Evolution of Regression IV: From OLS to GPS™, MARS®, CART®, TreeNet® and RandomForests®

March 2013
Dan Steinberg
Mikhail Golovnya
Salford Systems
Course Outline

• Regression Problem – quick overview
• Classical OLS – the starting point
• RIDGE/LASSO/GPS – regularized regression
• MARS® – adaptive non-linear regression

Previous Webinars:

Today’s Webinar: Hands On Session

• CART® Regression tree
• Bagger and Random Forest® CART ensembles
• TreeNet® Stochastic Gradient Boosting
• Hybrid TreeNet/GPS models
Boston Housing Data Set

• Concerns the housing values in Boston area
• Combined information from 10 separate governmental and educational sources to produce this data set
• 506 census tracts in City of Boston for the year 1970

  o **Goal:** study relationship between quality of life variables and property values
  o **MV** median value of owner-occupied homes in tract ($1,000's)
  o **CRIM** per capita crime rates
  o **NOX** concentration of nitric oxides (pp 10 million)
  o **AGE** percent built before 1940
  o **DIS** weighted distance to centers of employment
  o **RM** average number of rooms per house
  o **LSTAT** % lower status of the population
  o **RAD** accessibility to radial highways
  o **CHAS** borders Charles River (0/1)
  o **INDUS** percent non-retail business
  o **TAX** property tax rate per $10,000
  o **PT** pupil teacher ratio
## Running Score: Test Sample MSE

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS Regression Splines</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso/ Regularized</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
</tbody>
</table>
Regression Tree

- Decision tree is a nonparametric learning machine with capability for classification and regression (select methods)
- Stepwise procedure in which predictors enter the model one at a time
- In 1975 Jerome H. Friedman labeled the methodology *Recursive Partitioning* (for binary target)
- Independent work, also 1975, by Leo Breiman and Charles Stone included regression
- Collaboration of the two groups led to CART tree
- Procedure works by carving a high dimensional data space into a small to moderate set of regions
- Produce a prediction for each region
Regression Tree
Out of the box results, no tuning of controls

9 regions (terminal nodes)

Test MSE= 17.296
Grow tree to maximum possible size: here learn sample $R^2 = 1.0$
Then prune back to find best performing sub-tree on test sample
Maximal tree is almost certainly overfit yet frequently offers decent performance on new data
Regression Tree

• Generally the single regression tree is not expected to perform outstandingly
• Works by carving a high dimensional data space into mutually exclusive and collectively exhaustive regions
• Single prediction (mean for the region) is generated for every region
  o Our optimal tree above produces only 9 distinct outputs
  o For any input record there are only 9 possible predicted values
• Linear regression typically produces a unique response for every input record (for rich enough data)
Regression Tree Representation of a Surface High Dimensional Step function

Should be at a disadvantage relative to other tools. Can never be smooth. But always worth checking.
Use model to simulate impact of a change in predictor
Here we simulate separately for every training data record and then average
For CART trees is essentially a step function
May only get one “knot” in graph if variable appears only once in tree

We used TreeNet to grow a single CART-style tree and then extracted the graphs from TreeNet’s graphing functionality
Repeated Runs Different 20% Test Samples

100 repetitions dividing data random into 80% learn, 20% test
Percentiles: 5th = 12.73  95th MSE = 32.32, median 20.66
# Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Repeated 100 20% Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
</tbody>
</table>
## Monte Carlo/Parametric Bootstrap

<table>
<thead>
<tr>
<th>Target</th>
<th>Predicted</th>
<th>Residual</th>
<th>Constructed Target*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>Y1_hat</td>
<td>e5</td>
<td>Y1_hat + e5</td>
</tr>
<tr>
<td>Y2</td>
<td>Y2_hat</td>
<td>e22</td>
<td></td>
</tr>
<tr>
<td>Y3</td>
<td>Y3_hat</td>
<td>e9</td>
<td></td>
</tr>
<tr>
<td>Y4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\ldots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yn</td>
<td>Yn_hat</td>
<td>en</td>
<td></td>
</tr>
</tbody>
</table>

