Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 10: Random Forests (Draft: version 0.9.1)

Topics to be covered:

- Random forest recipe
- Why do random forests work?
- Ramifications of random forests
  - Out-of-bag error estimate
  - Variable importance
  - Proximity
  - Missing value imputation
  - Outliers
  - Unsupervised learning
  - Balancing prediction error

The method of random forests proposed by Breiman is still one of the best machine learning tools available today. In this lecture, we introduce random forests and give some theoretic analysis on why the method works. The basic reference is the paper by Breiman, which we follow closely [1].

1 Random Forests Recipe

Let $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ be the data, where $x^{(i)} \in \mathfrak{X}$ and $y^{(i)} \in \mathfrak{Y}$. As usual, $\mathfrak{X}$ is the input (feature) space with d features and $\mathfrak{Y}$ is the output (label) space. In case of regression, it is assumed that $\mathfrak{Y} = \mathbb{R}^n$; for classification, $\mathfrak{Y}$ is assumed to be a discrete finite set of K elements. As done always throughout this course, we assume there is a probability measure $P = P_{X,Y}$ on $\mathfrak{X} \times \mathfrak{Y}$, from which the data points $(x^{(i)}, y^{(i)})$ are drawn as IID samples. In our random forests discussion, we will fix $m \ll d$. Typically we use $m = \sqrt{d}$. But there is no hard and fast rule for how to fix m. We also fix constants $N_{\min}$ and B, and an impurity measure (Gini, entropy, misclassification rate, etc.) to be used for this recipe. The random forests recipe goes as follows:

Random Forest Recipe

Do for $k = 1$ to B

- Draw a bootstrap resample $\mathfrak{D}(k)$ from $\mathfrak{D}$. Let $OOB(k)$ be the set of data points not in $\mathfrak{D}(k)$. ("OOB" stands for *"out-of-bag"*.) Similarly we call $\mathfrak{D}(k)$ the k-th *"in-bag"* set and its elements "in-bag" data or data points; and $OOB(k)$ the k-th *"out-of-bag"* set. Thus $\mathfrak{D}$ is the disjoint union of the two, i.e.

$$\mathfrak{D} = \mathfrak{D}(k) \cup OOB(k)$$

- Grow, i.e. construct, a tree T_k using $\mathfrak{D}(k)$ by applying the following splitting rule:
  - At each node, randomly select m features (out of total d features)
  - Split the node by choosing the split that decreases the impurity the most among all possible splits using only these m selected features
  - Stop splitting the node if it contains fewer than $N_{\min}$ elements, where $N_{\min}$ is a predetermined number.
  - Do not prune

- From each tree T_k, get the predictor φ_k for $k = 1, \cdots, B$. (Our usual notation convention is $\hat{\varphi_k}$, but since it is understood in context, we drop the "hat" symbol and use φ_k.)

Get the final predictor $\zeta(x)$:

- For regression:

$$\zeta(x) = \mathrm{Av}_k\{\varphi_k(x)\} = \frac{1}{B}\sum_{k=1}^{B}\varphi_k(x) \tag{1}$$

- For classification:

$$\zeta(x) = \mathrm{Plur}_k\{\varphi_k(x)\} = \underset{j\in\mathcal{Y}}{\mathrm{argmax}}|\{k : \varphi_k(x) = j\}|. \tag{2}$$

Remark. *For ordinal response (output) variables, which we always assume are discrete, one takes the median rather than average as a definition of $\zeta(x)$.*

2 Why do random forests work?

In this section we present Breiman's argument [1] on why random forests work. He makes somewhat stronger assumptions here and there, and his resulting estimates are somewhat weaker. Nonetheless it reveals the inner workings of random forests, and moreover it sheds light on why and how the random forest recipe is so designed to make the correlation of tree predictors as small as possible by resorting to deliberate randomization.

Let us write $\varphi_k(x)$ gotten out of $\mathfrak{D}(k)$ as $\varphi(x, \theta_k)$. Here, θ_k stands for all those things that were involved in the construction of $\varphi_k(x)$ such as the random selection of m features at each node and the training set $\mathfrak{D}(k)$ itself. In general, treating θ as a random variable, we write $\varphi(x, \theta)$. It should be noted that we don't need to specify the distribution of θ, as it is used only as a theoretical background. But, at least in theoretical sense, we can talk about the expectation $E_\theta\varphi(x, \theta)$ of $\varphi(x, \theta)$ with respect to θ and the probability $P_\theta(\varphi(X, \theta) = j)$, etc.

2.1 Case A: Regression

As usual, we can regard $\mathrm{Av}_k\varphi_k(x) = \dfrac{1}{B}\sum_{k=1}^{B}\varphi(x,\theta_k)$ as an estimator of $E_\theta\varphi(x,\theta)$. In this sense, with appropriate preparation, it is not hard to show that

$$\mathrm{Av}_k\varphi_k(x) \to E_\theta\varphi(x,\theta),$$

almost surely for θ. (See [1].)

