Random Forests

Carlo Tomasi

October 29, 2018

Decision trees can represent arbitrarily complex hypothesis spaces $\mathcal{H}$. For instance, one can subdivide the unit interval $[0, 1]$ on the real line into segments of length ϵ for any $\epsilon > 0$ with a deep enough tree that splits each parent segment in half. In multiple dimensions, to subdivide $[0, 1]^d$ into small hypercubes of side ϵ, build a tree that interleaves d interval-splitting trees, one per dimension. One can then assign a separate value to each hypercube, thereby approximating any desired distribution function to any degree.

Because of their expressiveness, decision trees must be curbed lest they overfit. The complexity of individual decision trees is typically controlled by pruning them after expansion [6]. Empirical evidence shows that a better alternative is to use random forest predictors, which combine the predictions of several trees through a voting scheme.

A *random forest* [5] is a predictor that consists of a collection of decision trees $h_m(\mathbf{x})$ for $m = 1, \ldots, M$ that depend on independent identically distributed sets of random parameters. Each tree is trained on a different view of the data, and casts a unit vote for the label (for classification) or value (for regression) associated to input $\mathbf{x}$. Votes are aggregated by a suitable summary function: Majority for classification, mean or median for regression.

Of course, multiple votes would be useless if they all agreed. Because of this consideration, several ways have been proposed and compared to each other [2, 7] to ensure that the votes that different trees cast are independent of each other. These include the following:

Bagging, that is, applying the bootstrap resampling method we saw in an earlier note to train different trees with different bags out of T. The bags are made of training samples drawn independently and uniformly at random from T with replacement [4].

Boosting, in which the random subsets of samples are drawn in sequence, each subset is drawn from a distribution that favors samples on which previous predictors in the sequence failed, and predictors are given a vote weight proportional to their performance [8].

Arcing, similar to boosting but without the final weighting of votes [3].

Random forests combine bagging with random feature selection for each node of every tree in the forest [1, 5]. Bagging ensures that different trees have different views of the data. Random feature selection means that instead of picking *the best* dimension d on which to split at any given node of any given tree, the dimension d is chosen at random.

The combination has proven very effective in compromising between expressiveness and generalization. In Breiman's words,

i Its accuracy is as good as boosting and sometimes better.

ii It's relatively robust to outliers and noise.

iii It's faster than bagging or boosting alone.

iv It gives useful internal estimates of statistical risk, strength, correlation and variable importance.

v It's simple and easily parallelized.

Algorithm 1 summarizes how to train a random forest and use it for prediction. Using Breiman's training method, the function $\mathrm{findSplit}$ we developed for decision trees is modified to pick the feature component index j on which to split *at random*. This random selection is in contrast with the optimal choice of j, that is, the one that maximizes the decrease in impurity. Randomness is preferred over optimality in the splitting rule, since randomness increases the diversity of the trees in the forest and decreases the statistical risk as a consequence. Typical values of M, the number of trees, are in the tens or hundreds, and this hyper-parameter can be optimized by cross-validation. The size $|S|$ of each subset S is equal to the size N of the entire training set.

0.1 Out-of-Bag Estimate of the Statistical Risk

Recall that the goal of machine learning is not to minimize the training risk, but rather the statistical risk of the predictor, that is, the expected loss over samples drawn out of the (unknown) model distribution $p(\mathbf{x}, y)$. Interestingly, bagging enables a way to estimate the statistical risk of the random forest predictor as follows.

We saw that when drawing a set (or *bag*) B of N samples uniformly at random and with replacement out of the training set T, about 37% of the samples are left out of B (and an equal fraction of samples are repetitions). Let now $B_1, \ldots, B_M$ be the M bags used to train the M predictors $h_1, \ldots, h_M$ in a random forest. As mentioned above, each bag contains N samples drawn out of T with replacement, where $N = |T|$. An *out-of-bag predictor* h_{oob} *that works only on training data* can be constructed by letting predictor h_m provide a value for a training sample if and only if the sample is *not* in B_m, and then taking a summary (majority, mean, median) of the vote:

$$\forall \mathbf{x} \text{ such that } (\mathbf{x}, y) \in T\,, \quad h_{\mathrm{oob}}(\mathbf{x}) = \mathrm{summary}(\{h_m(\mathbf{x}) \,|\, m = 1, \ldots, M \text{ and } (\mathbf{x}, y) \notin B_m\})\ .$$