Prediction and residual come from baseline model  
Shuffle the residuals within the column (OK if residuals are just noise)  
Create new synthetic target: Predicted value plus scrambled residual  
Gives us a “new” data set. Same X matrix (predictors) by target Y has been altered
SAVING the LEARN/TEST PARTITION

• It can be convenient to be able to save the flag used internally by SPM to partition the data
• This is most easily done via a command script such as

  USE BOSTON.CSV
  PARTITION TEST=.20
  CATEGORY
  MODEL MV
  SAVE “BOSTON_LT20.CSV” /COMPLETE
  CART GO

• Later when we need to SELECT just the test records we will work with this version of the data
Bagger (Bootstrap Aggregation)

• Breiman introduced the influential “Bagger” in 1996 to solve the automatic model generation problem
• Intuition: Imagine that we have an unlimited supply of data
  o Draw a random sample from the master data (of good size)
    • Sample could be stratified to oversample rare target class
  o Grow a regression tree
  o Draw a new, different, random sample from the master data
  o Grow a new regression tree
  o Process can be continued indefinitely as each draw will result in a different sample and typically a different (at least slightly) tree
  o Combine results into a single prediction for any input data record
Bagger Mechanism

• Generate a reasonable number of bootstrap samples
  o Breiman started with numbers like 50, 100, 200
• Grow a standard CART tree on each sample
• Use the **unpruned** tree to make predictions
  o Pruned trees yield inferior predictive accuracy for the ensemble
• Simple voting for classification
  o Majority rule voting for binary classification
  o Plurality rule voting for multi-class classification
  o Average predicted target for regression models
• Will result in a much smoother range of predictions
  o Single tree gives same prediction for all records in a terminal node
  o In bagger records will have different patterns of terminal node results
• Each record likely to have a **unique score** from ensemble
Bagger Via CART Ensembles

Ensembles Engine (CART only)

- Ensemble Method: Bagging
- Number of trees to ensemble: 100
- Analysis Method: CART Ensembles

100 trees requested
Bagger via BATTERY BOOTSTRAP

Engine is CART

<table>
<thead>
<tr>
<th>Model Setup</th>
<th>Categorical</th>
<th>Force Split</th>
<th>Constraints</th>
<th>Testing</th>
<th>Select Cases</th>
<th>Best Tree</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
</tr>
<tr>
<td>ATOM</td>
<td>BOOTSTRAP</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
<td>Battery</td>
</tr>
</tbody>
</table>

Battery Types
- ATOM
- CVBIN
- CVR
- DATASHIFT
- DEPTH
- DRAW
- FLIP
- KEEP
- LOVO
- MCT
- MINCHILD
- MODELS
- NVT

Selected Batteries
- BOOTSTRAP

Battery Options
- Number of folds in cross-validation
- Proximity
- Quiet
- Automatic Best Predictor Discovery
- After Building a Model
- Analysis Method
- Number of Predictors
- TreeNet Tree Size
- Random Forests
- Variable Importance
- Sampling
- Save Results
- Bootstrap Setup
- General
- Bootstrap models
- Tree/Model Pruning
- Random Forests
- Variable Importance
- Sampling
- Save Results

Engine is CART
Bagger via Bootstrapping: Test MSE=9.545

Score TEST partition using 100 tree ensemble
Predictions From First 6 Trees

First 10 records of TEST partition
First 6 Bagger Trees, each tree grown to maximal size unpruned
No feedback, learning, or tuning derived from test partition
Predictions vary as each tree is built on a different bootstrap sample

Bagger outputs the AVERAGE prediction for a regression model
TEST MSE=9.545
Use model to simulate impact of a change in predictor
Here we simulate separately for every training data record and then average
For CART trees is essentially a step function
May only get one “knot” in graph if variable appears only once in tree

