Machine Learning Basic Understanding

By- Sudhanshu Saxena

Supervised Learning

ENSEMBLE Learning

What we are going to learn

- Basic Concepts
- Boosting
- Bagging
- Combination of Methods

Basic Concept

One major task of machine learning, pattern recognition and data mining is to construct good models from data sets. A “data set” generally consists of feature vectors, where each feature vector is a description of an object by using a set of features.

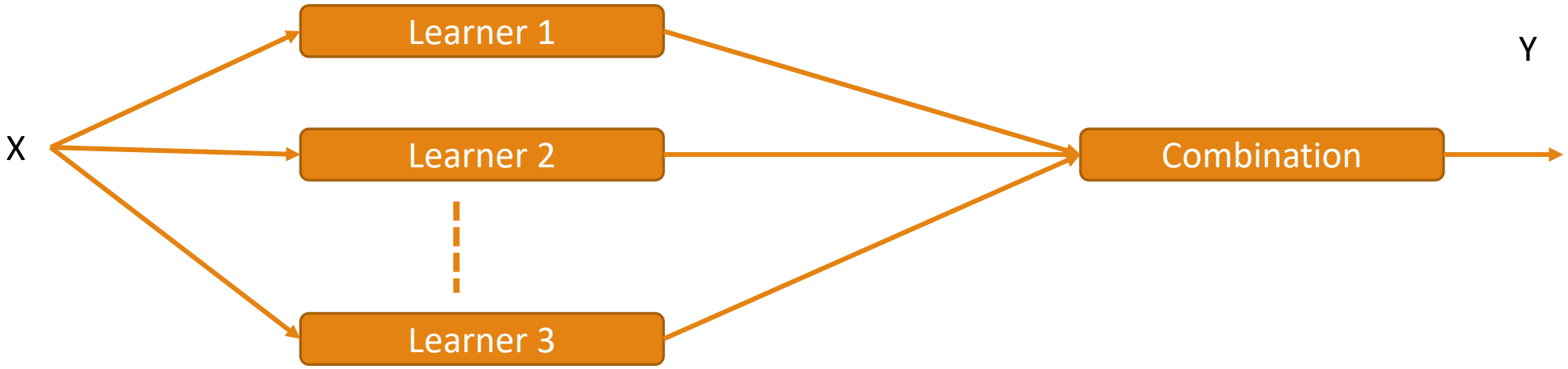
Ensemble methods train multiple learners to solve the same problem. In contrast to ordinary learning approaches which try to construct one learner from training data, ensemble methods try to construct a set of learners and combine them. Ensemble learning is also called committee-based learning, or learning multiple classifier systems

Basic Concept

An ensemble contains a number of learners called base learners. Base learners are usually generated from training data by a base learning algorithm which can be decision tree, neural network or other kinds of learning algorithms. Most ensemble methods use a single base learning algorithm to produce homogeneous base learners, i.e., learners of the same type, leading to homogeneous ensembles, but there are also some methods which use multiple learning algorithms to produce heterogeneous learners, i.e., learners of different types, leading to heterogeneous ensembles. In the latter case there is no single base learning algorithm and thus, some people prefer calling the learners individual learners or component learners to base learners

Basic Concept

The generalization ability of an ensemble is often much stronger than that of base learners. Actually, ensemble methods are appealing mainly because they are able to boost weak learners which are even just slightly better than random guess to strong learners which can make very accurate predictions. So, base learners are also referred to as weak learners.



Bootstrapping

In machine learning, the bootstrap method refers to random sampling with replacement. This sample is referred to as a resample. This allows the model or algorithm to get a better understanding of the various biases, variances and features that exist in the resample. Taking a sample of the data allows the resample to contain different characteristics then it might have contained as a whole. Demonstrated in figure 1 where each sample population has different pieces, and none are identical. This would then affect the overall mean, standard deviation and other descriptive metrics of a data set. In turn, it can develop more robust models.

