Bagging-Boosted Classification Trees & Random Forest

Karl Winterling, Nick Sullivan, Anthony Gusman

August 3, 2016

Introduction

"Is it cancer?" A serious question that medical professionals strive to answer. To do this requires:

- Collection of data on cancerous and normal tissue
- Identification of their key characteristics
- Application to classify new observations

Data is often recorded via DNA microarrays, which encode the relative gene expression levels of cancer and normal samples. Each sample consists of thousands of genes making analysis difficult.

How do we detect cancer based on the data?

Project Goal

Goal

Detect cancer from gene expression levels in DNA microarray datasets using *decision tree ensemble methods*.

Accomplishing this goal would provide the basis for valuable diagnostic tests. Such tests could provide doctors with the means to more accurately diagnose cancer, leading to earlier detection and saving lives.

Artificial “Intelligence”

- Media hype since 1950’s.
- Simulated AI vs. “real” AI.
- Machine Learning: The program can “learn” to “make better decisions” based on looking at patterns.
- Programs “inspired” by biology.
- Singularity: A program learns how to improve itself and becomes smarter than any human. Good SF at least.

Machine Learning Applications

- Spam filters.
- Autonomous drones.
- Facial recognition.
- Netflix recommendations.
- Video games.
- Education (e.g., assign homework based on quiz performance).
- Marketing (Google ads).
- Psychotherapy (machine-based Cognitive Behavioral Therapy).
- Crime modeling.

Mathematics

Definitions

Data N vectors X_i of class y_i

$D = \{(X_1, y_1), ..., (X_N, y_N)\}$

Learning Set Vectors with known classes

$L = \{(X_{1_L}, y_{1_L}), ..., (X_{n_L}, y_{n_L})\}$

Test Set Vectors (unknown classes)

$T = \{X_{1_T}, ..., X_{n_T}\}$

Classifier Function C built from learning set L. Denote $C(\cdot, L)$.
Given vector $\mathbf{x}$ of class y, predicted class is $C(\mathbf{x}, L)$.

Cardinality The number of vectors in a set A is denoted by $|A|$.

*For simplicity, we assume that $y_i \in \{-1, 1\}$.

Classification Trees

Classification Tree

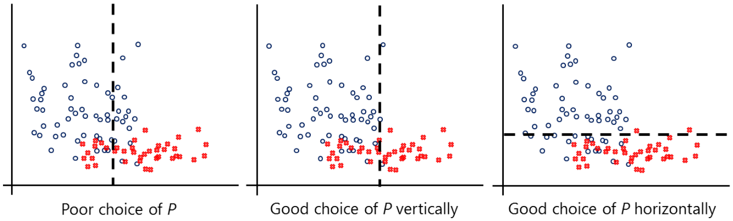
Given a learning set L, a tree classifier $C(\cdot, L)$ recursively partitions the set locally maximizing the purity at each level.

Two popular metrics exist to measure purity:

(1) Information gain & Entropy (ID3)

(2) **Gini impurity (CART)**

The tree is grown by choosing partitions which yield purer subsets.



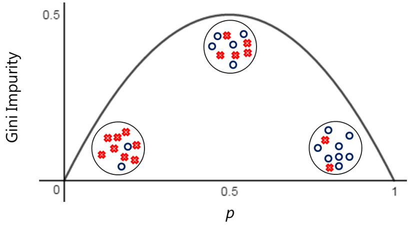
Gini: A measure of diversity

Definition

Suppose A contains J classes and let p_i be the probability of choosing a vector of class i. Then, the *Gini impurity* of A is the probability of misclassifying a vector:

$$I_{\text{gini}} = \sum_i p_i(1 - p_i)$$

Visualization for two classes



This sum is maximized when $p_i = 1/k$.

Intuitively, for $k = 2$, a set with $I_{\text{gini}} = 0$ (i.e., all one class) is trivial to classify, while a set with $I_{\text{gini}} = 0.5$ (i.e., equal amounts of each class) is classified by a coin toss.

A tree scans over each feature and splits at the point of minimum impurity.

Random Forest & Bagging

Random Forest

Definition

A random forest is a classifier consisting of a collection of tree-structured classifiers $\{C_j(\mathbf{x}, L_j),\ j = 1, \ldots\}$ where the $\{L_j\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$. ((Breiman 2001) word-for-word with notation adapted)

The random forest classifier is defined as the mode of classifiers, or

$$C(\mathbf{x}, L) = \operatorname{sign}\left(\sum_{j=1}^{K} C_j(\mathbf{x}, L_j)\right).$$

Construction

Random forests possess two layers of randomness:

(1) **Bootstrap Replicates:** The observations $\mathbf{x}_i$ in each L_j are selected uniformly with replacement from L, usually until $|L_j| = |L|$.

(2) **Feature Selection:** Each observation $\mathbf{x}_i$ consists of m features. At each node of every tree, a random subset of these features is chosen *without* replacement. The number of features chosen is denoted F.

When $F = m$ this procedure is known as *bagging*.

Additive Methods (Should) Reduce Error

- Use the error model,

$$\mathrm{Err}(x_0) = \left[\mathrm{Bias}\left(\hat{f}(x_0)\right)\right]^2 + \mathrm{Var}\left(\hat{f}(x_0)\right) + {\sigma_\epsilon}^2$$

- Assuming each classifier has relatively equal variance and correlation,

$$\begin{aligned}\mathrm{Var}\left(\hat{f}(x_0)\right) &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cor}\left(\hat{f}_i(X), \hat{f}_j(X)\right) \\ &= \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2.\end{aligned}$$

- Increasing n and decreasing correlation should reduce the variance.
- Caveat: Depends on data, describes regression models more accurately.

Boosting

Boosting: Motivation

- Lone Starr, Barf, Dot Matrix, and President Skroob are applied math graduate teaching staff at Starfleet Academy.
- Each follows a modified "Mary Sue" character archetype: each is perfect in every way but grossly incompetent in certain subject areas
- Problem: Given a True/False math question, find a strategy that takes an answer from each person and minimizes the probability of error.
- If incompetencies are disjoint, an answer must be correct if two people agree.
- Otherwise, we need a more complicated strategy.

