Learning Imbalanced Data with Random Forests

Chao Chen (Stat., UC Berkeley)
chenchao@stat.Berkeley.EDU

Andy Liaw (Merck Research Labs)
andy_liaw@merck.com

Leo Breiman (Stat., UC Berkeley)
leo@stat.Berkeley.EDU

Interface 2004, Baltimore
Outline

• Imbalanced data
• Common approaches and recent works
• “Balanced” random forests
• “Weighted” random forests
• Some comparisons
• Conclusion
Imbalanced Data

• Data for many classification problems are inherently imbalanced
  – One large, “normal” class (negative) and one small/rare, “interesting” class (positive)
  – E.g.: rare diseases, fraud detection, compound screening in drug discovery, etc.

• Why is this a problem?
  – Most machine learning algorithms focus on overall accuracy, and “break down” with moderate imbalance in the data
  – Even some cost-sensitive algorithms don’t work well when imbalance is extreme
Common Approaches

• Up-sampling minority class
  – random sampling with replacement
  – strategically add cases that reduce error

• Down-sampling majority class
  – random sampling
  – strategically omit cases that do not help

• Cost-sensitive learning
  – build misclassification cost into the algorithm

• Down-sampling tends to work better empirically, but loses some information, as not all training data are used
Recent Work

- One-sided sampling
- SMOTE: Synthetic Minority Oversampling Technique (Chawla et al, 2002)
- SMOTEBoost
- SHRINK
Random Forest

• A supervised learning algorithm, constructed by combining multiple decision trees (Breiman, 2001)
• Draw a bootstrap sample of the data
• Grow an un-pruned tree
  – At each node, only a small, random subset of predictor variables are tried to split that node
• Repeat as many times as you’d like
• Make predictions using all trees
“Balanced” Random Forest

- Natural integration of down-sampling majority class and ensemble learning
- For each tree in RF, down-sample the majority class to the same size as the minority class
- Given enough trees, all training data are used, so no loss of information
- Computationally efficient, since each tree only sees a small sample
“Weighted” Random Forest

• Incorporate class weights in several places of the RF algorithm:
  – Weighted Gini for split selection
  – Class-weighted votes at terminal nodes for node class
  – Weighted votes over all trees, using average weights at terminal nodes

• Using weighted Gini alone isn’t sufficient
Performance Assessment

### Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Predicted Positive</th>
<th>Predicted Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive</strong></td>
<td>True Positive</td>
<td>False Negative</td>
</tr>
<tr>
<td><strong>Negative</strong></td>
<td>False Positive</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

- **True Positive Rate (TPR):** \(\frac{TP}{TP + FN}\)
- **True Negative Rate (TNR):** \(\frac{TN}{TN + FP}\)
- **Precision:** \(\frac{TP}{TP + FP}\)
- **Recall:** same as TPR
- **g-mean:** \((TPR \times TNR)^{1/2}\)
- **F-measure:** \(\frac{(2 \times \text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}\)
# Benchmark Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of Var.</th>
<th>No. of Obs.</th>
<th>% Minority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oil Spill</td>
<td>50</td>
<td>937</td>
<td>4.4</td>
</tr>
<tr>
<td>Mammograph</td>
<td>6</td>
<td>11183</td>
<td>2.3</td>
</tr>
<tr>
<td>SatImage</td>
<td>36</td>
<td>6435</td>
<td>9.7</td>
</tr>
</tbody>
</table>
## Oil Spill Data

<table>
<thead>
<tr>
<th>Method</th>
<th>TPR</th>
<th>TNR</th>
<th>Precision</th>
<th>G-mean</th>
<th>F-meas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-sided sampling</td>
<td>76.0</td>
<td>86.6</td>
<td>20.5</td>
<td>81.13</td>
<td>32.3</td>
</tr>
<tr>
<td>SHRINK</td>
<td>82.5</td>
<td>60.9</td>
<td>8.85</td>
<td>70.9</td>
<td>16.0</td>
</tr>
<tr>
<td>SMOTE</td>
<td>89.5</td>
<td>78.9</td>
<td>16.4</td>
<td>84.0</td>
<td>27.7</td>
</tr>
<tr>
<td>BRF</td>
<td>73.2</td>
<td>91.6</td>
<td>28.6</td>
<td>81.9</td>
<td>41.1</td>
</tr>
<tr>
<td>WRF</td>
<td>92.7</td>
<td>82.4</td>
<td>19.4</td>
<td>87.4</td>
<td>32.1</td>
</tr>
</tbody>
</table>