Bootstrapping

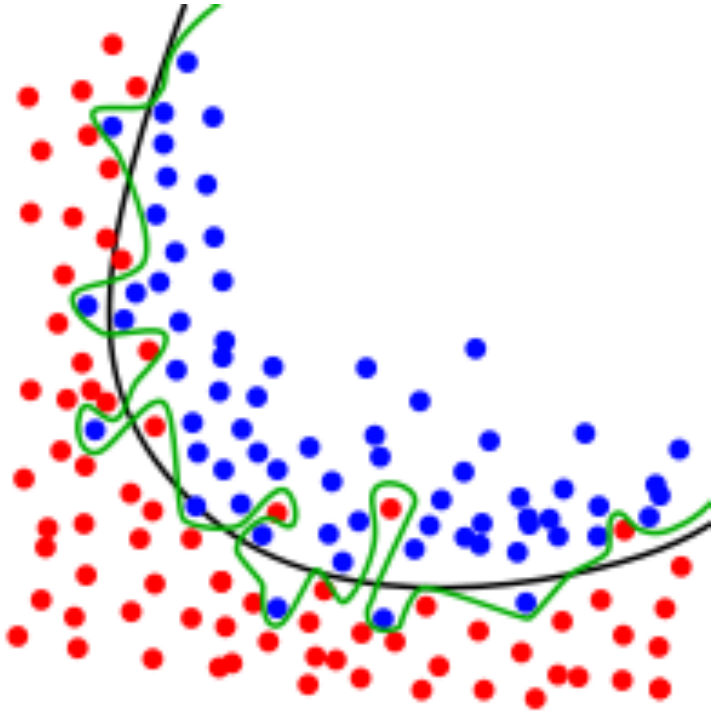
Bootstrapping is also great for small size data sets that can have a tendency to overfit. In fact, we recommended this to one company who was concerned because their data sets were far from “Big Data”. Bootstrapping can be a solution in this case because algorithms that utilize bootstrapping can be more robust and handle new data sets depending on the methodology chosen(boosting or bagging)

The reason to use the bootstrap method is because it can test the stability of a solution. By using multiple sample data sets and then testing multiple models, it can increase robustness. Perhaps one sample data set has a larger mean than another, or a different standard deviation. This might break a model that was overfit, and not tested using data sets with different variations.

Bagging

Bagging actually refers to (Bootstrap Aggregators). Most any paper or post that references using bagging algorithms will also reference Leo Breiman who wrote a paper in 1996 called “Bagging Predictors”.

Where Leo describes bagging as:
“Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor.”
What Bagging does is help reduce variance from models that are might be very accurate, but only on the data they were trained on. This is also known as overfitting.
Overfitting is when a function fits the data too well. Typically this is because the actual equation is much too complicated to take into account each data point and outlier.



Bagging

Another example of an algorithm that can overfit easily is a decision tree. The models that are developed using decision trees require very simple heuristics. Decision trees are composed of a set of if-else statements done in a specific order. Thus, if the data set is changed to a new data set that might have some bias or difference in spread of underlying features compared to the previous set. The model will fail to be as accurate. This is because the data will not fit the model as well(which is a backwards statement anyways).