See appendix to learn how to get these plots
Bagger Partial Dependency Plot

Averaging over many trees allows for a more complex dependency
Opportunity for many splits of a variable (100 large trees)
Jaggedness may reflect existence of interactions
## Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
<tr>
<td>Bagged CART</td>
<td>9.545</td>
<td></td>
<td>12.79</td>
</tr>
</tbody>
</table>
Variations on the Bagger

- Via CART
  - BATTERY BOOTSTRAP TEST=EXPLORE
- Via CART Ensembles (legacy version)
- Via TreeNet put into independent tree mode
  - Random half sample approximation to bootstrap sample
- Via Random Forests with all predictors available at every node

<table>
<thead>
<tr>
<th>Method</th>
<th>Test MSE</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CART BATTERY BOOTSTRAP</td>
<td>9.49</td>
<td>Unpruned trees</td>
</tr>
<tr>
<td>CART Ensembles</td>
<td>9.64</td>
<td>Unpruned trees</td>
</tr>
<tr>
<td>RandomForests in Bagger mode</td>
<td>8.90</td>
<td>Unpruned trees</td>
</tr>
<tr>
<td>TreeNet in Bagger Mode</td>
<td>8.85</td>
<td>Half-sample</td>
</tr>
</tbody>
</table>
RandomForests: Bagger on Steroids

• Leo Breiman was frustrated by the fact that the bagger did not perform better. Convinced there was a better way
• Observed that trees generated bagging across different bootstrap samples were surprisingly similar
• How to make them more different?
• Bagger induces randomness in how the rows of the data are used for model construction
• Why not also introduce randomness in how the columns are used for model construction
• Pick a random subset of predictors as candidate predictors – a new random subset for every node

• Breiman was inspired by earlier research that experimented with variations on these ideas
• Breiman perfected the bagger to make RandomForests
RF Main Innovation Counter-Intuitive

- Picking splitter at random appears to be a recipe for poor trees
- Indeed the RF model may not perform so well if the splitter is chosen completely at random
  - At every node choose the variable that splits the node at random
  - Better to allow at least some opportunity for optimization by considering several potential splitters
- By limiting access to a different subset of predictors at each node we guarantee that trees will be different across the different bootstrap samples
- Performance of individual trees on holdout data might be inferior to bagger trees but ensemble will do better
RF Model using defaults: Test MSE=8.286
# Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
<tr>
<td>Bagged CART</td>
<td>9.545</td>
<td></td>
<td>12.79</td>
</tr>
<tr>
<td>RF Defaults</td>
<td>8.286</td>
<td></td>
<td>12.84</td>
</tr>
</tbody>
</table>

BATTERY PARTITION reruns a model a specified number of times using a different random division of the data into learn and test partitions

In some cases we had to run this experiment “manually” via a command script as the BATTERY is not available for all model types
Varying PREDs
Try every value from 1 to 13

Purely-at-random RF yields inferior results but still impressive (Test MSE=11.15)

Test Sample results

OOB Results
At optimum OOB PREDs=6
Test MSE=8.002
# Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
<tr>
<td>Bagged CART</td>
<td>9.545</td>
<td></td>
<td>12.79</td>
</tr>
<tr>
<td>RF Defaults</td>
<td>8.286</td>
<td></td>
<td>12.84</td>
</tr>
<tr>
<td>RF PREDs=6</td>
<td>8.002</td>
<td></td>
<td>12.05</td>
</tr>
</tbody>
</table>
RandomForests Strengths

• RandomForests can be used for
  o Classification (binary or multi-class classification)
  o Regression (prediction of a continuous target)
  o Clustering/Density estimation

• Robust behavior relatively insensitive to dirty data

• Easy to use with few control parameters

• Graphical displays of key model insights

• RandomForests can be effectively used with wide data sets
  o Possibly many more columns than rows
  o High speed even with millions of predictors

• Naturally parallelizable as every tree grown independently

• Retains advantages offered by most regression trees

• Very strong variable (predictor) selection
Stochastic Gradient Boosting (TreeNet)

