Interpretable Image Classification Using Sparse Oblique Decision Trees

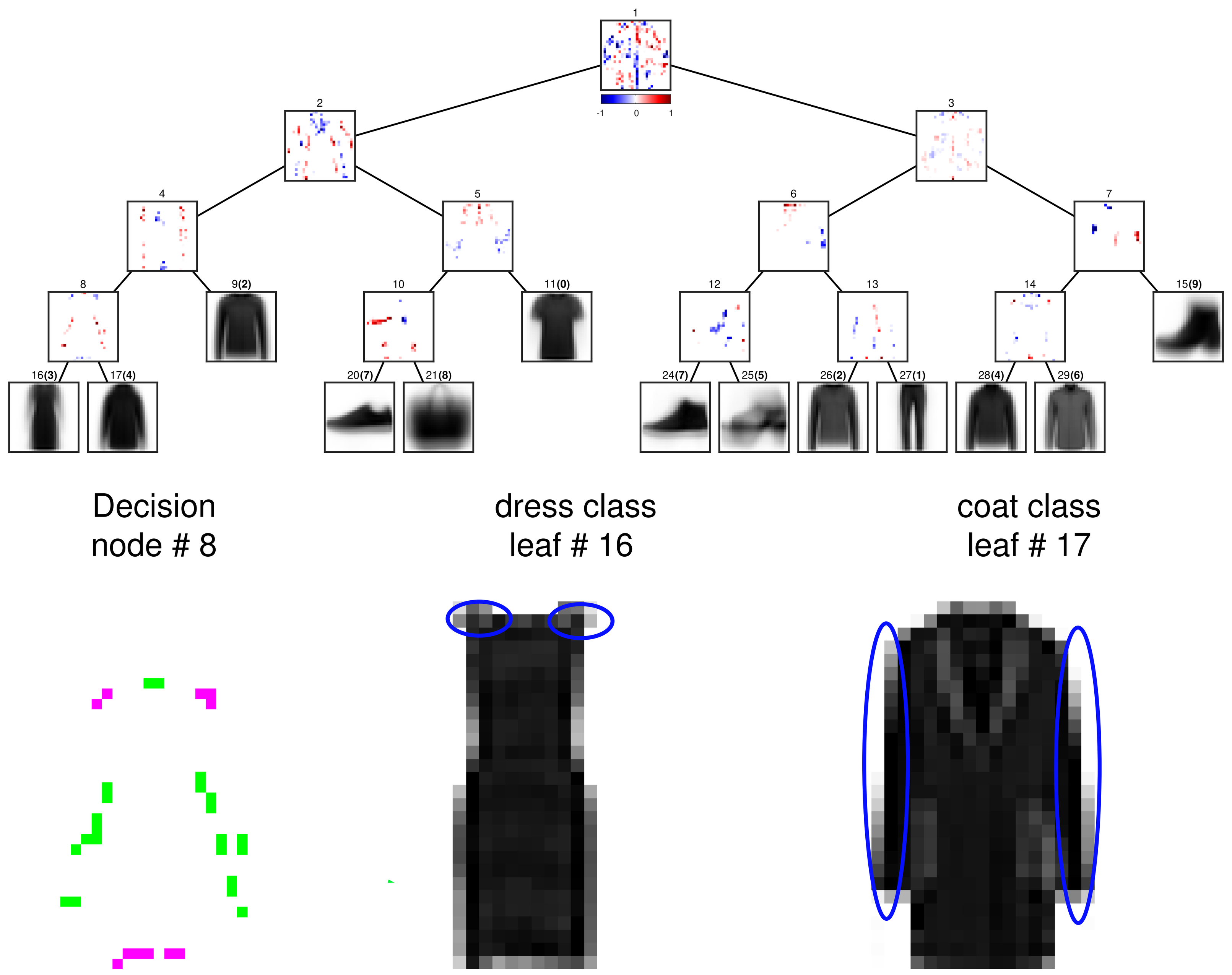
#4975

Suryabhan Singh Hada and **Miguel Á. Carreira-Perpiñán**,
Dept. CSE, UC Merced

1 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.

