From Trees to Forests and Rule Sets
A Unified Overview of Ensemble Methods

Giovanni Seni
G.Seni@ieee.org
PDF Solutions, Inc., Santa Clara University

John Elder
elder@datamininglab.com
Elder Research, Inc.

13th Intl. Conf. on Knowledge Discovery and Data Mining (KDD 2007)

Tutorial Goals

- Convey the essentials of an immensely useful modeling methodology
- Explain two recent developments
  – Importance Sampling
  – Rule Ensembles
- Provide the foundation for further study/research
- Reference resources in R
- Show real examples

Overview

- In a Nutshell, Examples & Timeline
- Predictive Learning
- Decision Trees
  - Regression tree induction
  - Desirable data mining properties
  - Limitations
- Model Selection
  - Bias-Variance Tradeoff
  - Cost-complexity pruning
  - Cross-Validation
  - Regularization via shrinkage (LASSO)

Overview (2)

- Ensemble Methods
  - Ensemble Learning & Importance Sampling (ISLE)
  - Generic Ensemble Generation
  - Bagging, Random Forest, AdaBoost, MART
- Rule Ensembles
- Interpretation
- Example from Industry

Ensemble Methods in a Nutshell

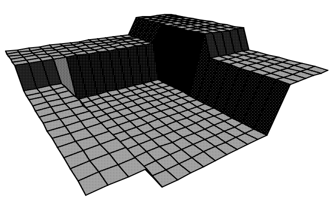
- "Algorithmic" statistical procedure
- Based on combining the fitted values from a number of fitting attempts
- Loosely related to:
  - Iterative procedures
  - Bootstrap procedures
- Original idea: a "weak" procedure can be strengthened if it can operate "by committee"
  - e.g., combining low-bias/ high variance procedures

Ensemble Methods In a Nutshell (2)

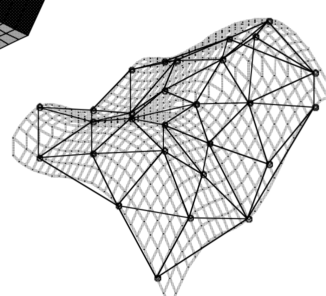
- Shown to perform extremely well in a variety of scenarios
- Shown to have desirable statistical properties
- Accompanied by interpretation methodology

Examples
Data Mining Products

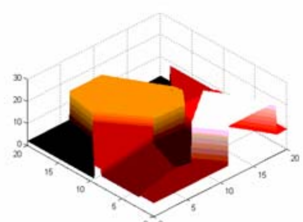
Examples
Algorithm Response Surfaces



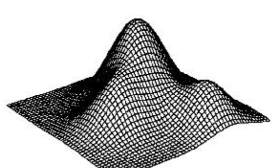
Decision Tree



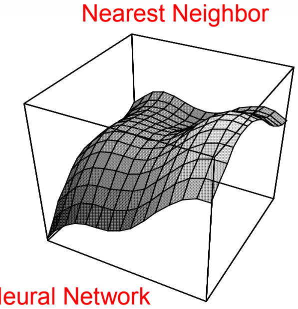
Delaunay Triangles
(or Hinging Hyperplanes)



Nearest Neighbor

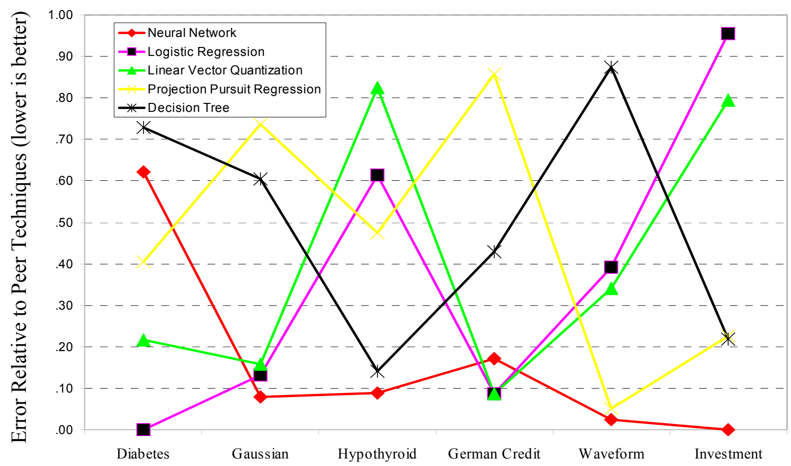


Kernel

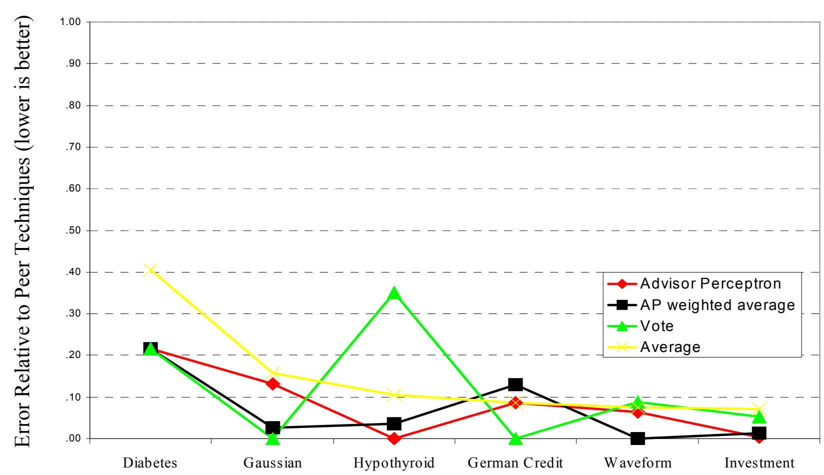


Neural Network
(or Polynomial Network)

Examples
Relative Performance

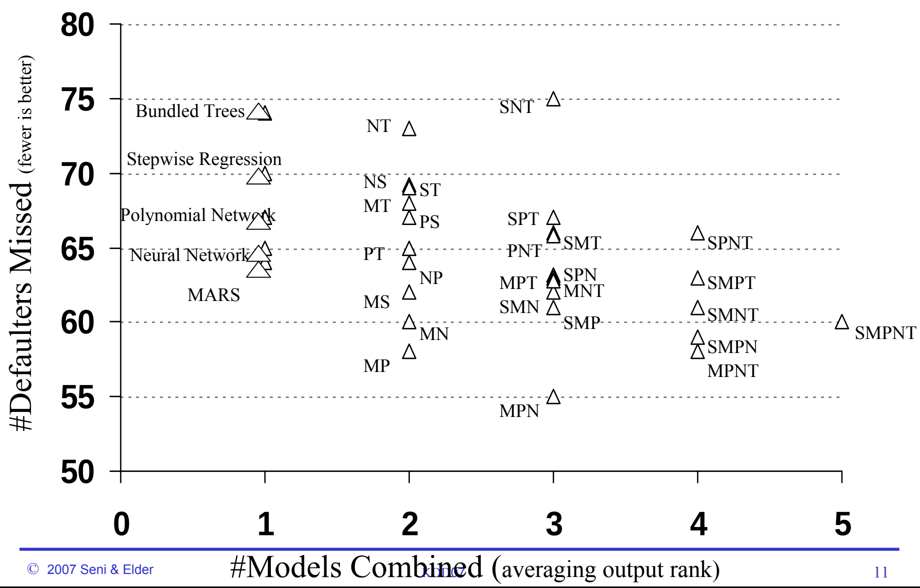


Examples
"Bundling" Improves Performance



Examples

Credit Scoring Performace



Timeline

- CART (Breiman, Friedman, Stone, Olshen, 1983)
- Bagging (Breiman, 1996)
  - Random Forest (Ho, 1995; Breiman 2001)
- AdaBoost (Freund, Schapire, 1997)
- Boosting – a statistical view (Friedman, Hastie, Tibshirani, 2000)
  - Gradient Boosting (Friedman, 2001)
  - Stochastic Gradient Boosting (Friedman, 1999)
- Importance Sampling Learning Ensembles (ISLE) (Friedman, Popescu, 2003)

Timeline (2)

- Lasso (Tibshirani, 1996)
  - Garotte (Breiman, 1995)
  - LARS (Efron, 2004)
  - Gradient directed regularization for linear regression (Friedman, 2004)
  - Boosted Lasso (Zhao, 2005)
  - Elastic Net (Zou, Hastie, 2005)
- Rule Ensembles (Friedman, Popescu, 2005)
  - Stochastic Discrimination (Kleinberg, 2000)

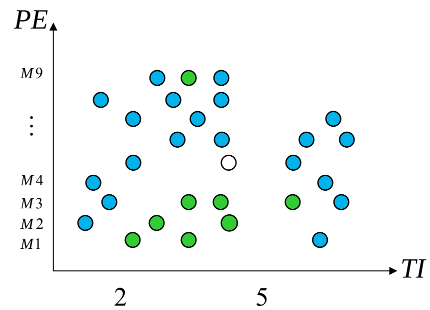
Overview

- In a Nutshell & Timeline
- ➢ Predictive Learning
- Decision Trees
  - Regression tree induction
  - Desirable data mining properties
- Model Selection
  - Bias-Variance Tradeoff
  - Cost-complexity pruning
  - Cross-Validation
  - Regularization via shrinkage (LASSO)

Predictive Learning
Example

- A simple data set

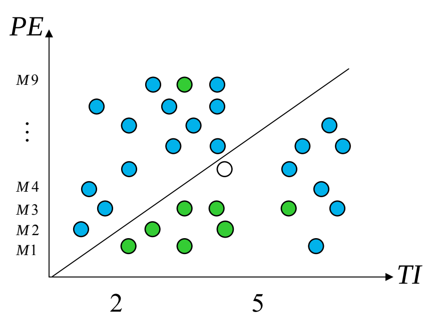
TI	*PE*	*Response*
1.0	*M2*	good
2.0	*M1*	bad
...	...	...
4.5	*M5*	**?**



- Modeling Goals:
  - Descriptive: summarize existing data in understandable and actionable way
  - Predictive: what is the "response" (e.g., class) of new point ○?

Predictive Learning
Example (2)

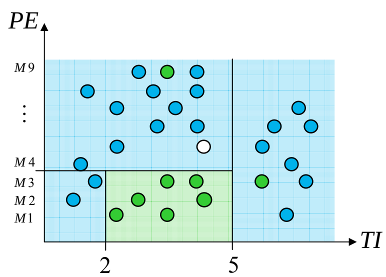
- Ordinary Linear Regression (OLR):

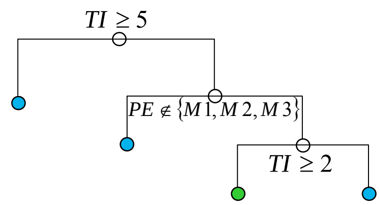


- Model: $\hat{F}(\mathbf{x}) = a_0 + \sum_{j=1}^{n} a_j x_j$ $\quad \hat{F}(\mathbf{x}) \geq 0 \begin{cases} \text{●} \\ \text{else ●} \end{cases}$ $\Rightarrow$ Not flexible enough

Predictive Learning
Example (3)

- Decision Trees





- Descriptive model: $TI \in [2,5]$ AND $PE \in \{M1, M2, M3\} \Rightarrow bad$
  ELSE $good$

Predictive Learning
Procedure Summary

- Given "training" data $D = \{y_i, x_{i1}, x_{i2}, \cdots, x_{in}\}_1^N = \{y_i, \mathbf{x}_i\}$
  - y_i, x_{ij} are measured values of attributes (properties, characteristics) of an object
  - y_i is the "response" (or output) variable
  - x_{ij} are the "predictor" (or input) variables
  - D is a random sample from some unknown (joint) distribution

- Build a functional model $\hat{y} = F(x_1, x_2, \cdots, x_n) = F(\mathbf{x})$
  - Offers adequate and interpretable description of how the inputs affect the outputs

Predictive Learning
Procedure Summary (2)

- *Model*: underlying functional form sought from data

$$\hat{F}(\mathbf{x}) = \hat{F}(\mathbf{x};\mathbf{a}) \in \mathcal{F}$$ family of functions indexed by **a**

- *Score criterion*: judges (lack of) quality of fitted model

$$\widehat{R}(\mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{F}(\mathbf{x}_i;\mathbf{a}))$$ e.g., OLR : $\frac{1}{N}\sum_{i=1}^{N}\left(y_i - a_0 - \sum_{j=1}^{n} a_j x_j\right)^2$

- *Search Strategy*: minimization procedure of criterion

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} \widehat{R}(\mathbf{a})$$

OLR: direct matrix algebra

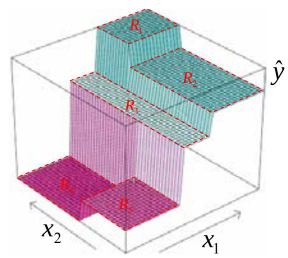
Trees, NN, Clustering : heuristic iterative algorithm

Overview

- In a Nutshell & Timeline
- Predictive Learning
- Decision Trees
  - Regression tree induction
  - Desirable data mining properties
  - Limitations
- Model Selection
  - Bias-Variance Tradeoff
  - Cost-complexity pruning
  - Cross-Validation
  - Regularization via shrinkage (LASSO)

Decision Trees
Induction Overview

- Model: $\hat{y} = T(\mathbf{x}) = \sum_{m=1}^{M} \hat{c}_m I(\mathbf{x} \in \hat{R}_m)$



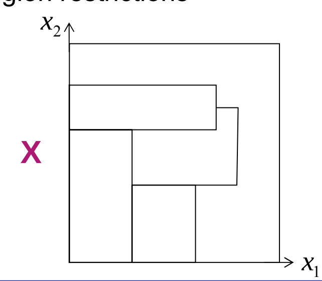
  - R_m : (hyper) rectangles in input space induced by tree cuts
  - c_m : estimated response within each rectangle is constant
  - $I(\cdots)$: indicator function; 1 if its argument is true

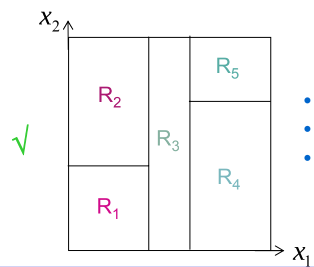
Decision Trees
Induction Overview (2)

- Score criterion:
  - Regression "loss"
    - Squared error: $L(y, \hat{y}) = (y - \hat{y})^2$
    - Absolute error: $L(y, \hat{y}) = | y - \hat{y} |$
  - Prediction "risk"
    - Average loss over all predictions – $R(T) = E_{y,\mathbf{x}} L(y, T(\mathbf{x}))$
- Goal: find tree T with lowest prediction risk

Decision Trees
Induction Overview (3)

- Search
  - Goal: $\{\hat{c}_m, \hat{R}_m\}_1^M = \arg\min_{\{c_m, R_m\}_1^M} \sum_{i=1}^{N} \Big[y_i - \overbrace{\sum_{m=1}^{M} c_m I(\mathbf{x}_i \in R_m)}^{\hat{y}_i = T(\mathbf{x}_i)} \Big]^2$ (least squares)
  - Unrestricted optimization with respect to $\{R_m\}_1^M$ is difficult!
  - Region restrictions





- Disjoint
- Cover space
- "Simple"

Decision Trees
Growing Algorithm

- Greedy Iterative procedure
  - Starting with a single region -- i.e., all given data
  - At the m-th iteration:

for each region R
  for each attribute x_j in R
    for each possible split s_j of x_j
      record change in <u>score</u> when we partition R into R^l and R^r
Choose (x_j, s_j) giving maximum improvement to fit
Replace R with R^l; add R^r

  - i.e., Forward stagewise additive procedure
  - When should we stop?

