6.0 Why RF?

- Single decision tree does not perform well, but it is super fast.
- Random forest is an ensemble classifier that
  - combines many decision trees,
  - outputs the class that is the mode of the class's output by individual trees

6.1 Algorithm

6.1.1 Training

Given:

- D: The set of training data.
  - $D = \{\langle X_1, y_1\rangle, \langle X_2, y_2\rangle, \langle X_3, y_3\rangle, \cdots, \langle X_N, y_N\rangle\}$, where
  - $\forall i \in [1, N], X_i = \{x_{i1}, x_{i2}, \cdots, x_{iM}\}$.
- N: The number of training data.
  - $|D| = N$
- M: The number of variables in each data.
  - $\forall i \in [1, n],\ |X_i| = M$

Do:

- For each tree:
  - ***Bootstrap*** to get n data from entire N training data.
  - The selected n data forms the training set $TR \subset D$.
  - The unselected $N - n'$ data forms the testing set $TE \subset D$.
  - The bootstrap process is performed in a "replace-inclusive" way.
  - The n data sets may contain same ones. That is, $n' \leq n$.
  - For each node:
  - Randomly select m feature variables, calculate the best split based on these m variables in the training set TR.

  Note:
- Each tree is fully grown and not pruned.

6.1.2 Bootstrapping 引导

Do:

- Reserve a tiny set TR as the training set.
- Reserve a large set TE as the test set.
- Do n times:
  - Select a sample $\langle X_i, y_i \rangle \in D$
  - Copy this sample into TR.
  - Put this sample back to D.

  Output:
- Training set TR, where $|TR| = n$.
- Test set TE, where $|TE| = N - n', n' \in [1, n]$.

  Note:
- Since we only copy whatever we selected from D and put it back again in each iteration, it is possible that we pick a $\langle X_i, y_i \rangle$ from D twice.
- Suppose we have $N - n'$ records left in D that's never been chosen, then $n' \leq n$.

6.1.3 Selection of n and m

How to select n?

- Build trees until the error no longer decreases.

How to select m?

- Try to recommend defaults.
  - Half of them, twice of them, ...
  - Pick the best.

6.2 Pros and Cons

6.2.1 Advantages

Performance

- High accuracy.
- High efficiency.
  - Especially on large datasets.

Good for high dimensional samples

- Good tolerance on high dimension data samples.
    - Handle thousands of input variables without variable deletion.
- Highlights important variables.

Good in error estimate and exception handling

- Generates internal unbiased estimate of generalization error during the building process of the forest.
- Effective in estimating missing data.
    - Maintains accuracy when a large proportion of the data is missing.
- Balances error in datasets with unbalanced class population.