We define the *prediction error (risk or generalization error) of the forest* by

$$PE^*(forest) = E_{X,Y}|Y - E_\theta\varphi(Y,\theta)|^2, \tag{3}$$

to which $E_{X,Y}[Y - \mathrm{Av}_k\varphi(X,\theta_k)]^2$ is seen to converge. Define the *average (expected) prediction error of individual tree* by

$$PE^*(tree) = E_\theta E|Y - \varphi(X,\theta)|^2. \tag{4}$$

Here E stands for the expectation $E_{X,Y}$ with respect to the measure $P_{X,Y}$, and in this lecture we usually drop the subscript X,Y. It is empirically observed that the random forests have very small bias, so we assume $E[Y - E_\theta\varphi(x,\theta)] = 0$. But, here, let us make a stronger assumption that

$$E[Y - \varphi(X,\theta)] = 0$$

for all θ. Then

$$\begin{aligned}
PE^*(forest) &= E|Y - E_\theta\varphi(X,\theta)|^2 \\
&= E|E_\theta[Y - \varphi(X,\theta)]|^2 \\
&= E\left\{E_\theta[Y - \varphi(X,\theta)] \cdot E_{\theta'}[Y - \varphi(X,\theta')]\right\} \\
&= EE_{\theta,\theta'}(Y - \varphi(X,\theta)) \cdot (Y - \varphi(X,\theta')) \\
&= E_{\theta,\theta'}E(Y - \varphi(X,\theta)) \cdot (Y - \varphi(X,\theta')) \\
&= E_{\theta,\theta'}\mathrm{Cov}(Y - \varphi(X,\theta), Y - \varphi(X,\theta')),
\end{aligned}$$

where the second equality uses the fact that Y is independent of θ.

Using the covariance-correlation formula, we have

$$\begin{aligned}
&\mathrm{Cov}(Y - \varphi(x,\theta), Y - \varphi(x,\theta')) \\
= \;& \rho(Y - \varphi(x,\theta), Y - \varphi(x,\theta'))std(Y - \varphi(x,\theta))std(Y - \varphi(x,\theta')) \\
= \;& \rho(\theta,\theta')sd(\theta)sd(\theta'),
\end{aligned}$$

where $\rho(\theta, \theta')$ is the correlation between $Y - \varphi(x, \theta)$ and $Y - \varphi(x, \theta')$, and $sd(\theta) = std(Y - \varphi(x, \theta))$ is the standard deviation of $Y - \varphi(x, \theta)$. Therefore, we have

$$PE^*(forest) = E_{\theta,\theta'}\{\rho(\theta, \theta')sd(\theta)sd(\theta')\}.$$

Define the weighted average correlation $\bar{\rho}$ by

$$\bar{\rho} = \frac{E_{\theta,\theta'}\{\rho(\theta, \theta')sd(\theta)sd(\theta')\}}{(E_\theta sd(\theta))^2}. \tag{5}$$

Note that $|\bar{\rho}| \leq 1$. By the definition of PE^*, we have

$$\begin{aligned} PE^*(forest) &= \bar{\rho}(E_\theta sd(\theta))^2 \\ &\leq \bar{\rho}E_\theta(sd(\theta))^2 \\ &= \bar{\rho}E_\theta E|Y - \varphi(X, \theta)|^2 \\ &= \bar{\rho}PE^*(tree) \end{aligned}$$

Theorem 1. *Assume $E[Y - \varphi(X, \theta)] = 0$ for all θ. Then*

$$PE^*(forest) \leq \bar{\rho}PE^*(tree)$$

Remark. *Note that $\bar{\rho}$ is a weighted average correlation between $\varphi(X, \theta)$'s. So the essence of what this theorem says is that the smaller the correlation is, the smaller the prediction error becomes. Recall that when the nodes are split in each tree T_k, we use only a small fraction of all available variables (features). Unless the data is truly pathological, it is reasonable to assume that two sets of features that have not much overlap between them should in general have a small correlation and the random forests' node splitting strategy is designed to reduce the correlation this way.*

Theorem 1 above is due to Breiman. But there is a much simpler analysis to the same effect. To do that, first recall that it was empirically verified that the random forests have very small bias. So the risk (prediction error or generalization error) is basically captured by the variance.

Now let us make a few simplifying assumptions. First assume that $\varphi_1, \cdots, \varphi_B$ as random variables have the same law with finite standard de-

viation σ. Let ρ be the correlation between any two of them. Then we have

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{B}\sum_{k=1}^{B}\varphi_k\right) &= \frac{1}{B^2}\sum_{k,m}\mathrm{Cov}(\varphi_k,\varphi_m) = \frac{1}{B^2}\sum_{k\neq m}\mathrm{Cov}(\varphi_k,\varphi_m) + \frac{1}{B^2}\sum_{k}\mathrm{Var}(\varphi_k) \\
&= \frac{1}{B^2}(B^2-B)\rho\sigma^2 + \frac{1}{B^2}B\sigma^2 \\
&= \rho\sigma^2 + \frac{\sigma^2}{B}(1-\rho).
\end{aligned}$$

Note that the first term $\rho\sigma^2$ is small if the correlation ρ is small, which can be achieved if the number m of candidate variables for each node split is much smaller than the total number d of all variables. (See the Remark above.) The second term gets small if B is large. i.e. if we take a large number of trees.

2.2 Case B: Classification

Let us now look at why the random forest reduces the prediction error in the case of classification. First recall that for the classifier

$$\begin{aligned}
\zeta(X) &= \underset{k}{\mathrm{Plur}}\varphi(X,\theta_k) \\
&= \underset{j}{\mathrm{argmax}}\frac{1}{B}|\{k:\varphi(X,\theta_k)=j\}|,
\end{aligned}$$

which is an estimator of

$$\underset{j}{\mathrm{argmax}}P_\theta(\varphi(X,\theta)=j).$$

In fact, with appropriate preparation, we can show that $|\{k:\varphi(X,\theta_k)=j\}|$ converges almost surely to $P_\theta(\varphi(X,\theta)=j)$ as $k\to\infty$, thereby implying

$$\underset{j}{\mathrm{argmax}}\frac{1}{B}|\{k:\varphi(X,\theta_k)=j\}| \to \underset{j}{\mathrm{argmax}}P_\theta(\varphi(X,\theta)=j),$$

almost surely. (An interested reader should consult [1].)

