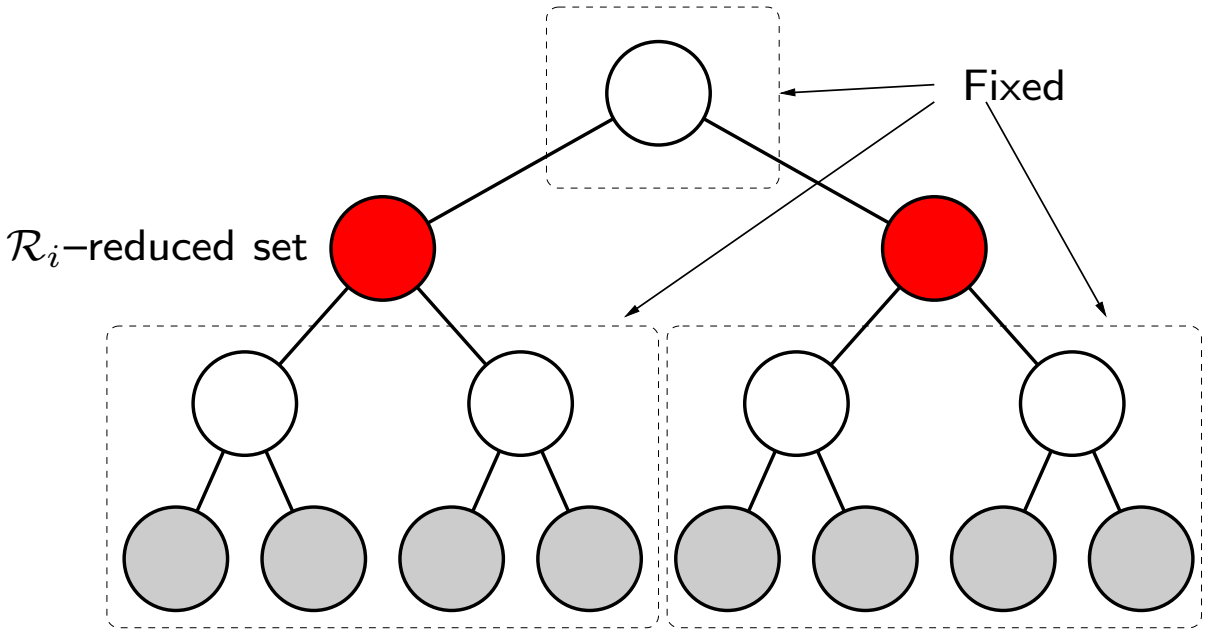
Consider any pair of nodes i and j. Assume the parameters of all other nodes ($\boldsymbol{\Theta}_{\text{rest}}$) are fixed. If nodes i and j are not descendants of each other, then $E(\boldsymbol{\Theta})$ can be rewritten as:

$$E(\boldsymbol{\Theta}) = E_i(\boldsymbol{\theta}_i) + E_j(\boldsymbol{\theta}_j) + E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$$

In other words, the separability condition states that any set of non-descendant nodes of a tree can be optimized independently. Note that $E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$ can be treated as a constant since we fix $\boldsymbol{\Theta}_{\text{rest}}$.

TAO: separability of nodes

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



TAO: optimizing over leaves

A set of non-descendant nodes are all the leaves. Optimizing over the parameters of one leaf is given by the following theorem.

