Machine Learning, Lecture 12: Random Forests

S. Nõmm

[1] Department of Computer Science, Tallinn University of Technology

April 21, 2016

Introduction

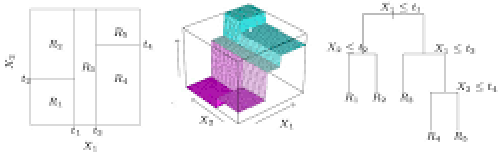
- Regression trees (reminder).
- Bootstrapping, Bagging and Boosting.
- Random forests

Tree based methods

- Tree-based methods partition the feature space into a set of rectangles and fit a simple model (like a constant in each one).
- Let Y be continuous response (dependent) variable and X_1, X_2 denote continues independent variables (taking values between 0 and 1).
- In order to simplify to recursive binary partitions.

Reference "The elements of statistical learning" by hastie Tibishiriani and Friedman. Springer 2011.

- An example (in regression case):



Tree based methods

1. Split at $X_1 = t_1$.
2. Split the region $X_1 \leq t_1$ at $X_2 = t_2$ and the region $X_1 > t_1$ at $X_1 = t_3$.
3. Split the region $X_1 > t_3$ at $X_2 = t_4$.

- Result: Partition in to R_1, R_2, R_3, R_4, R_5.
- The corresponding regression model predicts Y with the constant c_m in region R_m.

$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}$$

How to grow

- Let us suppose that the data consists of N p-dimensional inputs and a response variables. Also we have a partition into M regions and response is modeled as a constant c_m in each region.

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

- Adopt sum of the squares as the minimization criterion. Then best $\hat{c_m}$

$$\hat{c_m} = \overline{(y_i | x_i \in R_m)}.$$

- Greedy algorithm is used to find best binary partition. For the variable j and splitting point s define the pair of half planes.

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j > s\}.$$

How to grow

- Greedy algorithm is used to find best binary partition. For the variable j and splitting point s define the pair of half planes.

$$R_1(j,s) = \{X|X_j \leq s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j > s\}.$$

- Splitting variable j and splitting point s should solve:

$$\min_{j,s}\left[\min_{c_1} \sum_{x_1 \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2\right]$$

- Inner minimization is solved by

$$\hat{c}_1 = \overline{(y_i|x_i \in R_1(j,s))} \quad \text{and} \quad \hat{c}_2 = \overline{(y_i|x_i \in R_2(j,s))}.$$

- Once the best split is found, data is partitioned into the two regions, then the procedure is repeated on all the resulting regions.

How to grow

- Tree size is the tuning parameter governing complexity of the model. (Very large tree might overfit the data, very small - might not capture the important structure).
- Possible ways to choose tree size:
  - Split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold.
  - Grow the large tree with the stopping criteria minimum node size is reached. Then apply cost-complexity pruning.

Cost complexity pruning

- Define as subtree $T \subset T_0$, whereas T_0 denotes "large tree" grown with stopping criteria of minimum node size reached.
- The idea is to collapse a number of its internal (non-terminal) nodes.
- Let node m represent the region R_m and let $|T|$ denote the number of terminal nodes in T

$$\begin{aligned} N_m &= \#\{x_i \in R_m\} \\ \hat{c}_m &= \frac{1}{N_m} \sum_{x_i \in R_m} y_i, \\ Q_m(T) &= \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2, \end{aligned}$$

  $Q_m(T)$ is referred as *node impurity measure*.
- Cost-complexity criterion:

$$C_\alpha = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|.$$

Cost complexity pruning

- The idea is to find, for each α the subtree $T_{alpha} \supset T_0$ to minimize $C_\alpha(T)$.
- $\alpha > 0$ is the tuning parameter, whereas large values of α result in a smaller trees and vice verse.

Classification trees

- Node impurity measure $Q_m(T)$ does not suitable for classification.
- In a node m representing the region R_m with N_m observations let:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{I}(y_i = k).$$

the proportion of class k observations in node m.