Before we proceed, let us give the following definition:

Definition 1. (Margin Function)

$$mr(X,Y) = P_\theta(\varphi(X,\theta)=Y) - \max_{j\neq Y}P_\theta(\varphi(X,\theta)=j).$$

Note that $P_\theta(\varphi(X,\theta) = Y)$ is the probability that the tree predictor $\varphi(X,\theta)$ predicts the class label of X is Y. Similarly, $P_\theta(\varphi(X,\theta) = j)$ is the probability that the tree predictor φ predicts the class label of X is j. Observe that, in the limit as $k \to \infty$,

(i) if $mr(X,Y) \geq 0$, then the random forest classifier f will predict that the class label of X is Y

(ii) if $mr(X,Y) < 0$, then the random forest classifier f will predict that the class label of X is something other than Y

Therefore the classification error occurs when and only when $mr(X,Y) < 0$. (We ignore the arbitrariness that comes with a tie.) Hence the prediction error probability PE^* of the random forest classifier is

$$PE^* = P(mr(X,Y) < 0). \tag{6}$$

Define the mean (expectation) s of mr by

$$s = E[mr(X,Y)]. \tag{7}$$

Assume $s > 0$. In other words, we are assuming our random forests predictor is on average more often right than wrong. Then by Lemma 1 below, we have

$$PE^* \leq \frac{\mathrm{Var}(mr)}{s^2}. \tag{8}$$

Lemma 1. *Let U be a random variable and let s be any positive number. Then*

$$\mathrm{Prob}[U < 0] \leq \frac{E|U-s|^2}{s^2}.$$

Proof. By the Chebyshev (Markov) inequality, we have

$$\begin{aligned} E|U-s|^2 &\geq s^2\mathrm{Prob}(|U-s| \geq s) \\ &= s^2\{\mathrm{Prob}(U-s \geq s) + \mathrm{Prob}(U-s \leq -s)\} \\ &\geq s^2\mathrm{Prob}(U \leq 0) \\ &\geq s^2\mathrm{Prob}(U < 0). \end{aligned}$$

□

Now define

$$\hat{j}(X,Y) = \underset{j \neq Y}{\operatorname{argmax}} P_\theta(\varphi(x,\theta) = j).$$

Then

$$\begin{aligned} mr(X,Y) &= P_\theta[\varphi(X,\theta) = Y] - P_\theta[\varphi(X,\theta) = \hat{j}(X,Y)\,] \\ &= E_\theta[\mathbb{I}(\varphi(X,\theta) = Y) - \mathbb{I}(\varphi(X,\theta) = \hat{j}(X,Y))\,] \end{aligned}$$

Let

$$rmg(\theta, X, Y) = \mathbb{I}(\varphi(X,\theta) = Y) - \mathbb{I}(\varphi(X,\theta) = \hat{j}(X,Y)).$$

Then we have

$$mr(X,Y) = E_\theta[rmg(\theta, X, Y)]$$

Therefore

$$\begin{aligned} & \mathrm{Var}(mr) \\ = \;& E(E_\theta rmg(\theta,X,Y))^2 - (E\,E_\theta rmg(\theta,X,Y))^2 \\ = \;& EE_\theta rmg(\theta,X,Y)E_{\theta'}rmg(\theta',X,Y) - E\,E_\theta rmg(\theta,X,Y)\,E\,E_{\theta'}rmg(\theta',X,Y) \\ = \;& EE_{\theta,\theta'}\{rmg(\theta,X,Y)rmg(\theta',X,Y)\} - E_\theta E\,rmg(\theta,X,Y)E_{\theta'}E\,rmg(\theta',X,Y) \\ = \;& E_{\theta,\theta'}\Big\{E\,[rmg(\theta,X,Y)rmg(\theta',X,Y)] - E\,rmg(\theta,X,Y)E\,rmg(\theta',X,Y)\Big\} \\ = \;& E_{\theta,\theta'}\mathrm{Cov}\big(rmg(\theta,X,Y), rmg(\theta',X,Y)\big) \end{aligned}$$

Now write

$$\begin{aligned} & \mathrm{Cov}\big(rmg(\theta,X,Y), rmg(\theta',X,Y)\big) \\ = \;& \rho\big(rmg(\theta,X,Y), rmg(\theta',X,Y)\big) std(rmg(\theta,X,Y)) std(rmg(\theta',X,Y)) \\ = \;& \rho(\theta,\theta') sd(\theta) sd(\theta'), \end{aligned}$$

where $\rho(\theta,\theta')$ is the correlation between $rmg(\theta,X,Y)$ and $rmg(\theta',X,Y)$, and $sd(\theta) = std(rmg(\theta,X,Y))$ is the standard deviation of $rmg(\theta,X,Y)$.