• SGB is a revolutionary data mining methodology first introduced by Jerome H. Friedman in 1999
• Seminal paper defining SGB released in 2001
  o Google scholar reports more than 1600 references to this paper and a further 3300 references to a companion paper
• Extended further by Friedman in major papers in 2004 and 2008 (Model compression and rule extraction)
• Ongoing development and refinement by Salford Systems
  o Latest version released 2013 as part of SPM 7.0
• TreeNet/Gradient boosting has emerged as one of the most used learning machines and has been successfully applied across many industries
• Friedman’s proprietary code in TreeNet
Gradient Boosting Methodology: Key points

• Trees are *usually* kept small (2-6 nodes common)
  o However, should experiment with larger trees (12, 20, 30 nodes)
  o Sometimes larger trees are surprisingly good
• Updates are small (downweighted). Update factors can be as small as .01, .001, .0001.
  o Do not accept the full learning of a tree (small step size, also GPS style)
  o Larger trees should be coupled with slower learn rates
• Use random subsets of the training data in each cycle. Never train on all the training data in any one cycle
  o Typical is to use a random half of the learn data to grow each tree
TreeNet Gradient Boosting Model Setup

Differ from defaults only in selecting Least Squares Loss and TRESSES=1000
Best TreeNet results frequently require thousands of trees
Gradient Boosting Results (LOSS=LS)

Progress of model as trees are added
Best MSE and best MAD typically occur with different sized models (N Trees)
Essentially default settings yields Test MSE= 7.417 (bit better yet allowing 1200 trees)
# Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
<tr>
<td>Bagged CART</td>
<td>9.545</td>
<td></td>
<td>12.79</td>
</tr>
<tr>
<td>RF Defaults</td>
<td>8.286</td>
<td></td>
<td>12.84</td>
</tr>
<tr>
<td>RF PREDS=6</td>
<td>8.002</td>
<td></td>
<td>12.05</td>
</tr>
<tr>
<td>TreeNet Defaults</td>
<td>7.417</td>
<td>8.67</td>
<td>11.02</td>
</tr>
</tbody>
</table>

Using cross-validation on learn partition to determine optimal number of trees and then scoring the test partition with that model: TreeNet MSE=8.523
Tuning Gradient Boosting Regression

- Huber loss function uses squared error for the smaller errors
- For larger errors incremental loss is based on absolute error
- Robust form of regression less sensitive to outliers
Vary HUBER Threshold: Best MSE = 6.71

Vary threshold where we switch from squared errors to absolute errors.
Optimum when the 5% largest errors are not squared in loss computation.
Yields best MSE on test data. Sometimes LAD yields best test sample MSE.
Gradient Boosting Partial Dependency Plots

LSTAT

NOX
## Running Score

<table>
<thead>
<tr>
<th>Method</th>
<th>20% random</th>
<th>Parametric Bootstrap</th>
<th>Battery Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>27.069</td>
<td>27.97</td>
<td>23.80</td>
</tr>
<tr>
<td>MARS</td>
<td>14.663</td>
<td>15.91</td>
<td>14.12</td>
</tr>
<tr>
<td>GPS Lasso</td>
<td>21.361</td>
<td>21.11</td>
<td>23.15</td>
</tr>
<tr>
<td>CART</td>
<td>17.296</td>
<td>17.26</td>
<td>20.66</td>
</tr>
<tr>
<td>Bagged CART</td>
<td>9.545</td>
<td></td>
<td>12.79</td>
</tr>
<tr>
<td>RF Defaults</td>
<td>8.286</td>
<td></td>
<td>12.84</td>
</tr>
<tr>
<td>RF PREDS=6</td>
<td>8.002</td>
<td></td>
<td>12.05</td>
</tr>
<tr>
<td>TreeNet Defaults</td>
<td>7.417</td>
<td>8.67</td>
<td>11.02</td>
</tr>
<tr>
<td>TreeNet Huber</td>
<td><strong>6.682</strong></td>
<td>7.86</td>
<td>11.46</td>
</tr>
<tr>
<td>TN Additive</td>
<td>9.897</td>
<td></td>
<td>10.48</td>
</tr>
</tbody>
</table>