Decision Trees
Desirable Data Mining Properties

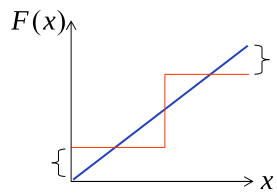
- Ability to deal with irrelevant inputs
  - i.e., automatic variable subset selection
  - Measure anything you can measure
  - Score provided for selected variables ("importance")
- No data preprocessing needed
  - Naturally handle all types of variables
    - numeric, binary, categorical
  - Invariant under monotone transformations: $x_j = g_j(x_j)$
    - Variable scales are irrelevant
    - Immune to bad x_j-distributions (e.g., outliers)

Decision Trees
Desirable Data Mining Properties (2)

- Computational scalability
  - Relatively fast: $O(nN \log N)$
- Missing value tolerant
  - Moderate loss of accuracy due to missing values
  - Handling via "surrogate" splits
- "Off-the-shelf" procedure
  - Few tunable parameters
- Interpretable model representation
  - Binary tree graphic

Decision Trees
Limitations

- Discontinuous piecewise constant model



  - In order to have many splits you need to have a lot of data
  - In high-dimensions, you often run out of data after a few splits
- Data fragmentation
  - Each split reduces training data for subsequent splits

Decision Trees
Limitations (2)

- Not good for <u>low</u> interaction $F^*(\mathbf{x})$
  - e.g., $F^*(\mathbf{x}) = a_o + \sum_{j=1}^{n} a_j x_j$ is worst function for trees

    $= \sum_{j=1}^{n} f_j^*(x_j)$ (no interaction, additive)

  - In order for x_l to enter model, must split on it
    - Path from root to node is a <u>product</u> of indicators

- Not good for $F^*(\mathbf{x})$ that has dependence on many variables

Decision Trees
Limitations (3)

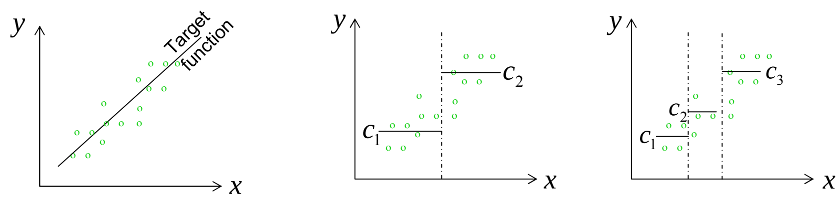
- High variance caused by greedy search strategy (local optima)
  - i.e., Small changes in data (sampling fluctuations) can cause big changes in tree
  - Errors in upper splits are propagated down to affect all splits below it
  - Very deep trees might be questionable
  - Pruning is important
- What to do next?
  - Use other methods when possible
  - Fix-up trees: use "ensembles“
    - Maintain advantages while dramatically increasing accuracy

Overview

- In a Nutshell & Timeline
- Predictive Learning
- Decision Trees
  - Regression tree induction
  - Desirable data mining properties
  - Limitations
- ➢ Model Selection
  - Bias-Variance Tradeoff
  - Cost-complexity pruning
  - Cross-Validation
  - Regularization via shrinkage (LASSO)

Model Selection
What is the “right” size of a tree?

- Dilemma
  - If tree (# of regions) is too small, then piecewise constant approximation is too crude (bias) $\Rightarrow$ increased errors
  - If tree is too large, then it fits the training data too closely (overfitting, increased variance) $\Rightarrow$ increased errors



 KDD07 31

Model Selection
Bias–Variance Decomposition

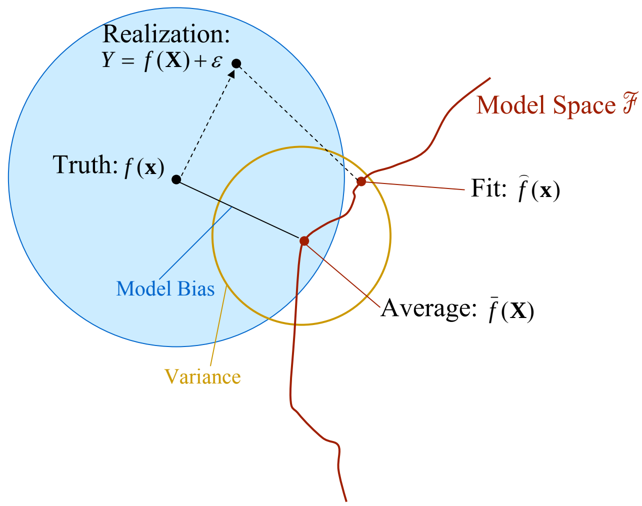
- Suppose $Y = f(\mathbf{X}) + \varepsilon$ where $\varepsilon \sim N(0, \sigma^2)$
- Consider “idealized” aggregate estimator: $\bar{f}(\mathbf{X}) = \mathrm{E}\ \hat{f}_D(\mathbf{X})$
  - Average over all possible data sets
- Prediction error of fit $\hat{f}(\mathbf{X})$ at a point $\mathbf{X} = \mathbf{x}_0$ using *squared-error* loss is decomposable:

$$
\begin{aligned}
Err(\mathbf{x}_0) &= \mathrm{E}\left[Y - \hat{f}(\mathbf{x}_0) \mid \mathbf{X} = \mathbf{x}_0\right]^2 \\
&= \mathrm{E}\left[f(\mathbf{x}_0) - \hat{f}(\mathbf{x}_0)\right]^2 + \sigma^2 \\
&= \mathrm{E}\left[f(\mathbf{x}_0) - \bar{f}(\mathbf{x}_0) + \bar{f}(\mathbf{x}_0) - \hat{f}(\mathbf{x}_0)\right]^2 + \sigma^2 \\
&= \mathrm{E}\left[\bar{f}(\mathbf{x}_0) - f(\mathbf{x}_0)\right]^2 + \mathrm{E}\left[\hat{f}(\mathbf{x}_0) - \bar{f}(\mathbf{x}_0)\right]^2 + \sigma^2 \\
&= \left[\bar{f}(\mathbf{x}_0) - f(\mathbf{x}_0)\right]^2 + \mathrm{E}\left[\hat{f}(\mathbf{x}_0) - \bar{f}(\mathbf{x}_0)\right]^2 + \sigma^2 \\
&= Bias^2(\hat{f}(\mathbf{x}_0)) + Var(\hat{f}(\mathbf{x}_0)) + \sigma^2
\end{aligned}
$$

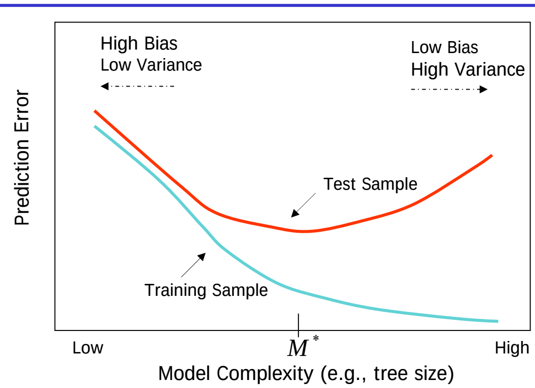
  - Typically, when bias is low, variance is high and vice-versa

 KDD07 32

Model Selection
Bias–Variance Schematic



Model Selection
Bias–Variance Tradeoff



– Right sized tree, M^*, when test error is at a minimum

– Overfitting is exacerbated when model (fitting procedure) is highly flexible

Model Selection

Tree Pruning

- Augmented tree score criterion: $\hat{R}_\alpha(T) = \hat{R}(T) + \alpha \cdot |T|$
  - $\hat{R}(T)$: "risk" of tree – how good it fits the data
    analogous to the residual sum of squares
  - $|T|$: size of tree – number of terminal nodes
    analogous to the model degrees of freedom
  - α : complexity parameter – cost of adding another var to the model

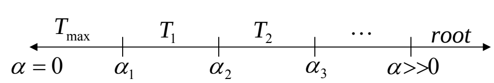
  $\Rightarrow$ Complexity term penalizes for the increased variance associated with more complex model

- Goal: find $T_\alpha^* = \min_T \hat{R}_\alpha(T)$

Model Selection

Tree Pruning (2)

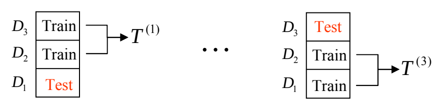
- *Cost-Complexity* Pruning (Cont.)
  - α is a "meta" parameter that controls the degree of stabilization
    - Larger values provides increased regularization $\Rightarrow$ more stable estimates
    - $\alpha = 0$: largest tree ($T_{\max}$) $\Rightarrow$ least stable estimates
    - $\alpha = \infty$: stump (root only) $\Rightarrow$ deterministic
    - Varying α produces a nested (finite) sequence of subtrees



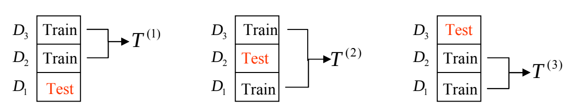
    - Choose the value of α, α^*(and thus $T_{\alpha^*}^*$), to minimize <u>future</u> risk $\hat{R}(T_\alpha)$

Model Selection
Cross–Validation

- Error on the training data is not a useful estimator
  - If test set is not available, need alternative method
- Combine measures of fit to obtain “honest” estimate of model’s “quality”
  - Each measure is constructed from random, non-overlapping subset of the data
  - E.g., 3-fold CV



Model Selection
Cross–Validation (2)



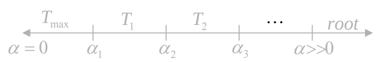
- $T^{(v)}$: tree built on $D - D_v$
- $\upsilon(i) : \{1,\ldots,N\} \mapsto \{1,\ldots,V\}$: indexing function
- $\hat{R}^{CV} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, T^{\upsilon(i)}(\mathbf{x}_i)\right)$: cross-validation estimate of prediction error

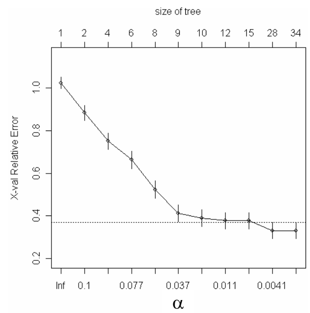
  i.e., average of the *fit measures* over the V splits

Model Selection
Cross−Validation (3)

- Set of models $\{T^{(v)}(\mathbf{x})\}$ is indexed by tuning parameter α, and we have

$$\hat{R}^{CV}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, T_\alpha^{\upsilon(i)}(\mathbf{x}_i)\right)$$



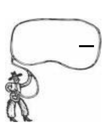


- Idea generalization: can "averaging" over a set of trees help correct for overly optimistic trees?
  – i.e., combine fitted values instead of measures of fit ⇒ Ensembles

Model Selection
Regularization via Shrinkage (*Lasso* − Tibshirani, 1996)

- Consider linear model $F(\mathbf{x}) = a_0 + \sum_{j=1}^{n} a_j x_j$
- Standard LR coefficients estimation $\{\hat{a}_j\} = \underset{\{a_j\}}{\arg\min} \sum_{i=1}^{N} L\left(y_i, \sum_{j=1}^{n} a_j x_{ij}\right)$
- Shrinking: continuous variable selection method

$$\{\hat{a}_j^{Lasso}\} = \underset{\{a_j\}}{\arg\min} \sum_{i=1}^{N} L\left(y_i, \sum_{j=1}^{n} a_j x_{ij}\right) + \lambda \sum_{j=1}^{n} |a_j|$$



  – "penalty" term provides a (deterministic) value that is independent of the particular random sample

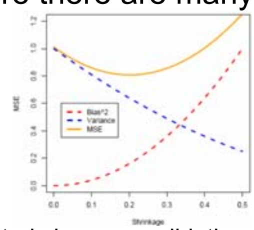
  ⇒ stabilizing influence on the criterion being minimized

  – promotes reduced variance of the estimated coefficient values

Model Selection
Regularization via Shrinkage (2)

- Lasso solutions nonlinear in y, and a quadratic programming algorithm is used to compute them

- L1 penalty leads to sparse solutions where there are many predictors with $a_j = 0$
  - Often improves prediction accuracy



  - Easier to interpret
  - "Bet on sparsity" principle

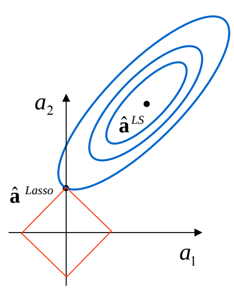
- λ is a "tuning" parameter $\Rightarrow$ Typically estimated via cross-validation
  - $\lambda = 0$: no regularization… least stable estimates
  - $\lambda > 0$: continuous coefficient "paths"

Model Selection
Regularization via Shrinkage (3)

- Lasso 2D estimation illustration
  - $L(\cdot)$: least-squares -- i.e.,

$$\{\hat{a}_0, \hat{a}_1, \hat{a}_2\} = \arg\min_{\{a_j\}} \sum_{i=1}^{N} \left(y_i - a_0 - a_1 x_{i1} - a_2 x_{i2}\right)^2 + \lambda\left(|a_1| + |a_2|\right)$$

  - Error function contours and constraint region



If first point where contours hit constraint region occurs at a corner, then one parameter a_j is zero

Model Selection
Regularization via Shrinkage (4)