Boosting: Motivation (Complex Decision Example)

- Our friends feel particularly lazy today.
- Suppose three use decision trees trained on a data set.
- President Skroob rolls 6 7-sided Dungeons & Dragons dice and says "yes" if and only if 3 or more share a prime factor.
- What is the optimal strategy? Which learning sets should you give the people for their decision trees?

Boosting

Definition

A booster is a classifier consisting of a collection of weighted classifiers $\{\alpha_t C(\mathbf{x}, L_t),\ t = 1, \ldots\}$ where the $\{L_t\}$ are formed via distribution D_t dependent on loss function $\epsilon_t\big(\mathbf{y}, C_{t-1}(\mathbf{x}, L_{t-1})\big)$ and each classifier casts a vote for the most popular class at input $\mathbf{x}$. (Adapted from Freund & Schapire 1999)

The strong boosting classifier C is determined by the weighted sum

$$C(\mathbf{x}, L) = \text{sign}\left(\sum_{t=1}^{K} \alpha_t C_t(\mathbf{x}, L_t)\right).$$

Boosting

Let H_{0i} be the event that tissue sample $\mathbf{x}_i$ is cancerous.

- **Input:** weak learner algorithm $\mathcal{L}$ (classification error $\epsilon < 1/2$), learning set $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $y_i = 1$ for "yes" (H_{0i} is true) and $y_i = -1$ for "no" (H_{0i} is false), number of "boosts" K.
- **Output:** Strong classifier C.

Boosting Algorithm

- Let D_1 define a probability distribution so that $D_{1,i}$ is the probability of choosing $\mathbf{x}_i$ in a sample.
- Do the following K times
  - Use D_t to sample the $\mathbf{x}_i$ with replacement to produce a learning set $L_t = \{(\mathbf{x}_i, y_i)\}$
  - Train $\mathcal{L}$ on L_t to produce a classifier C_t.
  - Determine the error of C_t on the entire set by comparing each $C_t(\mathbf{x}_i)$ to y_i and weighting by D_t.
  - Get α_t, where $\alpha_t = 0$ means C_t is a fair coin flip and higher α_t means C_t is better.
  - Use the error to weight C_t and add it to the strong classifier C.
  - Update D_t to D_{t+1} so that L_{t+1} contains more points that C_t classified incorrectly.

Error ϵ_t of C_t & Classifier Weight α_t

Let

$$\epsilon_t = \sum_{C_t(\mathbf{x}_i) \neq y_i} D_{ti} \qquad \text{(the sum over all misclassified points)}$$

Then, we define the classifier weight to be

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \log\left(\sqrt{\epsilon_t^{-1} - 1}\right).$$

Remarks:

- Well-defined if $\epsilon_t < 1/2$ (i.e., the weak learner algorithm $\mathcal{L}$ is better than classifying by flipping a coin), so $\alpha_t > 0$.
- A bigger α_t means that ϵ_t is smaller. Thus, better classifiers get larger weights. Think of α_t as a measure of how much C_t knows or a confidence level.

Updating Sampling Distribution D_t

Update as

$$D_{t+1,i} = Z \times D_{ti} \times \begin{cases} \exp(-\alpha_t) & \text{if } C_t(\mathbf{x}_i) = y_i, \\ \exp(\alpha_t) & \text{if } C_t(\mathbf{x}_i) \neq y_i, \end{cases}$$

where Z is a normalization constant.

Remark:

- C_{t+1} needs to work on what C_t got wrong.

Classification

The strong boosting classifier C is defined as the weighted sum

$$C(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{K} \alpha_t C_t(\mathbf{x})\right)$$

Remarks:

- We give more weight to a learner C_t if ϵ_t is smaller (and therefore α_t is larger, but we're careful not to make α_t too big).
- If a classifier is similar to a coin flip, α_t is close to 0.

Forward Stagewise Additive Modeling

Fact

Let G_m be the strong classifier at step m. Then AdaBoost is equivalent to,

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, G_{m-1}\left(\mathbf{x}_i\right) + \beta b\left(\mathbf{x}_i, \gamma\right)\right)$$

$$G_m\left(\mathbf{x}_i\right) = G_{m-1}\left(\mathbf{x}_i\right) + \beta_m b\left(\mathbf{x}_i, \gamma_m\right)$$

where $L(y, C(\mathbf{x})) = \exp\left(-yC(\mathbf{x})\right)$ measures the classification error of C.

Extremely Random Trees

ExtraTree

Random Forests are Random and Extremely Random Trees takes it one step further:

- The one key difference between Extra Trees and Random forests is in the selection of a threshold for the partition of the learning set.
- Instead of exhaustively searching for the threshold that creates the maximum purity of partitions for each feature, ExtraTrees randomly picks a threshold for each feature and then splits for maximum purity.
- This results in a method which trains faster and therefore able to take a data set to a classifier quicker than any discussed method.

Solution Process

Datasets

Dataset Properties				10-fold CV	
Dataset	Genes	Samples	Ratio	Train	Test
Liver	5520	181	76:105	163	18
Bladder	6688	125	22:103	112	13
Leukemia	7129	73	25:48	66	7
Colon	2000	62	22:40	56	6
R. Prostate	243	139	52:87	125	14

- Five datasets, 1000s of genes, about 100 samples
- Training and test sets formed with 10-fold cross-validation

