

Random Forests

Albert A. Montillo, Ph.D.

University of Pennsylvania, Radiology
Rutgers University, Computer Science

Guest lecture: Statistical Foundations of Data Analysis
Temple University
4-2-2009

Outline

1. Problem definition
2. Decision trees
3. Why is random forest needed?
4. First randomization: Bagging
5. Second randomization: Predictor subsets
6. Putting it all together: RF algorithm
7. Practical considerations
8. Sample results
9. Additional information for free* *almost
10. Comparisons: random forest vs svm, boosting..
11. Conclusions & summary

Problem definition

random forest =

learning ensemble consisting of a bagging of un-pruned decision tree learners with a randomized selection of features at each split.

Decision trees are the individual learners that are combined

Decision trees ... one of most popular learning methods
commonly used for data exploration

One type of decision tree is called CART...classification
and regression tree *(Breiman 1983)*

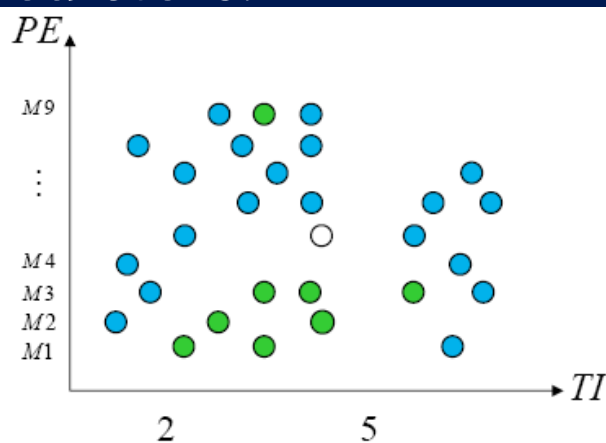
CART ... greedy, top-down binary, recursive partitioning
that divides feature space into sets of disjoint rectangular
regions.

- Regions should be pure wrt response variable.
- Simple model is fit in each region – majority vote for classification, constant value for regression

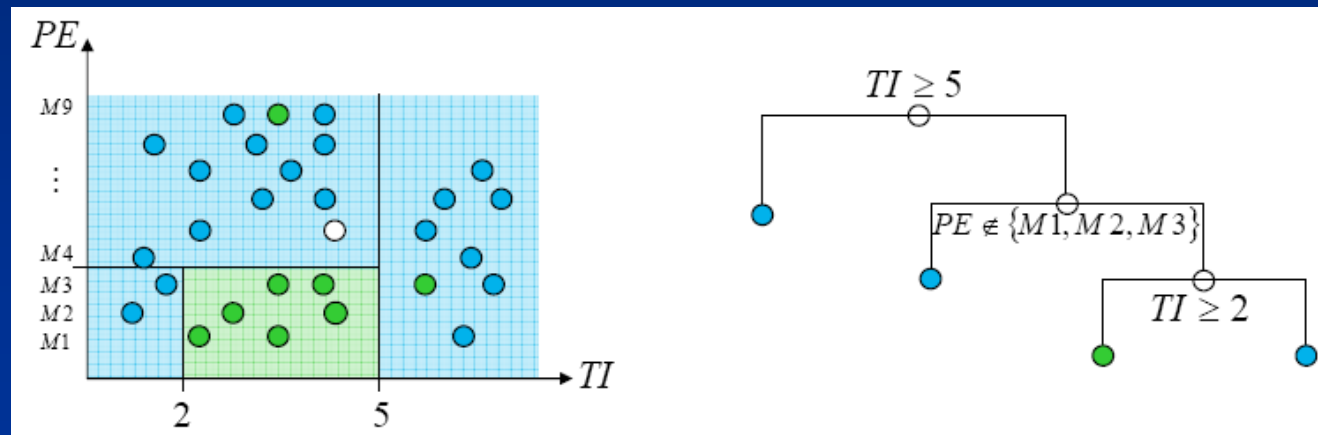
Decision trees involve greedy, recursive partitioning

Simple dataset with two predictors:

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



Greedy, recursive partitioning along TI and PE :