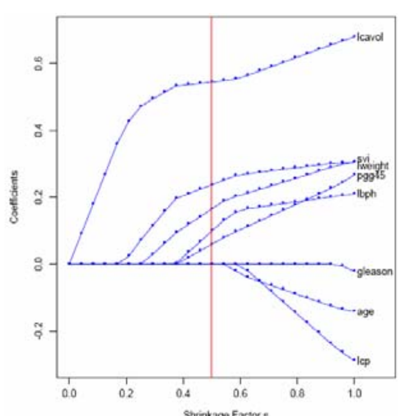
- Forward stagewise linear regression (Hastie et al, 2001)
  - Iterative strategy producing solutions that closely approximates the effect of the lasso
    - After $M < \infty$ iterations, many $\hat{a}_j$ will be 0
    - In general, $|\hat{a}_j(M)| < |\hat{a}_j^{LS}|$
    - $M \approx 1/\lambda$

Initialize $\hat{a}_j = 0$; $\varepsilon > 0$; M large
For $m = 1$ to M {
// Select predictor that best fits current residuals
$$\left(\alpha^*, j^*\right) = \arg\min_{\{\alpha, j\}} \sum_{i=1}^{N} \left[y_i - \sum_{l=1}^{n} \hat{a}_l x_{il} - \alpha x_{ij} \right]^2$$
// Increment/decrement a_{j^*} by infinitesimal amount
$$a_{j^*} \leftarrow a_{j^*} + \varepsilon \cdot sign(\alpha^*)$$
}
write $\hat{F}(\mathbf{x}) = \sum_{j=1}^{n} \hat{a}_j x_j$

  - See also LARS (Efron, et al 2004), Gradient directed regularization for linear regression (Friedman, 2005)

Model Selection
Regularization via Shrinkage (5)

- Lasso coefficient “paths” example:

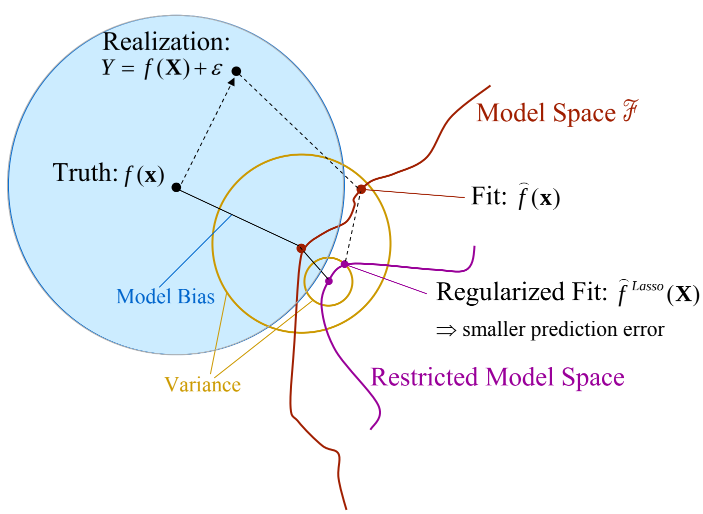


  - $s \approx 1/\lambda$
  - see R package "lars"

Model Selection
Regularization Summary

- What is *regularization*?
  - "any part of model building which takes into account – implicitly or explicitly – the finiteness and imperfection of the data and the limited information in it, which we can term "variance" in an abstract sense" [Rosset, 2003]
- Forms of regularization
  - Explicit constraints on model complexity
    - Constrained models can equivalently be cast as penalized ones
  - Implicit through incremental building of the model
  - Choice of robust loss functions

Model Selection
Regularization Summary (2)



Overview

- Ensemble Methods
  - Ensemble Learning & Importance Sampling (ISLE)
  - Generic Ensemble Generation
  - Bagging, Random Forest, AdaBoot, MART
- Rule Ensembles
- Interpretation

Ensemble Methods
Learning

- Model: $F(\mathbf{x}) = c_0 + \sum_{m=1}^{M} c_m T_m(\mathbf{x})$
  - $\{T_m(\mathbf{x})\}_1^M$: "basis" functions (or "base learners")
  - i.e., linear model in a (very) high dimensional space of derived variables
- Learner characterization: $T_m(\mathbf{x}) = T(\mathbf{x}; \mathbf{p}_m)$
  - $\mathbf{p}_m$: a specific set of joint parameter values – e.g., split definitions at internal nodes and predictions at terminal nodes
  - $\{T(\mathbf{x}; \mathbf{p})\}_{\mathbf{p} \in P}$: function class – i.e., set of all base learners of specified family

Ensemble Methods
Learning (2)

- Two-step process – approximate solution to

$$\{\hat{c}_m, \hat{\mathbf{p}}_m\}_o^M = \min_{\{c_m, \mathbf{p}_m\}_o^M} \sum_{i=1}^{N} L\left(y_i, c_0 + \sum_{m=1}^{M} c_m T_m(\mathbf{x}; \mathbf{p}_m)\right)$$

  – Step 1: Choose points $\{\mathbf{p}_m\}_1^M$

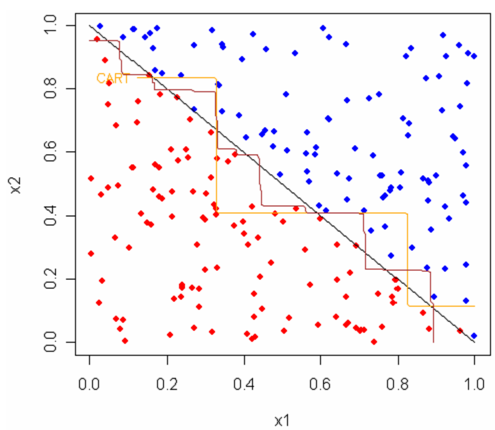
    - i.e., select $\{T_m(\mathbf{x})\}_1^M \subset \{T(\mathbf{x};\mathbf{p})\}_{\mathbf{p}\in P}$

  – Step 2: Determine weights $\{c_m\}_0^M$

    - e.g., via regularized LR

Ensemble Methods
Example

- 2 features, 2 classes



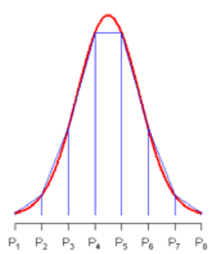
⇒ Another example in Appendix 1

Ensemble Methods

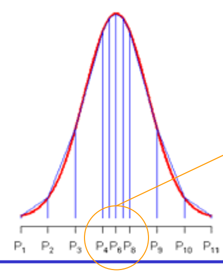
Importance Sampling (Friedman, 2003)

- How to judiciously choose the "basis" functions (i.e., $\{\mathbf{p}_m\}_1^M$)?
- Goal: find "good" $\{\mathbf{p}_m\}_1^M$ so that $F(\mathbf{x}; \{\mathbf{p}_m\}_1^M, \{a_m\}_1^M) \cong F^*(\mathbf{x})$
- Connection with numerical integration:
  - $\int_{\mathrm{P}} I(\mathbf{p})\,\partial\mathbf{p} \approx \sum_{m=1}^{M} w_m I(\mathbf{p}_m)$

$$F(\mathbf{x}) = \int_{\mathrm{P}} a(\mathbf{p}) T(\mathbf{x};\mathbf{p})\,d\mathbf{p}$$
$$\cong \sum_{m=1}^{M} w_m a(\mathbf{p}_m) T(\mathbf{x};\mathbf{p}_m)$$
$$= \sum_{m=1}^{M} c_m T(\mathbf{x};\mathbf{p}_m)$$



vs.



Accuracy improves when we choose more points from this region…

Ensemble Methods

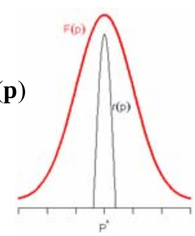
Importance Sampling (2)

- *Parameter importance* measure: $r(\mathbf{p})$
  - Indicative of relevance of $\mathbf{p}$
  - Naive approach: $r(\mathbf{p}_m)$ *iid* -- i.e., uniform
  - Better idea: inversely related to $\mathbf{p}_m$'s "risk"
    - i.e., $T(\mathbf{x};\mathbf{p}_m)$ has high error $\Rightarrow$ lack of relevance of $\mathbf{p}_m$
  - Single point vs. group importance
    - i.e., with/out knowledge of the other points that will be used
    - Sequential approximation: $\mathbf{p}$'s relevance judged in the context of the (fixed) previously selected points
  - Characterization of $r(\mathbf{p})$: "center" and "width"

Ensemble Methods

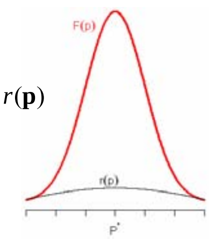
Importance Sampling (3)

Narrow $r(\mathbf{p})$



$\mathbf{p}^* = \arg\min_{\mathbf{p}} Risk(\mathbf{p})$

Broad $r(\mathbf{p})$



- Ensemble $\{T(\mathbf{x};\mathbf{p}_m)\}_1^M$ of "strong" base learners – i.e., all with $Risk(\mathbf{p}_m) \approx Risk(\mathbf{p}^*)$
- $T(\mathbf{x};\mathbf{p}_m)$'s yield similar highly correlated predictions $\Rightarrow$ unexceptional performance

- Diverse ensemble – i.e., predictions are not highly correlated with each other
- However, many "weak" base learners – i.e., $Risk(\mathbf{p}_m) >> Risk(\mathbf{p}^*)$ $\Rightarrow$ poor performance

$\Rightarrow$ The empirically observed strength-correlation tradeoff can be understood in terms of the width of $r(\mathbf{p})$

Ensemble Methods

Importance Sampling (4)

- *Heuristic sampling strategy* $\hat{r}(\mathbf{p})$: sampling around $\mathbf{p}^*$ by iteratively applying small perturbations to existing problem structure
  - Generating ensemble members $T_m(\mathbf{x}) = T(\mathbf{x};\mathbf{p}_m)$

$$
\begin{array}{l}
\text{For } m = 1 \text{ to } M \ \{ \\
\quad \mathbf{p}_m = PERTURB_m\{\arg\min_{\mathbf{p}} E_{xy} L(y, T(\mathbf{x};\mathbf{p}))\} \\
\}
\end{array}
$$

  - $PERTURB\{\cdot\}$ is a (random) modification of any of
    - Data distribution – e.g., by re-weighting the observations
    - Loss function – e.g., by modifying its argument
    - Search algorithm (used to find $\min_p$)
  - Width of $\hat{r}(\mathbf{p})$ is controlled by degree of perturbation

Ensemble Methods

Generic Ensemble Generation (Friedman, 2003)

- Step 1: Choose $\{\mathbf{p}_m\}_1^M$:

Forward Stagewise Fitting Procedure

$$\begin{aligned}
&F_0(\mathbf{x}) = 0\\
&\text{For } m = 1 \text{ to } M \ \{\\
&\quad \mathbf{p}_m = \arg\min_{\mathbf{p}} \sum_{i \in S_m(\eta)} L\big(y_i, F_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i; \mathbf{p})\big)\\
&\quad T_m(\mathbf{x}) = T(\mathbf{x}; \mathbf{p}_m)\\
&\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot T_m(\mathbf{x})\\
&\}\\
&\text{write } \{T_m(\mathbf{x})\}_1^M
\end{aligned}$$

Modification of data distribution

Modification of loss function ("sequential" approximation)

- Algorithm control: L, η, υ
  - $S_m(\eta)$: random sub-sample of size $\eta \le N$ $\Rightarrow$ impacts ensemble "diversity"
  - $F_{m-1}(\mathbf{x}) = \upsilon \cdot \sum_{k=1}^{m-1} T_k(\mathbf{x})$: "memory" function ($0 \le \upsilon \le 1$)

Ensemble Methods

Generic Ensemble Generation (2)

- Step 2: Choose quadrature coefficients $\{c_m\}_0^M$:
  - Given $\{T_m(\mathbf{x})\}_{m=1}^M = \{T(\mathbf{x}; \mathbf{p}_m)\}_{m=1}^M$, coefficients are obtained by a regularized linear regression

$$\{\hat{c}_m\} = \arg\min_{\{c_m\}} \sum_{i=1}^{N} L\left(y_i,\ c_0 + \sum_{m=1}^{M} c_m T_m(\mathbf{x}_i)\right) + \lambda \sum_{m=1}^{M} |c_m|$$

  - Regularization here helps reduce bias (in addition to variance) of the model
  - New iterative fast algorithms for various loss functions
    - "Gradient Direct Regularization…", Friedman & Popescu, 2004

Ensemble Methods
Bagging (Breiman, 1996)

- Bagging = Bootstrap Aggregation

  - $L(y,\hat{y}) = (y-\hat{y})^2$
  - $\upsilon = 0 \;\Rightarrow$ no memory
  - $\eta = N/2$
  - $T_m(\mathbf{x}) \;\Rightarrow$ are large un-pruned trees
  - $c_o = 0, \quad \{c_m = 1/M\}_1^M$
    i.e., not fit to the data (avg)

$$
\begin{aligned}
&F_0(\mathbf{x}) = 0\\
&\text{For } m = 1 \text{ to } M \;\{\\
&\quad \mathbf{p}_m = \arg\min_{\mathbf{p}} \sum_{i \in S_m(\eta)} L\big(y_i, F_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i;\mathbf{p})\big)\\
&\quad T_m(\mathbf{x}) = T(\mathbf{x};\mathbf{p}_m)\\
&\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot T_m(\mathbf{x})\\
&\}\\
&\text{write } \{T_m(\mathbf{x})\}_1^M
\end{aligned}
$$

  - i.e., perturbation of the data distribution only
  - Parallelizable
  - See R command *ipred*:ipredbagg

Ensemble Methods
Bagging – Example

- Simulated data
  - Predictor vars:

$$
n = 5 \qquad p(\mathbf{x}) = N\left(\begin{bmatrix}0\\ \vdots \\ 0\end{bmatrix}, \begin{bmatrix}1 & 0.95 & 0.95\\ 0.95 & \ddots & 0.95\\ 0.95 & 0.95 & 1\end{bmatrix}\right)
$$

  - Response var:

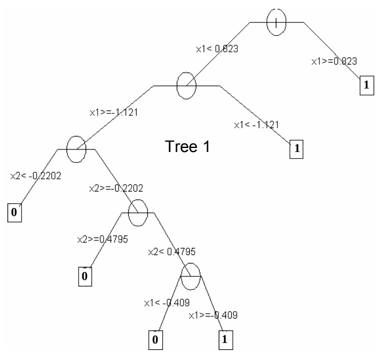
two-class; generated according to: $P(y=1 \mid x_1 \le 0.5) = 0.2$, $P(y=1 \mid x_1 > 0.5) = 0.8$ $\Rightarrow$ Bayes error rate : 0.2

  - Training: $N = 30$
    - 200 bootstrap samples
  - Testing: $N = 2000$

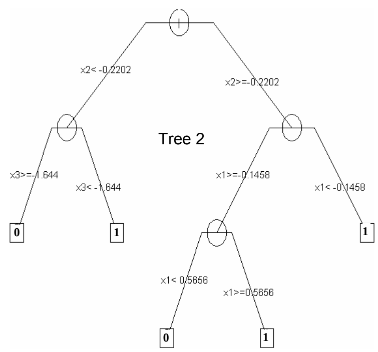
Ensemble Methods

Bagging – Example (2)

- Bootstrap trees



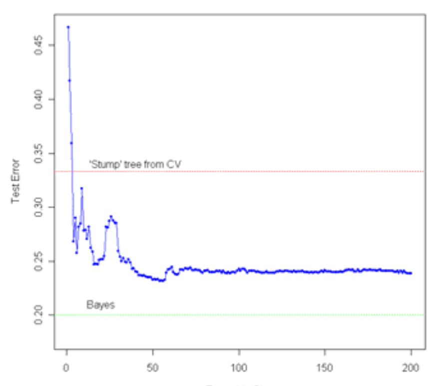
. . .