Numerical Implementation

Python 3.5

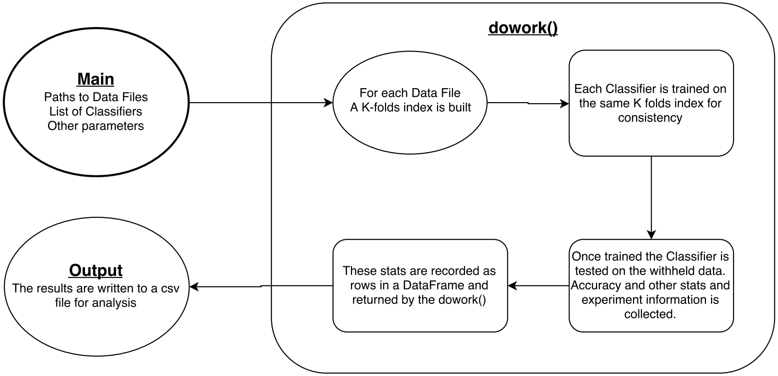
- Flexible interpreted language
- Many packages specializing in numerical and scientific computing (numpy, scipy, pandas)

SciKit-Learn

- Open source machine learning library
- Contains models, preprocessing, and validation tools

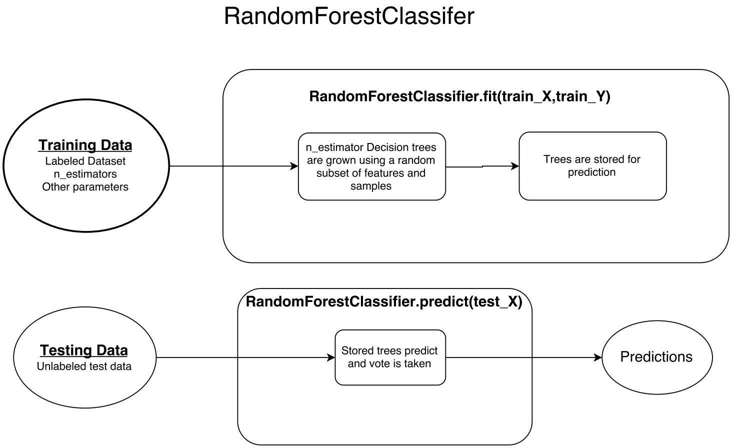
Numerical Implementation

General Dataflow in sklearnmodels.py



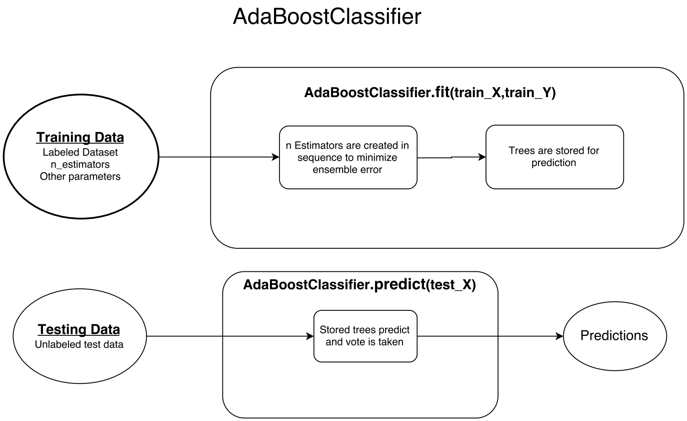
dowork()
Main
Paths to Data Files
List of Classifiers
Other parameters
For each Data File
A K-folds index is built
Each Classifier is trained on the same K folds index for consistency
Once trained the Classifier is tested on the withheld data. Accuracy and other stats and experiment information is collected.
These stats are recorded as rows in a DataFrame and returned by the dowork()
Output
The results are written to a csv file for analysis

Numerical Implementation

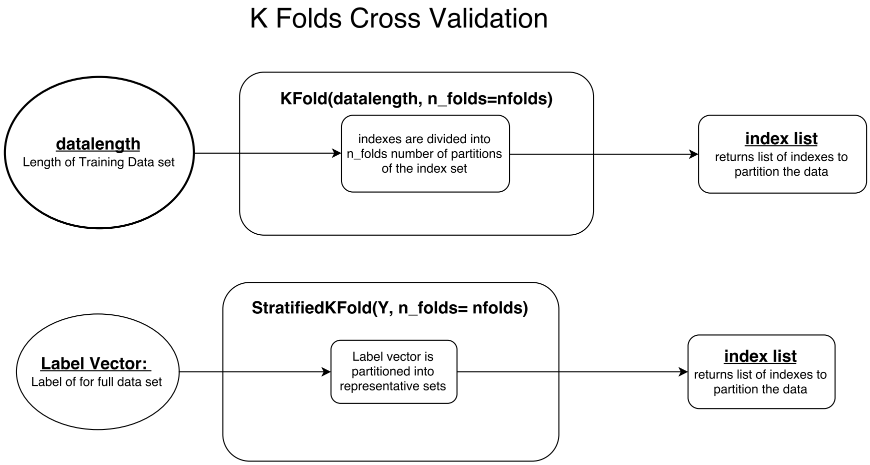


RandomForestClassifer
RandomForestClassifier.fit(train_X,train_Y)
Training Data
Labeled Dataset
n_estimators
Other parameters
n_estimator Decision trees are grown using a random subset of features and samples
Trees are stored for prediction
RandomForestClassifier.predict(test_X)
Testing Data
Unlabeled test data
Stored trees predict and vote is taken
Predictions

Numerical Implementation


AdaBoostClassifier
AdaBoostClassifier.fit(train_X,train_Y)
Training Data
Labeled Dataset
n_estimators
Other parameters
n Estimators are created in sequence to minimize ensemble error
Trees are stored for prediction
AdaBoostClassifier.predict(test_X)
Testing Data
Unlabeled test data
Stored trees predict and vote is taken
Predictions

Numerical Implementation


K Folds Cross Validation
datalength
Length of Training Data set
KFold(datalength, n_folds=nfolds)
indexes are divided into n_folds number of partitions of the index set
index list
returns list of indexes to partition the data
Label Vector:
Label of for full data set
StratifiedKFold(Y, n_folds= nfolds)
Label vector is partitioned into representative sets
index list
returns list of indexes to partition the data

Case Studies

(1) Optimal data interaction for the methods?

- Bootstrap sampling with or without replacement (Random forest)
- Number of random features (Random forest)
- Stratified sampling (Random forest, AdaBoost)
    - Modification of cross-validation
    - Preserves class proportions

(2) Improvements as number of trees increases?