Bootstrap: assessing the accuracy of the parameter estimate

- The bootstrap is a general tool for assessing statistical accuracy.
- Let us suppose that model M fit to a set of training data. Let $\mathbf{Z} = (z_1, \ldots, z_N)$ denote the training set, where $z_i = (y_i, z_i)$.
- The basic idea is to draw B (for example $B = 100$) the datasets with replacements from the training data. From the bootstrap sampling we may estimate any aspect of the distribution $S(\mathbf{Z})$.
- Variance:

$$\widehat{\mathrm{Var}}\big[S(\mathbf{Z})\big] = \frac{1}{B-1}\sum_{b=1}^{B}\big(S(\mathbf{Z}^{*b}) - \bar{S}^{*}\big)^2.$$

- Estimate prediction error:

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L\big(y_i, \hat{f}^{*b}(x_i)\big).$$

Bagging: improving the estimate or prediction

.

- As before $\mathbf{Z}$ is the training data.
- Let $\hat{f}(x)$ is the prediction at input x.
- Bootstrap aggregation or *bagging* averages this prediction over a collection of bootstrap samples.
- The *bagging* estimate is defined by:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$

- is the a Monte Carlo estimate of the true bagging estimate, approaching is as $B \to \infty$.
- The bagging estimate will differ from the original one only when letter is nonlinear or adaptive function.
- The case of Trees. Each bootstrapped tree will typically involve different features compared to the original ones. And might have different number of terminal nodes.The bagged estimate is the average prediction at x from this B trees.

Boosting

.

- One of the most powerful techniques introduced during last twenty years.
- Combine the outputs of many "weak" classifiers to produce powerful "committee".
- The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of data. thereby producing a sequence of weak classifiers. $G_m(x), m = 1, \ldots, M$.
- The predictions of all of them are then combined through a weighted majority vote to produce final prediction:

$$G(x) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right).$$

Ada Boost. M1

.

1. Initialize the observation weights $w_i = 1/N$, $i = 1, \dots, N$.
2. For $m = 1 : M$
   - (a) Fit classifier $G_m(x)$ to the training data using weights w_i.
   - (b) Compute:

$$\mathrm{err}_m = \frac{\displaystyle\sum_{i=1}^{m} w_i \mathbb{I}(y_i \neq G_m(x_i))}{\displaystyle\sum_{i=1}^{N} w_i}$$

   - (c) Compute $a_m = \log((1 - \mathrm{err}/\mathrm{err}_m$.
   - (d) Set $w_i \leftarrow w_i \exp[\alpha_m \mathbb{I}(y_i \neq G_m(x_i))]$, $i = 1, \dots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{i=1}^{M} \alpha_m G_m(x)\right]$.

Random Forests

.

- Bagging works well for high-variance low bias procedures.
- Random forests is a modification of bagging that builds a large collection of de-correlated trees, and then averages them.
- Random forest are similar to boosting in terms of training and tuning.

Random Forests

.

1. For $b = 1 : B$
   - (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   - (b) Grow a random forest tree T_b to the bootstrapped data, by recursively repeating followng steps for the each terminal node of the tree, until minimum node size $n_{\min}$ is reached.
     - i. Select $variables$ at random from the p variables.
     - ii.pick the best variable/split point among the m.
     - iii. Split the node into two daughter nodes.
2. Output the ensemble trees $\{T_b\}_1^B$

To make a prediction a a new point x:
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.
Classification: Let $\hat{C}_{\text{rf}}(x)$ be the class prediction of the bth random forest tree. Then $\hat{C}_b(x) = \text{majority} \quad \text{vote}\{\hat{C}_b(x)\}_1^B$.

Random Forests

.

- For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
- For classification, the default value for m is $p/3$ and the minimum node size is five.
- Increasing B does not cause the random forest sequence to overfit.