2. Reduced problem over a leaf

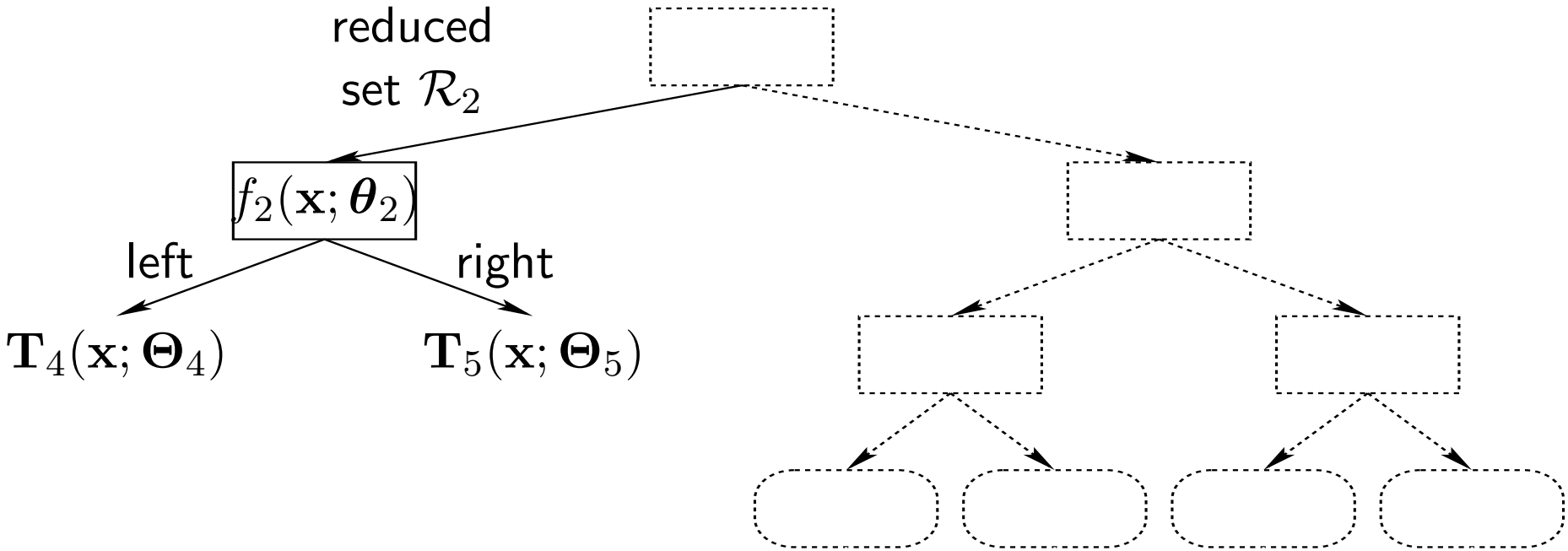
If i is a leaf, the optimization of $E(\boldsymbol{\Theta})$ over $\boldsymbol{\theta}_i$ can be equivalently written as:

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

The reduced set $\mathcal{R}_i$ contains the training instances that reach leaf i. Each leaf i has a predictor function $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ that produces the actual output. Therefore, solving the reduced problem over a leaf i amounts to fitting the leaf's predictor $\mathbf{g}_i$ to the instances in its reduced set to minimize the original loss (e.g. squared error).

TAO: optimizing over decision nodes

An example of a set of non-descendant nodes are all the decision nodes at the same depth:



Here, $\mathcal{R}_i$ is the reduced set of node i and (assuming binary trees) $f_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left},\texttt{right}\}$ is a decision function in node i which sends instance $\mathbf{x}_n$ to the corresponding child of i. We consider oblique trees, having hyperplane decision functions "go to right if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$" (where $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$).

TAO: optimizing over decision nodes (cont.)

The reduced problem over a decision node can be written as a weighted 0/1 loss binary classification problem on the node's reduced set instances:

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} \overline{L}_{in}(\overline{y}_{in}, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha\, \phi_i(\boldsymbol{\theta}_i)$$

where the weighted 0/1 loss $\overline{L}_{in}(\overline{y}_{in}, \cdot)$ for instance $n \in \mathcal{R}_i$ is defined as $\overline{L}_{in}(\overline{y}_{in}, y) = l_{in}(y) - l_{in}(\overline{y}_{in})\ \forall y \in \{\texttt{left}, \texttt{right}\}$, where $\overline{y}_{in} = \arg\min_y l_{in}(y)$ is a "pseudolabel" indicating a child which gives the lowest value of the regression loss L for instance $\mathbf{x}_n$ under the current tree.

