RF + RLSC

Kari Torkkola
Motorola
Intelligent Systems Lab
Tempe, AZ, USA
Kari.Torkkola@motorola.com

Eugene Tuv
Intel
Analysis and Control Technology
Chandler, AZ, USA
eugene.tuv@intel.com
RF + RLSC

- Random Forests (RF) for feature selection
- Regularized Least Squares Classifiers (RLSC)
- Stochastic ensembles of RLSCs
Why Random Forests for Feature Selection?

• Basic idea: Train a classifier, then extract features that are important to the classifier
• Features are not chosen in isolation!
• RF is extremely fast to train
• Allows for mixed data types, missing values
Random Forests for Feature Selection - How?

- **RF**
  - Trains a large forest of decision trees
  - Samples the training data for each tree
  - Samples the features to make each split
  - Error estimation from out-of-bag cases
  - Proximity measures, importance measures, ...

- **An Importance Measure**
  - A split in a tree by using a particular variable results in a decrease of the gini index
  - Sum of these decreases over the forest ranks features by importance
Challenge Examples

**Madelon**
- 500 variables, training set has 2000 cases
- Constructed 500 trees
- Variable importance has a clear cut-off point at 19 variables
- Validation set: 600 cases
  - The top 19 variables are the same, but the cut-off point is not that clear

**Dexter**
- 20000 variables, 300 cases in both the training and the validation sets
- Top 50 variables from both sets are 70% shared (stability)
Why Ensembles of RLSCs as Classifiers?

- Why not just use RF? – The base learner is not good enough!
- RLSC solves a simple linear problem

Given data \((x_i, y_i)_{i=1}^m\), find \(f : X \rightarrow Y\) that generalizes:

1. Choose a kernel, such as \(K(x, x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}\),
2. \(f(x) = \sum_{i=1}^m c_i Kx_i(x)\), where \(c_i\) is a solution to \((m\gamma I + K)c = y\)

- Square loss function works well in binary classification (Poggio, Smale, et al.)
- Use minimum regularization (just to guarantee solution) to reduce bias, sample cases to produce diversity in base learners
Things to worry about with RLSC Ensembles

- Kernel and its parameters?
- How many classifiers in the ensemble?
- What fraction of data to use to train each?
- How much to regularize (if at all)?
- Determine all of the above by cross-validation
Future Directions

- RF as one type of supervised kernel generator using the pairwise similarities
- Similarity between 2 cases could be defined (for a single tree) as total number of common parent nodes, normalized by level of the deepest case, and summed up for the ensemble
- Minimum number of common parents to define nonzero similarity is another parameter acting like width in Gaussian kernels.
- Works for any type of data (numeric, categorical, mixed, missing values)!
- Feature selection bypassed altogether!

Arcene: Gaussian kernel

<table>
<thead>
<tr>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Arcene: Supervised kernel

<table>
<thead>
<tr>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

• RF: Fast and robust feature selection
• RLSC: linear problem-solving
• Supervised kernels
• What we don’t know…