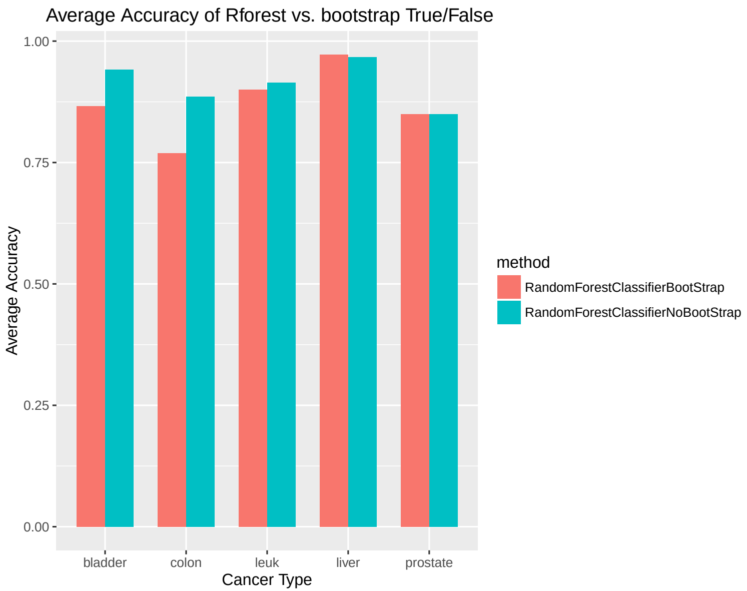
- Decision tree baseline

(3) Which ensemble method performs best?

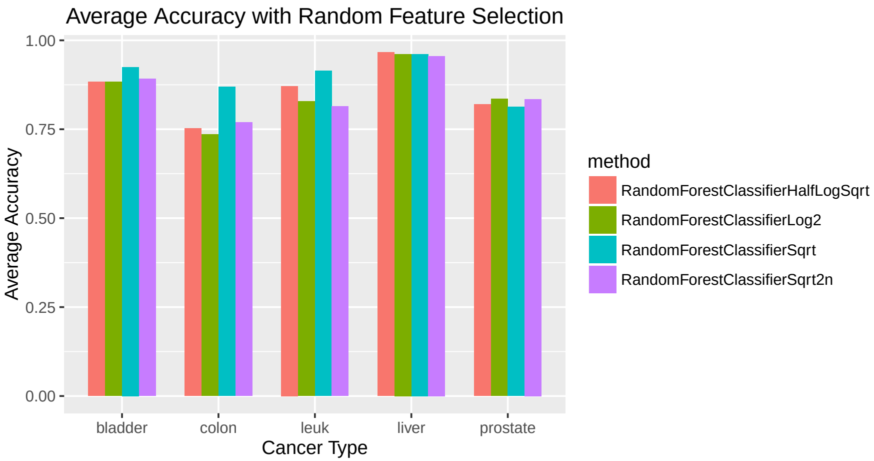
- Compare to outside methods

Findings

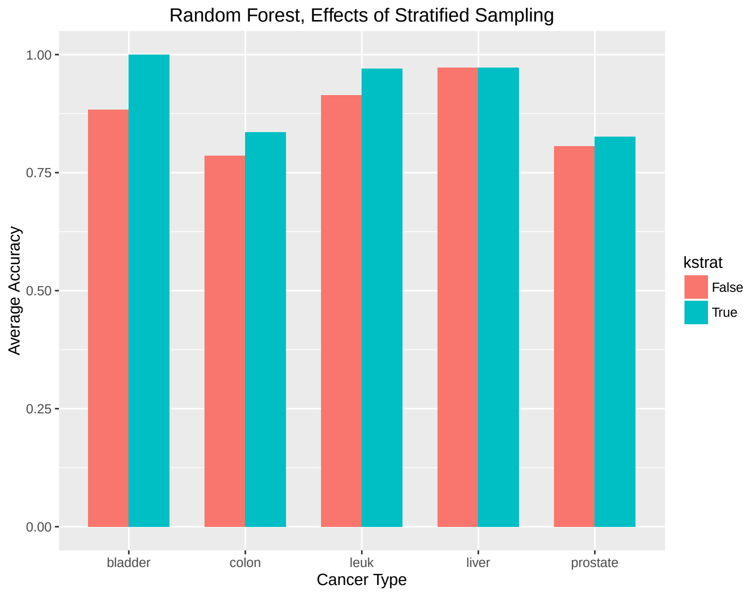
Case Study 1: Bootstrap w/ or w/o replacement