If we had used cross-validation to determine the optimal number of trees and then used those to score test partition the TreeNet Default model MSE=8.523
Interactions

• When the impact of the change in a predictor $X_i$ depends on the value of another predictor $X_j$
• Classical statistical models usually start with creating a new feature defined as the product $X_i \times X_j$
• Extraordinarily difficult to discover, refine, select interactions into classical models
• Trees generate interactions naturally (perhaps too many and too readily)
• Technically there is an interaction between $X_i$ and $X_j$ if the two predictors appear on the same branch of a tree
• Still need to check to see if in simulations there really is any material dependency that qualifies as an interaction
Preventing An Interaction

- Friedman suggested that one can develop models without interactions by just growing 2-node trees.
- If this technique is used one must compensate for the restricted learning allowed in each tree by growing many more trees.
- Salford offers other ways to prevent interactions.
- Preventing interactions in the TreeNet model substantially increases the test sample MSE to 9.897.
Interaction Detection with TreeNet

- TreeNet offers diagnostics and formal tests for the existence of interactions and their strengths
- Build a model without interactions (for example, limiting depth of the tree to one split only)
- Compare results with a more flexible model (trees allowed to grow deeper)
- TreeNet offers reports based on these comparisons
TreeNet Interaction Strength Measures
LOSS=LS

- TreeNet Interactions
- Whole Variable Interactions
  - Measure Predictor

<table>
<thead>
<tr>
<th>Interaction Strength</th>
<th>Predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.18831 LSTAT</td>
<td></td>
</tr>
<tr>
<td>8.99670 DIS</td>
<td></td>
</tr>
<tr>
<td>7.83481 RM</td>
<td></td>
</tr>
<tr>
<td>6.67131 NOX</td>
<td></td>
</tr>
<tr>
<td>6.18011 CRIM</td>
<td></td>
</tr>
<tr>
<td>4.75381 AGE</td>
<td></td>
</tr>
<tr>
<td>4.08043 B</td>
<td></td>
</tr>
<tr>
<td>3.82925 PT</td>
<td></td>
</tr>
<tr>
<td>2.39673 TAX</td>
<td></td>
</tr>
<tr>
<td>1.52854 INDUS</td>
<td></td>
</tr>
<tr>
<td>0.95982 RAD</td>
<td></td>
</tr>
<tr>
<td>0.82921 CHAS</td>
<td></td>
</tr>
<tr>
<td>0.27512 ZN</td>
<td></td>
</tr>
</tbody>
</table>

Interaction strength is a measure of the difference between predictions generated by a flexible and additive model.

For a specific 2-way interaction the differences are measured over the joint range of the pair of predictors in question.

Statistical tests are required to eliminate “spurious” interactions.
# Top Ranked 2-way interactions

