Random forest

From Wikipedia, the free encyclopedia

Random forests is a notion of the general technique of random decision forests[1][2] that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.[3]:587–588

The algorithm for inducing Breiman's random forest was developed by Leo Breiman[4] and Adele Cutler,[5] and "Random Forests" is their trademark.[6] The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho[1][2] and Amit and Geman [7] in order to construct a collection of decision trees with controlled variance.

The selection of a random subset of features is an example of the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.[8][9][10]

Contents

History

The general method of random decision forests was first proposed by Ho in 1995,[1] who established that forests of trees splitting with oblique hyperplanes, if randomly restricted to be sensitive to only selected feature dimensions, can gain accuracy as they grow without suffering from overtraining. A subsequent work along the same lines [2] concluded that other splitting methods, as long as they are randomly forced to be insensitive to some feature dimensions, behave similarly. Note that this observation of a more complex classifier (a larger forest) getting more accurate nearly monotonically is in sharp contrast to the common belief that the complexity of a classifier can only grow to a certain level before accuracy being hurt by overfitting. The explanation of the forest method's resistance to overtraining can be found in Kleinberg's theory of stochastic discrimination.[8][9][10]

The early development of Breiman's notion of random forests was influenced by the work of Amit and Geman[7] who introduced the idea of searching over a random subset of the available decisions when splitting a node, in the context of growing a single tree. The idea of random subspace selection from Ho[2] was also influential in the design of random forests. In this method a forest of trees is grown, and variation among the trees is introduced by projecting the training data into a randomly chosen subspace before fitting each tree. Finally, the idea of randomized node optimization, where the decision at each node is selected by a randomized procedure, rather than a deterministic optimization was first introduced by Dietterich.[11]

The introduction of random forests proper was first made in a paper by Leo Breiman.[4] This paper describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and bagging. In addition, this paper combines several ingredients, some previously known and some novel, which form the basis of the modern practice of random forests, in particular:

1. Using out-of-bag error as an estimate of the generalization error.
2. Measuring variable importance through permutation.

The report also offers the first theoretical result for random forests in the form of a bound on the generalization error which depends on the strength of the trees in the forest and their correlation.

Algorithm

Preliminaries: decision tree learning

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie *et al.*, because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate.[3]:352

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, because they have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.[3]:587–588 This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

Tree bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a decision or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x')$$

or by taking the majority vote in the case of decision trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the *out-of-bag error*: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.[12] The training and test error tend to level off after some number of trees have been fit.

From bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho.[13]

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split.[3]:592 For regression problems the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default.[3]:592

Extensions

Adding one further step of randomization yields *extremely randomized trees*, or ExtraTrees. These are trained using bagging and the random subspace method, like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Instead of computing the locally *optimal* feature/split combination (based on, e.g., information gain or the Gini impurity), for each feature under consideration, a random value is selected for the split. This value is selected from the feature's empirical range (in the tree's training set, i.e., the bootstrap sample)[14]

Properties

Variable importance

Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way. The following technique was described in Breiman's original paper[4] and is implemented in the R package randomForest.[5]

The first step in measuring the variable importance in a data set $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).

To measure the importance of the j-th feature after training, the values of the j-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the j-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.

Features which produce large values for this score are ranked as more important than features which produce small values.

This method of determining variable importance has some drawbacks. For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Methods such as partial permutations[15][16] and growing unbiased trees[17] can be used to solve the problem. If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favored over larger groups.[18]

Relationship to nearest neighbors

A relationship between random forests and the k-nearest neighbor algorithm (k-NN) was pointed out by Lin and Jeon in 2002.[19] It turns out that both can be viewed as so-called *weighted neighborhoods schemes*. These are models built from a training set $\{(x_i, y_i)\}_{i=1}^n$ that make predictions $\hat{y}$ for new points x' by looking at the "neighborhood" of the point, formalized by a weight function W:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i.$$

Here, $W(x_i, x')$ is the non-negative weight of the i'th training point relative to the new point x'. For any particular x', the weights must sum to one. Weight functions are given as follows:

- In k-NN, the weights are $W(x_i, x') = \frac{1}{k}$ if x_i is one of the k points closest to x', and zero otherwise.
- In a tree, $W(x_i, x') = \frac{1}{k'}$ if x_i is one of the k' points in the same leaf as x', and zero otherwise.

Since a forest averages the predictions of a set of m trees with individual weight functions W_j, its predictions are

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right)\, y_i.$$

This shows that the whole forest is again a weighted neighborhood scheme, with weights that average those of the individual trees. The neighbors of x' in this interpretation are the points x_i which fall in the same leaf as x' in at least one tree of the forest. In this way, the neighborhood of x' depends in a complex way on the structure of the trees, and thus on the structure of the training set. Lin and Jeon show that the shape of the neighborhood used by a random forest adapts to the local importance of each feature.[19]

Unsupervised learning with random forests

As part of their construction, RF predictors naturally lead to a dissimilarity measure between the observations. One can also define an RF dissimilarity measure between unlabeled data: the idea is to construct an RF predictor that distinguishes the “observed” data from suitably generated synthetic data.[4][20] The observed data are the original unlabeled data and the synthetic data are drawn from a reference distribution. An RF dissimilarity can be attractive because it handles mixed variable types well, is invariant to monotonic transformations of the input variables, and is robust to outlying observations. The RF dissimilarity easily deals with a large number of semi-continuous variables due to its intrinsic variable selection; for example, the "Addcl 1" RF dissimilarity weighs the contribution of each variable according to how dependent it is on other variables. The RF dissimilarity has been used in a variety of applications, e.g. to find clusters of patients based on tissue marker data.[21]