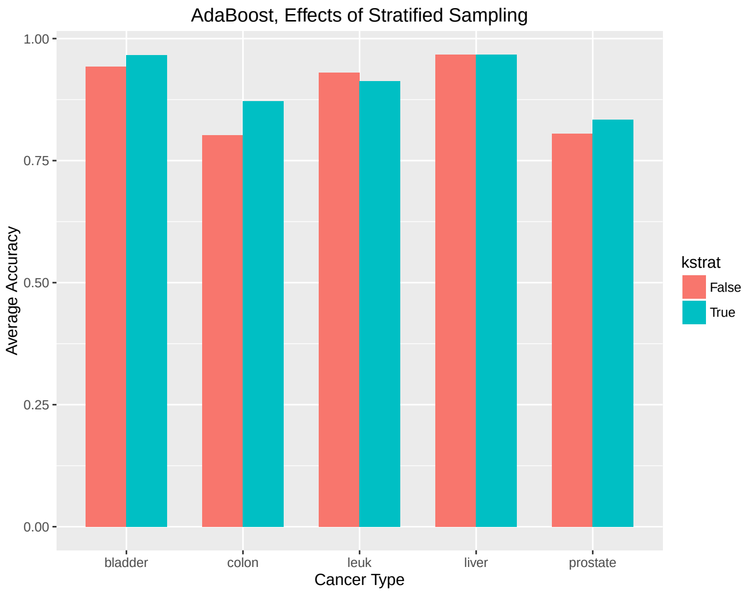
Case Study 1: Number of features



Case Study 1: Stratified sampling improves performance

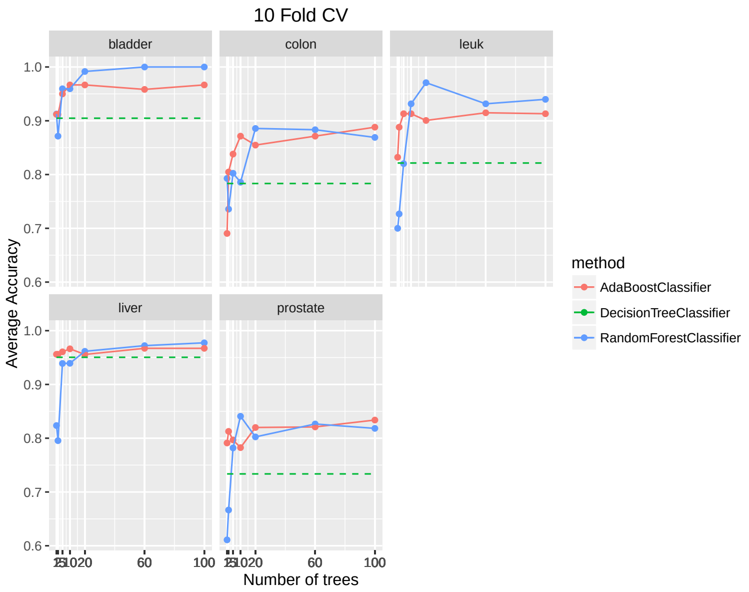


Case Study 1: Stratified sampling improves performance

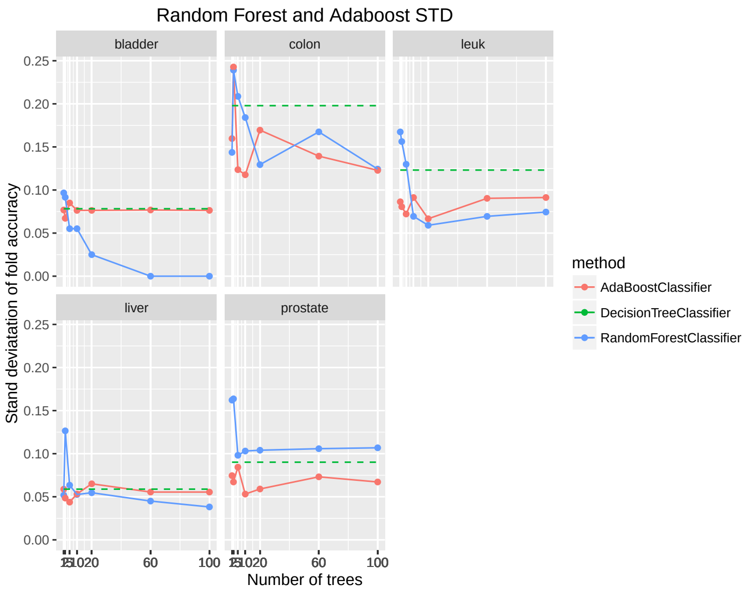


Case Study 2: Number of Trees

Case Study 2: Accuracy increases with number of trees



Case Study 2: Variance decreases with number of trees



Case Study 3: Which method is best?

Case Study 3: Comparison of Ensemble Methods

Classification Average Accuracy

Method	Bladder	Colon	Leukemia	Liver	R. Prostate
Optimal BPR LS	**100%**	87.1%	**95.89%**	**97.24%**	
AdaBoost	96.67%	88.81%	91.31%	96.72%	**83.38%**
Bagging	95.83%	81.67%	93.75%	96.16%	81.16%
Extra Random	**100%**	**93.33%**	95.65%	97.22%	82.00%
Random Forest	99.17%	83.57%	94.40%	96.69%	82.65%

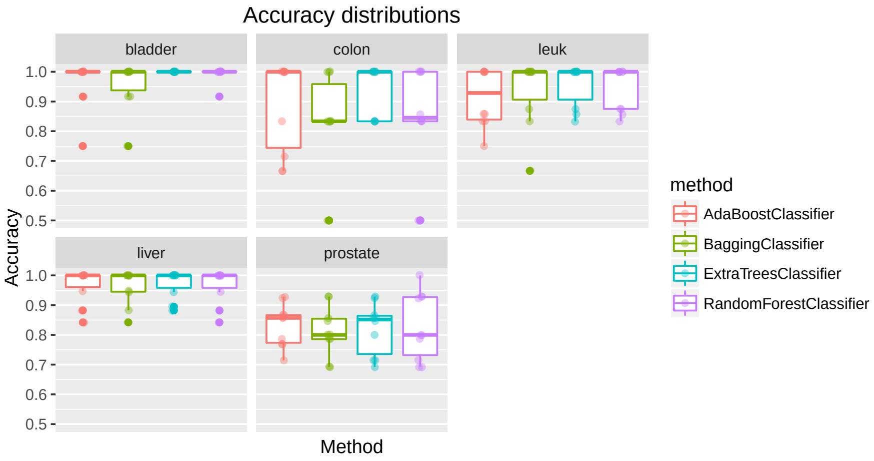
In terms of a ranked vote:

(1) Optimal BPR LS & Extra Randomized Trees

(2) AdaBoost & Random Forest

(3) Bagging

Case Study 3: Ensemble Methods Distribution



Conclusion

- Tree ensemble methods are valid for detecting cancer
- They are very fast
- Multiple runs generate a “distribution” of predictions
- Develop “pre-screening tests”

Future Work

- Effects of preprocessing data on tree-ensemble methods
- Continued evaluation on more data sets
- Optimization of extremely randomized trees

Questions?

