Report date 15/10/2024

Paper report:

**Overview:**

HedgeCut consists in a modified version of Extremely Randomized Trees (ERT), that allows low latency (about 0.1ms) training data removal. The system has been thought for production environments where low latency and high throughput are required. The model is easily trainable and doesn't lose accuracy when compared with similar standard models such as Random Forest or standard ERTs.

The unlearnable model has all the benefits of unlearnable models compared to standard retrained models.

The whole idea goes around the slip robustness concept which defines how robust is a split to training data removal (i.e. is it still gonna have a high Gini gain even after some training data has been removed?).

During the initial HedgetCut training, while building the split nodes, the training algorithm prefers robust nodes and tries to generate robust splits.

The split generation is one of the key part of HedgeCuts: random selection of K features and generates split candidates, for categorical features we check if the category belongs to a subset of the possible categories, for continuous data we select a random cut point from one of the distribution quantiles (5th, 10th, 15th, … percentiles), data for the distribution and for the set of categories to pick the subset from is computed on the whole dataset and not on the local data only. Then statistics and Gini gain is computed for each split. The best split candidate (highest gini gain) robustness is tested against the remaining splits. If the split is non robust then the process starts again up to B times (B is user defined), if we keep getting non robust splits that split will be marked as non robust, the generated but not picked splits are saved (with the respective subtree) and the learning will move on.

When data needs to be removed the unlearning algorithm will update the splits statistics since it needs to remove the data that are being unlearnt from the stats. If one of the other splits has a better Gini score after the data removal then that new split replaces the old split, and we move on.

The split robustness is defined as "the number of records that can be removed from the data without affecting the split decision".

The model is super scalable and parallelizable since every tree in the ensemble is independent.

The model is also very fast because we store all the subtrees of potential splits that we may replace.

**Experimental metrics and scenarios:**

In the experiments they measure:

* Unlearning time of HedgeCut vs retrain time (comparison with other models)
* Throughput (prediction/second) of only prediction or of prediction with some unlearning requests here and there
* Accuracy of retrained vs unlearnt models (Hedgecut retrained vs hedgecut unlearnt and other models retrained vs hedgecut unlearnt)
* Training time
* B parameter effect on accuracy and training runtime
* ε parameter effect on accuracy and training time and fraction of robust nodes vs non robust

The experiments (scenarios) they work on:

* Income dataset
* Heart disease dataset
* Credit dataset
* Recidivism dataset
* Purchase behavior dataset
* Run on a single machine , in the times the loading time is not included

**Benchmarked against:**

Decision tree, random forest, ERT

**Limitations:**

* Doesn't work on regression scenarios
* Missing online learning

**Repo:**

<https://github.com/schelterlabs/hedgecut>

**Authors:**

**BIFOLD**

Sebastian and Stefan, Ted Dunning

**Year:**

2021