Therefore

$$\mathrm{Var}(mr) = E_{\theta,\theta'}[\rho(\theta,\theta')sd(\theta)sd(\theta')].$$

Define the weighted average correlation $\bar{\rho}$ of $rmg(\theta,X,Y)$ and $rmg(\theta',X,Y)$ by

$$\bar{\rho} = \frac{E_{\theta,\theta'}[\rho(\theta,\theta')sd(\theta)sd(\theta')]}{E_{\theta,\theta'}[sd(\theta)sd(\theta')]}. \tag{9}$$

Note that $|\bar{\rho}| \leq 1$. Since $sd(\theta) = sd(\theta')$, we have

$$\mathrm{Var}(mr) = \bar{\rho}\{E_\theta sd(\theta)\}^2 \leq \bar{\rho} E_\theta [sd(\theta)]^2. \tag{10}$$

Now

$$E_\theta [sd(\theta)]^2 = E_\theta E[rmg(\theta, X, Y)]^2 - E_\theta [E\, rmg(\theta, X, Y)]^2.$$

On the other hand, since $s = E(mr(X, Y)) = E E_\theta rmg(\theta, X, Y) = E_\theta E\, rmg(\theta, X, Y)$, we have

$$\begin{aligned} s^2 &= \{E_\theta E\, rmg(\theta, X, Y)\}^2 \\ &\leq E_\theta [E\, rmg(\theta, X, Y)]^2. \end{aligned}$$

Therefore

$$E_\theta [sd(\theta)]^2 \leq E_\theta E[rmg(\theta, X, Y)]^2 - s^2. \tag{11}$$

Note that $rmg(\theta, X, Y) = \mathbb{I}(\text{something}) - \mathbb{I}(\text{somthing else})$, which implies that the only values $rmg(\theta, X, Y)$ can have are $0, \pm 1$. Therefore we have

$$E_\theta E[rmg(\theta, X, Y)]^2 \leq 1.$$

Now combining this with (11), (10) and (8), we can prove the following theorem.

Theorem 2. *Let $\bar{\rho}$ be the weighted average correlation defined by (9) and s the mean of $mr(X, Y)$ defined by (7). Assume $s > 0$. Then the prediction error of the random forest classifier satisfies the following inequality.*

$$PE^* \leq \bar{\rho} \frac{1 - s^2}{s^2}. \tag{12}$$

Remark. *It should be noted that the estimate (12) is not sharp. Nonetheless, it shows how crucially the predictor error depends on the correlation and we can conclude that if the correlation can be made small, the prediction error will become small too. That is the rationale behind the strategy of utilizing only a small fraction of the available variables at each node of the tree.*

To estimate the error, we need the following definition:

Definition 2. *For $j \in \mathfrak{Y}$, define*

$$Q(x, j) = \frac{\sum_k \mathbb{I}(\varphi(x, \theta_k) = j : (x, y) \in OOB(k)))}{\sum_k \mathbb{I}((x, y) \in OOB(k)))}.$$

Now fix a generic data point $(x, y) \in \mathfrak{D}$. Thus (x, y) may or may not be in $\mathfrak{D}(k)$. Then $Q(x, j)$ is an estimator for $P_\theta(\varphi(x, \theta) = j)$. Therefore $Q(x, y) - \max_{j \neq y} Q(x, j)$ is an estimator for

$$mr(x, y) = P(\varphi(x, \theta) = y) - \max_{j \neq y} P(\varphi(x, \theta) = j).$$

Now let $\mathfrak{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ be a given dataset. Then the error $PE^* = P(mr(X, Y) < 0)$ can be estimated by the following:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(mr(x^{(i)}, y^{(i)}) < 0) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(Q(x^{(i)}, y^{(i)}) < \max_{j \neq y^{(i)}} Q(x^{(i)}, j)).$$

3 Ramifications of random forests

We now show how to exploit the random forests to extract some important side information. Most of the materials in this section are taken from the manuscript of Breiman and Cutler [2].

3.1 Out-of-Bag (OOB) error estimate

Recall that at stage k, for $k = 1, \cdots, B$, of the random forest construction, roughly $1/3$ of $\mathfrak{D}$ is left out from the bootstrap sample $\mathfrak{D}(k)$, and this left-out set is denoted by $OOB(k)$. So the data set $\mathfrak{D}$ is a disjoint union of $\mathfrak{D}(k)$ and $OOB(k)$. For each k, using $\mathfrak{D}(k)$, construct a tree from which a predictor $\varphi_k(x)$ is derived. Since $OOB(k)$ is not used in the construction of $\varphi_k(x)$, it can be used to test $\varphi_k(x)$. This fact can be utilized to obtain an error estimate which is a good proxy for the generalization error.

The procedure goes as follows. For each data point $x^{(i)}$, compute $\varphi_k(x^{(i)})$ *only if* $x^{(i)}$ is in $OOB(k)$ for all $k = 1, \cdots, B$. Since $x^{(i)}$ is in $OOB(k)$ for roughly $1/3$ of all stages, $\varphi_k(x^{(i)})$ is computed only for some $k = 1, \cdots, B$. Nonetheless these computed results $\varphi_k(x^{(i)})$ for each $x^{(i)}$ can be aggregated accordingly. Namely, for the regression problem, take the average of the computed values $\varphi_k(x^{(i)})$; for the classification problem, take the majority voting of the computed $\varphi_k(x^{(i)})$, which is denoted by $\alpha(x^{(i)})$. Note that it is a good proxy for $\zeta(x^{(i)})$, the value of the final predictor of the random forest (see the recipe at the beginning of this lecture); but one must also note that $\alpha(x^{(i)})$ is computed as if $x^{(i)}$ were not in the dataset in the first place. Note

also that $\alpha(x^{(i)})$ is computed for all $i = 1, \cdots, N$, if B is big enough. In fact, this is a probabilistic statement, since the probability of any single data point belonging in all of $\mathfrak{D}(k)$ for $k = 1, \cdots, B$ is very small if B is big enough.