Ensemble Methods

Bagging – Example (3)

- Test Error



  – Trees have high variance on these data due to the correlation in the predictors

  – Bagging succeeds in smoothing out this variance and hence reducing test error

Ensemble Methods

Bagging – Why it helps?

- Under $L(y,\hat{y}) = (y-\hat{y})^2$ averaging reduces variance and leaves bias unchanged
- Consider "idealized" bagging (aggregate) estimator: $\bar{f}(\mathbf{x}) = \mathrm{E}\,\hat{f}_Z(\mathbf{x})$
  - $\hat{f}_Z$ fit to bootstrap data set $Z = \{y_i, x_i\}_1^N$
  - Z is sampled from actual population distribution (not training data)
  - We can write:

$$\begin{aligned}\mathrm{E}\left[Y - \hat{f}_Z(\mathbf{x})\right]^2 &= \mathrm{E}\left[Y - \bar{f}(\mathbf{x}) + \bar{f}(\mathbf{x}) - \hat{f}_Z(\mathbf{x})\right]^2 \\ &= \mathrm{E}\left[Y - \bar{f}(\mathbf{x})\right]^2 + \mathrm{E}\left[\hat{f}_Z(\mathbf{x}) - \bar{f}(\mathbf{x})\right]^2 \\ &\geq \mathrm{E}\left[Y - \bar{f}(\mathbf{x})\right]^2\end{aligned}$$

$\Rightarrow$ true population aggregation never increases mean squared error!

$\Rightarrow$ Bagging will often decrease MSE…

Ensemble Methods

Random Forest (Breiman, 2001)

- Bagging + algorithm randomizing
  - Subset splitting

    As each tree is constructed…
    - Draw a random sample of predictors before each node is split

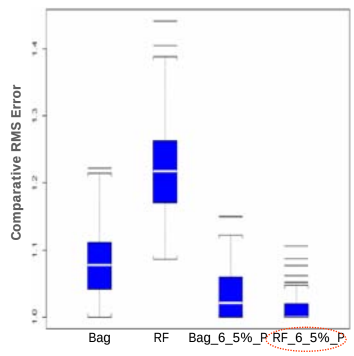
      $n_s = \lfloor \log_2(n) + 1 \rfloor$
    - Find best split as usual but selecting only from subset of predictors

  $\Rightarrow$ Increased diversity among $\{T_m(\mathbf{x})\}_1^M$ – i.e., wider $\hat{r}(\mathbf{p})$
    - Width (inversely) controlled by n_s
  - Speed improvement over Bagging

Ensemble Methods

Random Forest (2)

- Potential improvements:
  - Different data sampling strategy (not fixed)
  - Fit quadrature coefficients to data



- Simulation study with 100 different target functions (Popescu, 2005)
- xxx_6_5%_P : 6 terminal nodes trees
  5% samples without replacement
  Post-processing – i.e., using estimated "optimal" quadrature coefficients
- xxx_6_5%_P : Significantly faster to build!

- See R command randomForest

Ensemble Methods

AdaBoost (Freund & Schapire, 1997)

- AdaBoost = Adaptive Boosting
  - $L(y,\hat{y}) = \exp(-y \cdot \hat{y}); \quad y \in \{-1, 1\}$
  - $\upsilon = 1 \;\Rightarrow$ *Sequential* sampling
  - $\eta = N \;\Rightarrow$ Data perturbed via observation weights
  - $T_m(\mathbf{x}) \;\Rightarrow$ Any "weak" learner
  - $c_o = 0, \quad \{c_m\}_1^M \;\Rightarrow$ Sequential partial regression coefficients
  - $\hat{y} = sign(F_M(\mathbf{x})) = sign\left(\sum_{m=1}^{M} c_m T_m(\mathbf{x})\right)$

$F_0(\mathbf{x}) = 0$
For $m = 1$ to M {
  $(c_m, \mathbf{p}_m) = \arg\min_{c,\mathbf{p}} \sum_{i \in S_m(\eta)} L\left(y_i, F_{m-1}(\mathbf{x}_i) + c \cdot T(\mathbf{x}_i; \mathbf{p})\right)$
  $T_m(\mathbf{x}) = T(\mathbf{x}; \mathbf{p}_m)$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot c_m \cdot T_m(\mathbf{x})$
}
write $\{c_m, T_m(\mathbf{x})\}_1^M$

- "Real" AdaBoost : extension to regression case (Friedman, 2000)
- See R command *boost*:adaboost

Ensemble Methods
AdaBoost (2)

- Equivalence to Forward Stagewise Fitting Procedure

$$
\begin{array}{l}
\text{observation weights: } w_i^{(0)} = 1/N \\
\text{For } m = 1 \text{ to } M \ \{ \\
\quad \text{a. Fit a classifier } T_m(\mathbf{x}) \text{ to training data with } w_i^{(m)} \\
\quad \text{b. Compute} \\
\qquad err_m = \dfrac{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(\mathbf{x}_i))}{\sum_{i=1}^N w_i^{(m)}} \\
\quad \text{c. Compue } \alpha_m = \log((1 - err_m)/err_m) \\
\quad \text{d. Set } w_i^{(m+1)} = w_i^{(m)} \cdot \exp[\alpha_m \cdot I(y_i \neq T_m(\mathbf{x}_i)] \\
\} \\
\text{Output } sign\left(\sum_{m=1}^M \alpha_m T_m(\mathbf{x})\right)
\end{array}
$$

$$
\begin{array}{l}
F_0(\mathbf{x}) = 0 \\
\text{For } m = 1 \text{ to } M \ \{ \\
\quad (c_m, \mathbf{p}_m) = \arg\min_{c,\mathbf{p}} \sum_{i \in S_m(\eta)} L\left(y_i, F_{m-1}(\mathbf{x}_i) + c \cdot T(\mathbf{x}_i; \mathbf{p})\right) \\
\quad T_m(\mathbf{x}) = T(\mathbf{x}; \mathbf{p}_m) \\
\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot c_m \cdot T_m(\mathbf{x}) \\
\} \\
\text{write } \{c_m, T_m(\mathbf{x})\}_1^M
\end{array}
$$

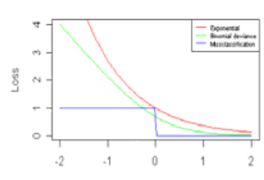
- We need to show $\mathbf{p}_m = \arg\min_{\mathbf{p}} (\cdot)$ is equivalent to line a. above
- $c_m = \arg\min_{c} (\cdot)$ is equivalent to line c.
- How weights $w_i^{(m)}$ are derived/updated

Appendix2

Ensemble Methods
AdaBoost (3)

- Why exponential loss?
  - Implementation convenience
  - Meaningful population minimizer: $F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \mathrm{E}_{Y|\mathbf{x}}\left(e^{-Y \cdot F(\mathbf{x})}\right) = \frac{1}{2} \log \frac{\Pr(Y=1|\mathbf{x})}{\Pr(Y=-1|\mathbf{x})}$
    - i.e., half the log-odds of $\Pr(Y=1|\mathbf{x})$ Appendix3
    - Equivalently, $\Pr(Y=1|\mathbf{x}) = \dfrac{1}{1+e^{-2 \cdot F^*(\mathbf{x})}}$
    - Same minimizer obtained with (negative) binomial log-likelihood

      $\mathrm{E}_{Y|\mathbf{x}}\left(-l(Y, F(\mathbf{x})\right) = \mathrm{E}_{Y|\mathbf{x}}\left(\log(1 + e^{-2Y \cdot F(\mathbf{x})}\right)$
    - However, exponential loss is less robust in noisy settings
    - No natural generalization to K classes



Ensemble Methods

Gradient Boosting (Friedman, 2001)

- Boosting with any differentiable loss criterion
  - General $L(y, \hat{y})$ and y
  - $\upsilon = 0.1 \Rightarrow$ *Sequential* sampling
  - $\eta = N/2$
  - $T_m(\mathbf{x}) \Rightarrow$ Any "weak" learner
  - $c_o = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$
  - $\{c_m\}_1^M \Rightarrow$ "shrunk" sequential partial regression coefficients
  - $\hat{y} = F_M(\mathbf{x})$

$F_0(\mathbf{x}) = c_0$
For $m = 1$ to M {
  $(c_m, \mathbf{p}_m) = \arg\min_{c,\mathbf{p}} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + c \cdot T(\mathbf{x}_i; \mathbf{p}))$
  $T_m(\mathbf{x}) = T(\mathbf{x}; \mathbf{p}_m)$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot c_m \cdot T_m(\mathbf{x})$
}
write $\{(\upsilon \cdot c_m), T_m(\mathbf{x})\}_1^M$

Appendix 4

– See R command gbm

Ensemble Methods

Gradient Boosting / MART

- MART = Multiple Additive Regression Trees
  – $T_m(\mathbf{x}) \Rightarrow$ *J*–terminal node regression tree ($J_m = J \ \forall m$)
    - *J* is a meta-parameter that controls interaction order allowed by approximation

      $J = 2 \Rightarrow$ "main-effects" only

      $J = 3 \Rightarrow$ two-variable interactions allowed (i.e., second-order effects)

      $\cdots \Rightarrow$ value of *J* should reflect dominant interaction level of target function

  – Adding $T_m(\mathbf{x})$ to model is like adding *J* separate (basis) functions

$$T\left(\mathbf{x}; \{b_j, R_j\}_1^J\right) = \sum_{j=1}^{J} b_j I(\mathbf{x} \in R_j) \ \Rightarrow \ F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + c_m \cdot T_m(\mathbf{x}) \ = \ F_{m-1}(\mathbf{x}) + c_m \cdot \sum_{j=1}^{J} b_{jm} I(\mathbf{x} \in R_{jm})$$

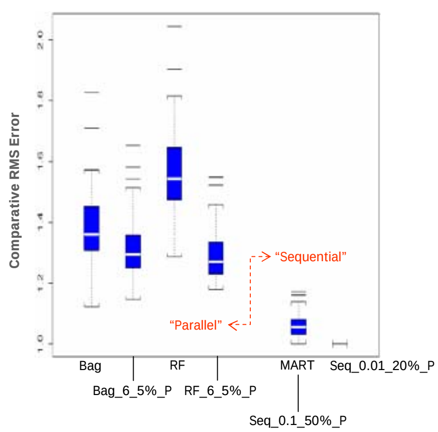
$$= F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \gamma_{jm} I(\mathbf{x} \in R_{jm}) \quad \text{with} \quad \gamma_{jm} = c_m \cdot b_{jm}$$

$\Rightarrow$ computational efficient update Appendix 5

Ensemble Methods
Gradient Boosting / MART (2)

- GB/MART doesn't choose quadrature coefficients in a separate step
  - i.e., ISLE recipe: $\{\hat{c}_m\} = \arg\min_{\{c_m\}} \sum_{i=1}^{N} L\left(y_i,\ c_0 + \sum_{m=1}^{M} c_m T_m(\mathbf{x}_i)\right) + \lambda \sum_{m=1}^{M} |c_m|$
- However, boosting is still understood as an "incremental forward stagewise regression" procedure with Lasso penalty (shrinkage controlled by υ)
  - Note similarity with Forward Stagewise Linear Regression procedure with $\{T_m(\mathbf{x})\}_1^M$ as predictors

Ensemble Methods
Parallel vs. Sequential Ensembles



- Simulation study with 100 different target functions (Popescu, 2005)
- xxx_6_5%_P : 6 terminal nodes trees
  5% samples without replacement
  Post-processing – i.e., using estimated "optimal" quadrature coefficients
- Seq_η_ν%_P : "Sequential" ensemble
  6 terminal nodes trees
  η: "memory" factor
  ν% samples without replacement
  Post-processing

- Sequential ISLE tend to perform better than parallel ones
  - Consistent with results observed in classical Monte Carlo integration

Overview

- Ensemble Methods
  - Ensemble Learning & Importance Sampling (ISLE)
  - Generic Ensemble Generation
  - Bagging, Random Forest, AdaBoot, MART
- Rule Ensembles
- Interpretation

Ensemble Methods

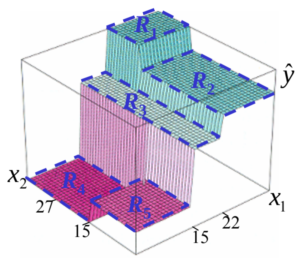
Rule Ensembles (Friedman & Popescu, 2005)

- Ensemble Recap:
  - Model: $F(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m f_m(\mathbf{x})$
  - $\{f_m(\mathbf{x})\}_1^M$: "basis" functions (or "base learners")
    - Derived predictors capture non-linearities and interactions
  - 2-stage fitting process:
    - i. generate basis functions
    - Ii. Post fit to the data via regularized regression
  - Simple representation offered by a single tree no longer available

Ensemble Methods

Rule Ensembles (2)