## Mammography Data

<table>
<thead>
<tr>
<th>Method</th>
<th>TPR</th>
<th>TNR</th>
<th>Precision</th>
<th>G-mean</th>
<th>F-mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIPPER</td>
<td>48.1</td>
<td>99.6</td>
<td>74.7</td>
<td>69.2</td>
<td>58.1</td>
</tr>
<tr>
<td>SMOTE</td>
<td>62.2</td>
<td>99.0</td>
<td>60.5</td>
<td>78.5</td>
<td>60.4</td>
</tr>
<tr>
<td>SMOTE-Boost</td>
<td>62.6</td>
<td>99.5</td>
<td>74.5</td>
<td>78.9</td>
<td>68.1</td>
</tr>
<tr>
<td>BRF</td>
<td>76.5</td>
<td>98.2</td>
<td>50.5</td>
<td>86.7</td>
<td>60.8</td>
</tr>
<tr>
<td>WRF</td>
<td>72.7</td>
<td>99.2</td>
<td>69.7</td>
<td>84.9</td>
<td>71.1</td>
</tr>
</tbody>
</table>

## Satimage Data

<table>
<thead>
<tr>
<th>Method</th>
<th>TPR</th>
<th>TNR</th>
<th>Precision</th>
<th>G-mean</th>
<th>F-meas</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIPPER</td>
<td>47.4</td>
<td>97.6</td>
<td>67.9</td>
<td>68.0</td>
<td>55.5</td>
</tr>
<tr>
<td>SMOTE</td>
<td>74.9</td>
<td>91.3</td>
<td>48.1</td>
<td>82.7</td>
<td>58.3</td>
</tr>
<tr>
<td>SMOTE-Boost</td>
<td>67.9</td>
<td>97.2</td>
<td>72.7</td>
<td>81.2</td>
<td>70.2</td>
</tr>
<tr>
<td>BRF</td>
<td>77.0</td>
<td>93.6</td>
<td>56.3</td>
<td>84.9</td>
<td>65.0</td>
</tr>
<tr>
<td>WRF</td>
<td>77.5</td>
<td>94.6</td>
<td>60.5</td>
<td>85.6</td>
<td>68.0</td>
</tr>
</tbody>
</table>

A Simple Experiment: 2Norm

- Fix size of one class at 100, vary the size of other class among 5e3, 1e4, 5e4 and 1e5
- Train both WRF and BRF, predict on same size test set
  - WRF: use reciprocal of class ratio as weights
  - BRF: draw 100 from each class w/replacement to grow each tree
- With usual prediction, BRF has better false negative rate; WRF has better true positive rate
- Compare cumulative gain to see difference
Comparing Cumulative Gain

![Graph comparing cumulative gain with different ratios and lines for WRF and BRF.](image)
To Wrap Up…

• We propose two methods of learning imbalanced data with random forests
  – BRF: down-sampling majority in each tree
  – WRF: incorporate class weights in several places

• Both show improvements over existing methods

• The two are about equally effective on real; hard to pick a winner

• Need further study to see if/when/why one works better than the other
Free Software

• Random Forest (Breiman & Cutler): Fortran code, implements WRF, available at
  
  $$\text{http://stat-www.berkeley.edu/users/breiman/RandomForests/}$$

• randomForest (Liaw & Wiener): add-on package for R (based on the Fortran code above), implements BRF, available on CRAN
  
  (e.g.: $$\text{http://cran.us.r-project.org/src/contrib/PACKAGES.html}$$)
Acknowledgment

• Adele Cutler (Utah State)
• Vladimir Svetnik, Chris Tong, Ting Wang (BR)
• Matt Wiener (ACSM)