Consistency of Online Random Forests
Misha Denil, David Matheson and Nando de Freitas
University of British Columbia, Computer Science

Motivation
- Despite extensive use, very little is known about the mathematical properties of random forest algorithms. When do they converge, and why?
- Theoretical works typically focus on stylized versions of the algorithms used in practice.
- Online tree models have been around for a long time (e.g. Hoeffding trees).
- Online random forests have seen use recently in computer vision.
- Our contribution: A memory efficient online algorithm with provable consistency.

Random Forest Theory Publications

Stream Partitioning
- Split data randomly into two streams as it arrives:
  - Structure points influence the structure of the tree (cell partitions).
  - Estimation points estimate class membership probabilities in the leaves.
- Stream assignment happens uniformly at random in each tree.
- Could have an additional null stream but it doesn’t seem to help in practice.

Leaf Splitting Rules
- Rule 1: Refuse to create new leafs with fewer than \(n(d)\) (estimation) points.
- Rule 2: If the information gain from a split is less than \(r\) refuse to choose that split.
- Rule 3: If there are more than \(n(d)\) points in the leaf to be split then ignore Rule 2.
  - The first rule ensures that leafs are not split too often, so we eventually have a good estimate of the probability in each leaf.
  - The second rule discriminates between eligible splits based on a greedy heuristic.
  - The third rule ensures that no branch of the tree ever stops growing completely.

Memory Management
- Growing trees online is memory intensive. The bottleneck is storing statistics for candidate splits (these dwarf the cost of storing the rest of the tree).
- Each leaf requires \(O(\text{candidate dimensions} \times \text{candidate split points} \times \text{number of classes})\).
- Offline trees do not have this problem.
- We use a very simple idea from Hoeffding trees.
- Pick a fixed size for the fringe. Leaves in the fringe are active, the rest are inactive.
- For active leafs, store
  - the full splitting statistics.
- For inactive leafs, store
  - an estimate of \(p = \mathbb{P}(X \in A)\)
  - an estimate of \(e = \mathbb{P}(g(X) \neq Y | X \in A)\)
- The product of these two is an upper bound on the possible improvement from splitting \(A\).
- When a leaf is split a place in the fringe opens up. The inactive leaf with the largest improvement bound is activated to take its place.

Leaf Splitting Mechanism
- When a leaf is created, choose \(\min(1 + \text{Poisson}(\lambda, D))\) distinct candidate dimensions.
- Choose candidate split points by projecting the first \(m\) (structure) points into each candidate dimension.
- When a new structure point arrives:
  - Update structural statistics in each candidate child.
  - Optionally split if criteria are met.
- When a new estimation point arrives:
  - Update estimation statistics in each candidate child.
  - Update the predictor in the leaf.
- Below: Example of two trees processing a single data point.

Active Leaf

Data point
Structure Statistics
Estimation Statistics

Small Experiments
- Left: Compare the accuracy of the forest to the trees on a simple synthetic problem. Even with consistent base classifiers there is a significant benefit to averaging in finite time.
- Right: Comparison between online and offline performance on the USPS data set. Both online forests use 10 passes through the data.

Kinect Experiments
- Task: Assign a body part label to each pixel in a depth image.
- Left: Generate pairs of 640x480 resolution depth and body part images by rendering random poses from the CMU mocap dataset (depth / ground truth / predictions).
- Sample 50 pixels for each body part class from each pose for training.
- Centre: Each split thresholds the depth difference between two pixels described by two offsets from the pixel being classified. Candidate pairs of offsets are sampled from a 2d Gaussian distribution with variance 75.0.
- Right: Comparison between our algorithm and Saffari (2009). Limiting the fringe size to 1000 nodes we require 1.6GB for leaf statistics. Saffari (2009) requires 10GB with a fixed depth of 8.

Code
- Code for all experiments:
- General purpose random forest library:
  - https://github.com/david-matheson/rftk

Proof outline
- If the base classifier is consistent then the ensemble is consistent.
  - Sufficient to prove a single tree is consistent.
- If a classifier is consistent conditioned on \(I \in \mathbb{I}\) and \(\nu(I) = 1\) then the classifier is consistent without conditioning on \(I\).
  - Use \(I\) as an infinite sequence partitioning data into structure and estimation points.
  - Require each stream infinitely long.
  - Reduce to several single class problems by mapping \((X, Y) \rightarrow (X, \mathbb{P}[Y = k])\)
  - Apply theorem from Devroye 1996.
    - Requires: \(N^*(A_i(X)) \rightarrow \infty\) and \(\text{diam}(A_i(X)) \rightarrow 0\)
  - First condition: follows from splitting mechanism plus assuming \(X\) has a density.
  - Second condition: show leafs will be split infinitely often and that the size of a leaf is reduced each time it is split.
    - Bound time before a single split, iterate to bound time to arbitrary number of splits.
    - Show that expected size of first dimension shrinks after a split.
  - Extension to a bounded fringe: Sufficient to show that for any inactive leaf, the probability it has not been activated by time \(t\) goes to 0 as \(t\) grows.
  - Bound number of splits as a function of number of data points with Hoeffding bounds.

Consistency of Online Random Forests

Saffari et al. (2009)