Decision trees: model, score criterion, search strategy

Model of underlying functional form sought from data

general



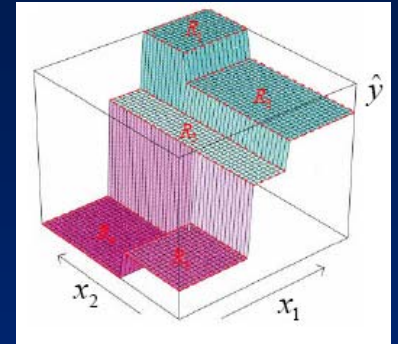
decision trees

$$\hat{F}(\mathbf{x}) = \hat{F}(\mathbf{x}; \mathbf{a}) \in \mathcal{F}$$

Training set: $D = \{y_i, x_{i1}, x_{i2}, \dots, x_{ik}\}_1^N = \{y_i, \mathbf{x}_i\}$

where $k = \# \text{predictors}$ and $N = \# \text{samples}$

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



Score criterion to judge quality of fit of model

$$\hat{R}(\mathbf{a}) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{F}(\mathbf{x}_i; \mathbf{a}))$$

Squared error: $L(y, \hat{y}) = (y - \hat{y})^2$

Search strategy to minimize the score criterion

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

$$\{\hat{c}_m, \hat{R}_m\}_1^M = \arg \min_{\{c_m, R_m\}_1^M} \sum_{i=1}^N \left[y_i - \sum_{m=1}^M c_m I(\mathbf{x}_i \in R_m) \right]^2$$