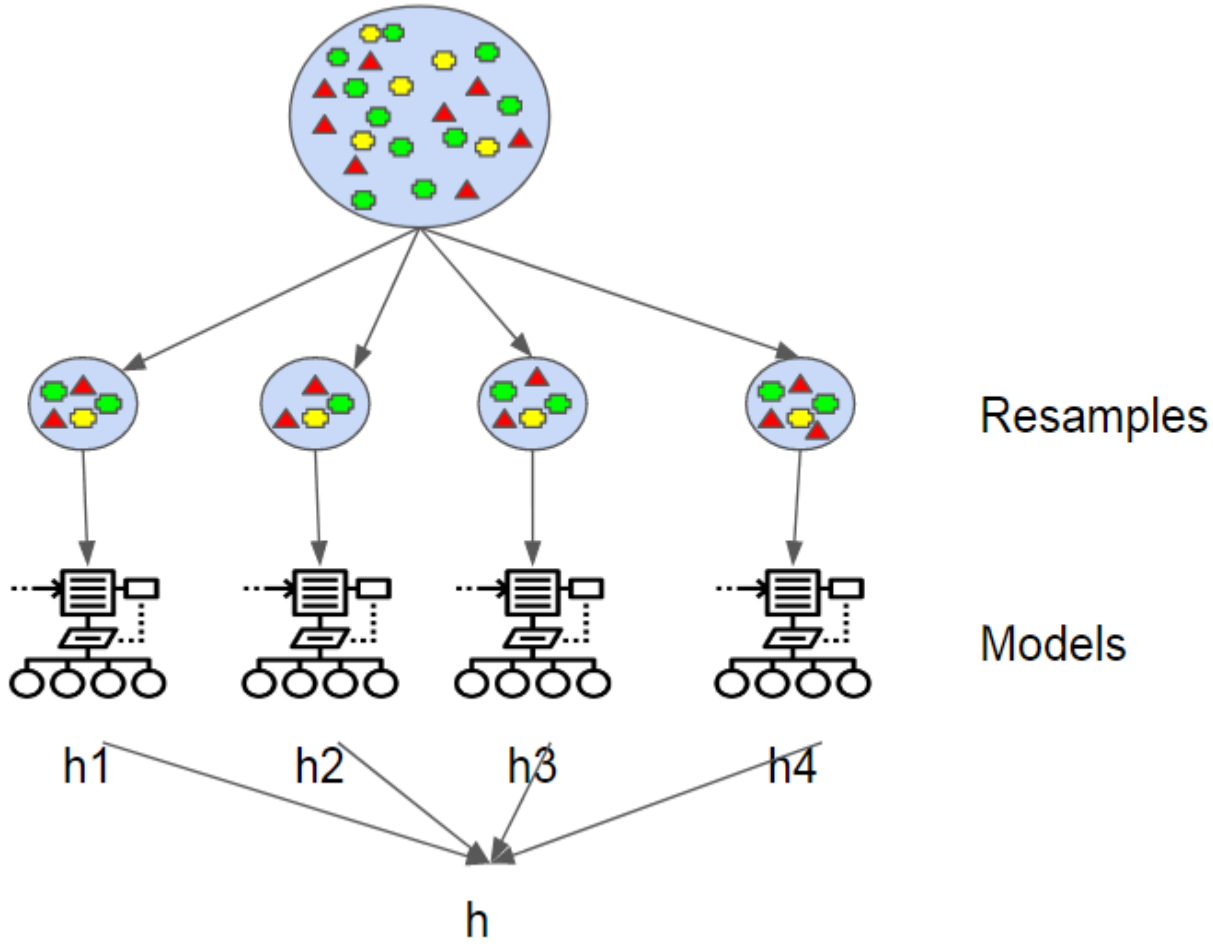
Bagging gets around this by creating it's own variance amongst the data by sampling and replacing data while it tests multiple hypothesis(models). In turn, this reduces the noise by utilizing multiple samples that would most likely be made up of data with various attributes(median, average, etc).

Bagging

Once each model has developed a hypothesis. The models use *voting for classification* or *averaging for regression*.

This is where the “Aggregating” in “Bootstrap Aggregating” comes into play. Each hypothesis has the same weight as all the others.

Essentially, all these models run at the same time, and vote on which hypothesis is the most accurate.
This helps to decrease variance i.e. reduce the overfit.



Boosting

The term boosting refers to a family of algorithms that are able to convert weak learners to strong learners. Intuitively, a weak learner is just slightly better than random guess, while a strong learner is very close to perfect performance. The birth of boosting algorithms originated from the answer to an interesting theoretical question posed by Kearns and Valiant. That is, whether two complexity classes, weakly learnable and strongly learnable problems, are equal. This question is of fundamental importance, since if the answer is positive, any weak learner is potentially able to be boosted to a strong learner, particularly if we note that in real practice it is generally very easy to obtain weak learners but difficult to get strong learners. Schapire proved that the answer is positive, and the proof is a construction, i.e., boosting

Boosting

The general boosting procedure is quite simple. Suppose the weak learner will work on any data distribution it is given, and take the binary classification task as an example; that is, we are trying to classify instances as positive and negative. The training instances in space X are drawn i.i.d. from distribution D, and the ground-truth function is f. Suppose the space X is composed of three parts X1, X2 and X3, each takes 1/3 amount of the distribution, and a learner working by random guess has 50% classification error on this problem. We want to get an accurate (e.g., zero error) classifier on the problem, but we are unlucky and only have a weak classifier at hand, which only has correct classifications in spaces X1 and X2 and has wrong classifications in X3, thus has 1/3 classification error. Let's denote this weak classifier as h1. It is obvious that h1 is not desired.

