What’s better than a tree?
Random Forest

- Can often improve performance of decision tree classifiers using a set of decision trees (a forest)
- Each tree trained on a random subset of training data
- Classify a data instance using all trees
- Combine answers to make classification
  - E.g., vote for most common class
1. Many trees are created using random subsets of features and bootstrapped data.

2. Each tree votes by predicting target class.

3. Votes are tallied to reach the final prediction.
Wisdom of the Crowd

Statistician Francis Galton observed a 1906 contest to guess an ox’s weight at a country fair. 800 people entered. He noted that their average guess (1,197lb) was very close to the actual weight (1,198lb).

When getting human annotations training data for machine learning, standard practice is get ≥ 3 annotations and take majority vote.

cf. abbreviation (short for Latin: confer/conferatur) refer reader to other material to make a comparison
Random Forests Benefits

• Decision trees not the strongest modeling approach
• Random forests make them much stronger
• => more robust than a single decision tree
  – Limit overfitting to given dataset
  – Reduce errors due to training data bias
  – Stable performance if some noise added to training data
Bagging

• Idea can be used on any classifier!
• Improve classification by combining classifications of randomly selected training subsets
• Bagging = **Bootstrap aggregating ensemble** meta-algorithm that can improve stability & accuracy of algorithms for statistical classification and regression
• Helps avoid overfitting
• AKA ensembling
**Bagging**

Bootstrap Aggregation

Data → Bootstrapped Data → Models → Voting → Outcome

- Data
- Bootstrap Data
- Models
- Voting
- Outcome

**Diagram:**

- Data
- Bootstrapped Data
- Models
- Voting
- Outcome

- Train model
- Voting

- Sampled with replacement

- Chris Albon
Choosing training data subsets

• **Classic bagging**: select random subset of training instances **with replacement**

• **Pasting**: select random subset of training instances (i.e., without replacement)

• **Random Subspaces**: use all training instances, but with a random subset of features

• **Random Patches**: random subset of instances and random subset of features

• **Best?** depends on problem, training data, algorithm
Examples

• Two examples using Weka
  – UCI Auto mpg prediction dataset
  – UCI Adult income prediction dataset

• RandomForest improves over J48 for the smaller dataset, but not for the larger

• Takeaway: more data is always best
UCI Auto MGP Dataset

398 instances with 8 attributes from 1983:

Predict MPG from other 7 attributes

Arff training data (240); test data (132)
Avg F1 = 0.98 very high!
Avg F1 = 1.0 perfect!
100% ... Wait, What?

• Results are too good to be true!
  – Something must be wrong

• ML results tend to be asymptotic
  – Asymptotic lines approach a curve but never touch

• Closer you get to F1=1.0, the harder it is to improve

• What did we do wrong?
Results are too good

• Relatively small dataset allows construction of a DT model that does very well
• Using Random Forest still got perfect results!
• We trained and tested on the same data!
• Very poor methodology since it overfits to this particular training set
• This training dataset has a separate test data set
  – We can also try 10-fold cross validation
Avg F1 = 0.843 good
Avg F1 = 0.867 better
New AUTO MPG Results

• Using an independent test set shows more realistic balanced F1 score of \(0.843\)
• Using Random Forest raises this to \(0.867\)
• While the increase is not large, it is probably statistically significant
• F1 scores this high are difficult to increase dramatically
  – Human scores for many tasks are often in this range (i.e., 0.8–0.9)
UCI Adult Census Income Dataset

~49K instances with 15 attributes from 1994:
1. >50K: binary; age: continuous. workclass:
   Private, Self-emp-not-inc, Self-emp-inc, Federal-
   gov, Local-gov, State-gov, Without-pay, Never-
   worked. fnlwgt: continuous. education: Bachelors,
   Some-college, 11th, HS-grad, Prof-school, ...