Appendix

(CS1a) Bootstrap: WR, WOR	
Data	All
kfolds	10
Stratified	False
Train/Test Fixed	True
Methods	Random forest
ntrees	100
nfeatures	$\sqrt{n}$

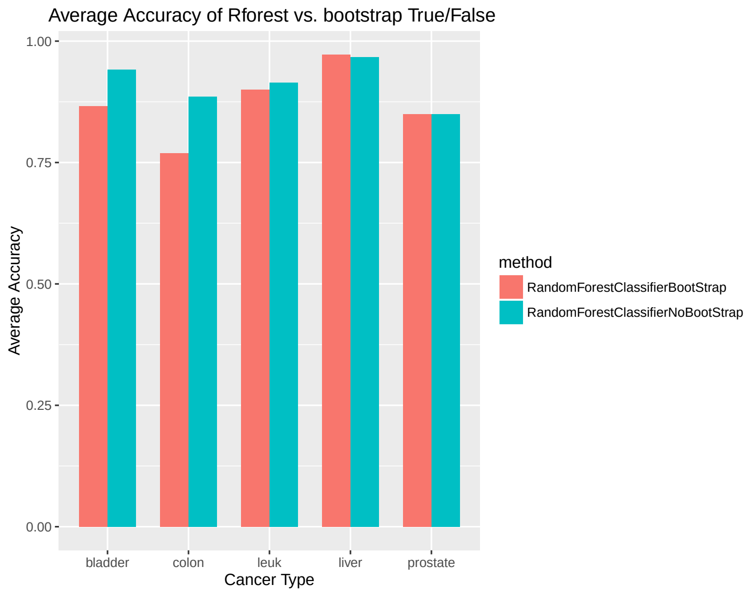
(CS1b) Nfeatures: $\log_2 n$, $\frac{1}{2}(\log_2 n + \sqrt{n})$, $\sqrt{n}$, $\sqrt{2n}$	
Data	All
kfolds	10
Stratified	False
Train/Test Fixed	True
Methods	Random forest
ntrees	100

(CS1c) Stratified: True, False	
Data	All
kfolds	10
Train/Test Fixed	False
Methods	Random forest, AdaBoost
ntrees	100
nfeatures	$\sqrt{n}$, max

(CS2) ntrees: 1, 2, 5, 10, 20, 60, 100	
Data	All
kfolds	10
Stratified	True
Train/Test Fixed	True
Methods	Rforest, Boost, dtree
ntrees	100, 100, 1
nfeatures	$\sqrt{n}$, max, max

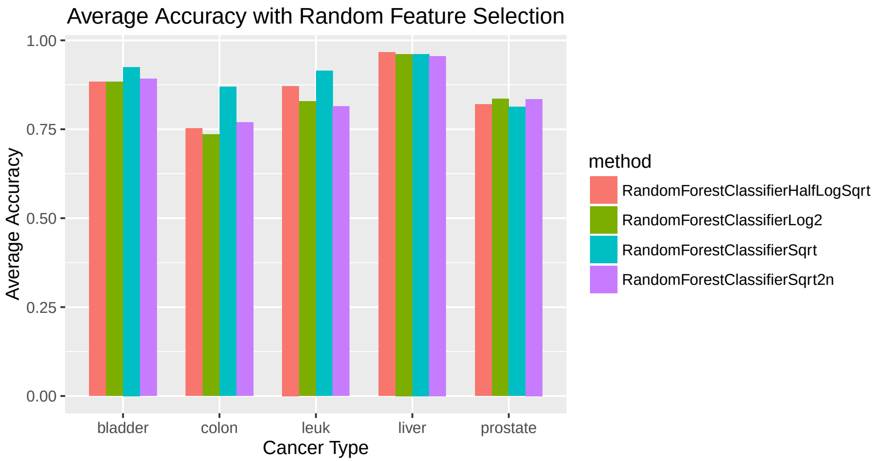
(CS3) Perf: Boost, Bag, Rforest, ETrees	
Data	All
kfolds	10
Stratified	True
Train/Test Fixed	True
ntrees	100
nfeatures	max, max, $\sqrt{n}$, $\sqrt{n}$

Case Study 1: Bootstrap w/ or w/o replacement



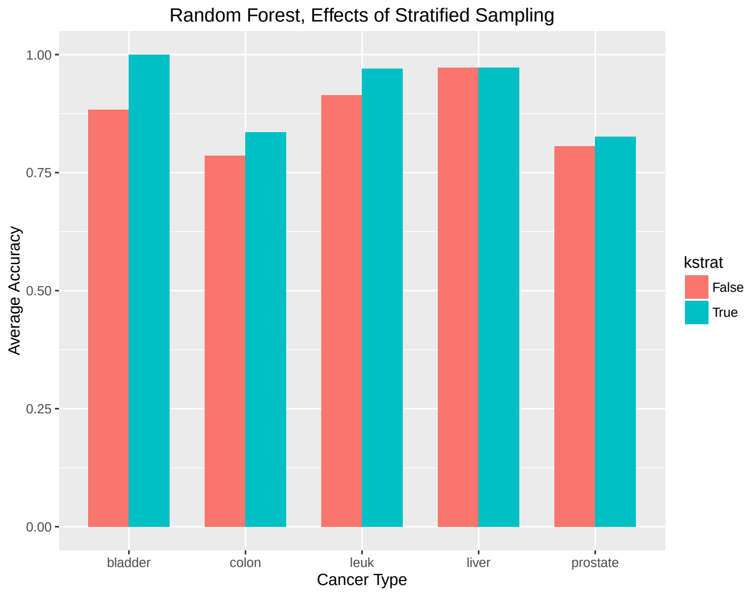
Bootstrap?	Bladder	Colon	Leuk	Liver	Prostate
True	86.67%	76.90%	90.00%	97.22%	84.95%
False	94.17%	88.57%	91.43%	96.67%	84.95%

