Motivation and summary

- Interpreting the image datasets is a difficult task, as each image contains a lot of irrelevant data. This makes it hard to understand what part of the image is important or what common concept defines a particular category of the class.

- We address these issues by using sparse oblique trees as a tool to understand the given image dataset. 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. We can learn accurate enough sparse oblique trees with the tree alternating optimization (TAO) algorithm.

Interpret dataset using sparse oblique tree

- Train a sparse oblique tree using TAO and pick the sparsity parameter such that the resultant tree is as sparse as possible but remains accurate enough. Next, inspect the weights ($w$) of the decision nodes to extract relevant features from the dataset.

- Write weight vector $w$ and input $x$ as $w = (w_0, w_- w_+)$ and $x = (x_0, x_-, x_+)$, where $w_0 = 0$, $w_- < 0$ and $w_+ > 0$ contain the zero, negative and positive weights in $w$, and $x$ is arranged accordingly. Call $S_0$, $S_-$ and $S_+$ the corresponding sets of indices in $w$.

  - If $x$ goes right, we represent the feature selected as a binary vector $\mu_+ \in \{0, 1\}^d$, containing ones only at $S_+$.
  - If $x$ goes right, we represent the feature selected as a binary vector $\mu_- \in \{0, 1\}^d$, containing ones only at $S_-$.

- We call $\mu^+$ and $\mu^-$ the NODE-FEATURES, where location of one represents features selected by $w$. We use NODE-FEATURES to interpret the dataset as follows:

  - 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 the selected.

  - For features specific to a given input $x$: Repeat the process as above, but only for the leaf containing the input $x$. Next, keep only those features that are active in the $x$.

- We can plot these features to visualize what concept is captured by these features.