Dynaforest: High-Level Pseudocode

Notation and Setup

- *n_trees*: Number of trees in the forest.
- *window*: Number of trees to consider in each optimization step.
- *max_depth*: Maximum depth of each tree.
- *min_samples*: Minimum number of samples required to allow a split.
- *feature_subsampling_pct*: Fraction of the feature set considered for each split.
- *bootstrapping*: Boolean; if true, each tree is trained on a bootstrap sample of the data.
- $X \in \mathbb{R}^{N \times d}$: Training features (with N samples and d features).
- $y \in \mathbb{R}^N$: Training target values.

1. Initialization

(a) Read hyperparameters: $n\_trees, window, max\_depth, min\_samples, feature\_subsampling\_pct, bootstrapping$.

(b) Compute the number of features to consider in each split:

$$\text{num\_features\_considering} = \max\Big(\lfloor d \cdot \text{feature\_subsampling\_pct} \rfloor, 1\Big).$$

2. Fit Phase

Input: Training set (X, y).

(a) **Initialize Trees**:

(i) For $i = 1$ to n_trees:

i. If *bootstrapping* is `True`, draw a bootstrap sample (X_i, y_i) of the same size as (X, y).
ii. Randomly select num_features_considering features from the d available.
iii. Create a new tree T_i (initially a *stump*).
iv. Call get_best_split(X_i, y_i) on T_i to find the best root split.
v. Split T_i once, creating left and right children.
vi. Store T_i in the forest.
vii. Record T_i's predictions on the full training set.

(b) **Iterative Optimization**:

(i) **Repeat until no improvement**:

i. Randomly pick a subset of *window* trees, $\{T_{i_1}, T_{i_2}, \ldots, T_{i_{window}}\}$.
ii. For each tree T_{i_k} in this subset:
A. Compute the temporary ensemble mean of the *other window* $- 1$ trees:

$$\hat{y}_{-k} = \frac{1}{\text{window} - 1} \sum_{j \in \{i_1, \ldots\}, j \neq i_k} T_j(X).$$

B. Call get_best_split$(T_{i_k}, \hat{y}_{-k})$ to evaluate potential error reduction.
C. Store the *error reduction* for T_{i_k}.
iii. If all recorded error reductions ≤ 0, **stop** (no further gain).
iv. Otherwise, choose T_{best} with the **largest positive** error reduction.
v. Split T_{best} on its best split candidate.
vi. Update T_{best}'s predictions on the training set.

3. Predict Phase

Input: Test data X_{test}.

(a) For each tree T_i in the forest:

$$\hat{y}_i = T_i(X_{\text{test}}).$$

(b) Compute the ensemble prediction by averaging:

$$\hat{y}_{\text{ensemble}} = \frac{1}{n\_trees} \sum_{i=1}^{n\_trees} \hat{y}_i.$$

(c) Return $\hat{y}_{\text{ensemble}}$ as the final prediction.

4. Notes on the Algorithm

- **Forest-Level Splitting:** Unlike standard random forests, where each tree grows independently, the *Dynatree* approach selects which tree to split based on which candidate tree-split reduces the *ensemble* mean-squared error the most.
- **Reducing Bias & Correlation:**
  1. By involving partial ensemble predictions ($\hat{y}_{-k}$) when determining splits, the approach can better reduce overall bias.
  2. Bootstrapping and feature subsampling help ensure the trees remain partially de-correlated, reducing variance in the ensemble.
- **Stopping Criterion:** The process halts when additional splits in a random subset of trees no longer reduce the ensemble error.