For hyperplane nodes (oblique trees), this is NP-hard, but can be approximated by using a convex surrogate loss (we use the logistic loss). Hence, if ϕ_i is an ℓ_1 norm, this requires solving an ℓ_1-regularized logistic regression.

Pseudocode for training a single TAO tree

TAO repeatedly alternates optimizing over sets of nodes while monotonically decreasing the objective function.

input training set; initial tree $\mathbf{T}(\cdot;\boldsymbol{\Theta})$ of depth Δ
$\mathcal{N}_0,\dots,\mathcal{N}_\Delta \leftarrow$ nodes at depth $0,\dots,\Delta$, respectively
generate $\mathcal{R}_1 \leftarrow \{1,\dots,N\}$ using initial tree
repeat
  for $d = \Delta$ **down to** 0
    parfor $i \in \mathcal{N}_d$
      if i is a leaf **then**
        $\boldsymbol{\theta}_i \leftarrow$ fit a regressor/classifier (const, linear, neural net, etc.)
          $\mathbf{g}_i$ on reduced set $\mathcal{R}_i$
      else
        generate pseudolabels $\overline{y}_n$ for each point $n \in \mathcal{R}_i$
        $\boldsymbol{\theta}_i \leftarrow$ fit a binary classifier on $\mathcal{R}_i$
  update $\mathcal{R}_i$ for each node
until stop
prune dead subtrees of $\mathbf{T}$
return $\mathbf{T}$

Our approach

- Convert raw text data to TF-IDF features.
- Train a sparse oblique tree on TF-IDF features using TAO and pick the sparsity parameter such that the resultant tree is as sparse as possible but remains accurate enough.
- Use the weights of decision nodes to extract relevant TF-IDF features from the dataset.

Extracting features using weights of decision nodes

- Consider a decision node i in T with decision rule:
  - "if $\mathbf{w}_i^T \mathbf{x} + b_i \geq 0$ then go to right child, else go to left child"
    - $\mathbf{w}_i \in \mathbb{R}^D$ is the weight vector.
    - $b_i \in \mathbb{R}$ is the bias.
    - $\mathbf{x} \in \mathbb{R}^D$ is the input.
- Write $\mathbf{w}$ as: $\mathbf{w} = (\mathbf{w}_0 \ \mathbf{w}_- \ \mathbf{w}_+)$
  - $\mathbf{w}_0 = 0$.
  - $\mathbf{w}_- < 0$.
  - $\mathbf{w}_+ > 0$.
- Call $\mathcal{S}_0$, $\mathcal{S}_-$ and $\mathcal{S}_+$ the corresponding sets of indices in $\mathbf{w}$.

Extracting features using weights of decision nodes

- Consider input $\mathbf{x} \in \mathbb{R}^D$:
  - If $\mathbf{x}$ goes right, we represent the feature selected as a binary vector $\mu_+ \in \{0,1\}^d$, containing ones only at $\mathcal{S}_+$.
  - If $\mathbf{x}$ goes left, we represent the feature selected as a binary vector $\mu_- \in \{0,1\}^d$, containing ones only at $\mathcal{S}_-$.
- We call μ^+ and μ^- the NODE-FEATURES, where location of one represents features selected by $\mathbf{w}$.