Comparing $\alpha(x^{(i)})$ with the given output $y^{(i)}$ this way, we can get a fairly good proxy for the error estimate of $x^{(i)}$ as if $x^{(i)}$ were not in the training set of the random forest. Repeating this for $i = 1, \cdots, N$, we can easily obtain the overall error of the random forest. This error estimate is called the out-of-bag (OOB) error estimate and we may regard it as a reasonably good proxy for the generalization error.

3.2 Variable importance.

3.2.1 Impurity importance

Recall that in the random forest recipe at each node the split decision is made to make the impurity decrease the most. So at that node, it is reasonable to presume the variable involved with the split decision is the most important one. The **impurity importance** of each variable is the addition of the absolute values of such decrease amounts for all splits in all trees in the forest every time the split is done using *that* variable. If the impurity measure is Gini, we call it the **Gini importance**; if it is entropy, we call it **entropy importance**, and so on.

3.2.2 Permutation importance

The permutation importance is based on the idea that the data value associated with an important variable must have a certain structural relevance to the problem at hand so that the effect of a random shuffling (permutation) of its data value must be reflected in the decrease in accuracy of the resulting predictor.

To clarify how the permutation is done, let us set up the notations. Let $\mathfrak{D}$ be a set of data defined by $\mathfrak{D} = \{x^{(1)}, \cdots, x^{(n)}\}$, where the i-th data $x^{(i)}$ is written in vector form as $x^{(i)} = [x_1^{(i)}, \cdots, x_d^{(i)}]^T$. We write $\mathfrak{D}$ in matrix form

as

$$\mathfrak{D} = \begin{pmatrix} x_{11} & \cdots & x_{ij} & \cdots & x_{1d} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{id} \\ \vdots & & \vdots & & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nd} \end{pmatrix},$$

in which $x_{ij} = x_j^{(i)}$ for all $i = 1, \cdots, n$ and $j = 1, \cdots, d$. So the i-th row represents the the i-th data.

Let π be a permutation on $\{1, \cdots, n\}$ denoted by

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix}.$$

Then the permuted data set $\mathfrak{D}(\pi, j)$ gotten from $\mathfrak{D}$ by applying π to the j-th column becomes

$$\mathfrak{D}(\pi, j) = \begin{pmatrix} x_{11} & \cdots & x_{\pi(1)j} & \cdots & x_{1d} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \cdots & x_{\pi(i)j} & \cdots & x_{id} \\ \vdots & & \vdots & & \vdots \\ x_{n1} & \cdots & x_{\pi(n)j} & \cdots & x_{nd} \end{pmatrix}.$$

Recall that the j-th column of $\mathfrak{D}$ is the values of the j-th feature running for all data points $i = 1, \cdots, N$. In $\mathfrak{D}(\pi, j)$ these values in the j-th by column is shuffled by the permutation π. Let us use the following notation:

$$x^{(i)}(\pi, j) = (x_{i1} \cdots x_{\pi(i)j} \cdots x_{id}).$$

Example. *Let*

$$\mathfrak{D} = \begin{pmatrix} 11 & 33 & 44 & 66 \\ 22 & 55 & 11 & 33 \\ 66 & 0 & 88 & 44 \end{pmatrix},$$

and let $\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$. *Then*

$$\mathfrak{D}(\pi, 3) = \begin{pmatrix} 11 & 33 & 11 & 66 \\ 22 & 55 & 88 & 33 \\ 66 & 0 & 44 & 44 \end{pmatrix}.$$

The permutation importance is based on the idea that if the variable (feature) x_j is important, the error using the randomly shuffled dataset $\mathfrak{D}(\pi, j)$ should be much worse than that using the correct dataset $\mathfrak{D}$. On the other hand, a feature (variable) has not much relevance or importance, if the error using $\mathfrak{D}(\pi, j)$ is more or less the same as the one with $\mathfrak{D}$.

▶ Permutation importance by OOB set
The permutation importance by OOB set measures how much the accuracy deteriorates due to permutation in the OOB sets. In what follows, let us first fix j. (Namely, we are computing the variable importance of the j-th variable.)

• Classification

For each k, let φ_k be the predictor associated with the tree T_k. Let π_k be a permutation on the k-th out-of-bag set $OOB(k)$. Define

$$C(\pi, j, k) = \sum_{(x^{(i)}, y^{(i)}) \in OOB(k)} \left\{ \mathbb{I}(\, y^{(i)} = \varphi_k(x^{(i)})\,) - \mathbb{I}(\, y^{(i)} = \varphi_k(x^{(i)}(\pi_k, j))\,) \right\},$$

which counts the number of times the data points in $OOB(k)$ are correctly classified minus the number of times the permuted data points in $OOB(k)$ are correctly classified.

Define

$$VI(j, k) = \frac{C(\pi_k, j, k)}{|OOB(k)|},$$

and finally define the permutation variable importance of the j-th variable x_j by

$$VI(j) = \frac{1}{B} \sum_{k=1}^{B} VI(j, k).$$

It is very reasonable to presume that the higher $VI(j)$ is, the more important x_j is. This can be called a "raw" score in the sense that it is not normalized so that it cannot be compared across different random forests. In order to do so, one must somehow present a normalized form. This can be done if $VI(j, k)$ are independent for $k = 1, \cdots, B$. Namely, one can compute the standard error and divide $VI(j)$ by the standard error. This new score is a

z score which can be claimed as a normalized score. If normality is further assumed, one can make various probabilistic statements.