$\hat{y}_i = T(\mathbf{x}_i)$

Decision trees: 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 score 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  $R^r$ 
```

Grow tree until minimum node size (e.g. 5) is reached.
Then prune tree using cost-complexity pruning.

Decision trees: avoid overfitting through pruning

Larger the tree, more likely of overfitting training data.

Pruning finds subtree that generalizes beyond training data

Essence is to trading off tree complexity (size) and goodness of fit to the data (node purity).

Efficient split yields pure children: R^l and R^r

Decision trees: Efficiency of split measured through node purity

Regression –simple case; use squared error impurity measure. Let $R_m \subseteq \mathcal{R}^d$ and N_m denote # samples falling into R_m . Impurity of R_m for subtree T is given by $Q_m(T)$:

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i,$$
$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

Classification – squared error not suitable. Instead use: misclassification error, Gini index, or deviance. [see ESL ch 9]

Pruning involves finding the subtree T that minimizes the cost complexity objective function:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

For details see [Breiman 1984]

Why RF? Decision trees: Advantages and limitations

Advantages:

Characteristic	Neural nets	SVM	Trees	MARS	k-NN, kernels
Natural handling of data of "mixed" type	●	●	●	●	●
Handling of missing values	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●
Computational scalability (large N)	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●
Interpretability	●	●	●	●	●
Predictive power	●	●	●	●	●

TABLE 10.1. Some characteristics of different learning methods. Key: ● = good, ● = fair, and ● = poor.

Limitations:

- Low prediction accuracy
- High variance

→ Ensemble, to maintain advantages while increasing accuracy !

Random forest: first randomization through bagging

Bagging = bootstrap aggregation *(Breiman 1996)*

Bootstrap sample = create new training sets by random sampling the given one $N' \leq N$ times with replacement

Bootstrap aggregation ... parallel combination of learners, independently trained on distinct bootstrap samples

Final prediction is the mean prediction (regression) or class with maximum votes (classification).

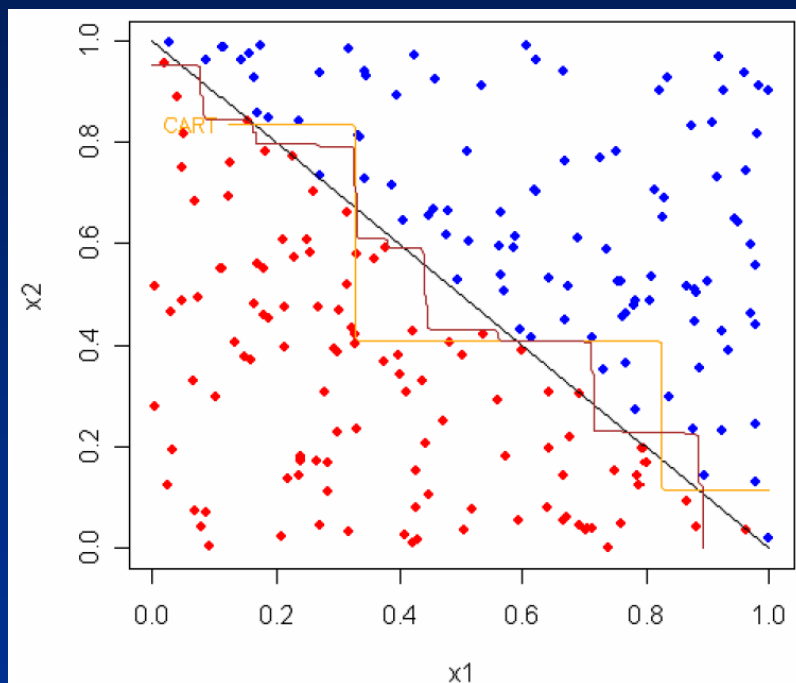
Using squared error loss, bagging alone decreases test error by lowering prediction variance, while leaving bias unchanged.

Bagging: reduces variance – Example 1

Two categories of samples: blue, red

Two predictors: x_1 and x_2

Diagonal separation .. hardest case for tree-based classifier

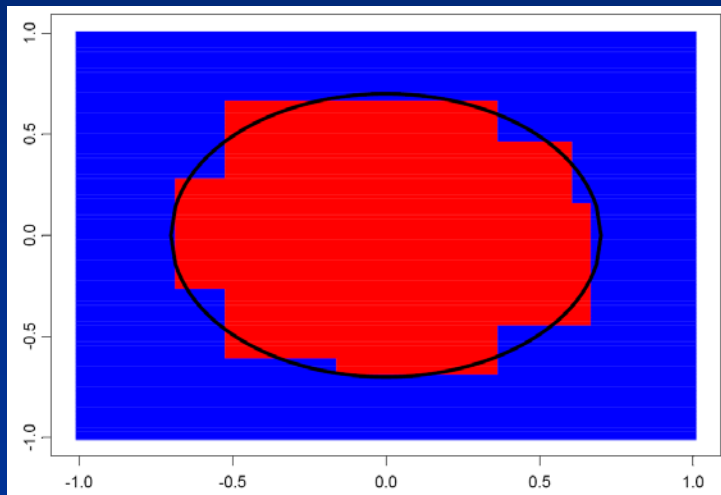
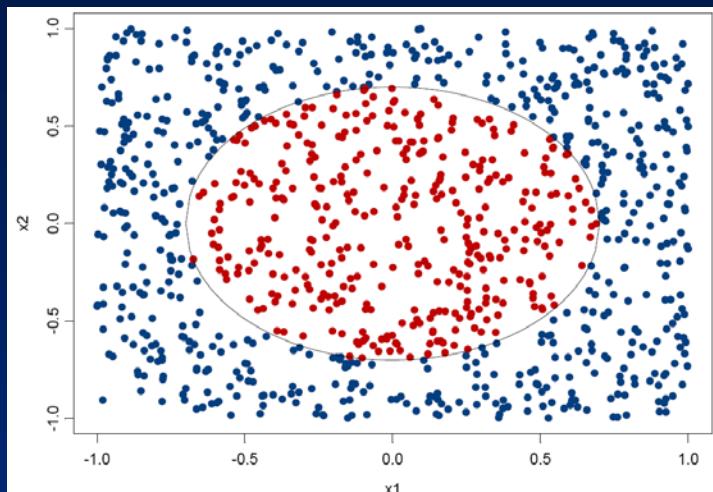


Single tree decision boundary in orange.

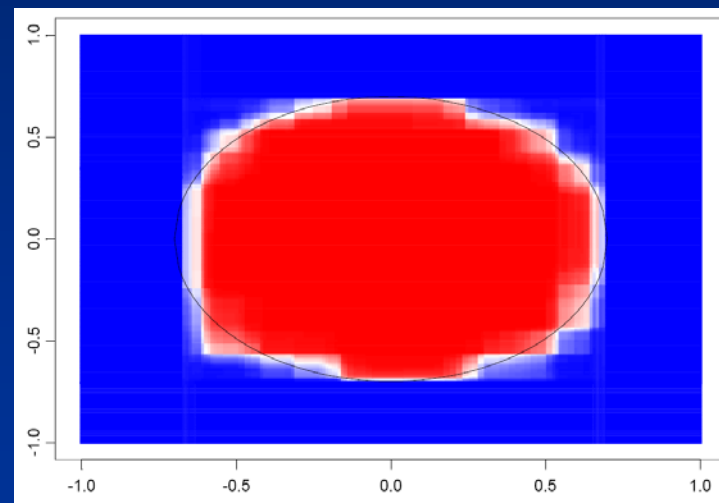
Bagged predictor decision boundary in red.

Bagging: reduces variance – Example 2

Ellipsoid separation →
Two categories,
Two predictors



Single tree decision boundary



100 bagged trees..

Random forest: second randomization through predictor subsets

Bagging alone utilizes the same full set of predictors to determine each split.

However random forest applies another judicious injection of randomness: namely by selecting a random subset of the predictors for each split *(Breiman 2001)*

Number of predictors to try at each split is known as m_{try} .
While this becomes new parameter, typically $m_{try} = \sqrt{k}$ (classification) or $m_{try} = k/3$ (regression) works quite well.
RF is not overly sensitive to m_{try}

Bagging is a special case of random forest where $m_{try} = k$

Random forest: second randomization through predictor subsets