Interpret dataset using NODE-FEATURES

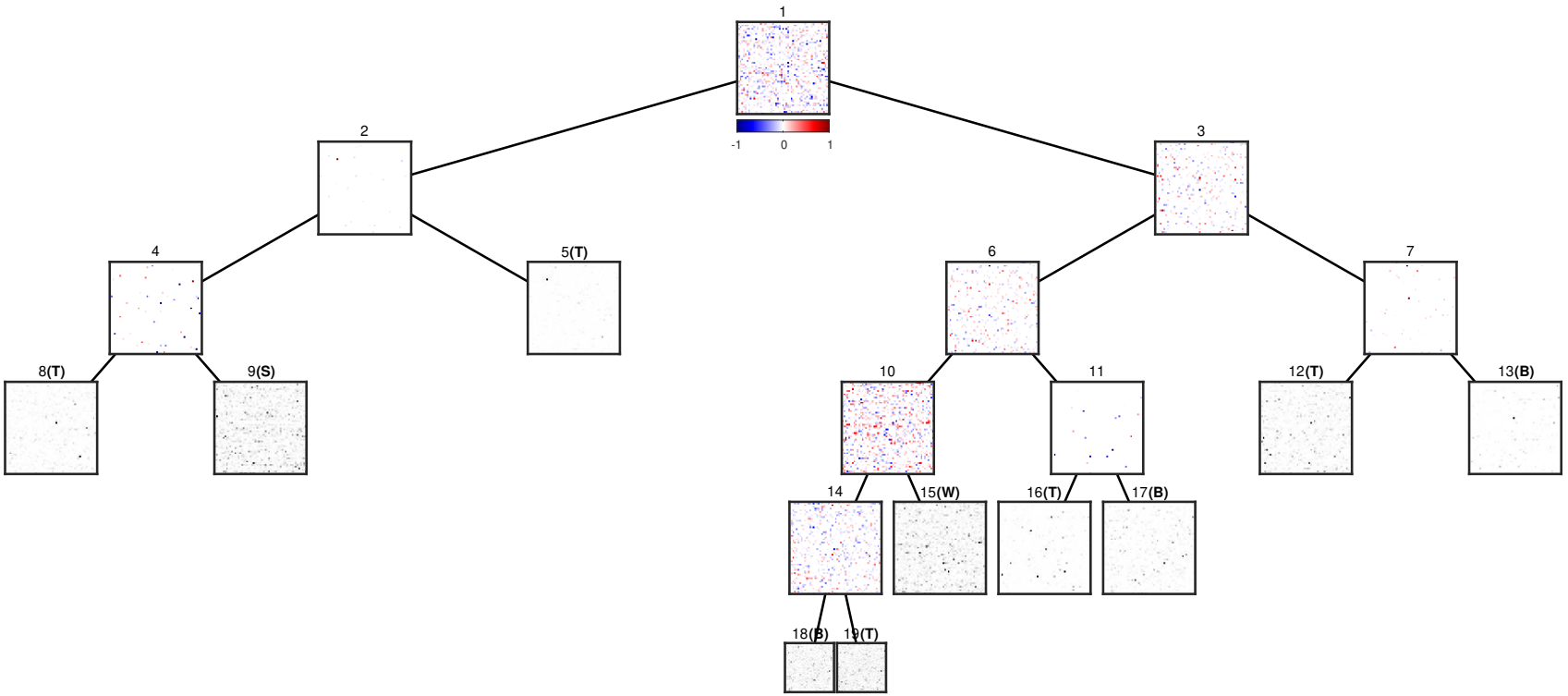
- For each decision node NODE-FEATURES represents the features related to left and right subtree. By using NODE-FEATURES, we can understand what set of features separate a group of classes.
- Features associated with a class k: for each node in the path from the root to leaf for class k collect NODE-FEATURES, and at the end take logical OR of all NODE-FEATURES. If there is more than one leaf for class k, take the union of all the features selected.
- For features specific to a given an input $\mathbf{x}$, repeat the process as above, but only for the leaf containing the input $\mathbf{x}$. Next, keep only those features that are active in the $\mathbf{x}$.

We can map the TF-IDF features to its corresponding word to interpret them.