Notation: Let us unpack this expression. Pick a sample $(\mathbf{x}, y)$ out of the training set T. For prediction, of course, we ignore the true label y. To form the prediction $h_{\mathrm{oob}}(\mathbf{x})$, we first ask the trees in the forest to provide their best estimates $\hat{y}$ of the prediction. However, we only accepts values from trees that were trained without using the sample $(\mathbf{x}, y)$. Therefore, h_{oob} lets tree number m provide a value $h_m(\mathbf{x})$ if and only if there is a sample $(\mathbf{x}, y)$ in the training set T for which

$$(\mathbf{x}, y) \notin B_m\ .$$

Finally, wee form the prediction $h_{\mathrm{oob}}(\mathbf{x})$ by summarizing the values provided by all participating trees. For data points $\mathbf{x}$ that do not come from T, the value of $h_{\mathrm{oob}}(\mathbf{x})$ is undefined.

The *out-of-bag risk* is then the risk of h_{oob} estimated on T', the set of samples out of T that were not used to train all the trees:

$$T' = \{t \in T \,|\, \exists\ m \text{ such that } t \notin B_m\}\ .$$

The set T' may be a proper set of T, because some of the samples in T may show up in all of the bags. These omni-present samples are not used to compute the out-of-bag risk, because no tree is allowed to vote on them. The out-of-bag risk is defined as

$$e_{\mathrm{oob}}(h, T') = \frac{1}{|T'|} \sum_{(\mathbf{x},y)\in T'} \ell(y, h_{\mathrm{oob}}(\mathbf{x}))\ .$$

This empirical risk can be shown to be an unbiased estimate of the random forest's statistical risk [5].

Algorithm 1 Training a random forest and using it for prediction

function $\phi \leftarrow \text{trainForest}(T, M)$ ▷ M is the desired number of trees
  $\phi =\leftarrow \emptyset$ ▷ The initial forest has no trees
  for $m = 1, \ldots, M$ **do**
    $S \leftarrow$ set of $|T|$ samples drawn uniformly at random out of T with replacement
    $\phi \leftarrow \phi \cup \{\text{trainTree}(S, 0)\}$
  end for
end function

function $\tau \leftarrow \text{trainTree}(S, \text{depth})$
    ▷ This function is the same as in the Algorithm used to train a decision tree, except that it calls findSplitR instead of findSplit
end function

function $[L, R, j, t] \leftarrow \text{findSplitR}(S)$
▷ *This function replaces* findSplit *in* trainTree *when training a random forest*
  $i_S \leftarrow \text{i}(S)$ ▷ $\text{i}(S)$ is the impurity of S. See text.
  $\Delta_{\text{opt}} \leftarrow -1$ ▷ At the end, Δ_{opt} will be the greatest decrease in impurity.
  $j \leftarrow$ integer drawn uniformly at random out of $\{1, \ldots, d\}$
  for $\ell = 1, \ldots, u_j$ **do** ▷ Loop on all thresholds for dimension j.
    $L \leftarrow \{\mathbf{x} \,|\, x_j \leq t_j^{(\ell)}\}$ ▷ The splitting thresholds $t_j^{(\ell)}$ for $j = 1, \ldots, N$ and $\ell = 1, \ldots, u_j$
    $R \leftarrow S \setminus L$ ▷ are assumed to have been precomputed (see text).
    $\Delta \leftarrow i_S - \frac{|L|}{|S|}\text{i}(L) - \frac{|R|}{|S|}\text{i}(R)$ ▷ See text for a faster way to compute Δ
    if $\Delta > \Delta_{\text{opt}}$ **then**
      $[\Delta_{\text{opt}}, L_{\text{opt}}, R_{\text{opt}}, d_{\text{opt}}, t_{\text{opt}}] \leftarrow [\Delta, L, R, j, t]$
    end if
  end for
  return $[L_{\text{opt}}, R_{\text{opt}}, d_{\text{opt}}, t_{\text{opt}}]$
end function

function $y \leftarrow \text{forestPredict}(\mathbf{x}, \phi, \text{summary})$
  $V = \{\}$ ▷ A set of values, one per tree, initially empty
  for $\tau \in \phi$ **do**
    $y \leftarrow \text{predict}(\mathbf{x}, \tau, \text{summary})$ ▷ The predict function for decision trees
    $V \leftarrow V \cup \{y\}$
  end for
  return $\text{summary}(V)$
end function

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms. *Machine Learning*, 36(1/2):105–139, 1999.

[3] L. Breiman. Arcing classifiers. Technical report, Statistics Department, University of California, Berkeley, CA, 1996.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[7] T. Dietterich. An experimental comparison on three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, 1996.