Variants

Instead of decision trees, linear models have been proposed and evaluated as base estimators in random forests, in particular multinomial logistic regression and naive Bayes classifiers.[22][23]

Libraries

Python (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) -- from scikit-learn library for Python

R (https://cran.r-project.org/web/packages/randomForest/index.html)-- randomForest package in R

Matlab (http://in.mathworks.com/help/stats/treebagger.html?refresh=true)-- from Statistics and Machine Learning Toolbox in Matlab

Spark (http://spark.apache.org/docs/latest/mllib-ensembles.html)-- Spark MLib

KNIME (https://www.knime.org/files/nodedetails/_labs_treeensemble_Tree_Ensemble_Learner.html) -- KNIME Tree Ensemble

See also

- Decision tree learning
- Gradient boosting
- Randomized algorithm
- Bootstrap aggregating (bagging)
- Ensemble learning
- Boosting
- Non-parametric statistics
- Kernel random forest

References

1. Ho, Tin Kam (1995). *Random Decision Forests* (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.
2. Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** (8): 832–844. doi:10.1109/34.709601.
3. Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.
4. Breiman, Leo (2001). "Random Forests". *Machine Learning* **45** (1): 5–32. doi:10.1023/A:1010933404324.
5. Liaw, Andy (16 October 2012). "Documentation for R package randomForest" (PDF). Retrieved 15 March 2013.
6. U.S. trademark registration number 3185828, registered 2006/12/19.
7. Amit, Yali; Geman, Donald (1997). "Shape quantization and recognition with randomized trees" (PDF). *Neural Computation* **9** (7): 1545–1588. doi:10.1162/neco.1997.9.7.1545.
8. Kleinberg, Eugene (1996). "An Overtraining-Resistant Stochastic Modeling Method for Pattern Recognition" (PDF). *Annals of Statistics* **24** (6): 2319–2349. doi:10.1214/aos/1032181157. MR 1425956.
9. Kleinberg, Eugene (2000). "On the Algorithmic Implementation of Stochastic Discrimination" (PDF). *IEEE Transactions on PAMI* **22** (5).
10. Kleinberg, Eugine. "Stochastic Discrimination and its Implementation".
11. Dietterich, Thomas (2000). "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization". *Machine Learning*: 139–157.
12. Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Springer. pp. 316–321.
13. Ho, Tin Kam (2002). "A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors" (PDF). *Pattern Analysis and Applications*: 102–112.
14. Geurts, P.; Ernst, D.; Wehenkel, L. (2006). "Extremely randomized trees" (PDF). *Machine Learning* **63**: 3. doi:10.1007/s10994-006-6226-1.
15. Deng,H.; Runger, G.; Tuv, E. (2011). *Bias of importance measures for multi-valued attributes and solutions*. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). pp. 293–300.
16. Altmann A, Tolosi L, Sander O, Lengauer T (2010). "Permutation importance:a corrected feature importance measure". *Bioinformatics*. doi:10.1093/bioinformatics/btq134.
17. Strobl, Carolin; Boulesteix, Anne-Laure; Augustin, Thomas (2007). "Unbiased split selection for classification trees based on the Gini index" (PDF). *Computational Statistics & Data Analysis*: 483–501.
18. Tolosi L, Lengauer T (2011). "Classification with correlated features: unreliability of feature ranking and solutions.". *Bioinformatics*. doi:10.1093/bioinformatics/btr300.
19. Lin, Yi; Jeon, Yongho (2002). *Random forests and adaptive nearest neighbors* (Technical report). Technical Report No. 1055. University of Wisconsin.
20. Shi, T., Horvath, S. (2006). "Unsupervised Learning with Random Forest Predictors". *Journal of Computational and Graphical Statistics* **15** (1): 118–138. doi:10.1198/106186006X94072.
21. Shi, T., Seligson D., Belldegrun AS., Palotie A, Horvath, S. (2005). "Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma". *Modern Pathology* **18** (4): 547–557. doi:10.1038/modpathol.3800322. PMID 15529185.

22. Prinzie, A., Van den Poel, D. (2008). "Random Forests for multiclass classification: Random MultiNomial Logit". *Expert Systems with Applications* **34** (3): 1721–1732. doi:10.1016/j.eswa.2007.01.029.
23. Prinzie, A., Van den Poel, D. (2007). Random Multiclass Classification: Generalizing Random Forests to Random MNL and Random NB, Dexa 2007, Lecture Notes in Computer Science, 4653, 349–358. (http://dx.doi.org/10.1007/978-3-540-74469-6_35)

External links

- Random Forests classifier description (https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm) (Site of Leo Breiman)
- Liaw, Andy & Wiener, Matthew "Classification and Regression by randomForest" R News (2002) Vol. 2/3 p. 18 (http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf) (Discussion of the use of the random forest package for R)
- Prinzie, Anita; Poel, Dirk (2007). "Random Multiclass Classification: Generalizing Random Forests to Random MNL and Random NB" (PDF). *Database and Expert Systems Applications*. Lecture Notes in Computer Science. p. 349. doi:10.1007/978-3-540-74469-6_35. ISBN 978-3-540-74467-2.
- C# implementation (http://semanticquery.com/archive/semanticsearchart/researchRF.html) of random forest algorithm for categorization of text documents supporting reading of documents, making dictionaries, filtering stop words, stemming, counting words, making document-term matrix and its usage for building random forest and further categorization.
- A python implementation (http://scikit-learn.org/stable/modules/ensemble.html) of the random forest algorithm working in regression, classification with multi-output support.
- AwesomeRandomForest - A list of random forest resources (http://jiwonkim.org/awesome-random-forest)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=700408350"

Categories: Classification algorithms | Ensemble learning | Decision trees

---