AG's news dataset

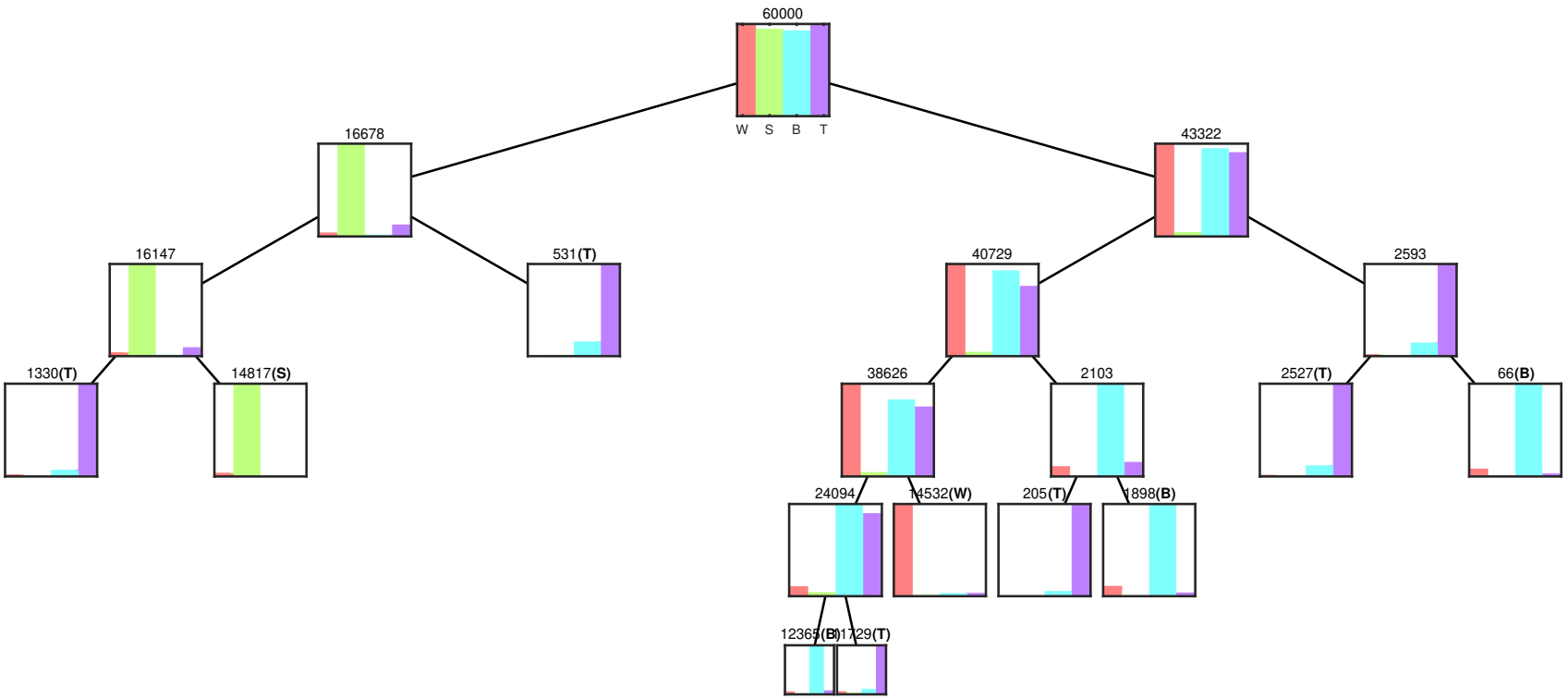
- AG is a collection of news articles, and these articles have been collected from more than 2 000 news sources.
- We use AG's news topic classification dataset containing four classes: world news, sports news, business news, and science/technology news.
- Dataset has 60 000 training and 7 600 test instances.
- We first transform the data into TF-IDF features, which creates a vocabulary size of 53 740, out of those, we choose 3 000 features using Chi-Squared feature selection.

Sparse oblique tree on AG's news dataset



W, **S**, **B**, and **T** represents world news, sports news, business news, and science/technology news class respectively.