- Trees as collection of conjunctive rules: $T_m(\mathbf{x}) = \sum_{j=1}^{J} \hat{c}_{jm} I(\mathbf{x} \in \hat{R}_{jm})$



$$R_1 \Rightarrow r_1(\mathbf{x}) = I(x_1 > 22) \cdot I(x_2 > 27)$$

$$R_2 \Rightarrow r_2(\mathbf{x}) = I(x_1 > 22) \cdot I(0 \le x_2 \le 27)$$

$$R_3 \Rightarrow r_3(\mathbf{x}) = I(15 < x_1 \le 22) \cdot I(0 \le x_2)$$

$$R_4 \Rightarrow r_4(\mathbf{x}) = I(0 \le x_1 \le 15) \cdot I(x_2 > 15)$$

$$R_5 \Rightarrow r_5(\mathbf{x}) = I(0 \le x_1 \le 15) \cdot I(0 \le x_2 \le 15)$$

  – These simple rules, $r_m(\mathbf{x}) \in \{0,1\}$, can be used as base learners

  – Main motivation is *interpretability*

Ensemble Methods

Rule Ensembles (3)

- Rule-based model: $F(\mathbf{x}) = a_0 + \sum_m a_m r_m(\mathbf{x})$
  – Still a piecewise constant model
    • Linear targets can still be problematic…

  – We can complement the non-linear rules with purely linear terms:

$$\boxed{F(\mathbf{x}) = a_0 + \sum_m a_m r_m(\mathbf{x}) + \sum_j b_j x_j}$$

    • Original continuous variables x_j can be replaced by their "winzorized" versions

Ensemble Methods
Rule Ensembles (4)

- Rule generation
  - Use some approximate optimization approach to solve

$$\mathbf{p}_m = \arg\min_{\mathbf{p}} \sum_{i \in S_m(\eta)} L\big(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i; \mathbf{p})\big)$$

    where $\mathbf{p}_m$ are the split definitions for rule $r_m(\mathbf{x})$

  - Take advantage of a decision tree ensemble
    - E.g., one rule for each terminal node in each tree $T_m(\mathbf{x})$
    - In the case of shallow trees (boosting), regions corresponding to non-terminal nodes can also be included
      - Each J –terminal node tree generates $2 \times (J-1)$ rules

Ensemble Methods
Rule Ensembles (5)

- Rule fitting
  - Linear regularized procedure

$$(\{\hat{a}_k\}, \{\hat{b}_j\}) = \arg\min_{\{a_k\},\{b_j\}} \sum_{i=1}^{N} L\Big(y_i,\ a_0 + \sum_{k=1}^{K} a_k r_k(\mathbf{x}_i) + \sum_{j=1}^{p} b_j x_{ij}\Big) + \lambda\Big(\sum_{k=1}^{K} |a_k| + \sum_{j=1}^{p} |b_j|\Big)$$

    - $K = \sum_{m=1}^{M} 2 \times (J_m - 1)$ total number of rules
    - $p \le n$ total number of linear terms
  - Tree size controls rule “complexity”
    - A J –terminal node tree can generate rules involving up to $(J-1)$ factors
    - Modeling J –order interactions requires rules with J or more factors

Overview

- Ensemble Methods
  - Ensemble Learning & Importance Sampling (ISLE)
  - Generic Ensemble Generation
  - Bagging, Random Forest, AdaBoot, MART
- Rule Ensembles
- ➢ Interpretation

Ensemble Methods
Interpretation

- Importance scores
  - Which particular variables are the most influential?
    - i.e., relative influence or contribution in predicting the response
  - Higher importance variables are more likely to be of interest
  - Detection of "masking"
- Interaction statistic
  - Which variables are involved in interactions with other variables?
  - Strength and degrees of those interactions
- Partial dependence plots
  - Nature of the dependence of the response on influential inputs

Appendix 6

Ensemble Methods

Interpretation − Example

- Simulated data
  - Predictor vars: $n = 10$

$$p(x_1 = -1) \;=\; p(x_1 = 1) \;=\; 1/2$$

$$p(x_m = -1) \;=\; p(x_m = 0) \;=\; p(x_m = 1) \;=\; 1/3 \qquad m = 2,\ldots,10$$

  - Response var:

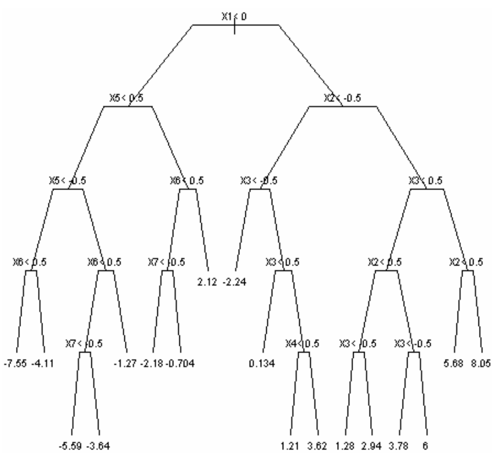
$$y = \begin{cases} 3 + 3 \cdot x_2 + 2 \cdot x_3 + x_4 + z & x_1 = 1 \\ -3 + 3 \cdot x_5 + 2 \cdot x_6 + x_7 + z & x_1 = -1 \end{cases} \qquad \begin{array}{c} z \sim N(0,2) \\ x_8, x_9, x_{10} \text{ are "noise"} \end{array}$$

  - Training: $N = 200$

Ensemble Methods

Interpretation − Example (2)

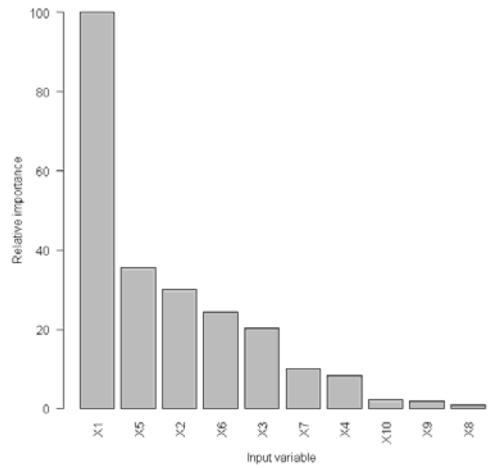
- Single tree vs. rule-ensemble



Imp	Coef	Supp	Rule
100	1.80	0.30	X1 = 1 & X2 ∈{0, 1}
95	-2.25	0.14	X1 = -1 & X5 ∈{ -1}
83	-1.59	0.24	X1 = 1 & X2 ∈{-1, 0} & X3 ∈{-1, 0}
71	-1.00	0.35	X1 = -1 & X6 ∈ {-1}
62	1.37	0.17	X1 = 1 & X2 ∈{0, 1} & X3 ∈ {0, 1}
57	1.25	0.35	X1 = -1 & X6 ∈ {-1, 0}
46	1.00	0.11	X1 = -1 & X5 ∈ {1} & X7 ∈ {0, 1}
42	0.91	0.51	X1 = 1 & X4 ∈ {1}
40	-0.67	0.51	X1= -1

Ensemble Methods
Interpretation − Example (3)

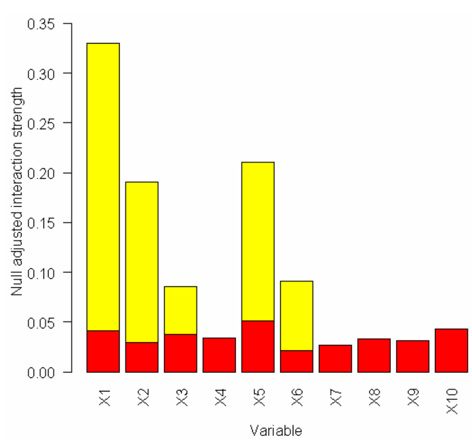
- Variable importance



  - See R package "rulefit"

Ensemble Methods
Interpretation − Example (4)

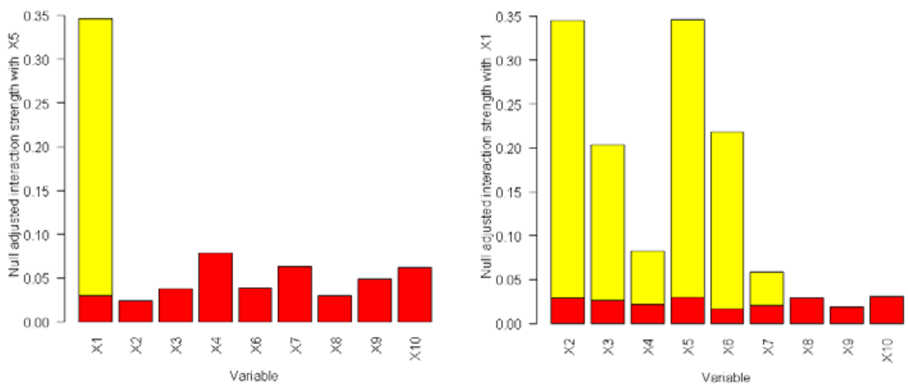
- Interaction statistic – "null" adjusted



Ensemble Methods

Interpretation – Example (5)

- Two-variable interaction statistic: $(X_5, *)$ and $(X_1, *)$

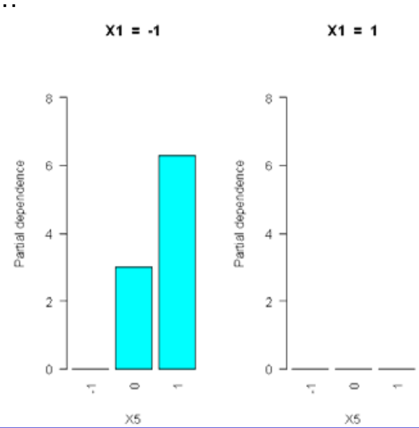


Ensemble Methods

Interpretation – Example (7)

- Two-Variable Partial Dependencies
  - Effect of x_j and x_k on response after accounting for the (average) effects of the other variables…

$$y = \begin{cases} 3 + 3 \cdot x_2 + 2 \cdot x_3 + x_4 + z & x_1 = 1 \\ -3 + 3 \cdot x_5 + 2 \cdot x_6 + x_7 + z & x_1 = -1 \end{cases}$$



Ŧ: Here translated to have a minimum value of zero

References

1. Breiman, L., Friedman, J.H., Olshen, R., and Stone, C. (1993). *Classification and Regression Trees*. Chappman & Hall/CRC.
2. Breiman, L. (1995). Better Subset Regression Using the Nonnegative Garrote. *Technometrics*, 37(4):373-384.
3. Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 26:123-140.
4. Breiman, L. (1998). *Arc*ing Classifiers. *Annals of Statistics* 26(2):801-849.
5. Breiman, L. (2001). Random Forests, random features. Technical Report, University of California, Berkeley.
6. Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least Angle Regression. *Annals of Statistics*, 32(2):407–499.
7. Elder, J. (2003). The Generalization Paradox of Ensembles. *Journal of Computational and Graphical Statistics*, 12(4):853-864.
8. Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proc. of the 13th Intl. Conference*, Morgan Kauffman, San Francisco, 148-156.
9. Freund, Y. and Schapire, R.E. (1997). A decision theoretical generalization of on-line learning and an application to boosting. *Journal of Computer and systems Sciences*, 55(1):133-68.
10. Friedman, J.H. (1999). Stochastic gradient boosting. Technical Report Department of Statistics, Stanford University.

References (2)

11. Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 28:337-40.
12. Friedman, J. (2001). Greedy function approximation: the gradient boosting machine, *Annals of Statistics*, 29:1189–1232.
13. Friedman, J. and Popescu, B. E. (2003). Importance Sampled Learning Ensembles. Technical Report, Stanford University.
14. Friedman, J. and Popescu, B. E. (2004). Gradient directed regularization for linear regression and classification. Technical Report, Stanford University.
15. Friedman, J. and Popescu, B. E. (2005). Predictive learning via rule ensembles. Technical Report, Stanford University.
16. Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning* (ESL) – Data Mining, Inference and Prediction. Springer
17. Ho, T.K. (1995). Random Decision Forests. *Proc. of the 3rd Intl. Conference on Document Analysis and Recognition,* 278-282.
18. Kleinberg, E. (2000). On the Algorithmic Implementation of Stochastic Discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473-490.
19. Rosset, S. (2003). Topics in Regularization and Boosting. PhD. Thesis, Stanford university.

References (3)

20. Seni, G., Yang, E. and Akar, S. (2007). Yield Modeling with Rule Ensembles. *Proc. of the 18th IEEE/SEMI Advanced Semiconductor Manufacturing Conference*.

21. Tibshirani, R. Regression shrinkage and selection via the lasso (1996). *J. Royal Statistics Society B.*, 58:267-288.

22. Zhao, P. and Yu, B. (2005). Boosted lasso. Proc. of the SIAM Intl. Conference on Data Mining, Newport Beach, CA, 35-44.

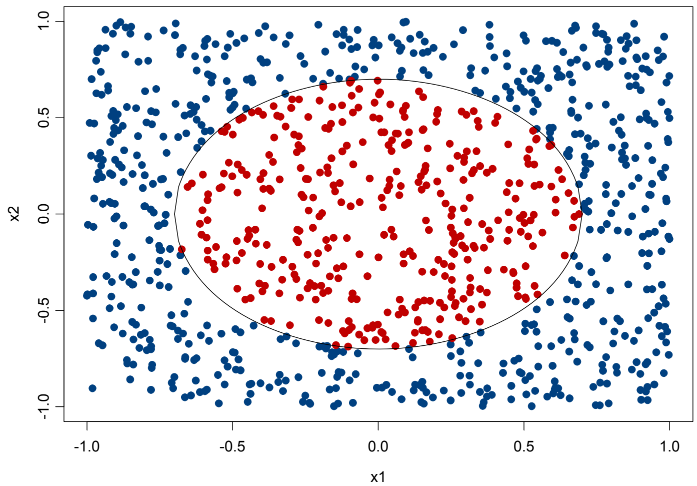
23. Zou, H. and Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. Keynote Address, SIAM workshop on Variable Selection, Newport Beach, CA.