<table>
<thead>
<tr>
<th>Predictor: LSTAT</th>
<th>Measure1</th>
<th>Measure2</th>
<th>Predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.95186</td>
<td>7.44771</td>
<td>RM</td>
<td></td>
</tr>
<tr>
<td>5.51983</td>
<td>11.23505</td>
<td>DIS</td>
<td></td>
</tr>
<tr>
<td>4.05403</td>
<td>7.49177</td>
<td>AGE</td>
<td></td>
</tr>
<tr>
<td>3.96962</td>
<td>7.18046</td>
<td>NOX</td>
<td></td>
</tr>
<tr>
<td>3.87788</td>
<td>7.61404</td>
<td>CRIM</td>
<td></td>
</tr>
<tr>
<td>1.33469</td>
<td>2.37310</td>
<td>PT</td>
<td></td>
</tr>
<tr>
<td>0.87894</td>
<td>1.67121</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>0.72915</td>
<td>1.41017</td>
<td>INDUS</td>
<td></td>
</tr>
<tr>
<td>0.64022</td>
<td>1.28559</td>
<td>RAD</td>
<td></td>
</tr>
<tr>
<td>0.48919</td>
<td>0.93228</td>
<td>TAX</td>
<td></td>
</tr>
<tr>
<td>0.28317</td>
<td>0.55915</td>
<td>ZN</td>
<td></td>
</tr>
<tr>
<td>0.11276</td>
<td>0.22112</td>
<td>CHAS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predictor: RM</th>
<th>Measure1</th>
<th>Measure2</th>
<th>Predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.95185</td>
<td>7.44771</td>
<td>LSTAT</td>
<td></td>
</tr>
<tr>
<td>2.22036</td>
<td>4.50192</td>
<td>PT</td>
<td></td>
</tr>
<tr>
<td>2.05789</td>
<td>4.55433</td>
<td>NOX</td>
<td></td>
</tr>
<tr>
<td>1.03393</td>
<td>2.35649</td>
<td>DIS</td>
<td></td>
</tr>
<tr>
<td>0.96500</td>
<td>2.24020</td>
<td>AGE</td>
<td></td>
</tr>
<tr>
<td>0.68364</td>
<td>1.54068</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>0.58447</td>
<td>1.36044</td>
<td>CRIM</td>
<td></td>
</tr>
<tr>
<td>0.56944</td>
<td>1.30544</td>
<td>CHAS</td>
<td></td>
</tr>
<tr>
<td>0.47589</td>
<td>1.06081</td>
<td>TAX</td>
<td></td>
</tr>
<tr>
<td>0.40533</td>
<td>0.91243</td>
<td>INDUS</td>
<td></td>
</tr>
<tr>
<td>0.28889</td>
<td>0.67370</td>
<td>RAD</td>
<td></td>
</tr>
<tr>
<td>0.03818</td>
<td>0.08853</td>
<td>ZN</td>
<td></td>
</tr>
</tbody>
</table>
Parametric Bootstrap
Evaluate True Strength of Interactions

Blue bar: Mean difference in strength measure (flexible – additive)
Red bar: Standard Deviation of strength measure (in additive model)
Hybrid Models: ISLE

- There are many styles of hybrid model
- We focus on just one: Importance Sampled Learning Ensembles
- We start with an ensemble of trees and treat each of them as a new feature
  - Every tree transforms the one or more predictors into a single output
  - Create a data set in which every tree of the ensemble is a "variable"
  - Could be a very large number of such new features (thousands)
- Now use regularized regression to select and reweight the trees
  - Could yield a model with a much smaller set of trees
ISLE Ensemble Compression

GPS regularized regression progress in dashed line
TreeNet Gradient Boosting ensemble solid line
In this example negligible improvement and slight compression
Some other Variations of Bagger

• Decision Forest: Keep training data set fixed but randomly select which predictors to use (a random half of all predictors for any tree)
  o Grow many trees and then combine via voting or averaging
• Bagged Decision Forest: Randomize both over the training records and the allowed predictors
  o Each data set is a different bootstrap sample
  o Each predictor set is a different random half of available predictors
• For binary classification problem vary priors systematically over a broad range
  o Data set remains unchanged
  o Important tree growing parameter is changed systematically
• Random Splitter selection from top $K$ best predictors (Dieterrich)
• Wagging: Exponential distribution assigns non-zero weights to all records
  o Webb and Zheng (2004) IEEE Transactions on Knowledge and Data Engineering
  o Inferior to bagging but designed for systems that must include all records in every tree
What Next?

• Evolution of Classification Modeling
  o Parallel to our regression series and focused on binary response models

• New approaches to Clustering with modern tools of data mining
  o Going beyond K-MEANS and hierarchical clustering

• Faster model development with staged modeling automation
  o How partial automation and pre-packaged experiments can cut days and even weeks off model development schedules

• Overview of Data Mining for beginners
Salford Predictive Modeler SPM

• Download a current version from our website http://www.salford-systems.com

• Version will run without a license key for 10-days

• Request a license key from unlock@salford-systems.com

• Request configuration to meet your needs
  o Data handling capacity
  o Data mining engines made available