The idea of boosting is to correct the mistakes made by h1. We can try to derive a new distribution D from D, which makes the mistakes of h1 more evident, e.g., it focuses more on the instances in X3. Then, we can train a classifier h2 from D . Again, suppose we are unlucky and h2 is also a weak classifier, which has correct classifications in X1 and X3 and has wrong classifications in X2. By combining h1 and h2 in an appropriate way (we will explain how to combine them in the next section), the combined classifier will have correct classifications in X1, and maybe some errors in X2 and X3.

Random Forest

What we will learn

1. What is the Random Forest algorithm?
2. How does it work? (Decision Tree, Random Forest)
3. What is the difference between Bagging and Random Forest?
4. Advantages and Disadvantages of Random Forest
5. Solving a Problem
6. Parameter Tuning in Random Forest

What is Overfitting?

Explaining your training data instead of finding patterns that generalize is what overfitting is. In other words, your model learns the training data by heart instead of learning the patterns which prevent it from being able to generalized to the test data. It means your model fits well to training dataset but fails to the validation dataset.

What is the Random Forest algorithm?

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

How does it work? (Decision Tree, Random Forest)

Given a data frame (n x p), a tree stratifies or partitions the data based on rules (if-else). Yes, a tree creates rules. These rules divide the data set into distinct and non-overlapping regions. These rules are determined by a variable's contribution to the homogeneity or pureness of the resultant child nodes (X2,X3).

the variable X1 resulted in highest homogeneity in child nodes, hence it became the root node. A variable at root node is also seen as the most important variable in the data set.

How does it work? (Decision Tree, Random Forest)

But how is this homogeneity or pureness determined? In other words, how does the tree decide at which variable to split?

How does it work? (Decision Tree, Random Forest)

In **regression trees** (where the output is predicted using the mean of observations in the terminal nodes), the splitting decision is based on minimizing RSS. The variable which leads to the greatest possible reduction in RSS is chosen as the root node. The tree splitting takes a **top-down greedy** approach, also known as *recursive binary splitting*. We call it "greedy" because the algorithm cares to make the best split at the current step rather than saving a split for better results on future nodes.

How does it work? (Decision Tree, Random Forest)

In **classification trees** (where the output is predicted using mode of observations in the terminal nodes), the splitting decision is based on the following methods:

Gini Index – It's a measure of node purity. If the Gini index takes on a smaller value, it suggests that the node is pure. For a split to take place, the Gini index for a child node should be less than that for the parent node.

Entropy – Entropy is a measure of node impurity. For a binary class (a,b), the formula to calculate it is shown below. Entropy is maximum at p = 0.5. For p(X=a)=0.5 or p(X=b)=0.5 means, a new observation has a 50%-50% chance of getting classified in either classes. The entropy is minimum when the probability is 0 or 1.

How does it work? (Decision Tree, Random Forest)

In a nutshell, every tree attempts to create rules in such a way that the resultant terminal nodes could be as pure as possible. Higher the purity, lesser the uncertainty to make the decision.

But a decision tree suffers from high variance. “High Variance” means getting high prediction error on unseen data. We can overcome the variance problem by using more data for training. But since the data set available is limited to us, we can use resampling techniques like bagging and random forest to generate more data.