However, for most applications, the above "raw" permutation importance $VI(j)$ is good enough.

Remark. $C(\pi, j, k)$ is not the only form that can be used. One can instead define it using the concept of "margin" as used in Section 2 above. The margin in short is the number of correct classifications subtracted by the biggest number among all wrong classifications. One can also use the ratio of the margin of unpermuted OOB data over the margin of the permuted OOB data. All these variations are more or less equivalent.

• Regression

For regression problem, the variable importance can also be defined similarly. Let us briefly outline how it is done.

For each k, let φ_k be the regressor associated with the tree T_k. Define

$$S(k) = \sum_{(x^{(i)},y^{(i)})\in OOB(k)} |y^{(i)} - \varphi_k(x^{(i)})|^2,$$

which is the square root of the sum of squares errors of φ_k when applied to $OOB(k)$. Let π_k be a permutation on the k-th out-of-bag set $OOB(k)$. Define also

$$S(\pi, j, k) = \sum_{(x^{(i)},y^{(i)})\in OOB(k)} |y^{(i)} - \varphi_k(x^{(i)}(\pi_k, j))|^2,$$

which is the square root of the sum of squares errors of φ_k when applied to a permuted $OOB(k)$ in which $x^{(i)}$ is replaced with $x^{(i)}(\pi_k, j)$. Then define

$$VI(j, k) = \frac{S(\pi_k, j, k)}{S(k)},$$

and finally define the permutation variable importance of the j- variable by

$$VI(j) = \frac{1}{B}\sum_{k=1}^{B} VI(j, k).$$

There are many variants one can think of. First in the definition of $VI(j)$ one may take the geometric mean of $VI(j,k)$ for $k = 1, \cdots, B$ instead of the arithmetic mean proposed above. One may also use the difference instead of the quotient in the definition of $VI(j,k)$. The definition of $S(k)$ and $S(\pi, j, k)$ can be modified to use the square root of the sum of squared errors. It is not well studied which variant works consistently better than the others.

▶ **Permutation importance by test set**

In case the dataset is abundantly available, it is better to split the dataset into the training set and the test set. One can then do the training of the random forest using the training set and use the test set to get the variable importance. To do so, one simply permutes the whole of the test set all at once and applies the above procedure. This is a much simpler procedure as one does not have to make the permutation for *every* out-of-bag set. On the other hand, care must be taken to avoid bias in the test set, say, by resorting to random sampling. One can perhaps take many smaller chunks of test sets and do the averaging.

3.3 Proximity

The proximity of two data points is a measure of how close they are. Knowledge of the proximity for all data points, if available, is very important information that can be exploited in many ways. For instance, it can be used in many nonlinear dimension reduction and clustering problems. If all the features are numeric, each data point is represented as a point in the input space $\mathbb{R}^d$. If so, one can easily devise a proximity measure that says that the smaller the Euclidean distance is, the higher the proximity. However, even in this all numeric input case, the issue is not that simple. For instance, a length can be measured in meters, in millimeters, or even in kilometers, each of which case may produce quite a different outcome. In other words, it all depends on the normalization, which sometimes is a subtle problem on its own. Moreover, if some feature are not numeric but abstract, say categorical, it is not clear how to define the proximity in the first place. Random forest, however, provides a handy and sensible way of defining the proximity.

Recall that random forest algorithm constructs a tree T_k for each bootstrap resample $\mathfrak{D}(k)$ for $k = 1, \cdots, B$. And from T_k is derived the k-th predictor φ_k in the following manner: given any input x, one can put it

down the tree T_k to eventually arrive at a terminal node, which is denoted by $\nu_k(x)$. This ν_k can be regarded as a terminal node operator that sends x to its terminal node $\nu_k(x)$. (This kind of operation is also called the tree run of T_k.) Once the terminal node $\nu_k(x)$ is found, $\varphi_k(x)$ can be computed by taking the majority vote of the values in the terminal node $\nu_k(x)$ in the case of classification and the average in case of regression. One must note that two input points with the same terminal nodes have the same φ_k values but the converse does not hold.

For each $k = 1, \cdots, B$, and for $i, j = 1, \cdots, N$, define the following k-th proximity $Prox_k(i, j)$ of two data points $x^{(i)}$ and $x^{(i)}$, regardless of whether in bag or out of bag, by

$$Prox_k(i,j) = \begin{cases} 1 & \text{if } \nu_k(x^{(i)}) = \nu_k(x^{(j)}) \\ 0 & \text{if } \nu_k(x^{(i)}) \neq \nu_k(x^{(j)}). \end{cases}$$

Note that this says that $Prox_k(i, j) = 1$ if and only if $x^{(i)}$ and $x^{(j)}$ end up in the same terminal node after the tree run on T_k.

Example 1. *Suppose* $\mathfrak{D} = \{x^{(1)}, \cdots, x^{(5)}\}$. *Assume*

$$\nu_k(x^{(1)}) = \nu_k(x^{(2)}) \neq \nu_k(x^{(3)}) = \nu_k(x^{(4)}) = \nu_k(x^{(5)}).$$

Then the $Prox_k$ *can be expressed in matrix form as:*

$$Prox_k = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The **proximity** is defined as the average of $Prox(i, j)$ as:

$$Prox(i,j) = \frac{1}{B} \sum_{k=1}^{B} Prox^{(k)}(i,j).$$

Note that $Prox(i, i) = 1$ and $0 \leq Prox(i, j) \leq 1$.