Overall the additional randomness:

- further reduces variance even on smaller sample set sizes, improving accuracy
 - however significantly lowering predictor set at each node might lower accuracy, particularly if there are few good predictors among many non-informative “predictors”
- Subset of predictors much faster to search than **all** predictors

Random forest algorithm

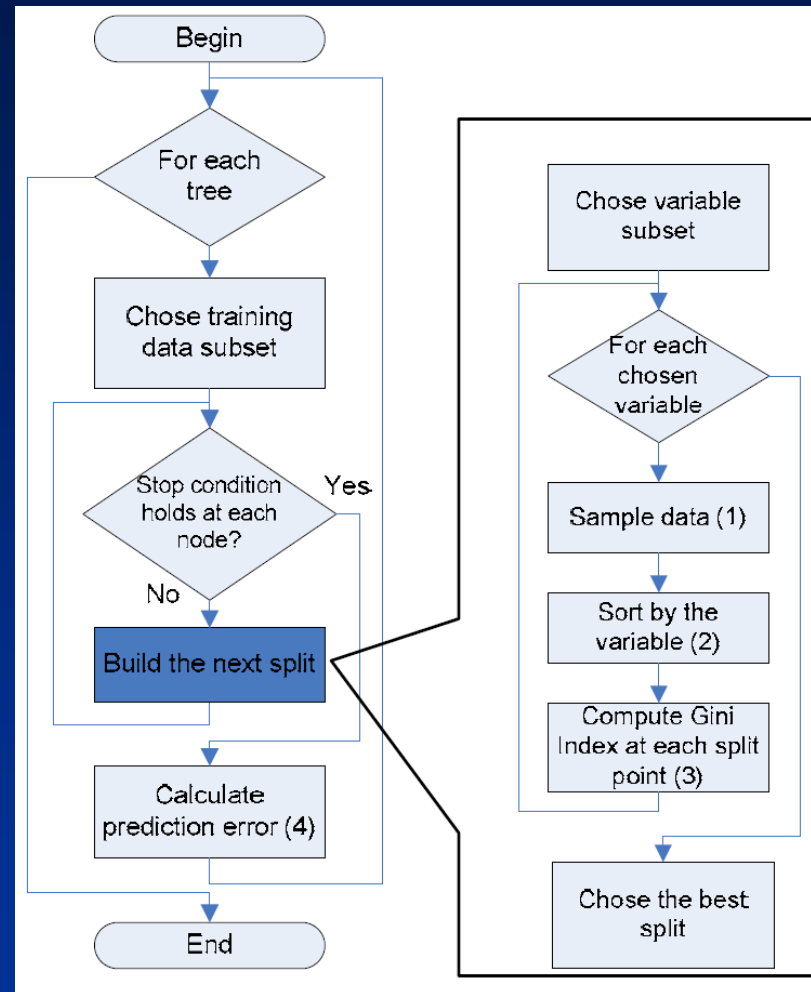
Let N_{trees} be the number of trees to build
for each of N_{trees} iterations

1. Select a new bootstrap sample from training set
2. Grow an un-pruned tree on this bootstrap.
3. At each internal node, randomly select m_{try} predictors and determine the best split using only these predictors.
4. Do not perform cost complexity pruning. Save tree as is, along side those built thus far.

Output overall prediction as the average response
(regression) or majority vote (classification) from all
individually trained trees

Random forest algorithm (flow chart)

For computer scientists:



Random forest: practical considerations

Splits are chosen according to a purity measure:

- e.g. squared error (regression), Gini index or deviance (classification) [see ESL ch 9]

How to select N_{trees} ?

- Build trees until the error no longer decreases

How to select m_{try} ?

- Try the recommended defaults, half of them and twice them and pick the best

Random forest: sample results

RF retains many benefits of decision trees ...

- Handles missing values, continuous and categorical predictors, and problems where $k \gg N$

while achieving results that decision trees cannot:

- RF does not require tree pruning to generalize
- Over variety of problem domains, RF improves prediction accuracy compared to single trees:

Test set misclassification error (%)

Data set	Forest	Single tree
Breast cancer	2.9	5.9
Ionosphere	5.5	11.2
Diabetes	24.2	25.3
Glass	22.0	30.4
Soybean	5.7	8.6
Letters	3.4	12.4
Satellite	8.6	14.8
Shuttle $\times 10^3$	7.0	62.0
DNA	3.9	6.2
Digit	6.2	17.1

Random forest: Additional information for free*

Estimating the test error:

- While growing forest, estimate test error from training samples
- For each tree grown, 33-36% of samples are not selected in bootstrap, called out of bootstrap (OOB) samples [Breiman 2001]
- Using OOB samples as input to the corresponding tree, predictions are made as if they were novel test samples
- Through book-keeping, majority vote (classification), average (regression) is computed for all OOB samples from all trees.
- Such estimated test error is very accurate in practice, with reasonable N_{trees}

Random forest: Additional information for free*

Estimating the importance of each predictor:

- Denote by \hat{e} the OOB estimate of the loss when using original training set, D .
- For each predictor x_p where $p \in \{1, \dots, k\}$
 - Randomly permute p^{th} predictor to generate a new set of samples $D' = \{(y_1, \mathbf{x}'_1), \dots, (y_N, \mathbf{x}'_N)\}$
 - Compute OOB estimate \hat{e}_k of prediction error with the new samples
- A measure of importance of predictor x_p is $\hat{e}_k - \hat{e}$, the increase in error due to random perturbation of p^{th} predictor

Random forest: Additional information for free*

Estimating the similarity of samples:

- During growth of the forest
 - Keep track of the number of times, samples x_i and x_j appear in the same terminal node
 - Normalize by N_{trees}
 - Store all normalized co-occurrences in a matrix, denoted as the proximity matrix.
- Proximity matrix can be considered a kind of similarity measure between any two samples x_i and x_j