Case Study 1: Number of features



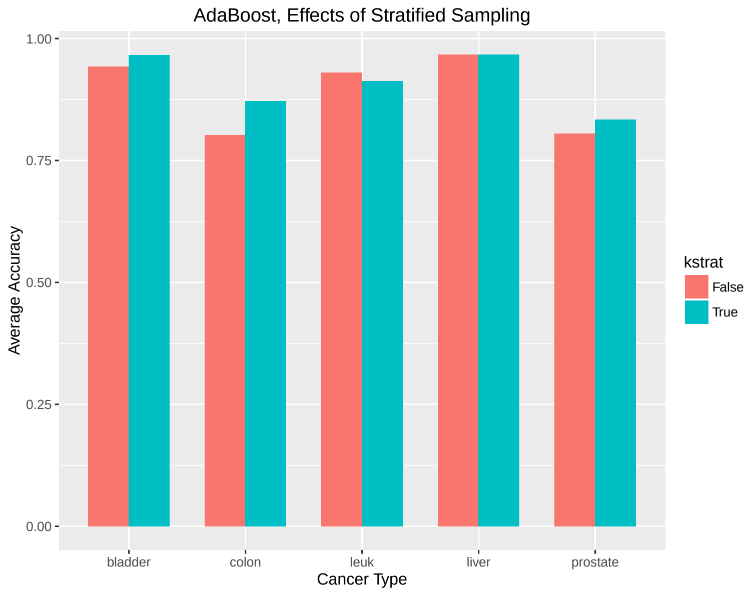
nfeatures	Bladder	Colon	Leuk	Liver	Prostate
HalfLogSqrt	88.33%	75.24%	87.14%	96.67%	82.03%
Log2	88.33%	73.57%	82.86%	96.11%	83.52%
Sqrt	92.50%	86.90%	91.43%	96.11%	81.32%
Sqrt2n	89.17%	76.90%	81.43%	95.56%	83.46%

Case Study 1: Stratified sampling improves performance



Stratified?	Bladder	Colon	Leuk	Liver	Prostate
False	88.33%	78.57%	91.43%	97.22%	80.66%
True	100%	83.57%	97.08%	97.22%	82.66%

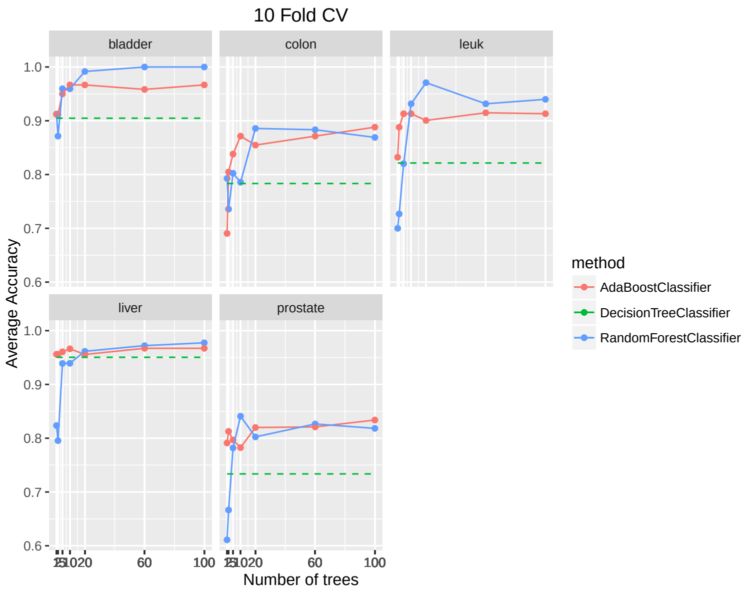
Case Study 1: Stratified sampling improves performance



Stratified?	Bladder	Colon	Leuk	Liver	Prostate
False	94.29%	80.24%	93.04%	96.70%	80.55%
True	96.67%	87.14%	91.31%	96.72%	83.38%

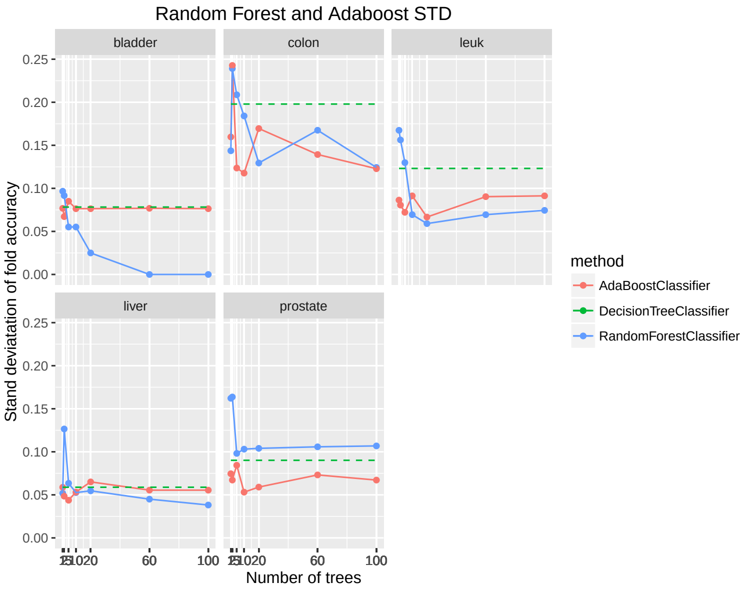
Case Study 2: Number of Trees

Case Study 2: Accuracy increases with number of trees



Method	Bladder	Colon	Leuk	Liver	Prostate
DecisionT	90.48%	78.33%	82.14%	95.05%	73.36%
RForest (100)	100%	86.90%	93.99%	97.74%	81.83%
AdaBoost (100)	96.67%	88.81%	91.31%	96.72%	83.38%