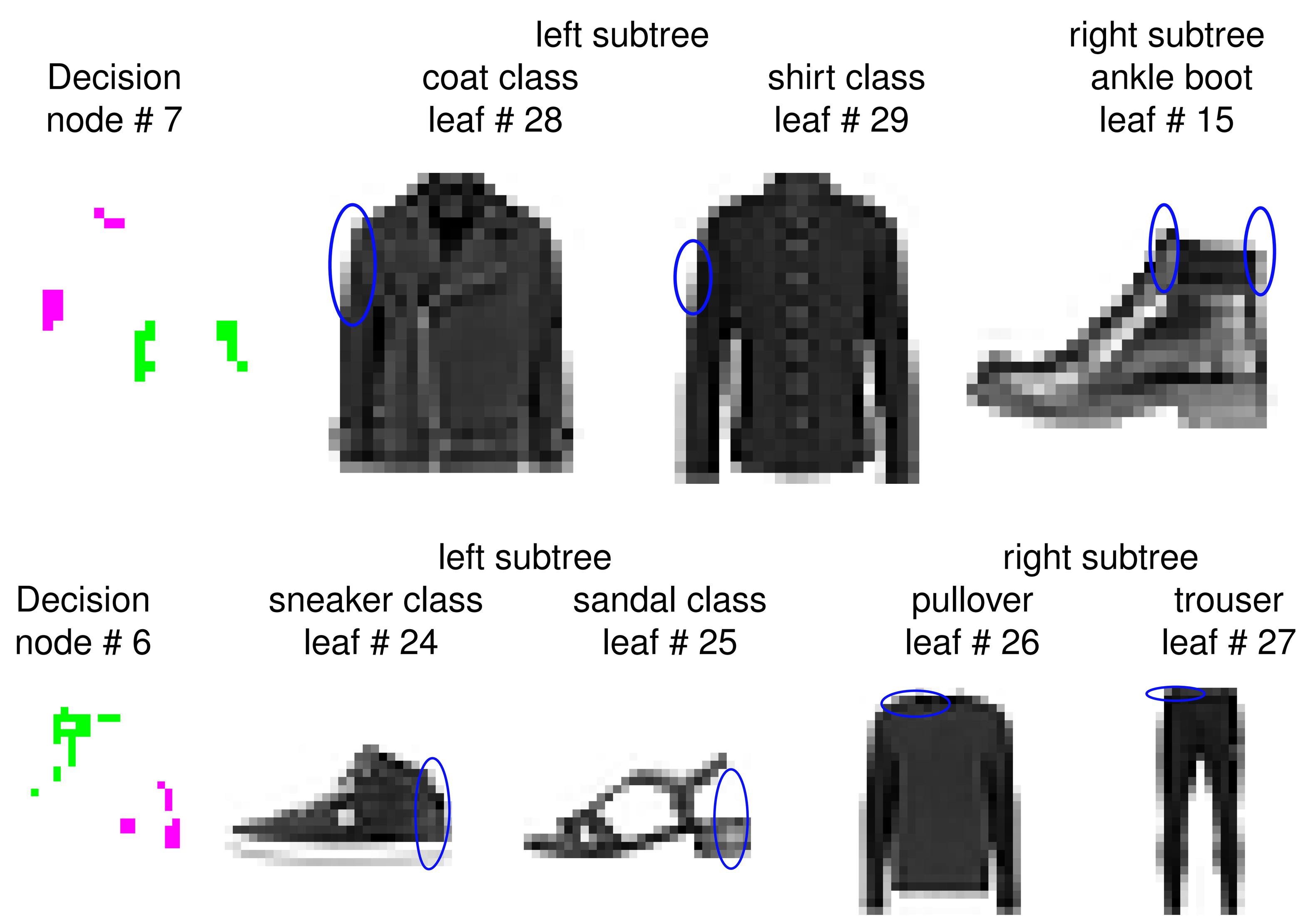
Fashion-MNIST dataset



2 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 $\mathbf{w} = (\mathbf{w}_0\ \mathbf{w}_-\ \mathbf{w}_+)$ and $\mathbf{x} = (\mathbf{x}_0\ \mathbf{x}_-\ \mathbf{x}_+)$, where $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{w}_- < \mathbf{0}$ and $\mathbf{w}_+ > \mathbf{0}$ contain the zero, negative and positive weights in **w**, and **x** is arranged accordingly. Call $\mathcal{S}_0$, $\mathcal{S}_-$ and $\mathcal{S}_+$ 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 $\mathcal{S}_+$.
  - If **x** goes right, 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 **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 selected.
  - For features specific to a given an 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.

Difference among group of classes



Feature selection for a given instance