Appendices

1. Visualizing Bagging and AdaBoost Decision Boundaries
2. On AdaBoost: Equivalence to Forward Stagewise Fitting Procedure
3. On AdaBoost: Prove population minimizer of exponential loss is the half the log-odds of $\Pr(Y=1\,|\,\mathbf{x})$
4. On Gradient Boosting: Solving for robust loss criterion
5. On MART: tree specific optimization
   - LAD-regression algorithm
6. Interpretation Statistics for Ensemble Methods
7. On Complexity and Generalized Degrees of Freedom

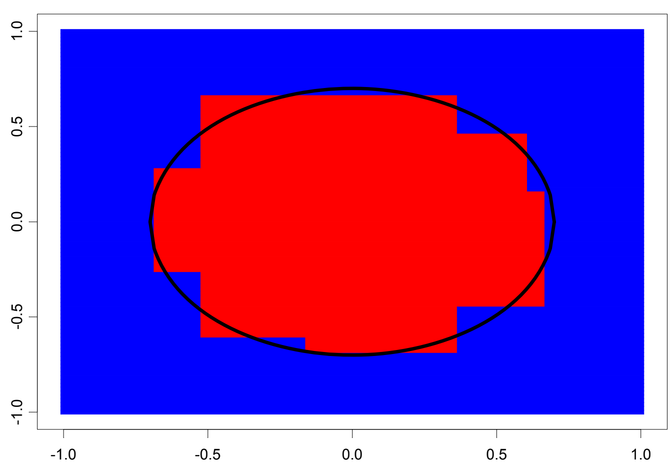
Appendix 1

Visualizing Bagging and AdaBoost (2-dimensional, 2-class example)



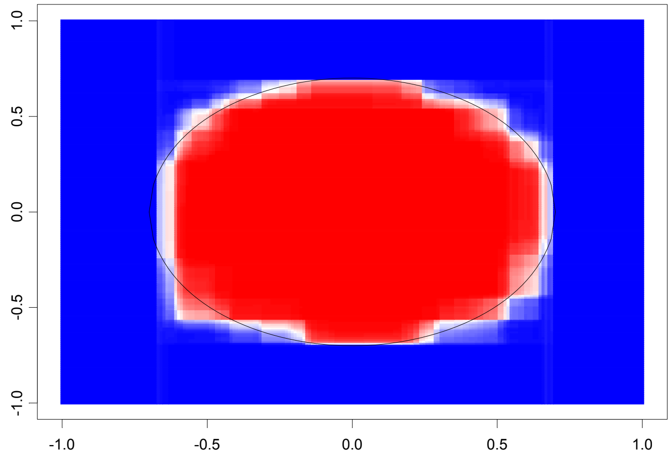
Appendix 1

Decision boundary of a single tree



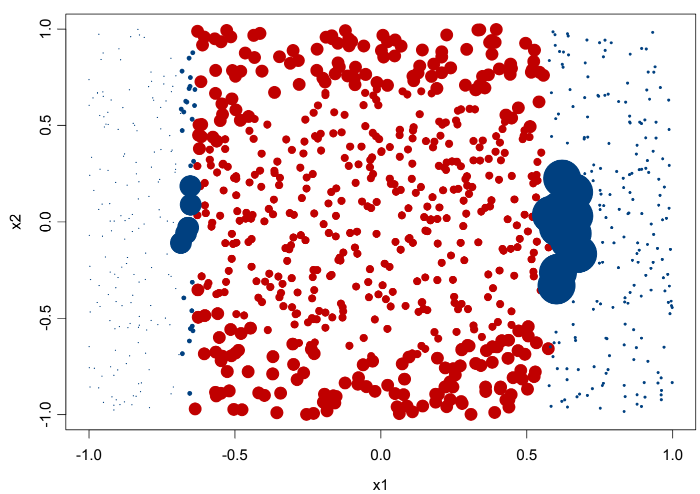
Appendix 1

100 bagged trees leads to smoother boundary

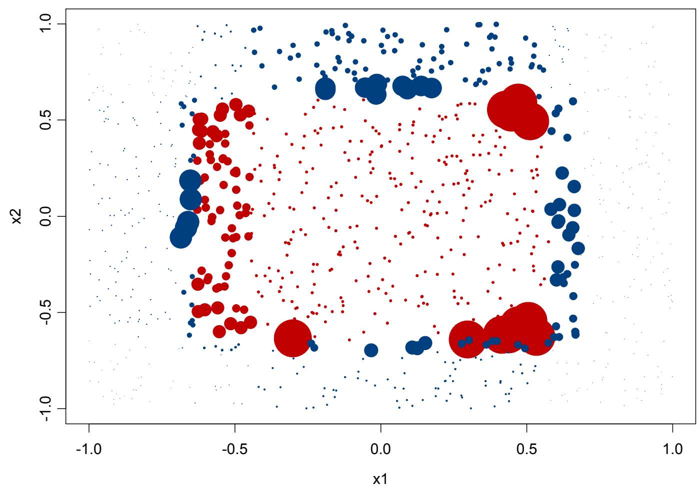


Appendix 1

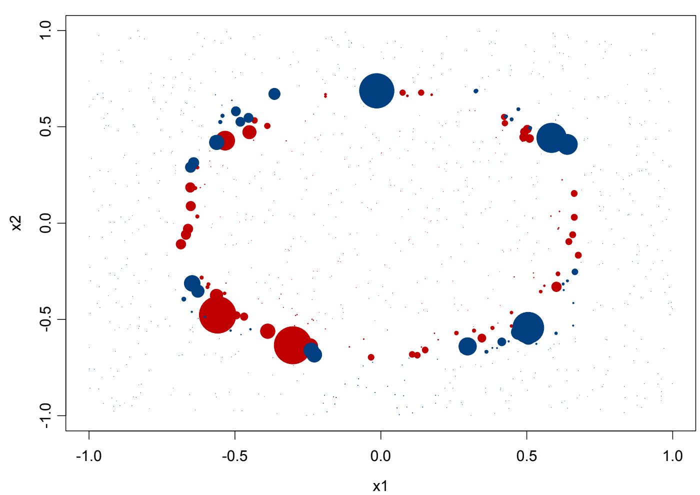
AdaBoost, after one iteration (CART splits, larger points have great weight)



Appendix 1
After 3 iterations of AdaBoost

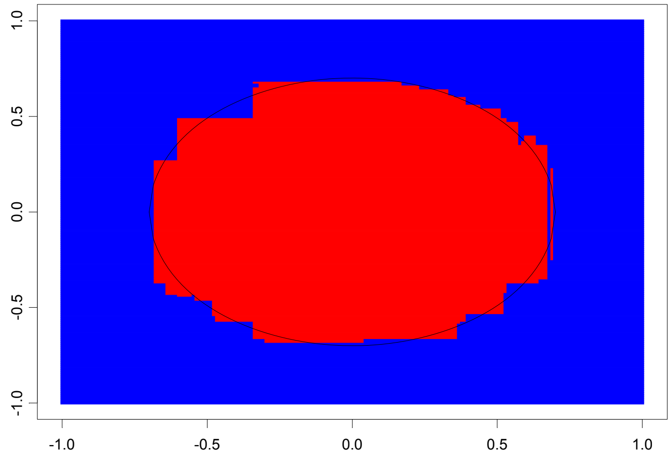


Appendix 1
After 20 iterations of AdaBoost



Appendix 1

Decision boundary after 100 iterations of AdaBoost



Appendix 2

On AdaBoost – Equivalence to FSF Procedure

- We have $(c_m, \mathbf{p}_m) = \arg\min_{c,\mathbf{p}} \sum_{i=1}^{N} L\big(y_i, F_{m-1}(\mathbf{x}_i) + c \cdot T(\mathbf{x}_i;\mathbf{p})\big)$

$$L(y,\hat{y}) = \exp(-y\cdot\hat{y}) \quad \Rightarrow \quad (c_m, \mathbf{p}_m) = \arg\min_{c,\mathbf{p}} \sum_{i=1}^{N} \exp\big(-y_i \cdot F_{m-1}(\mathbf{x}_i) \; - \; c \cdot y_i \cdot T(\mathbf{x}_i;\mathbf{p})\big)$$

$$= \arg\min_{c,\mathbf{p}} \sum_{i=1}^{N} w_i^{(m)} \cdot \exp\big(-c \cdot y_i \cdot T(\mathbf{x}_i;\mathbf{p})\big) \quad \text{with} \quad w_i^{(m)} = e^{-y_i F_{m-1}(\mathbf{x}_i)} \quad (1)$$

  - $w_i^{(m)}$ doesn't depend on c or $\mathbf{p}$, thus can be regarded as an observation weight
  - Solution to (1) can be obtained in two steps:
    - Step1: given c, solve for $T(\mathbf{x};\mathbf{p}_m)$

$$T_m = \arg\min_{T} \sum_{i=1}^{N} w_i^{(m)} \cdot \exp\big(-c \cdot y_i \cdot T(\mathbf{x}_i)\big) \quad \Rightarrow \quad T_m = \arg\min_{T} \left[\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq T(\mathbf{x}_i)) \right]$$

    - Step2: given T_m, solve for c

$$c_m = \arg\min_{c} \sum_{i=1}^{N} w_i^{(m)} \cdot \exp\big(-c \cdot y_i \cdot T_m(\mathbf{x}_i)\big) \quad \Rightarrow \quad c_m = \frac{1}{2}\log\frac{1-err_m}{err_m} \quad \text{where} \quad err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq T_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i^{(m)}}$$

Appendix 2
On AdaBoost – Equivalence to FSF Procedure (2)

- Step1: given c, solve for $T(\mathbf{x};\mathbf{p}_m)$

$$T_m = \arg\min_T \sum_{i=1}^N w_i^{(m)} \cdot \exp(-c \cdot y_i \cdot T(\mathbf{x}_i)) = \arg\min_T \left[e^{-c} \cdot \sum_{y_i = T(\mathbf{x}_i)} w_i^{(m)} + e^{c} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_i^{(m)} \right]$$

$$= \arg\min_T \left[e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} - e^{-c} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_i^{(m)} + e^{c} \cdot \sum_{y_i \neq T(\mathbf{x}_i)} w_i^{(m)} \right]$$

$$= \arg\min_T \left[\underbrace{\left(e^{c} - e^{-c}\right)}_{\text{constant}} \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T(\mathbf{x}_i)) + \underbrace{e^{-c} \cdot \sum_{i=1}^N w_i^{(m)}}_{\text{doesn't depend on T}} \right] = \arg\min_T \left[\sum_{i=1}^N w_i^{(m)} I(y_i \neq T(\mathbf{x}_i)) \right]$$

  - i.e., T_m is the classifier that minimizes the weighted error rate (line a.)

- Step 2: given T_m, solve for c

$$c_m = \arg\min_c \left[\left(e^{c} - e^{-c}\right) \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(\mathbf{x}_i)) + e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} \right]$$

$\Rightarrow$ compute and set to zero derivative with respect to c

Appendix 2
On AdaBoost – Equivalence to FSF Procedure (3)

- Step2: (cont)

$$\frac{\partial}{\partial c}\left(\left(e^{c} - e^{-c}\right) \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) + e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} \right) = \left(e^{c} + e^{-c}\right) \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) - e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} = 0$$

$$\Rightarrow e^{c} \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) + e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) - e^{-c} \cdot \sum_{i=1}^N w_i^{(m)} = 0$$

$$\Rightarrow (\text{dividing by } e^{-c}) \quad e^{2c} \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) + \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) - \sum_{i=1}^N w_i^{(m)} = 0$$

$$\Rightarrow e^{2c} \cdot \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i)) = \sum_{i=1}^N w_i^{(m)} - \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))$$

$$\Rightarrow e^{2c} = \frac{\sum_{i=1}^N w_i^{(m)} - \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))} \qquad \Rightarrow c = \frac{1}{2} \ln \frac{\sum_{i=1}^N w_i^{(m)} - \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}$$

and

$$\frac{1 - err_m}{err_m} = \frac{1 - \dfrac{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)}}}{\dfrac{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)}}} = \frac{\dfrac{\sum_{i=1}^N w_i^{(m)} - \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)}}}{\dfrac{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)}}} = \frac{\sum_{i=1}^N w_i^{(m)} - \sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}{\sum_{i=1}^N w_i^{(m)} I(y_i \neq T_m(x_i))}$$

Appendix 2

On AdaBoost – Equivalence to FSF Procedure (4)

- Finally, weight update $w_i^{(m+1)}$ expression:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + c_m \cdot T_m(\mathbf{x}) \quad \Rightarrow \quad w_i^{(m+1)} = e^{-y_i F_m(\mathbf{x}_i)} = e^{-y_i [F_{m-1}(\mathbf{x}_i) + c_m \cdot T_m(\mathbf{x}_i)]}$$

$$= e^{-y_i F_{m-1}(\mathbf{x}_i)} \cdot e^{-c_m \cdot y_i \cdot T_m(\mathbf{x}_i)} = w_i^{(m)} \cdot e^{-c_m \cdot y_i \cdot T_m(\mathbf{x}_i)}$$

$$-y_i \cdot T_m(\mathbf{x}_i) = 2I(y_i \neq T_m(\mathbf{x}_i)) - 1 \quad \Rightarrow \quad = w_i^{(m)} \cdot e^{2 \cdot c_m \cdot I(y_i \neq T_m(\mathbf{x}_i))} \cdot e^{-c_m}$$

$$= w_i^{(m)} \cdot e^{\alpha_m \cdot I(y_i \neq T_m(\mathbf{x}_i))} \cdot \underbrace{e^{-c_m}}_{\text{multiplies all weights… has no effect}}$$

  - $\alpha_{\mathrm{m}} = 2 \cdot c_m$ is the quantity in line 2c
  - Equivalence to d. $\surd$

$\Rightarrow$ AdaBoost minimizes the exponential loss with a forward-stagewise additive modeling approach!