- Can even use a clustering method to perform unsupervised learning from RF. For details see

[Liam A, 2002]

Comparisons: random forest vs SVMs, neural networks

Random forests have about same accuracy as SVMs and neural networks

RF is more interpretable

- Feature importance can be estimated during training for little additional computation
- Plotting of sample proximities
- Visualization of output decision trees

RF readily handles larger numbers of predictors.

Faster to train

Has fewer parameters

Cross validation is unnecessary : It generates an internal unbiased estimate of the generalization error (test error) as the forest building progresses

Comparisons: random forest vs boosting

Main similarities:

- Both derive many benefits from ensembling, with few disadvantages
- Both can be applied to ensembling decision trees.

Main differences:

- Boosting performs an exhaustive search for best predictor to split on; RF searches only a small subset
- Boosting grows trees in series, with later trees dependent on the results of previous trees; RF grows trees in parallel independently of one another.

Comparisons: random forest vs boosting

Which one to use and when...

- RF has about the same accuracy as boosting for classification, For continuous response, boosting appears to outperform RF
- Boosting may be more difficult to model and requires more attention to parameter tuning than RF.
- On very large training sets, boosting can become slow with many predictors, while RF which selects only a subset of predictors for each split, can handle significantly larger problems before slowing.
- RF will not overfit the data. Boosting can overfit (though means can be implemented to lower the risk of it).
- If parallel hardware is available, (e.g. multiple cores), RF embarrassingly parallel with out the need for shared memory as all trees are independent

Conclusions & summary: Random forests offers:



Fast fast fast!

- RF is fast to build. Even faster to predict!
- Practically speaking, not requiring cross-validation alone for model selection significantly speeds training by 10x-100x or more.
- Fully parallelizable ... to go even faster!

Automatic predictor selection from large # of candidates

- RF can be used for feature selection alone; or to streamlining other, slower learners

Resistance to over training

Ability to handle data without preprocessing

- data does not need to be rescaled, transformed, or modified
- resistant to outliers
- automatic handling of missing values

Cluster identification

- can be used to generate tree-based clusters through sample proximity

Interpretability

- Predictor importance, sample proximity and tree structure offer insights into data

References:

1984, Breiman L, Friedman J, Olshen R, Stone C, Classification and Regression Trees; Chapman & Hall; New York

1996, Breiman L, Bagging Predictors. Machine Learning 26, pp 123-140.

2001, Breiman L, Random Forests. Machine Learning, 45 (1), pp 5-32.

2001, Breiman L, Statistical modeling: the two cultures, Statistical Science 2001, Vol. 16, No. 3, 199-231

2001, Hastie T, Tibshirani R, Friedman J, The Elements of Statistical Learning; Springer; New York

2002, Salford Systems; Salford Systems White Paper Series, www.salford-systems.com

2002, Liaw A, Wiener M, Classification and Regression by Random Forest, R News, Vol 2/3, Dec

2002, Breiman L, Looking inside the black box, Wald Lecture Series II

2005, Cutler, Random Forests, Encyclopedia of Statistics in Behavioral Science, pp 1665–1667.

2006, Sandri M, Zuccolotto P, Variable selection using random forests, Data analysis, classification and the forward search, Springer, pp. 263-270

Thank you

... Questions?