Predict income >50k from 15 attributes

Arff data
Choose J48 -C 0.25 -M 2

Use training set

Size of the tree: 911

Time taken to build model: 2.64 seconds

Time taken to test model on training data: 0.16 seconds

Correctly Classified Instances 42803 87.6356 %
Incorrectly Classified Instances 6039 12.3644 %

Kappa statistic 0.6325
Mean absolute error 0.1861
Root mean squared error 0.3048
Relative absolute error 51.1076 %
Root relative squared error 71.4388 %
Total Number of Instances 48842

Detailed Accuracy By Class:

<table>
<thead>
<tr>
<th></th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
<th>PRC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;50K</td>
<td>0.631</td>
<td>0.046</td>
<td>0.810</td>
<td>0.631</td>
<td>0.710</td>
<td>0.600</td>
<td>0.907</td>
<td>0.792</td>
<td>&gt;50K</td>
</tr>
<tr>
<td>&lt;=50K</td>
<td>0.954</td>
<td>0.369</td>
<td>0.891</td>
<td>0.954</td>
<td>0.921</td>
<td>0.840</td>
<td>0.907</td>
<td>0.960</td>
<td>&lt;=50K</td>
</tr>
</tbody>
</table>

Weighted Avg. 0.876 0.369 0.891 0.954 0.921 0.840 0.907 0.960

Confusion Matrix:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>&lt;-- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>7375</td>
<td>4312</td>
<td>&gt;50K</td>
</tr>
<tr>
<td>b</td>
<td>1727</td>
<td>35428</td>
<td>&lt;=50K</td>
</tr>
</tbody>
</table>
Classifier: RandomForest -P 100 -l 100 -num-slots 1 -k 0 -M 1.0 -V 0.001 -S 1

Test options:
- Use training set

Classifier output:

Data mining with 100 iterations and base learner:
weka.classifiers.trees.RandomTree -k 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 15.17 seconds

Time taken to test model on training data: 6.52 seconds

Summary:

Correctly Classified Instances 48774 99.8608 %
Incorrectly Classified Instances 68 0.1392 %
Kappa statistic 0.9962
Mean absolute error 0.0737
Root mean squared error 0.1263
Relative absolute error 20.2565 %
Root relative squared error 29.6022 %
Total Number of Instances 48842

Detailed Accuracy By Class:

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCR</th>
<th>ROC Area</th>
<th>PRC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;50K</td>
<td>0.995</td>
<td>0.000</td>
<td>1.000</td>
<td>0.997</td>
<td>0.997</td>
<td>0.996</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>&lt;=50K</td>
<td>1.000</td>
<td>0.005</td>
<td>0.998</td>
<td>1.000</td>
<td>0.999</td>
<td>0.996</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.999</td>
<td>0.004</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
<td>0.996</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Confusion Matrix:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>--- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11624</td>
<td>63</td>
<td>a = &gt;50K</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>37150</td>
<td>b = &lt;=50K</td>
</tr>
</tbody>
</table>
Result

• Significant increase on F1 scores when both trained and evaluated on training set
• This is considered to be poor methodology since it overfits to the particular training set
Create train and test collection

• Train has ~95% of data, test 5%
• Trained models for J48 and random forest using train dataset
• Tested on test data set
• Results were that random forest was (at best) about the same as J48
• Large dataset reduced problem of overfitting, so random forest did not help
F = 0.856
F = 0.853
Conclusions

- Bagging helps, especially if training data adequate, but not as large as it should be
  - With lots of data, overfitting less of a problem, so bagging may not help
- While we explore it using decision trees, it can be applied to any classifier
  - Scikit-learn has a general module for bagging
- In general, using any of several ensemble approaches to classification often helpful
- Training neural networks uses a different approach (dropout) to control overfitting
Conclusions

• Wait, there’s more...

• A classification problem can change over time
  – E.g.: recognizing a spam message from its content and metadata

• We showed that an ensemble approach can detect a change in the nature of spam
  – Which tells us its time to retrain with new data
Recognizing Concept Drift

• Build ensemble of five models to classify spam comments left on a blog at time T1
• Note the relative level of agreement
• Detect when one of the models starts to diverge from the others with at time T2
  – Time to get new data and retrain
  – Examining disagreements can be enlightening
• Used temporal data spanning several years to prove effectiveness
  – E.g., spam focus shift from viagra to weight loss