Appendix 3

On AdaBoost – Population Minimizer

- Prove $F^*(\mathbf{x}) = \underset{F(\mathbf{x})}{\arg\min}\, \mathrm{E}_{Y|\mathbf{x}}\left(e^{-Y \cdot F(\mathbf{x})}\right) = \frac{1}{2} \log \frac{\Pr(Y=1|\mathbf{x})}{\Pr(Y=-1|\mathbf{x})}$

$$E(e^{-yf(x)}|x) = \Pr(Y=1|x) \cdot e^{-f(x)} + \Pr(Y=-1|x) \cdot e^{f(x)}$$

$$\Rightarrow \frac{\partial E(e^{-yf(x)}|x)}{\partial f(x)} = -\Pr(Y=1|x) \cdot e^{-f(x)} + \Pr(Y=-1|x) \cdot e^{f(x)}$$

$$\Rightarrow \frac{\partial E(e^{-yf(x)}|x)}{\partial f(x)} = 0 \quad \Rightarrow \quad \Pr(Y=-1|x) \cdot e^{f(x)} = \Pr(Y=1|x) \cdot e^{-f(x)}$$

$$\Rightarrow \quad \ln \Pr(Y=-1|x) + f(x) = \ln \Pr(Y=1|x) - f(x)$$

$$\Rightarrow \quad 2f(x) = \ln \Pr(Y=1|x) - \ln \Pr(Y=-1|x)$$

$$\Rightarrow \quad f(x) = \frac{1}{2} \ln \frac{\Pr(Y=1|x)}{\Pr(Y=-1|x)}$$

Appendix 4
On Gradient Boosting

- Solving for robust loss criterion (e.g., absolute loss, binomial deviance) requires use of a "surrogate", more convenient, $\tilde{L}(y,\hat{y})$

- Like before, we solve $(c_m,\mathbf{p}_m)=\arg\min_{c,\mathbf{p}}\sum_{i=1}^{N}L(y_i,F_{m-1}(\mathbf{x}_i)+c\cdot T(\mathbf{x}_i;\mathbf{p}))$ in two steps:
  - Step1: find $T(\mathbf{x};\mathbf{p}_m)$ … here we use $\tilde{L}(y,\hat{y})$

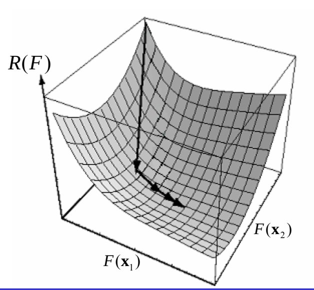
$$\mathbf{p}_m=\arg\min_{\mathbf{p}}\sum_{i=1}^{N}\tilde{L}(y_i,F_{m-1}(\mathbf{x}_i)+c\cdot T(\mathbf{x}_i;\mathbf{p}))$$

  - Step2: given T_m, solve for c

$$c_m=\arg\min_{c}\sum_{i=1}^{N}L(y_i,F_{m-1}(\mathbf{x}_i)+c\cdot T(\mathbf{x}_i;\mathbf{p}_m))$$

Appendix 4
On Gradient Boosting (2)

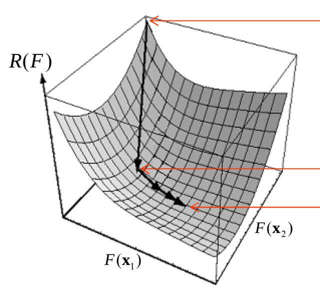
- $\tilde{L}(y,\hat{y})$ is derived from analogy to numerical optimization in function space
  - Learning: $\hat{F}=\arg\min_F R(F)$ – i.e., minimum "risk"
  - Each possible F is a "point" in $\Re^N$ – i.e., $\mathbf{F}=\langle F(\mathbf{x}_1),F(\mathbf{x}_2),\ldots,F(\mathbf{x}_N)\rangle$



Appendix 4

On Gradient Boosting (3)

- Steepest descent in function space ("non-parametric" view)



$\mathbf{F}_0$: initial guess

$\mathbf{F}_1 = \mathbf{F}_0 - \rho_1 \nabla R(F)$

$\widehat{\mathbf{F}}_M = \sum_{m=0}^{M} \mathbf{F}_m$

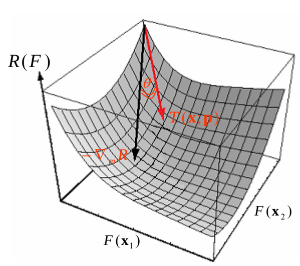
  - Gradient components $\nabla_m R = \begin{bmatrix} \partial R / \partial F(\mathbf{x}_1) \\ \cdots \\ \partial R / \partial F(\mathbf{x}_N) \end{bmatrix}_{F=F_{m-1}} = \begin{bmatrix} \partial L(y_1, F(\mathbf{x}_1)) / \partial F(\mathbf{x}_1) \\ \cdots \\ \partial L(y_N, F(\mathbf{x}_N)) / \partial F(\mathbf{x}_N) \end{bmatrix}_{F=F_{m-1}}$

  - Step size (line search): $\rho_m = \arg\min_\rho R(\mathbf{F}_{m-1} - \rho \nabla_m R)$ $\Rightarrow$ like "Step2" before

Appendix 4

On Gradient Boosting (4)

- Problem: $\widehat{\mathbf{F}}_M$ defined on training data only
  - Select parameterized function (defined for all $\mathbf{x}$): $T(\mathbf{x};\mathbf{p})$
  - Choose $\mathbf{p}$ so that $T(\mathbf{x};\mathbf{p})$ is most "parallel" to $\nabla R(F)$



- Geometric interpretation of correlation:

$$\cos\theta = corr\left(\{-\nabla R(F(\mathbf{x}))\}_{i=1}^{N}, \{T(\mathbf{x}_i;\mathbf{p})\}_{i=1}^{N}\right)$$

- Most highly correlated $T(\mathbf{x};\mathbf{p})$ is found from solution :

$$\mathbf{p}_m = \arg\min_{\beta,\mathbf{p}} \sum_{i=1}^{N} \left(-\nabla_m R(F(\mathbf{x}_i)) - \beta \cdot T(\mathbf{x}_i;\mathbf{p})\right)^2$$

  - Surrogate criterion $\tilde{L}$: $\tilde{y}_i = -\left.\dfrac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right|_{F=F_{m-1}}$ $\Rightarrow$ Least-squares minimization

Appendix 5

On MART

- Adding $T_m(\mathbf{x})$ to model is like adding *J* separate (basis) functions

$$T\left(\mathbf{x};\{b_j, R_j\}_1^J\right) = \sum_{j=1}^{J} b_j I(\mathbf{x} \in R_j) \;\Rightarrow\; F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + c_m \cdot T_m(\mathbf{x}) \;=\; F_{m-1}(\mathbf{x}) + c_m \cdot \sum_{j=1}^{J} b_{jm} I(\mathbf{x} \in R_{jm})$$

$$= F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \gamma_{jm} I(\mathbf{x} \in R_{jm}) \quad \text{with } \gamma_{jm} = c_m \cdot b_{jm}$$

- Since regions R_{jm} are disjoint, we can do separate updates in each terminal node
  - Step2: was: $c_m = \arg\min_c \sum_{i=1}^{N} L\left(y_i, F_{m-1}(\mathbf{x}_i) + c \cdot T(\mathbf{x}_i; \mathbf{p}_m)\right)$

    now: $\gamma_{jm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{jm}} L\left(y_i,\; F_{m-1}(\mathbf{x}_i) + \gamma\right)$

    i.e., optimal constant update in each terminal region

Appendix 5

On MART (2)

- Summary:

$F_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$

For $m = 1$ to M {

// Step1: find $T_m(\mathbf{x})$ using surrogate criterion

$$\tilde{y}_i = -\left.\frac{\partial L\left(y_i, F(\mathbf{x}_i)\right)}{\partial F(\mathbf{x}_i)}\right|_{F = F_{m-1}}$$

$$\left\{\hat{b}_{jm}, \hat{R}_{jm}\right\}_1^J = \arg\min_{\{b_{jm}, R_{jm}\}} \sum_{i=1}^{N} \left[\tilde{y}_i - \sum_{j=1}^{J} b_{jm} I\left(\mathbf{x}_i \in R_{jm}\right)\right]^2$$

// Step2: find coefficients

$$\hat{\gamma}_{jm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{jm}} L\left(y_i, F_{m-1}(\mathbf{x}_i) + \gamma\right)$$

// Update expansion

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot \sum_{j=1}^{J} \hat{\gamma}_{jm} I\left(\mathbf{x}_i \in \hat{R}_{jm}\right)$$

}

write $\hat{F}(\mathbf{x}) = F_m(\mathbf{x})$

Appendix 5
On MART (3)

- LAD-regression: $L(y,F)=|y-F|$
  - More robust than $(y-F)^2$
  - Resistant to outliers in y … trees already providing resistance to outliers in $\mathbf{x}$
  - Algorithm:

$$
\begin{aligned}
& F_0(\mathbf{x}) = median\,\{y_i\}_1^N \\
& \text{For } m = 1 \text{ to } M \ \{ \\
& \quad \text{// Step1 : find } T_m(\mathbf{x}) \\
& \quad \tilde{y}_i = sign(y_i - F_{m-1}(\mathbf{x}_i)) \\
& \quad \{\hat{R}_{jm}\}_1^J = J - \text{terminal node LS-regression tree}\left(\{\tilde{y}_i, \mathbf{x}_i\}_1^N\right) \\
& \quad \text{// Step2 : find coefficien ts} \\
& \quad \hat{\gamma}_{jm} = \underset{\mathbf{x}_i \in \hat{R}_{jm}}{median}\,\{y_i - F_{m-1}(\mathbf{x}_i)\}_1^N \qquad j = 1 \ldots J \\
& \quad \text{// Update expansion} \\
& \quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon \cdot \sum_{j=1}^{J} \hat{\gamma}_{jm} I\left(\mathbf{x}_i \in \hat{R}_{jm}\right) \\
& \}
\end{aligned}
$$

Appendix 6
Interpretation − Variable Importance

- Single tree measure (CART) : $\Im(x_l) = \sum_{t \in T} \hat{\mathrm{I}}\left(v(t), \tilde{s}_{v(t)}\right) \cdot I\left(v(t) = l\right)$
  - i.e., sum the "goodness of split" scores whenever x_l is used in a surrogate split
  - Sum is over all internal nodes in the tree $t \in T$
  - If x_l was used as the primary split in some node, then corresponding score $\hat{\mathrm{I}}(l, s_l)$ is used in the summation
  - Normalized measure: $\mathrm{Imp}(x_j) = 100 \cdot \Im(x_j) / \max_{1 \le l \le m} \Im(x_l)$
    - i.e., only the relative magnitudes of the $\Im(x_j)$ are interesting
- Tree ensemble generalization
  - Average over all of the trees: $\Im(x_j) = \frac{1}{M} \sum_{m=1}^{M} \Im(x_j; T_m)$

Appendix 6
Interpretation − Variable Importance (2)

- Rule ensemble measure:
  - Term (*global*) importance: absolute value of coefficient of standardized predictor
    - Rule term: $\Im_k = |\hat{a}_k| \cdot \sqrt{s_k \cdot (1 - s_k)}$ where rule's support is $s_k = \frac{1}{N}\sum_{i=1}^{N} r_k(\mathbf{x}_i)$
    - Linear term: $\Im_j = |\hat{b}_j| \cdot std(x_j)$
  - Term (*local*) importance: at *each* point $\mathbf{x}$ … absolute change in $\hat{F}(\mathbf{x})$ when term is removed
    - Rule term: $\Im_k(\mathbf{x}) = |\hat{a}_k| \cdot |r_k(\mathbf{x}) - s_k|$
    - Linear term: $\Im_j(\mathbf{x}) = |\hat{b}_j| \cdot |x_j - \bar{x}_j|$

$$\Im_l(\mathbf{x}) = \Im_l(\mathbf{x}) + \sum_{x_l \in r_k} \Im_k(\mathbf{x}) / size(r_k)$$

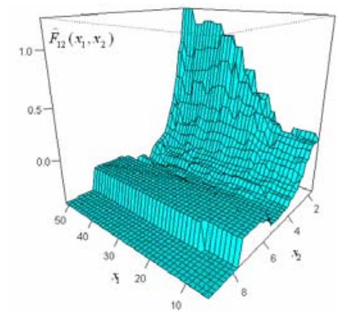
Appendix 6
Interpretation − Partial Dependence Plots (Friedman, 2001)

- Visualize the effect on $\hat{F}(\mathbf{x})$ of a small subset of the important input variables $\mathbf{x}_S \subset \{x_1, x_2, \ldots, x_n\}$
  - after accounting for the (average) effects of the other ("complement") variables $\mathbf{x}_C$ – i.e., $\mathbf{x}_S \cup \mathbf{x}_C = \{x_1, x_2, \ldots, x_n\}$
  - $\hat{F}(\mathbf{x}) = \hat{F}(\mathbf{x}_S, \mathbf{x}_C)$
  - Partial dependence on $\mathbf{x}_S$: $\hat{F}_S(\mathbf{x}_S) = \mathrm{E}_{\mathbf{x}_C} \hat{F}(\mathbf{x}_S, \mathbf{x}_C)$
  - Approximated by: $\hat{F}_S(\mathbf{x}_S) = \frac{1}{N}\sum_{i=1}^{N} F(\mathbf{x}_S, x_{iC})$

    $\{x_{1C}, \ldots, x_{NC}\}$ are the values of $\mathbf{x}_C$ occurring in the training data

Appendix 6
Interpretation − Partial Dependence Plots (2)

- Example



  - The shape of the function on either variable is affected by the values of the other, suggesting the presence of an interaction

Appendix 6
Interpretation − Interaction Statistic (Friedman, 2005)

- If x_j and x_k do not interact, then
  - $\hat{F}(\mathbf{x})$ can be expressed as sum of two functions: $\hat{F}(\mathbf{x}) = f_{\setminus j}(\mathbf{x}_{\setminus j}) + f_{\setminus k}(\mathbf{x}_{\setminus k})$

    i.e., $f_{\setminus j}(\mathbf{x}_{\setminus j})$ does not depend on x_j ; $f_{\setminus k}(\mathbf{x}_{\setminus k})$ is independent of x_k

  - Thus, partial dependence on $\mathbf{x}_S = \{x_j, x_k\}$ can be decomposed:

$$\hat{F}_{j,k}(x_j, x_k) = \hat{F}_j(x_j) + \hat{F}_k(x_k)$$

    i.e., sum of respective partial dependencies

- Test for the presence of (x_j, x_k) interaction

$$H_{jk}^2 = \sum_{i=1}^{N} \left[\hat{F}_{j,k}(x_{ij}, x_{ik}) - \hat{F}_j(x_{ij}) - \hat{F}_k(x_{ik})\right]^2 \Big/ \sum_{i=1}^{N} \hat{F}_{j,k}^2(x_{ij}, x_{ik})$$

Appendix 6
Interpretation − Interaction Statistic (2)

- If x_j does not interact with any other variable
  - $\hat{F}(\mathbf{x})$ can be expressed as sum of two functions: $\hat{F}(\mathbf{x}) = f_j(\mathbf{x}_j) + f_{\backslash j}(\mathbf{x}_{\backslash j})$

    where $f_j(x_j)$ is a function only of x_j
  - Thus, $\hat{F}(\mathbf{x}) = F_j(x_j) + F_{\backslash j}(\mathbf{x}_{\backslash j})$
- Test whether x_j interacts with *any* other variable