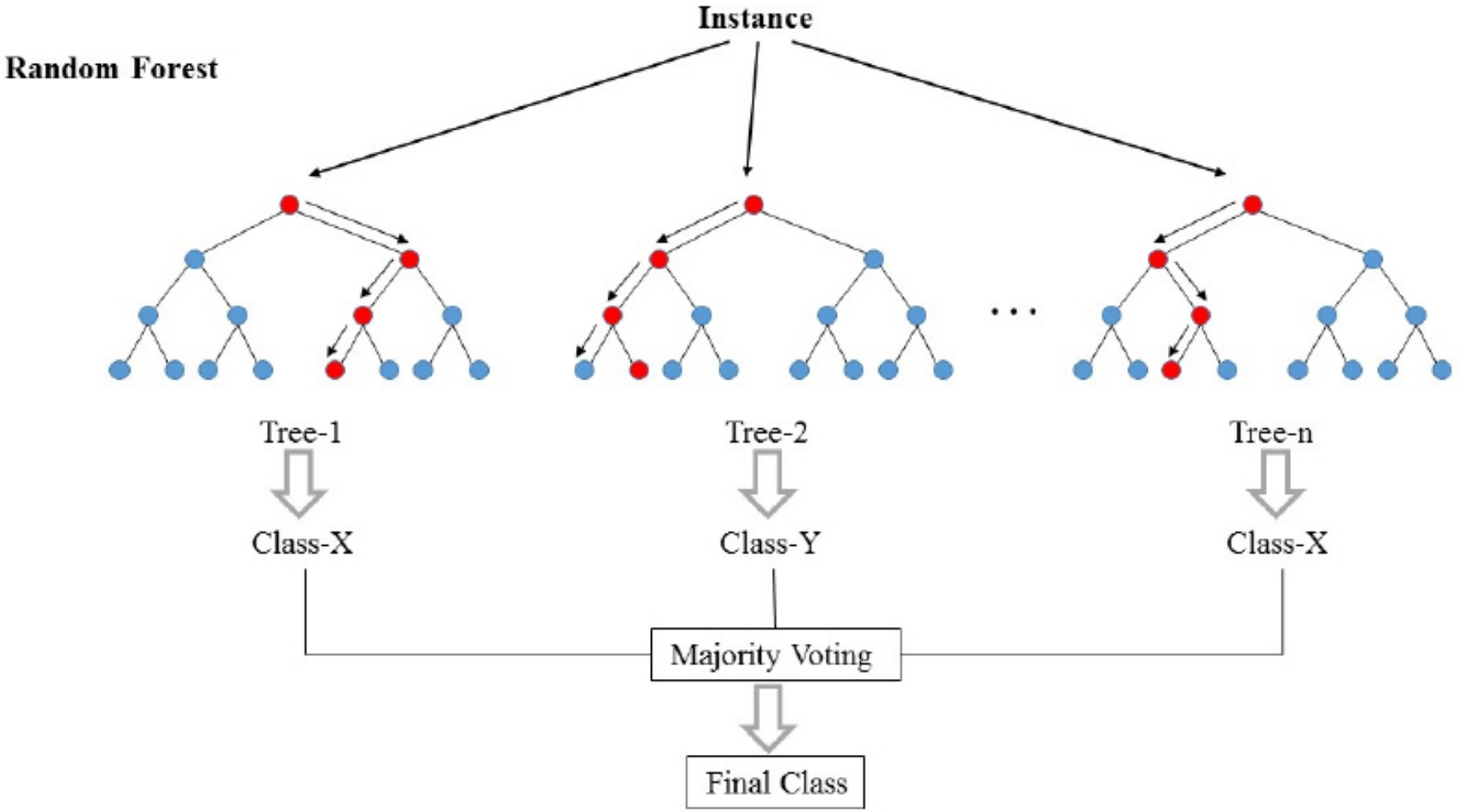
Random Forest

Bagging or bootstrap aggregation is a technique for reducing the variance of an estimated prediction function. Bagging seems to work especially well for high-variance, low-bias procedures, such as trees. For regression, we simply t the same regression tree many times to bootstrap- sampled versions of the training data, and average the result. For classification, a committee of trees each cast a vote for the predicted class.

Boosting was initially proposed as a committee method as well, although unlike bagging, the committee of weak learners evolves over time, and the members cast a weighted vote. Boosting appears to dominate bagging on most problems, and became the preferred choice.

Random forests (Breiman 2001) is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.

How Random Forest work



Advantages and Disadvantages:

An advantage of random forest is that it can be used for both regression and classification tasks and that it's easy to view the relative importance it assigns to the input features.

Random Forest is also considered as a very handy and easy to use algorithm, because it's default hyperparameters often produce a good prediction result. The number of hyperparameters is also not that high and they are straightforward to understand.

One of the big problems in machine learning is overfitting, but most of the time this won't happen that easy to a random forest classifier. That's because if there are enough trees in the forest, the classifier won't overfit the model.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

And of course Random Forest is a predictive modeling tool and not a descriptive tool. That means, if you are looking for a description of the relationships in your data, other approaches would be preferred.

Thank You!!!