▶ How to get a metric out of proximity

We now describe how to get a metric out of proximity.

(1) Define $S(i,j) = 1 - Prox(i,j)$, so $S(i,i) = 0$.

(2) If $Prox(i,j) = 1$ for some $i \neq j$. Eliminate one of x_i or x_j from the data or simply redefine $S(i,j) \leftarrow S(i,j) + \epsilon$, for some predefined small number ϵ.

(3) Repeat the above process until no pair with proximity 1 is left.

Now S is a reasonable distance between any two data. Breiman suggests $1 - Prox(i,j)$ is more like squared Euclidean distance. If so, one may define $S(i,j) = \sqrt{1 - Prox(i,j)}$ instead. Defined either way, S does not necessarily satisfy the triangle inequality. Say, if we let $S(1,2) = 0.1$, $S(2,3) = 0.2$, and $S(1,3) = 0.7$, then we see $S(1,3) > S(1,2) + S(2,3)$ violates the triangle inequality.

Definition. *A path between two points a and b is a succession of points linking a and b. Let $\gamma = (a_0, a_1, \cdots a_m)$ be a path from a_0 and a_m. Define*

$$S(\gamma) = S(a_0, a_1) + \cdots + S(a_{m-1}, a_m).$$

Define

$$d(x^{(i)}, x^{(j)}) = \inf_r \{S(\gamma) : \gamma \text{ is a path for } x^{(i)} \text{ to } x^{(j)}\}.$$

Check that

(1) $d(x^{(i)}, x^{(j)}) \geq 0$, and $d(x^{(i)}, x^{(j)}) = 0$ if and only if $x^{(i)} = x^{(j)}$

(2) $d(x^{(i)}, x^{(j)}) = d(x^{(j)}, x^{(i)})$

(3) $d(x^{(i)}, x^{(j)}) \leq d(x^{(i)}, x^{(k)}) + d(x^{(k)}, x^{(j)})$

Remark. *The above definition certainly defines a distance. But this tautological definition has such a high computational complexity that it is not very useful in practice.*

3.4 Missing value imputation

Real datasets are not perfect. They usually comes with many missing values in the input and sometimes even in the output. Since data is valuable it is not a good idea to throw away data points with missing values. To do so, one has to devise a way to fill in the missing ones and this process is usually called missing value imputation.

3.4.1 Missing input value imputation for the training set

▶ Mean/mode replacement (rough fill)

This is a fast and easy way to replace missing values. There are two cases.

(i) If x_j is numeric, then take as the imputed value the mean or median of the non-missing values of the j-th feature.

(ii) If x_j is categorical, then take the most frequent value (majority vote).

▶ Proximity-based imputation

This is a more elaborate process that yields better results. The procedure goes as follows.

(i) Do a rough fill (mean/mode imputation) for missing input values

(ii) Construct a random forest with the imputed input data and compute the proximity using this random forest

(iii) Let $x_{ij} = x_j^{(i)}$ be a missing value in the original dataset (before any imputation).

Case A: x_{ij} is numerical
Take the weighted average of the non-missing values weighted by the proximity i.e.,

$$x_{ij} = \frac{\sum_{k \in \mathfrak{N}_j} Prox(i,k) x_{kj}}{\sum_{k \in \mathfrak{N}_j} Prox(i,k)},$$

where $\mathfrak{N}_j = \{k \ : \ x_{kj} \text{ is non-missing}\}$ is the set of input data indices whose j-th feature value is non-missing.

Case B: x_{ij} is categorical
Replace x_{ij} with the most frequent value where the frequency is calculated with the proximity as weights

$$x_{ij} = \underset{\ell}{\operatorname{argmax}} \Big\{ \sum_{k \in \mathfrak{N}_j} Prox(i,k) \mathbb{I}(x_{kj} = \ell) \Big\}.$$

(iv) Repeat (*ii*) (*iii*) several times (typically $4 \sim 6$ times).

Remark. When the output value (label) is missing in the data of the training set, one can impute it in a way similar to what was done for missing labels in the test set. However, it often entails many complications, therefore it is better not to impute output values, if applicable.

3.4.2 Missing value imputation for the test set

It is in general good to use non-missing data only for the testing set whenever possible, because in testing, unlike training, untainted honest result is very important. If, however, imputation must be done, there two cases to consider.

(i) Case: label exists
Run down the test data point with missing output value through the random forest learned with the training set and compute the proximity of that test data point to the training data points. Then use the imputation method (iii) above using the training set.

(ii) Case: label does not exist
For classification problem, the label can be imputed as follows. Suppose there are K classes. For the test data point with missing label, create K data points with the same input but with K different output labels. Run these augmented test data points down the random forest and the label with the most vote is chosen as the imputed label.

3.5 Outliers

An outlier in a classification problem is a data point that is far away from the rest of the data with the same output label. To measure it, for the i-th data point, define

$$\bar{P}(i) = \sum_{k \in \mathfrak{L}(i)} Prox^2(i,k),$$

where $\mathfrak{L}(i) = \{k : y^{(k)} = y^{(i)}\}$ is the set of indices whose output label is the same as that of $y^{(i)}$.