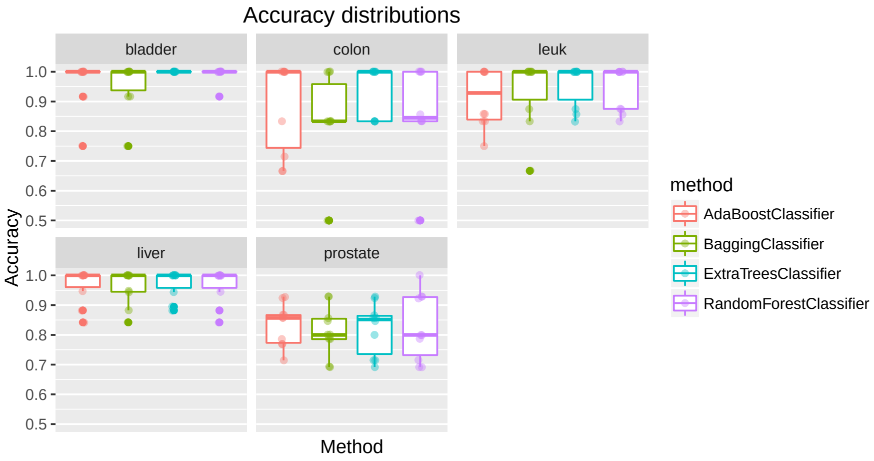
Case Study 2: Variance decreases with number of trees



Method	Bladder	Colon	Leuk	Liver	Prostate
DecisionT	0.0782	0.1979	0.1231	0.0589	0.0901
RForest (100)	0	0.1243	0.075	0.0382	0.1069
AdaBoost (100)	0.0764	0.1228	0.0913	0.0555	0.0672

Case Study 3: Which method is best?

Case Study 3: Ensemble Methods Distribution



Classification Standard Deviation

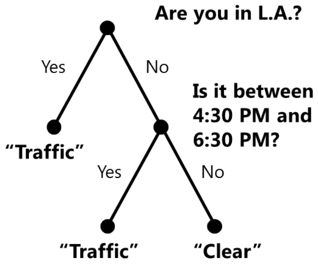
Method	Bladder	Colon	Leukemia	Liver	R. Prostate
AdaBoost	7.6%	14.4%	9.1%	5.5%	**6.7%**
Bagging	7.7%	17.4%	10.7%	5.5%	7.8%
Extra Random	**0%**	**8.2%**	**6.7%**	**4.5%**	8.2%
Random Forest	2.5%	18.3%	6.9%	5.6%	10.6%

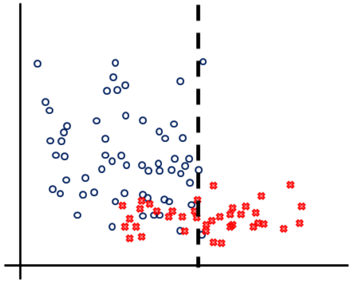
Overview of Statistical Methods

Statistical Methods (Overview)

Building Block: Classification tree

- Classifies by a series of yes/no questions
  - Ask questions that narrow your choices the most
- Involves greedily partitioning the observation space
- Easy to understand, but tends to overfit





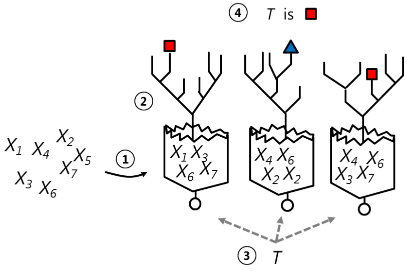
Statistical Methods (Overview)

Ensemble Methods: Operate on "weak" classifiers to produce "strong" classifiers

(1) Random forest

- k trees vote; plurality wins
- Each tree is grown from a random sampling of the data
- Aims to reduce variance without increasing bias

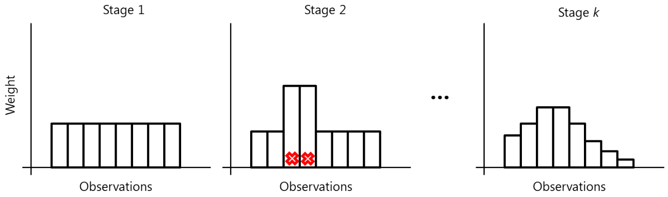
Note: Bagging (i.e., *B*ootstrap *Agg*regat*ing*) is a special case where observations are sampled with replacement by rolling an N-sided die m times for each learning subset



Statistical Methods (Overview)

(2) Boosting

- k weighted trees vote; plurality wins
- The trees evolve over k stages on reweighted data
- Aims to learn ‘hard’ observations



- More accurate trees receive greater vote

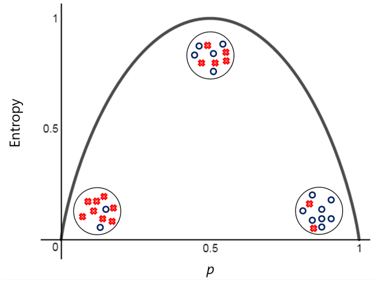
Entropy: A measure of diversity

Definition

Suppose A contains k classes of vectors with $|y_i|$ members in each class. Let $p_i = |y_i|/|A|$ be the proportion of class i observations. Then, the *entropy of A* is:

$$\text{Ent}(A) = -\sum_i p_i \log_2 p_i$$

Visualization for two classes



This sum is maximized when $p_i = 1/k$.

Intuitively, for $k = 2$, a set with $\text{Ent}(A) = 0$ (i.e., all one class) is trivial to classify, while a set with $\text{Ent}(A) = 1$ (i.e., equal amounts of each class) is impossible to classify.

Information Gain

We seek to minimize entropy to improve our classification ability. To this end, we define *information gain* which measures the change in entropy.

Information Gain

Suppose A is partitioned into subsets $A_1, ..., A_n$ by partition P. Let $q_j = |A_j|/|A|$ be the proportion of observations in A_j. Then, the *information gain of P* is:

$$\text{IG}(P) = \text{Ent}(A) - \sum_j q_j \,\text{Ent}(A_j)$$

E.g., given proposed set A^* and original set A,

$$\begin{aligned}\text{IG}(P) &= \Delta\text{Information} = \text{Information}(A^*) - \text{Information}(A) \\ &= \big(1 - \text{Ent}(A^*)\big) - \big(1 - \text{Ent}(A)\big) = \text{Ent}(A) - \text{Ent}(A^*).\end{aligned}$$

Maximizing the information gain thus corresponds to minimizing entropy.

A tree scans over each feature and splits at the point of minimum entropy.