Random Forests

Aaron Bobick
School of Interactive Computing
Administrivia

• Shray says the problem set is “close to done”

• Today chapter 15 of the Hastie book.

• Very few slides brought to you by Peter Orbanz with huge thanks to A. Criminisi...
Recall Bootstrap Method

• General tool for assessing statistical accuracy.
• Suppose we have a model to fit the training data:

\[ Z = \{(x_i, y_i), i = 1, \ldots, N\} \]

• The idea is to draw random samples with replacement of size \(N\) from the training data. This process is repeated \(B\) times to get \(B\) bootstrap datasets.
• Refit the model to each of the bootstrap datasets and examine the behavior of the fits over \(B\) replications.
Recall Bootstrap

The basic idea: Randomly draw datasets with replacement from the training data, each sample the same size as the original training set.
Boostrapping in regression

95% Conf

\[ \hat{\mu}(x) \pm 1.96 \cdot \hat{\text{se}}[\hat{\mu}(x)] \]

For N=200, 95% means 5th smallest and biggest
Recall Bagging

• Introduced by Breiman (Machine Learning, 1996).
• Acronym for ‘Bootstrap aggregation’.
• In regression, it averages the prediction over a collection of bootstrap samples, thus reducing the variance in prediction.
• For classification, a committee (or ensemble – later) of classifiers each cast a vote for the predicted class.
Recall Bagging for classification

• Let $\hat{G}$ be a classifier for a K-class response. Consider an underlying indicator vector function

$$\hat{f}(x) = (0, \ldots, 0, 1, 0, \ldots, 0)$$

the entry in the $i^{th}$ place is 1 if the prediction for $x$ is the $i^{th}$ class, such that

$$\hat{G}(x) = \arg \max_k \hat{f}(x).$$

• Then the bagged estimate $\hat{f}_{bag}(x) = (p_1, \ldots, p_K)$

where $p_k$ is the proportion of base classifiers predicting class $k$ at $x$ where $k = 1, \ldots, K$.

• Finally:

$$\hat{G}_{bag}(x) = \arg \max_k \hat{f}_{bag}(x).$$
Recall Bagging Trees

Notice the bootstrap trees are different than the original tree.
Good bagging

**FIGURE 8.10.** Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.
The key is independence

**FIGURE 8.11.** Simulated academy awards voting. 50 members vote in 10 categories, each with 4 nominations. For any category, only 15 voters have some knowledge, represented by their probability of selecting the “correct” candidate in that category (so \( P = 0.25 \) means they have no knowledge). For each category, the 15 experts are chosen at random from the 50. Results show the expected correct (based on 50 simulations) for the consensus, as well as for the individuals. The error bars indicate one standard deviation. We see, for example, that if the 15 informed for a category have a 50% chance of selecting the correct candidate, the consensus doubles the expected performance of an individual.
“The key is independence...”

• **Bias** is the difference between the expected value of a regressor/classifier and the actual value. If you build N trees from the same distribution of data you’d expect the same expected value for all. So the bias is what the bias is. You can’t really do anything about that.

• And for any given tree, the *variance* (the expected square value between the estimate and the truth) is what the variance is.

• But if the trees are independently constructed then the *average* of the trees will have the same bias but *reduced variance*.

• The goal of random forests is (according to AFB) is to produce trees that are as independent as possible.
From Bagging and Boosting to Random Forests

• Bagging works particularly well for trees, since trees have high variance but low bias (why?).

• But, boosting typically outperforms bagging with trees.

• Why? The main culprit is usually dependence: Boosting is better at reducing correlation between the trees than bagging is.

• **Random Forests**: modification of bagging with trees designed to reduce correlation.
  • Tree training optimizes each split over all dimensions.
  • Random forests choose a different subset of dimensions at each split.
  • Optimal split is chosen within the subset.
  • The subset is chosen at random out of all dimensions \{1, \ldots, m\}. 

Random Forest Algorithm

**Algorithm 15.1 Random Forest for Regression or Classification.**

1. For $b = 1$ to $B$:
   
   (a) Draw a bootstrap sample $Z^*$ of size $N$ from the training data.
   
   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\text{min}}$ is reached.
   
   i. Select $m$ 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 of trees $\{T_b\}_1^B$.

To make a prediction at a new point $x$:

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

**Classification:** Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) =$ majority vote $\{\hat{C}_b(x)\}_1^B$. 
FIGURE 15.1. Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each “step” in the figure corresponds to a change in a single misclassification (in a test set of 1536).
FIGURE 15.2. The results of 50 simulations from the “nested spheres” model in IR^10. The Bayes decision boundary is the surface of a sphere (additive). “RF-3” refers to a random forest with m = 3, and “GBM-6” a gradient boosted model with interaction order six; similarly for “RF-1” and “GBM-1.” The training sets were of size 2000, and the test sets 10,000.
Some RF details

• Typically $m = \sqrt{p}$ for classification. But this varies. “In practice the best values for this parameter will depend on the problem.” (Duh)

• “Out of bag samples” - to estimate test error, for each $x_i$ average/vote only the trees which were constructed from bootstrap samples that did not contain $x_i$.

• If the number of irrelevant variables is high, can be a problem. But not if enough good ones.
Noise variables... but...

**FIGURE 15.7.** A comparison of random forests and gradient boosting on problems with increasing numbers of noise variables. In each case the true decision boundary depends on two variables, and an increasing number of noise variables are included. Random forests uses its default value $m = \sqrt{p}$. At the top of each pair is the probability that one of the relevant variables is chosen at any split. The results are based on 50 simulations for each pair, with a training sample of 300, and a test sample of 500.
Tree size not that big a factor

**FIGURE 15.8.** The effect of tree size on the error in random forest regression. In this example, the true surface was additive in two of the 12 variables, plus additive unit-variance Gaussian noise. Tree depth is controlled here by the minimum node size; the smaller the minimum node size, the deeper the trees.
Analysis of RFs

• Assume each tree has a variance of $\sigma^2$

• If they were i.i.d. then the average of $B$ trees would be $\frac{\sigma^2}{B}$

• But they are not independent. Assume a (positive) pairwise correlation between trees of $\rho$. Then variance of the average is:

$$\rho \sigma^2 + \frac{1 - \rho}{B} \sigma^2$$

• As $B \to \infty$ variance is determined by $\rho$.

• BY selecting only $m$ each node to consider, decrease correlation.
Empirical effect of $m$

**FIGURE 15.9.** Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of $m$. The boxplots represent the correlations at 600 randomly chosen prediction points $x$. 
Relation to K-NN

- A deep(maximal) tree finds a “path” down to an individual training element. Much like KNN

**FIGURE 15.11.** Random forests versus 3-NN on the mixture data. The axis-oriented nature of the individual trees in a random forest lead to decision regions with an axis-oriented flavor.
Now for the tutorial...