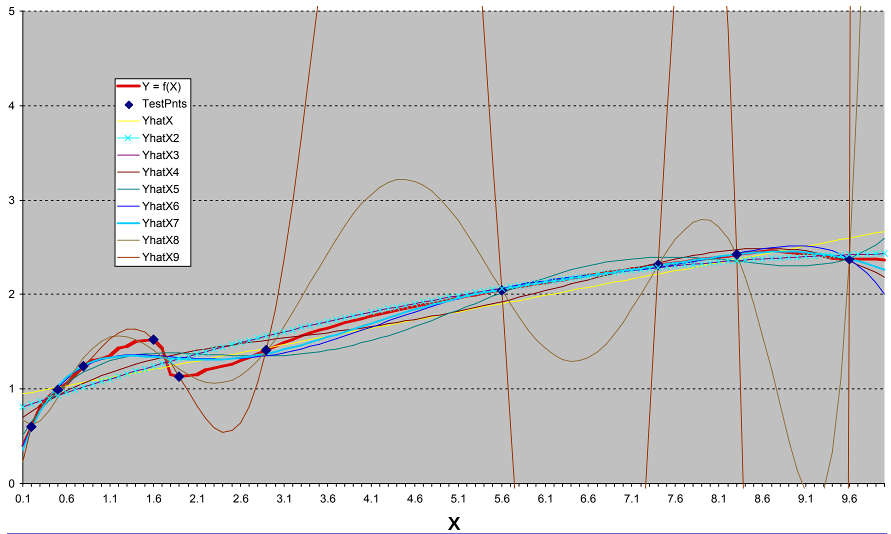
$$H_j^2 = \sum_{i=1}^{N} \left[\hat{F}(\mathbf{x}_i) - \hat{F}_j(x_{ij}) - \hat{F}_{\backslash j}(\mathbf{x}_{i\backslash j})\right]^2 \Big/ \sum_{i=1}^{N} \hat{F}^2(\mathbf{x}_i)$$

Appendix 7
Complexity of Ensembles

Outline:

- Ensembles generalizing well was a theoretical surprise
  - Importance Sampling helps us understand some behavior
- Chief danger of data mining: Overfit
- Occam's razor: regulate complexity to avoid overfit
- But, does the razor work?
  - counter-evidence has been gathering
- What if complexity is measured incorrectly?
- Generalized degrees of freedom (Ye, 1998)
- Experiments: single vs. bagged decision trees
- Summary: factors that matter

Appendix 7
Overfit models generalize poorly



Appendix 7
Occam's Razor

- *Nunquam ponenda est pluralitas sin necesitate*
  "Entities should not be multiplied beyond necessity"
  "If (nearly) as descriptive, the simpler model is more correct."
- But, gathering doubts:
  - Ensemble methods which employ *multiple models* (e.g., bagging, boosting, bundling, Bayesian model averaging
  - Nonlinear terms have higher (or lower) than linear effect
  - Much overfit is from *excessive search* (e.g., Jensen 2000), rather than over-parameterization
  - Neural network structures are fixed, but their degree of fit grows with time
- Domingos (1998) won KDD Best Paper arguing for its death
  - What if complexity is measured incorrectly?

Appendix 7
Target Shuffling can measure "over search"
(e.g., Battery MTC, Monte Carlo Target in CART β)

- Break link between target, *Y*, and features, *X*
  by shuffling *Y* to form *Ys*.
- Model new *Ys* ~ *f(X)*
- Measure quality of resulting (random) model
- Repeat to build distribution

-> Best (or mean) shuffled (i.e., useless) model
sets the baseline above which true model performance
may be measured

Appendix 7
Complexity can be measured by the *Flexibility* of the Modeling *Process*

- *Generalized Degrees of Freedom*, GDF (Ye, *JASA* 3/1998)
  - Perturb output, re-fit procedure,
    measure changes in estimates
- *Covariance Inflation Criterion*, CIC (Tibshirani & Knight, 1999)
  - Shuffle output, re-fit procedure,
    measure covariance between new and old estimates.
- Key step (loop around modeling procedure) reminiscent of *Regression Analysis Tool,* RAT (Faraway, 1991) -- where resampling tests of a 2-second procedure took 2 days to run.

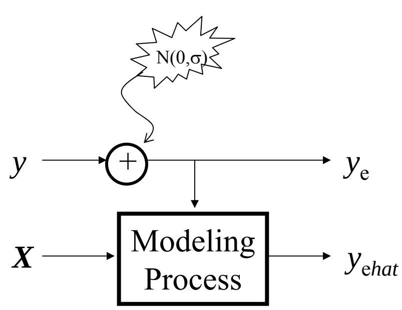
Appendix 7
Generalized Degrees of Freedom (GDF)

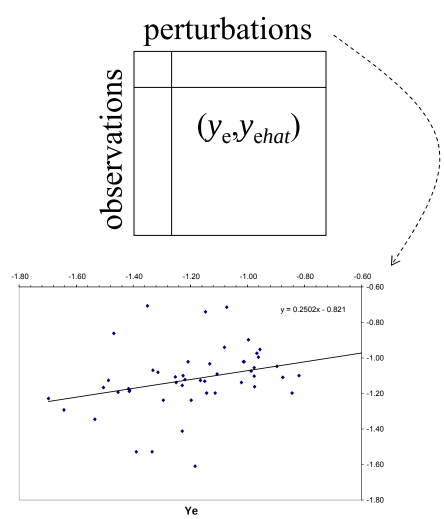
- #terms in Linear Regression (LR) = DoF, k
- Nonlinear terms (e.g., MARS) can have effect of ~3k (Friedman, Owen '91)
- Other parameters can have effects < 1
  (e.g., under-trained neural networks)

Procedure (Ye, 1998):

- For LR, k = trace(Hat Matrix) = $\Sigma\ \delta y_{hat} / \delta y$
- Define GDF to be sum of sensitivity of each fitted value, y_{hat}, to perturbations in the corresponding output, y. That is, instead of extrapolating from LR by counting terms, use alternate trace measure which is equivalent under LR.
- (Similarly, the effective degrees of freedom of a spline model is estimated by the trace of the projection matrix, S: $y_{hat} = Sy$)
- Put a y-perturbation loop around the entire modeling process (which can involve multiple stages)

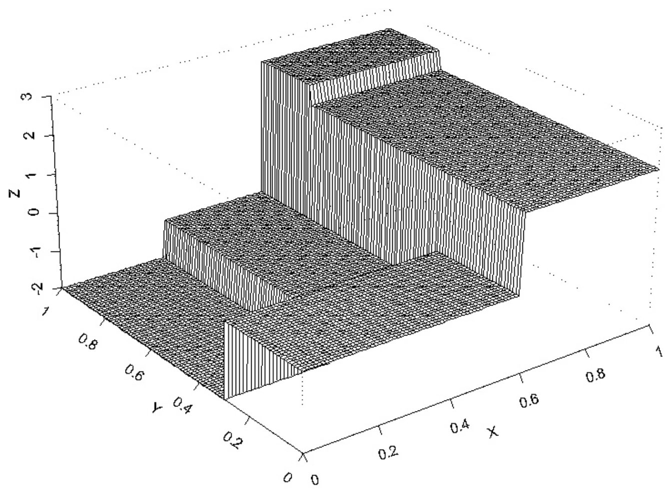
Appendix 7
GDF computation



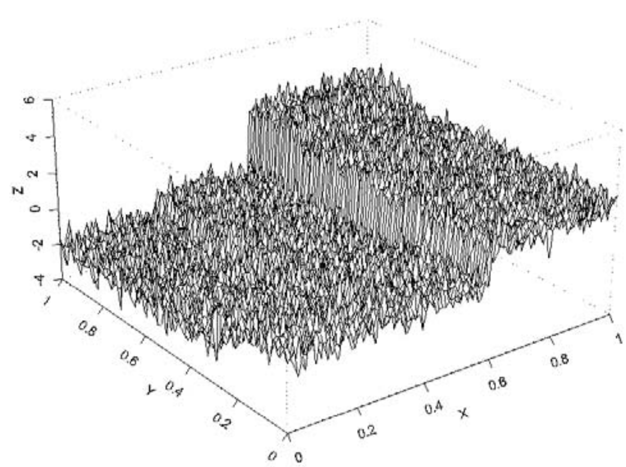


Ye robustness trick:
average, across perturbation runs, the sensitivities for a given observation.

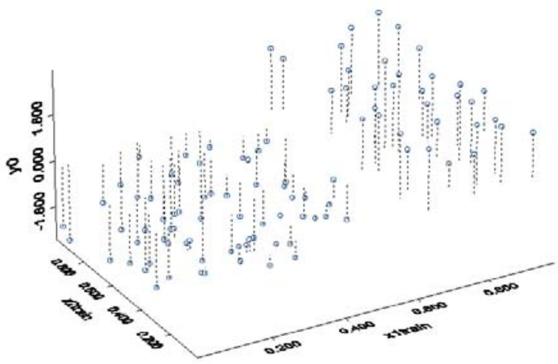
Appendix 7
Example: data surface is piecewise constant



Appendix 7
Additive N(0,.5) noise

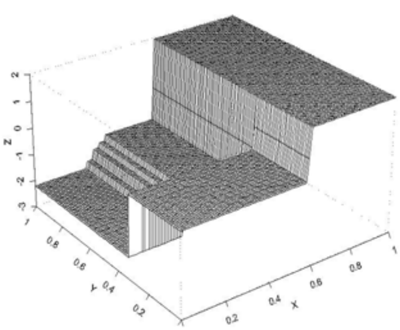


Appendix 7
100 random training samples

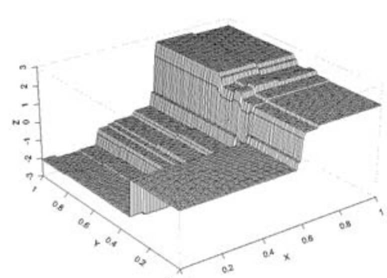


Appendix 7
Estimation Surfaces for bundles of 5 trees

4-leaf trees



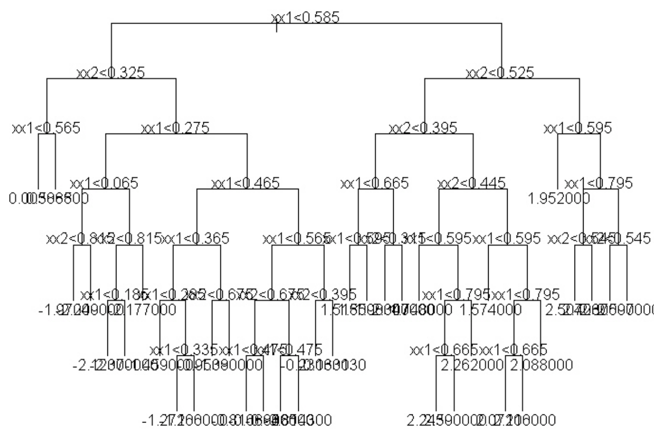
8-leaf trees
(some of the finer structure is real)



Bagging produces gentler stair-steps than raw tree
(illustrating how it generalizes better for smooth functions)

Appendix 7

Equivalent tree for 8-leaf bundle (25% pruned)

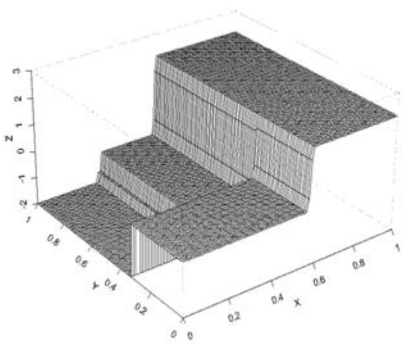


So a bundled tree is still a tree.
But is it as complex as it looks?

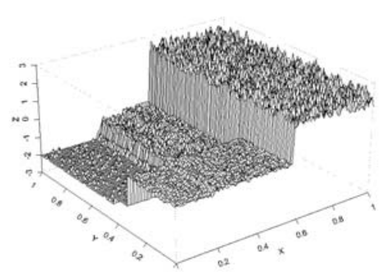
Appendix 7

Experiment: Introduce *selection noise* (an additional 8 candidate input variables)

Estimation surface for 5-bag of 4-leaf trees

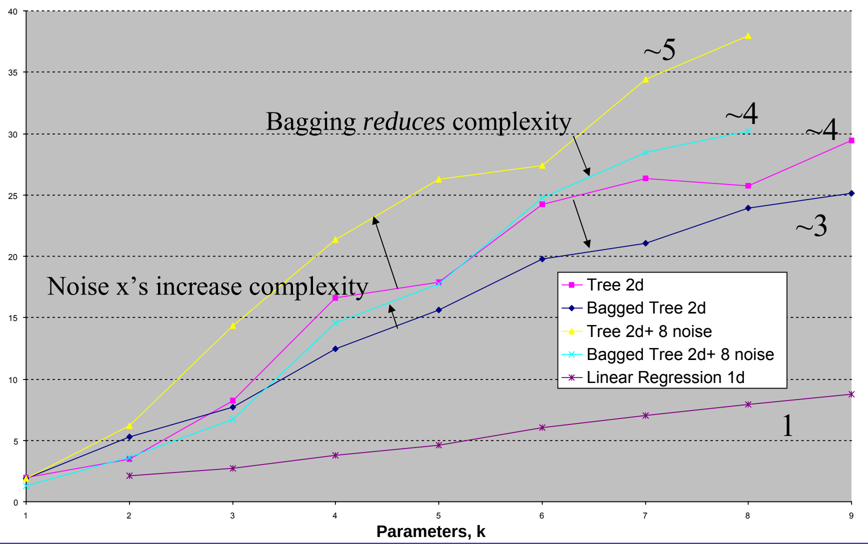


… for 8-leaf trees



Main structure here is clear enough for simple models to avoid noise inputs but their eventual use leads to a distribution of estimates on 2-d projection.

Appendix 7
Estimated GDF vs. #parameters



Appendix 7
Ensembles & Complexity Summary

- Bundling competing models improves generalization.
- Different model families are a good source of component diversity.
- If we measure complexity as *flexibility* (GDF)
  the classic relation between complexity and overfit is revived.
  - The more a modeling process can match an arbitrary change made to its output, the more complex it is.
  - Simplicity is not parsimony.
- Complexity increases with distracting variables.
- It is expected to increase with parameter power and search thoroughness, and decrease with priors, shrinking, and clarity of structure in data. Constraints (observations) may go either way…
- Model ensembles often have *less* complexity than their components.
- Diverse modeling procedures can be fairly compared using GDF