Histograms


60000
W S B T
16678
43322
16147
531(T)
40729
2593
1330(T)
14817(S)
38626
2103
2527(T)
66(B)
24094
14532(W)
205(T)
1898(B)
12365(B)
1729(T)

Features selected for each class

Class	Original # features	# features selected by the tree
world	2902	958
sports	2692	579
business	2817	1175
science/technology	2844	1618
Total	3000	1969

Data interpretation using tree

- The class sports is easier to separate from other classes. Tree only needs only 579 out of 2692 features.
- The class business is much harder to separate. The tree requires 1 618 out of 2 817 features. However, the tree categorized the instances into three groups.
  - leaf# 13.
  - leaf# 17.
  - leaf# 18.

Subgroups of business class: leaf # 13

- Decision node# 3 sends leaf# 13 to the right subtree and rest to the left subtree.
- Among all the features associated with right subtree (μ_3^+) is `halliburton`.
- Leaf# 13 contains news articles specific to "Halliburton" company only.
- Some of the features associated with this leaf and μ_3^+ are: {`oil, filed, Iraq, united, president,discovered`}; which makes sense as, `halliburton` is the name of a multinational oil company.

Subgroups of business class: leaf # 17

- Decision node# 6 separates leaf# 17 and 18. It sends leaf # 17 to the right subtree.
- Some features related to μ_6^+ are: {`market, CEO, chairman, debt, deficit, dollar, IPO, monetary, fiscal, capital, stock`}.
- This indicates that the leaf# 17 contains the business news that is specific to finance.
- Indeed, when we look at instances in leaf# 17, they are mostly finance-based news articles like:

  " *NEW YORK - Newly optimistic investors sent stocks sharply higher Friday, propelling the Dow Jones industrials 112 points higher, as new economic data showed strength in manufacturing and the departure of PeopleSoft Inc.'s chief executive raised hopes for a merger in the tech sector. The major indexes closed the week with respectable gains.*"

Subgroups of business class: leaf # 18

- Decision node# 6 separates leaf# 17 and 18. It sends leaf # 18 to the left subtree.
- Some features related to μ_6^- are: {`soldier, killing, piracy, coup, attack, threatening, sanction, UN`}.
- This indicates that the leaf# 18 contains the business news that is specific to world politics.
- Indeed, when we look at instances in leaf# 18, they are mostly world politics based business news articles like: "*NEW YORK (Reuters) - Oil prices ended above $50 a barrel on Friday amid concern over possible fighting between rebels and the military in Nigeria's oil-rich delta region.*"

Feature selection for an instance of the class sci/tech

Original text:

"*AP - The Norwegian hacker famed for developing DVD encryption-cracking software has apparently struck again #151; this time breaking the locks on Apple Computer Inc.'s wireless music streaming technology.*"

Feature selection for an instance of the class sci/tech

Features selected by tree for all classes:

apparently, dvd, wireless, struck, inc, technology, developing, time, apple, software, computer, breaking, music, hacker, ap

Feature selection for an instance of the class sci/tech

Features selected by tree for sci/tech class only:

`dvd, wireless, inc, technology, developing, time, apple, software, computer, music, hacker, ap`

Only tech-related features remained.

Feature selection for an instance of the class sci/tech

Features selected by tree the instance only:

`dvd, wireless, inc, technology, developing, apple, software, computer, music`

Conclusion

- Sparse oblique trees can be used as an accurate yet interpretable model.
- Weights of the decision nodes can explain the underlying difference between classes, sub-group of a class, or group of classes.
- Using the hierarchical structure of the oblique tree, we can extract features that are tailored not only to class but also for specific instances.

References

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

[2] 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.

[3] M. Gabidolla and M. Á. Carreira-Perpiñán. Pushing the envelope of gradient boosting forests via globally-optimized oblique trees. In *Proc. of the 2022 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'22)*, New Orleans, LA, June 19–24 2022.

[4] 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.

[5] A. Zharmagambetov, S. S. Hada, M. Gabidolla, and M. Á. Carreira-Perpiñán. Non-greedy algorithms for decision tree optimization: An experimental comparison. In *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.