eCornell

CHEAT SHEET

Random Forest

Algorithm Name	Random Forest
Description	Random Forest combines full-depth CART classifiers into an ensemble. The classifiers are trained independently (possibly in parallel) on data bootstraps with one small modification: for each split in each tree a small subset of k features is randomly sampled and all other features are ignored. This procedure reduces the variance of the final classifier as training trees on random features results in them being uncorrelated further.
Applicability	Classification and regression problems.
Assumptions	Data set is not too high dimensional; Similar inputs have similar labels.
Underlying Mathematical Principles	Full depth trees have high variance. Random Forests combine many high variance classifiers to create a low variance ensemble.
Hyperparameters	• Number of trees m to average. • Number of features k to sub-sample. Random Forests are famously insensitive to hyper-parameters. Default choices: m large enough until the predictions converge (m=100, or m =1000) $k = \sqrt{d}$, where d is the dimensionality of the input data
Setting	Regression or classification ($y_i \in \{+1, -1\}$). While **regression** trees return continuous values, we can still use them to solve **classification** problems with discrete labels. For example, we can return the sign of the output of the tree.
Out-of-bag Error	Because each tree is not learned on the full data but only a sub-sample, each tree has its own out-of-bag subset of the training data on which it was not trained. One can estimate the test error of a Random Forest classifier by computing the classification error of each data point obtained by averaging the predictions of those ensemble members that were **not** trained on this data point.
Ensemble	Combination of many classifiers (in this case, limited depth regression trees): $\hat{h}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} h_{D_i}(\mathbf{x})$

Pseudocode	`RandForests`$(D = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), m,\ k,\ n)$
	H $\leftarrow 0$;
	for t=1:m **do**
	  $D_t \leftarrow$ subset of size n randomly sampled from D with replacement;
	  $h_t \leftarrow CART(D_t, k, maxdepth = \infty)$
	  H $\leftarrow$ H $+ \frac{1}{T} h_t$;
	end
	return H

Prediction Confidence	Random Forests naturally provide prediction confidences. For example in classification settings you can output the percentage of trees that predicted the most common label as a statistic of confidence. In regression settings you can output the variance of the predictions of all ensemble members.
Feature Importance	Random Forests can naturally provide a measure of feature importance. For example, for each feature compute the average (or total) reduction in loss obtained on the training set through splitting on this particular feature.
Strengths	Random Forests are particularly well suited for applications with heterogeneous features (as CART trees are blazingly fast to evaluate). The big advantages of Random Forests are that they tend to not overfit to the training data, are amazingly insensitive to hyper-parameters, work naturally for regression and classification settings, provide feature importances, output prediction confidences, and that they provide an unbiased estimate of the testing error directly from the training set.
Weakness	Random Forests tends to excel in lower dimensional feature spaces (<1000 dimensions) and is often not well suited for very high and sparse feature spaces (e.g. bag-of-words text data). For very large data sets (n in the millions) Random Forests can become slow to evaluate, as the trees become too large.
XGBoost	XGBoost is a popular implementation of Random Forests and GBRT. It is highly optimized and one of the most robust and widely used machine learning algorithms in practice.