The raw outlier measure $mo(i)$for the i-th data point is defined as

$$mo(i) = N/\bar{P}(i).$$

This measure is big if the average proximity is small, i.e. the i-th data point is far away from other data points with the same label. Let ℓ be an output label. Look at the set of data with the same output label ℓ, and compute the median of these raw outlier measures and their absolute deviation from the median. Subtract the median from each raw measure, and divide by the absolute deviation to arrive at the final outlier measure. One can then use this outlier measure to discover outliers.

3.6 Unsupervised learning

Unsupervised learning by definition deals with data that has no y-part, i.e. no output labels. Namely the dataset is of the form $\mathfrak{D} = \{x^{(i)}\}_{i=1}^N$. One of the goals of unsupervised learning is to discover how features (variables) are correlated to each other, i.e. how they are dependent on each other. So it is a good idea to compare the given dataset with one whose features are uncorrelated to each other.

To do that, first assign label 1 to every data point in the original dataset $\mathfrak{D}$ so that $x^{(i)}$ becomes $(x^{(i)}, 1)$, for all $i = 1, \cdots, N$. Call its collection as

$$\mathfrak{D}_1 = \{(x^{(i)}, 1)\}_{i=1}^N.$$

Now create another dataset $\mathfrak{D}_2$ as follows. For each feature j, randomly select x'_j from the set of possible values the j-th feature can take and repeating for $j = 1, \cdots, d$, form a vector $x' = (x'_1, \cdots, x'_d)$. In this vector x', any dependency or correlation among features is destroyed. Assign label 2 to this vector. Do this N times to form a new dataset

$$\mathfrak{D}_2 = \{(x'^{(i)}, 2)\}_{i=1}^N.$$

Now merge these two datasets to form a new bigger dataset

$$\mathfrak{D} = \mathfrak{D}_1 \cup \mathfrak{D}_2.$$

With this new dataset $\mathfrak{D}$, we can do supervised learning using a random forest. Namely, do the usual things like construction of random forest and calculate the OOB error estimate. If the OOB error rate is big, say 40%, this method basically fails and there is not much to say. If, on the other hand, this error rate is low, the dependency structure of class 1 has some meaning, and one can do with it all sorts of things like missing value imputation and so on. Most importantly one can compute the proximity and with it one can do many things like clustering, dimension reduction and so on.

3.7 Balancing prediction error

In some data sets, some labels occur quite frequenly while other labels occur rather infrequently. For instance, in equipment failure data, since equipments don't fail that often, most of the data points should inevitably show no-failure and the failure cases must show up only rarely. In this kind of highly unbalanced data, the prediction error imbalance between classes may become a very important issue. For instance, suppose we come up with a silly predictor declaring no-failure prediction for all equipments. Since most equipment data has no-failure outcome anyway, this silly predictor must have very low error rate. (It has error only for those rare failure cases.) But this is certainly a very unsatisfactory result, as it has a 100% error rate for the cases we are interested in, i.e. the failure cases.

A similar behavior may happen for random forests, or for that matter, for any machine learning algorithm. The reason is that random forests, in trying to minimize the overall error rate, will try to keep the error rate low on the larger class (no-failure cases in our equipment failure example) while letting the smaller class (failure cases in our equipment failure example) have a larger error rate. This kind of imbalance must be detected by looking at the error rates of the individual classes.

A way of rectifying the imbalance is to use a different weight for each class. To illustrate, let us look at the node split example. Suppose there are two classes: class 1 and class 2; and class 1 is abundant and class 2 rare. Say, suppose there are 70 class 1 cases and 4 class 2 cases. Now let us assign $1 : w$ weight to class 1 over class 2 (for example, $w = 10$). According to this weighting scheme, the probability distribution changes as follows:

(i) unweighted (equally weighted) probability

$$\text{probability of class 1} = \frac{70}{70+4}$$

$$\text{probability of class 2} = \frac{4}{70+4}$$

(ii) $1 : w$ weighted probability

$$\text{probability of class 1} = \frac{70}{70+4*w}$$

$$\text{probability of class 2} = \frac{4 * w}{70 + 4 * w}$$

In other words, when it comes to calculating the probability, hence the impurity computation, one class 2 element is counted as w elements (w may not be a whole number). However, when counting the actual number of elements for the purpose of checking if the number of elements in a node is less than $N_{\min}$, this weighting scheme does not apply, i.e. each class 2 element is still counted as one.

Check that the Gini impurity of this node using the unweighted probability is

$$p(1-p) = \frac{70}{74} \times \frac{4}{74} \approx 0.05.$$

On the other hand, the Gini impurity using the weighted probability is

$$p(1-p) = \frac{70}{110} \times \frac{40}{110} \approx 0.23,$$

which indicates that in the second case the impurity is deemed much higher.

In general, when a split is done using the usual unweighted (equally weighted) probability, it may basically ignore the class 2 cases as its total contribution to the error is negligible compared to that of the class 1 cases. If, on the other hand, the weight is assigned as in the example above, the random forest is forced to go out of its way to try to keep as many class 2 cases in the same child node as possible after the split. This way, the error rate for class 2 will decrease, but the overall error rate may increase somewhat. What the correct weight has to be is still an open problem. One usually resorts to a trial-and-error approach of tinkering with various weights until a satisfactory error rate profile emerges.

References

[1] Breiman, L., *Random Forests*, Machine Learning 45, 5-32 (2001)

[2] Breiman, L. and Cutler, A., *Random Forests*, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm