# Data Science Lecture Notes 13

Donghui Yan

University of Massachusetts Dartmouth

# Outline

- Introduction
- Trees
- Random Forests
- The randomForest package

# Revisiting the classification problem

- Given a training sample $(X_1, Y_1), ..., (X_n, Y_n)$, we wish to learn the relationship $f : X \mapsto Y$ s.t.

$$\min_{f \in \mathcal{F}} \mathbb{E} l(f(X), Y)$$

for any loss metric $l(.,.)$ that we care

  - $X_i$ called features, $Y_i \in \{1, 2, ..., J\}$ labels
  - $\mathcal{F}$ is the function class that the target function $f$ resides

- Restrict $\mathcal{F}$ to recursively defined piecewise constant functions $\rightarrow$ trees.

# Tree-based classifiers

- Recursive partition of the space into disjoint regions (rectangles)
- Formally, *a piecewise constant function*

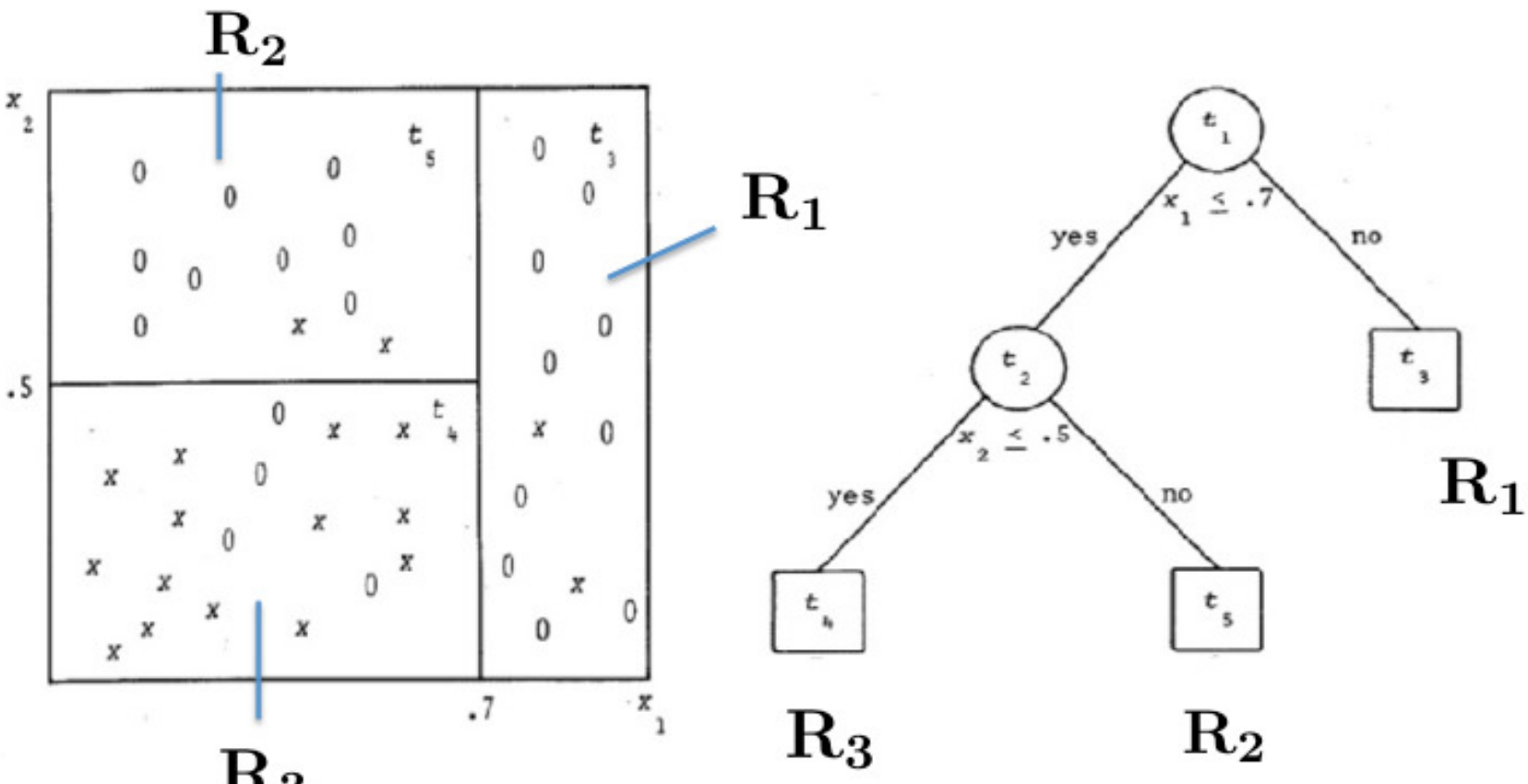
$$f(X) = \sum_{i=1}^{T} h(R_i).1_{\{X \in R_i\}}$$

where $\bigcup_{i=1}^{T} R_i$ is a partition of the space s.t. $\bigcap_{i=1}^{T} R_i = \emptyset$

- Classification by majority vote on the labels
  - Of all data points in the same leaf node as $X$
  - Or average on values for regression.

# Space partition and classification tree

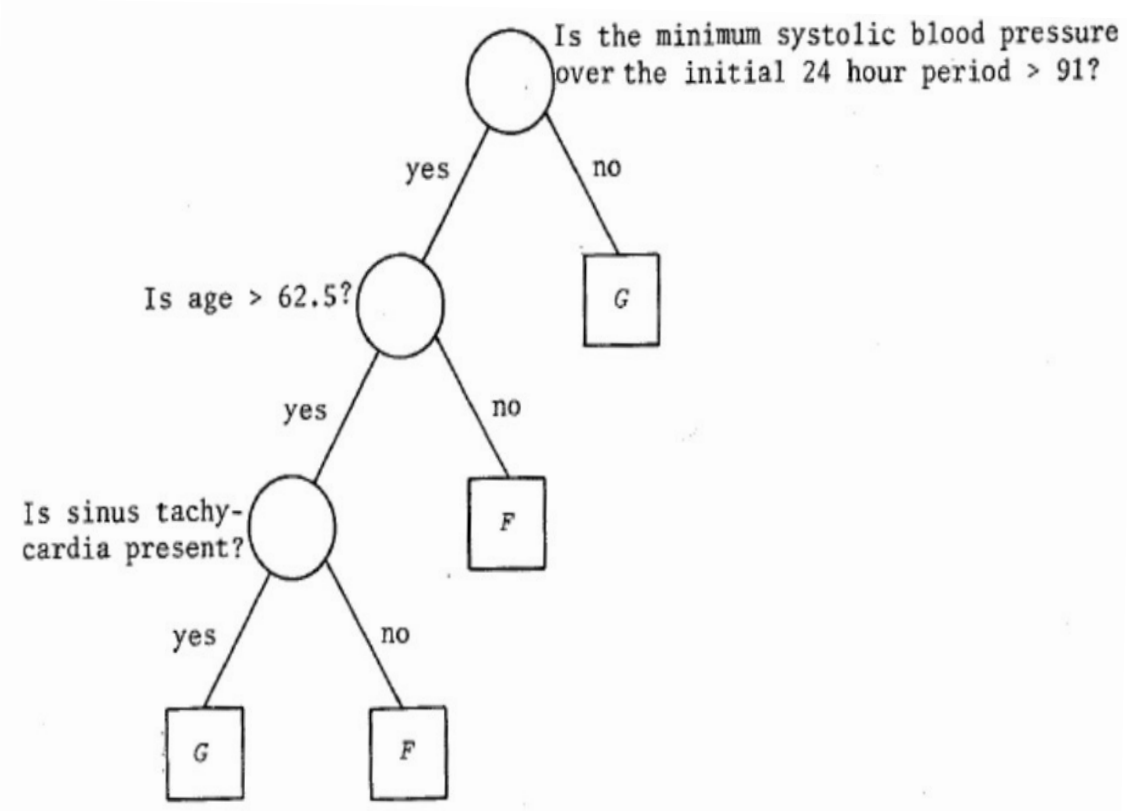
- $1^{st}$ partition: $R_1, (R_2 \cup R_3)$. $2^{nd}$ partition: $R_2, R_3$
- $h(R_1) =$ '0', $h(R_2) =$ '0', $h(R_3) =$'x'.



# A real example of a classification tree

Risk analysis of heart attack patients (UCSD Medical Center)

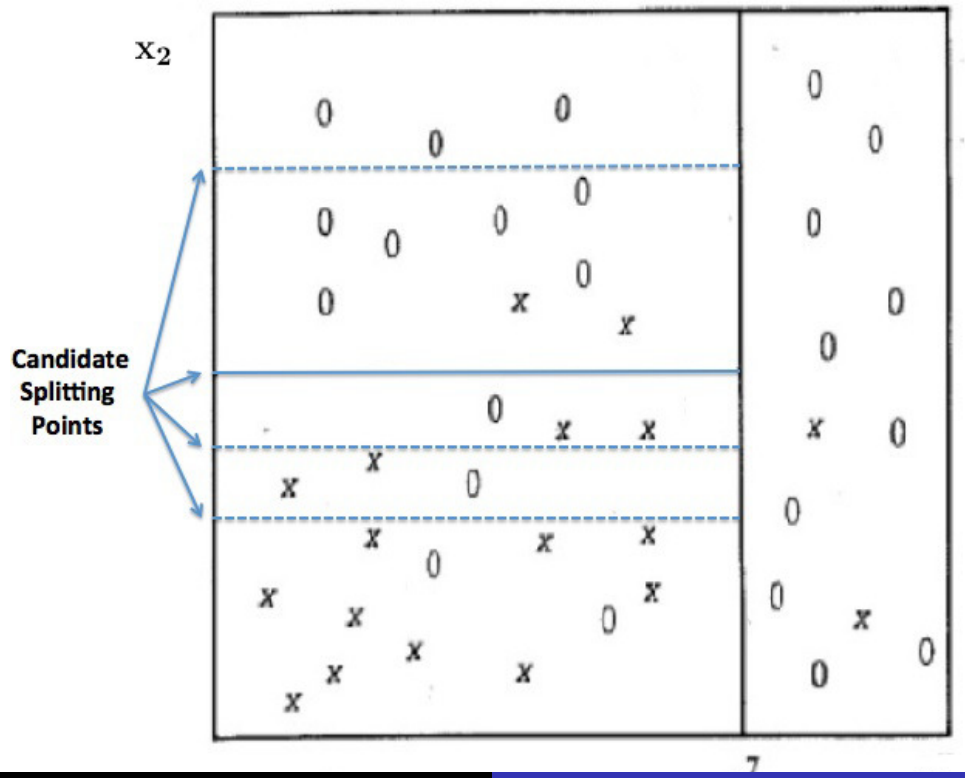
- G = "high risk", F = "low risk".



# Construction of a tree

A tree is built by *recursive splitting* of tree nodes until stop.

- The root node $\leftrightarrow$ the entire data space
- "Split" means partition the data corresponding to a tree node.



# Construction of a tree

- Construction of a tree revolves around 3 elements
  - The selection of splits (on which feature and at which point)
  - The stopping rule
  - The assignment of each terminal node to a class
    - Often related to the splitting criterion
- Additionally, there is the issue of how to prune a tree
  - Trade-off between predictive power and tree complexity.

# Construction of a tree (the basic algorithm)

- Root node $\leftrightarrow$ the training set
- Set up a list (initially empty), $L_{act}$, for active nodes
- Put the root node into the list
- Repeat the following until a stopping criterion is met
  - Scan through $L_{act}$ and gather candidate splitting points
  - Calculate reduction of node impurity for all candidates
  - Candidate leading to $max$ reduction of impurity $\rightarrow$ splitting point
  - Split the corresponding node and place its child nodes into $L_{act}$.

# Classification or prediction with a tree

- Let $X$ denote an observation in the test set
- Let the root node be the working node $N_w$
- Repeat the following until reaching a leaf node
  - Let $N_w^L$ be left child of $N_w$ and $N_w^R$ right child
  - Compare $X$ with the splitting point at node $N_w$
    - If $X$ is smaller then $N_w \leftarrow N_w^L$
    - Otherwise $N_w \leftarrow N_w^R$
- Majority vote over node $N_w$ and label $X$ accordingly.

# Splitting criteria

- Node impurity based
  - Gini (CART, 1984)
  - Entropy (J. Quinlan, 1983; Fayyad et al, 1992)
- Node separation based
  - Various clustering and SVM based criteria (Ho, 1998)
  - Linear discriminant analysis based (Loh et al, 1988-)
  - Margin tree (Tibshirani and Hastie, 2006)
- Miscellaneous
  - Mingers' statistic (1986), G statistic (Mingers, 1987), corrections to entropy (Marshall, 1986; Quinlan, 1986).

# Node impurity based criteria (continued)

- The "goodness-of-split" function has the form

$$\theta(s,t) = \phi(\boldsymbol{p}) - P_L \cdot \phi(\boldsymbol{p}_L) - P_R \cdot \phi(\boldsymbol{p}_R)$$

- Some commonly used impurity functions
  - Gini: $\phi(\boldsymbol{p}) = \sum_{j=1}^{J} p_j(1-p_j)$ (CART, 1984)
  - Entropy: $\phi(\boldsymbol{p}) = -\sum_j p_j \log p_j$ (C4.5, Fayyad et al, 1992)

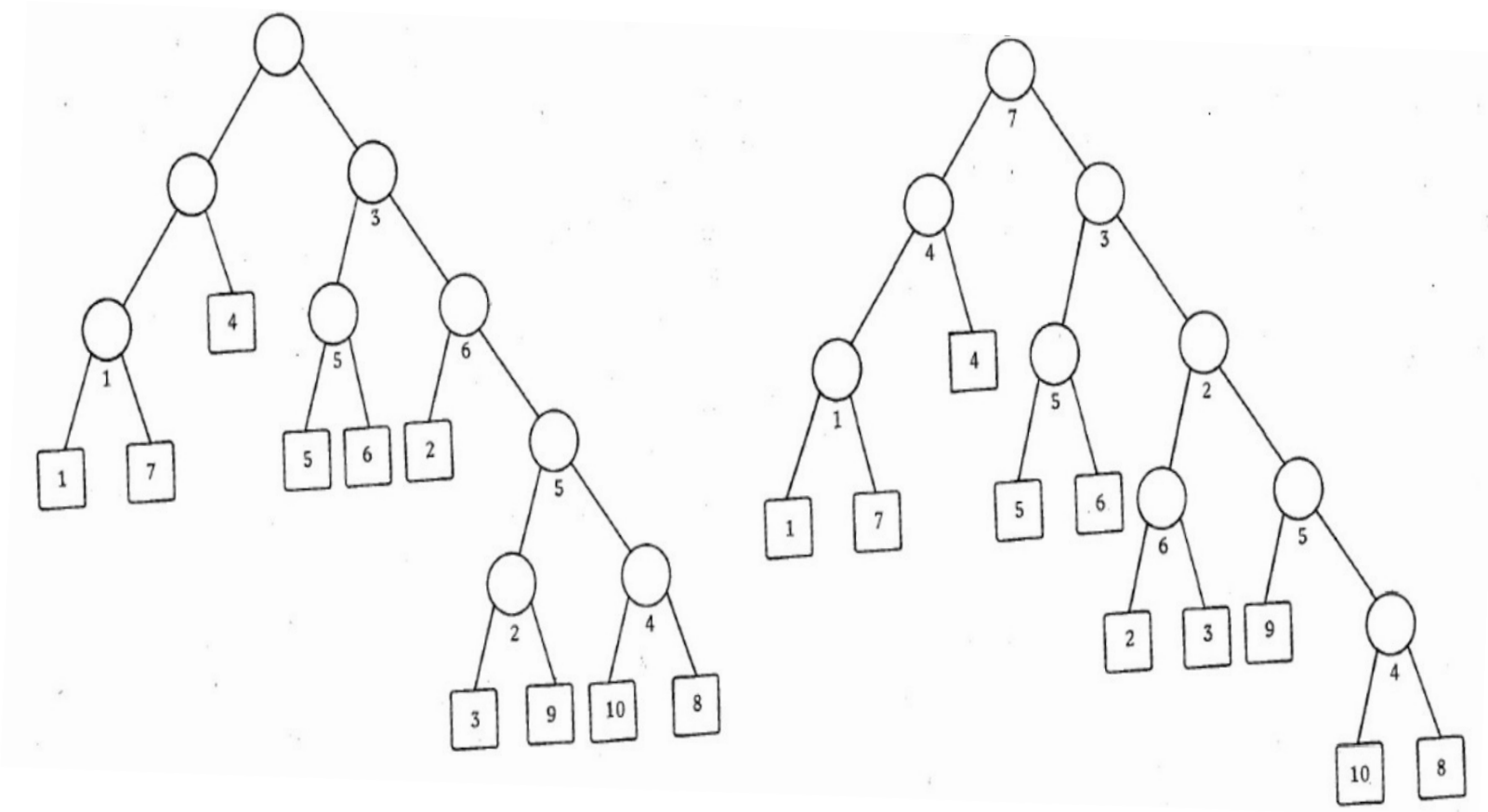
- Others
  - Twoing: $\theta(s,t) = \frac{P_L P_R}{4}\left[\sum_j |p_{jL} - p_{jR}|\right]^2$ (CART, 1984).

# Node impurity based criteria

**Theorem [Breiman, 1994].** Optimal splits do not split classes.

- Property of some splitting criteria
  - Gini sends the *largest* class to left node and the rest to the right
  - The entropy criterion split classes s.t. $|P_L - 0.5|$ is maximized, hence it produces relatively *balanced* trees
  - Twoing is similar to the entropy criterion.

# Example Gini-tree and Twoing-tree



# Some notable work

- CART (Breiman, Friedman, Olshen and Stone, 1984)
  - Split on multiple features or linear combination of features
  - Gini and Twoing for node splitting
- C4.5 and its variants (J. Quinlan, 1986-1989)
  - Entropy minimization
- CRUISE, QUEST and variants (Loh et al, 1988-)
  - Multi-way split of nodes
  - linear discriminant function for node split
- Multiple response
  - M. Segal (1992)
  - H. Zhang (1998).

# A historical note about CART

- pre-CART age
  - Regression trees date back to AID (Morgan and Sonquist, 1963)
- The development of CART
  - Work began in 1973 when Breiman and Friedman, independently, "reinvented the wheel" and used tree methods in classifiers
  - Breiman and Friedman joined forces and later joined by Stone
  - Olshen was an early user and then contributed in theory
  - Idea of a book in 1980 and published by 1984.

*"While the pregnancy has been rather prolonged, we hope that the baby appears acceptably healthy to the members of our statistical community" (CART)*

# Some practical issues

- How to choose the right-sized tree
  - Too big a tree tends to overfit
  - Too small would cause "lack of fit" (performance suffers)
  - How to grow a right sized tree?
    - "Grow and prune" (CART, Breiman et al 1984).
- How to deal with missing values
  - "Missing values" in data is a rule in real world
  - Solutions
    - Simply ignore all instance with missing attributes
    - Replace missing values with the mean or mode of the attribute over all instances with the same target value (Loh and Shih, 1997)
    - "Surrogate" split (Breiman et al 1984).

# Tree-based regression

- Essentially the same as the classification tree
- Except different choices of loss function
  - Least absolute deviation regression

$$l(\hat{f}, (X_i, Y_i)_{i=1}^{N_t}) = \sum_{i=1}^{N_t} |\hat{f}(X_i) - Y_i|$$

  - Least squared error

$$l(\hat{f}, (X_i, Y_i)_{i=1}^{n}) = \sum_{i=1}^{N_t} \left(\hat{f}(X_i) - Y_i\right)^2$$

    - The minimizer is average of points in same leaf node
  - Least squared error is favored for computational reason.

# Trees in R

- The "tree" package
  - Ripley (2015-)
  - Classification and regression trees based on CART
- The "rpart" package
  - Therneau, Atkinson, and Ripley (2004-)
  - Recursive partitioning for classification, regression and survival trees
  - Implemented most of CART.

# The Iris flower data

- Introduced by Fisher (1936) for discriminant analysis
- Three species of Iris with each 50 observations
  - Iris setosa, Iris versicolor and Iris virginica
  - Four features
    - The length and width of the sepals and petals.



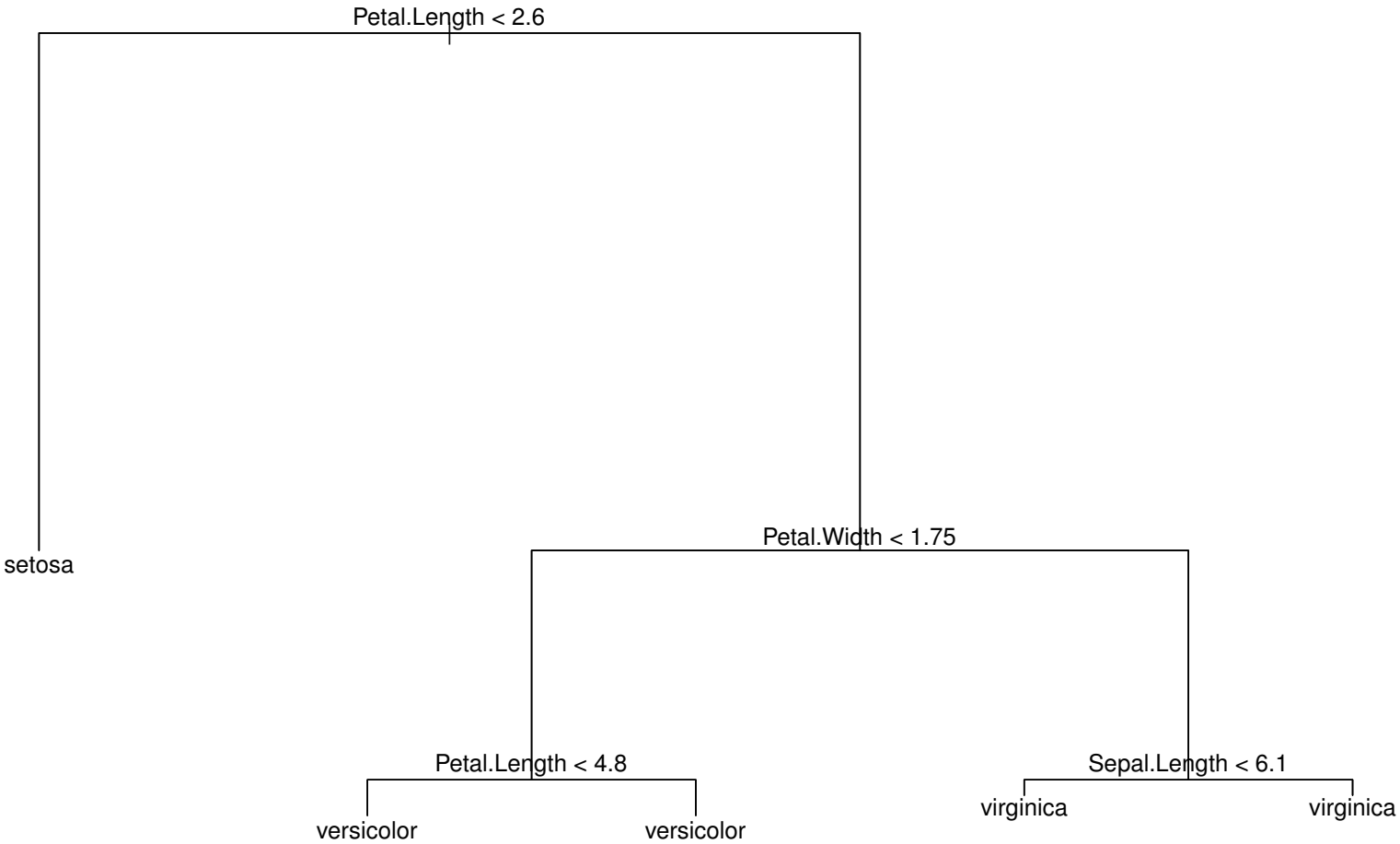
# R code on the Iris data (training)

```
##Uncomment if use this package for the first time
##install.packages("tree");
library(tree);

data(iris);
n<-nrow(iris);
idx<-sample(1:n, floor(n/2), replace=FALSE);
iris_train<-iris[idx,]; iris_test<-iris[-idx,];
labels<-class.ind(iris$Species);

myiris<-tree(Species ~ ., data=iris_train);
```

# The fitted tree on the Iris data



# R code on the Iris data (test)

```
test.cl <- function(true, pred) {
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}

conf<-test.cl(labels[-idx,], predict(myiris, iris_test));
acc<-sum(diag(conf))/sum(conf);
cat("The accuracy on the test set is", acc,"\n");
```

# Test on the Iris data

The confusion matrix is given by

```
          cres
true  1  2  3
   1 23  0  0
   2  0 23  0
   3  0  3 26
```

```
The accuracy on the test set is 0.96
```

- Accuracy = (23+23+26)/(23+23+26+3) = 0.96.

# Summary

- Flexible (continuous + categorical variables)
- Fast to construct and predict, 'good' interpretations
- Invariant to monotone transformations on individual variables
- Automatic incorporation of variable selection
- Not stable and prune to perturbations (noise) in the training set
  - *"Good but not great classifiers"* (Leo Breiman)

One way to turn into a 'great' classifier is by ensemble

- Boosting (Freund and Schapire, 1995)
- Bagging (Random Forests, Breiman (1996, 1999)).

# Random Forests (RF)

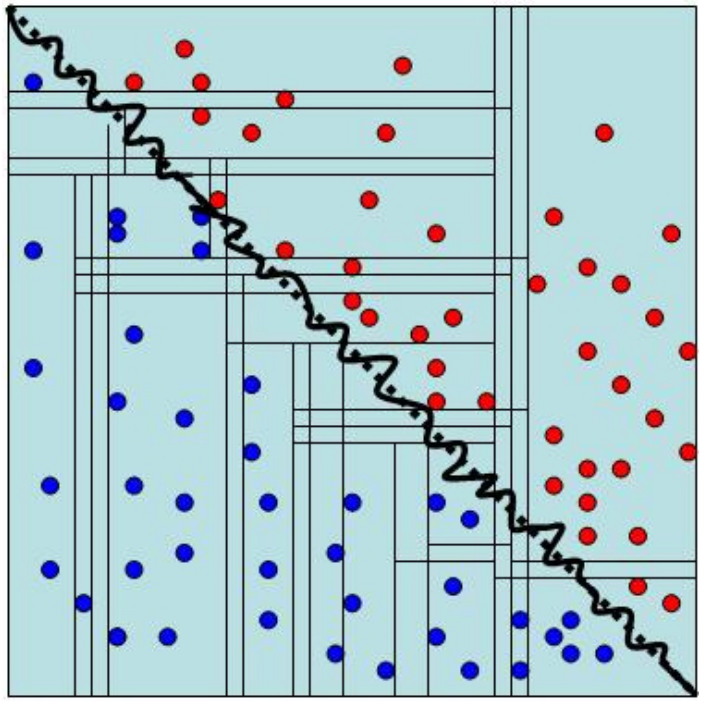
RF was proposed by Leo Breiman in 1999.

- Excellent empirical performance – comparable or better than the best classifiers such as Boosting and SVM
- Blazingly fast and often requires *little* parameter tuning
- Huge impacts in statistics, machine learning and many applied areas
- RF is *mysterious* in terms of theory and the behavior of RF is still not fully understood thus far.

# Ideas of RF

- Ensemble of tree-based classifiers (CART)
- The tree-based classifier is CART
  - Except that RF grows the tree to the *maximum*
    - Leaf node size 1 for classification
    - 5 for regression
- Ensemble implemented by bagging
  - Each tree built by sampling with replacement from data
  - Ensemble to reduce resulting variance
- Philosophy of RF
  - Ensemble of good CARTs (instead of the best).

# An illustration of RF



# An algorithmic description of RF

Two aspects of RF

- The growth of trees (focus of discussion)
- The prediction (easy, simply let a test example $X$ fall off each individual tree and then majority vote).

Some notations

- The training sample $\mathcal{S} = (X_i, Y_i)_{i=1}^n$
- Feature space $\mathcal{F} = \{1, ..., p\}$ or $\mathcal{F}_L = \{\sum_{i=1}^L \alpha_i f_i : \ \alpha_i \in \mathcal{U}[-1, 1], f_i \in \mathcal{F}\}$ for linear combinations of random variables.
- $mtry =$ # tries on different (sets of) features at each node
- $t =$ current node, $t_L =$ left subnode of $t$, $t_R =$ right subnode of $t$.

# The growth of a tree in RF

1. Sample with replacement from $\mathcal{S}$ to get $\mathcal{S}^{(k)} = (X_i^{(k)}, Y_i^{(k)})_{i=1}^n$.
2. Set $t \leftarrow \mathcal{S}^{(k)}$.
3. Recurse
   - 3.1. For $j = 1, ..., mtry$
     - Sample a feature $f$ from $\mathcal{F}$ and search for best splitting point $s$
     - Compute reduction of node impurity (or drop in oob) at node $t$

     End
   - 3.2. Keep the pair $(f, s)$ that reduce the node impurity the most
   - 3.3. Split node at $(f, s)$ to get $t = t_L \cup t_R$
   - 3.4. Set $t \leftarrow t_L$ or $t \leftarrow t_R$

   Until there is *only one point* left in the node.

# Performance of RF

- RF is comparable to Adaboost on 19 datasets (Breiman 1999)
- Compared to SVM, ANN, LR, NB, kNN, BSTDT, BAGDT w.r.t. metrics including accuracy, AUC, squared error loss
  - RF ranks the $2^{nd}$ (closely after boosted decision tree) on 11 low dimensional data (Caruana et al, 2005)
  - RF has the best overall performance on 11 datasets with dimension $761 \sim 685569$ (Caruana et al, 2008)
- RF performs much better (and much faster) than SVM and kNN on a Tissue Microarray image scoring problem where data has a dimension of 2601 (My own recent experience).

# Performance comparison (Breiman, 2001)

| | | Forest-RC | | |
|---|---|---|---|---|
| Data set | Adaboost | Selection | Two features | One tree |
| Glass | 22.0 | 24.4 | 23.5 | 42.4 |
| Breast cancer | 3.2 | 3.1 | 2.9 | 5.8 |
| Diabetes | 26.6 | 23.0 | 23.1 | 32.1 |
| Sonar | 15.6 | 13.6 | 13.8 | 31.7 |
| Vowel | 4.1 | 3.3 | 3.3 | 30.4 |
| Ionosphere | 6.4 | 5.5 | 5.7 | 14.2 |
| Vehicle | 23.2 | 23.1 | 22.8 | 39.1 |
| German credit | 23.5 | 22.8 | 23.8 | 32.6 |
| Image | 1.6 | 1.6 | 1.8 | 6.0 |
| Ecoli | 14.8 | 12.9 | 12.4 | 25.3 |
| Votes | 4.8 | 4.1 | 4.0 | 8.6 |
| Liver | 30.7 | 27.3 | 27.2 | 40.3 |
| Letters | 3.4 | 3.4 | 4.1 | 23.8 |
| Sat-images | 8.8 | 9.1 | 10.2 | 17.3 |
| Zip-code | 6.2 | 6.2 | 7.2 | 22.7 |
| Waveform | 17.8 | 16.0 | 16.1 | 33.2 |
| Twonorm | 4.9 | 3.8 | 3.9 | 20.9 |
| Threenorm | 18.8 | 16.8 | 16.9 | 34.8 |

# Performance comparison (Caruana et al, 2005)

*Table 3.* Normalized scores of each learning algorithm by problem (averaged over eight metrics)

| MODEL | COVT | ADULT | LTR.P1 | LTR.P2 | MEDIS | SLAC | HS | MG | CALHOUS | COD | BACT | MEAN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BST-DT | **.938** | .857 | **.959** | **.976** | .700 | .869 | **.933** | .855 | **.974** | **.915** | .878* | **.896*** |
| RF | .876 | .930 | .897 | .941 | **.810** | .907* | .884 | .883 | .937 | .903* | .847 | .892 |
| BAG-DT | .878 | .944* | .883 | .911 | .762 | .898* | .856 | **.898** | .948 | .856 | **.926** | .887* |
| BST-DT | .922* | .865 | .901* | .969 | .692* | .878 | .927 | .845 | .965 | .912* | .861 | .885* |
| RF | .876 | .946* | .883 | .922 | .785 | .912* | .871 | .891* | .941 | .874 | .824 | .884 |
| BAG-DT | .873 | .931 | .877 | .920 | .752 | .885 | .863 | .884 | .944 | .865 | .912* | .882 |
| RF | .865 | .934 | .851 | .935 | .767* | **.920** | .877 | .876 | .933 | .897* | .821 | .880 |
| BAG-DT | .867 | .933 | .840 | .915 | .749 | .897 | .856 | .884 | .940 | .859 | .907* | .877 |
| SVM | .765 | .886 | .936 | .962 | .733 | .866 | .913* | .816 | .897 | .900* | .807 | .862 |
| ANN | .764 | .884 | .913 | .901 | .791* | .881 | .932* | .859 | .923 | .667 | .882 | .854 |
| SVM | .758 | .882 | .899 | .954 | .693* | .878 | .907 | .827 | .897 | .900* | .778 | .852 |
| ANN | .766 | .872 | .898 | .894 | .775 | .871 | .929* | .846 | .919 | .665 | .871 | .846 |
| ANN | .767 | .882 | .821 | .891 | .785* | .895 | .926* | .841 | .915 | .672 | .862 | .842 |
| BST-DT | .874 | .842 | .875 | .913 | .523 | .807 | .860 | .785 | .933 | .835 | .858 | .828 |
| KNN | .819 | .785 | .920 | .937 | .626 | .777 | .803 | .844 | .827 | .774 | .855 | .815 |
| KNN | .807 | .780 | .912 | .936 | .598 | .800 | .801 | .853 | .827 | .748 | .852 | .810 |
| KNN | .814 | .784 | .879 | .935 | .633 | .791 | .794 | .832 | .824 | .777 | .833 | .809 |
| BST-STMP | .644 | **.949** | .767 | .688 | .723 | .806 | .800 | .862 | .923 | .622 | .915* | .791 |
| SVM | .696 | .819 | .731 | .860 | .600 | .859 | .788 | .776 | .833 | .864 | .763 | .781 |
| BST-STMP | .639 | .941 | .700 | .681 | .711 | .807 | .793 | .862 | .912 | .632 | .902* | .780 |
| BST-STMP | .605 | .865 | .540 | .615 | .624 | .779 | .683 | .799 | .817 | .581 | .906* | .710 |
| DT | .671 | .869 | .729 | .760 | .424 | .777 | .622 | .815 | .832 | .415 | .884 | .709 |
| DT | .652 | .872 | .723 | .763 | .449 | .769 | .609 | .829 | .831 | .389 | .899* | .708 |
| LR | .625 | .886 | .195 | .448 | .777* | .852 | .675 | .849 | .838 | .647 | .905* | .700 |

# Performance comparison (Caruana et al, 2008)

*Table 3.* Bootstrap analysis of rankings by average performance across problems

| AVG | 1ST | 2ND | 3RD | 4TH | 5TH | 6TH | 7TH | 8TH | 9TH | 10TH |
|---|---|---|---|---|---|---|---|---|---|---|
| RF | 0.728 | 0.207 | 0.053 | 0.011 | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| ANN | 0.054 | 0.184 | 0.293 | 0.248 | 0.117 | 0.070 | 0.023 | 0.010 | 0.000 | 0.000 |
| BSTDT | 0.059 | 0.228 | 0.195 | 0.208 | 0.179 | 0.079 | 0.040 | 0.011 | 0.000 | 0.000 |
| SVM | 0.036 | 0.175 | 0.196 | 0.194 | 0.162 | 0.099 | 0.091 | 0.043 | 0.005 | 0.000 |
| LR | 0.092 | 0.133 | 0.079 | 0.074 | 0.110 | 0.176 | 0.245 | 0.091 | 0.000 | 0.000 |
| BAGDT | 0.002 | 0.012 | 0.105 | 0.134 | 0.242 | 0.276 | 0.133 | 0.079 | 0.017 | 0.000 |
| BSTST | 0.009 | 0.014 | 0.028 | 0.071 | 0.088 | 0.115 | 0.296 | 0.358 | 0.021 | 0.000 |
| KNN | 0.021 | 0.047 | 0.052 | 0.059 | 0.087 | 0.160 | 0.122 | 0.177 | 0.263 | 0.013 |
| PRC | 0.000 | 0.000 | 0.000 | 0.001 | 0.013 | 0.024 | 0.050 | 0.231 | 0.682 | 0.000 |
| NB | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.012 | 0.987 |

# Related work and variants

- Some pre-RF variants
  - Random subspace method (T. Ho, 1995)
    - Grow a tree on data randomly projected over a subset of features
  - Randomized shallow trees (Amit and Geman, 1997)
    - Node split by random selection from many geometric features
  - Randomized trees (T. Dietterich, 1998)
    - Node split by a random choice over the best 20
- Perfect RF (Cutler et al, 1999)
  - Random feature selection and "random" split
- Greedy and Purely RF (Biau, Devroye and Lugosi, 2008).

# Random Forests in R

- The "randomForest" package
  - Fortran original by Leo Breiman and Adele Cutler
  - R port by Andy Liaw and Matthew Wiene
  - Classification and regression based on a forest of trees.

# The 'randomForest' function

```
randomForest(xtrain, y, xtest=xtest, ytest=ytest,
             ntree=NTREE, mtry=MTRY,
              replace=TRUE,classwt=NULL,cutoff,strata,
              sampsize=nrow(xtrain),
              nodesize=if(!is.null(y) && !is.factor(y)) 5 else 1,
              importance=FALSE, localImp=FALSE,nPerm=1,
              proximity=FALSE,oob.prox=FALSE,
              norm.votes=TRUE,do.trace=FALSE,
              keep.forest=!is.null(y) && is.null(xtest),
              corr.bias=FALSE, keep.onbag=FALSE);
```

- $ntree$ = # trees
- $mtry$ = # tries in selecting variables
- $node\ size$ = 5 if regression and 1 if classification.

# The spam filter example

- The goal is to design an automatic spam detector
- Information collected from 4601 email messages
  - For which one knows if it is *email* or *spam*
- Formulate as a supervised classification problem
  - Or (logistic) regression problem with discrete response
- Data available at *ftp.ics.uci.edu*
  - Donated by *George* Forman of HP Lab, Palo Alto, CA.

# The spam filter example

- Similar as typical applications, most importantly
  - What features to use?
- Follow the tradition of document/text processing
  - Use relative frequencies of 57 commonly used words
  - One instance of the popular 'Bag of words'
- Why may this help?

| | george | you | your | hp | free | hpl | ! | our | re | edu | remove |
|---|---|---|---|---|---|---|---|---|---|---|---|
| spam | 0.00 | 2.26 | 1.38 | 0.02 | 0.52 | 0.01 | 0.51 | 0.51 | 0.13 | 0.01 | 0.28 |
| email | 1.27 | 1.27 | 0.44 | 0.90 | 0.07 | 0.43 | 0.11 | 0.18 | 0.42 | 0.29 | 0.01 |

# Features breakdown

- 48 quantitative predictors
  - e.g., *business, address, internet, free, and george* (customizable for users)
- % characters that match one of 6 special characters
- Avg length of uninterrupted sequences of capital letters
- Length of longest uninterrupted sequence of capital letters
- Sum of length of uninterrupted sequences of capital letters.

# R code on the Spam data

```
##The spam data
##install.packages("randomForest");
library(randomForest);
x<-read.table("spam.Data",sep=",");
n<-nrow(x); p<-ncol(x);
x[,p]<-as.factor(x[,p]);

##Split the data according to HTF for comparison
T<-3065;
idx2<-sample(1:n,T, replace=FALSE);
xtrain<-x[idx2,1:(p-1)]; y<-x[idx2,p];
xtest<-x[-idx2,1:(p-1)]; ytest<-x[-idx2,p];
```

# R code on the Spam data (continued)

```
set.seed(111);
MTRY<-2*floor(sqrt(p)); NTREE<-100;
nc<-2; cutoff<-rep(1/nc,nc);
myrf<-randomForest(xtrain,y,xtest=xtest,ytest=ytest,
          ntree=NTREE, mtry=MTRY,
          replace=TRUE,classwt=NULL,cutoff,strata,
          sampsize=nrow(xtrain),
          nodesize=if(!is.null(y) && !is.factor(y)) 5 else 1,
          importance=FALSE, localImp=FALSE,nPerm=1,
          proximity=FALSE,oob.prox=FALSE,
          norm.votes=TRUE,do.trace=FALSE,
          keep.forest=!is.null(y) && is.null(xtest),corr.bias=FA
          keep.onbag=FALSE);
```

# Test on the Spam data

```
> print(myrf);
                       Number of trees: 100
No. of variables tried at each split: 8

        OOB estimate of  error rate: 5.32%
Confusion matrix:
     0    1 class.error
0 1809   59  0.03158458
1  104 1093  0.08688388
                Test set error rate: 4.3%
Confusion matrix:
    0   1 class.error
0 898  22  0.02391304
1  44 572  0.07142857
```

- Accuracy = (898+572)/(898+572+22+44) = 0.9570.

# German Credit data (numeric)

- Numeric version of data is used (a few indicator variables added)
- Response variable is whether the credit is *good or bad*
- Possible factors
  - Status of checking acct, duration(month), credit history, purpose
  - Credit amount, Savings acct/bonds, Present employment since
  - Installment rate in % disposable income, Personal status and sex
  - Other debtors/guarantors, Present residence since, Property, Age
  - Other installment plans, Housing, # existing credits at this bank
  - Job, # people being liable to provide maintenance for
  - Telephone, foreign worker.

# R code on the German credit data

```
##The German Credit data
##install.packages("randomForest");
library(randomForest);

x <- read.table("germanCreditNum.Data",header=FALSE,sep="");
n <- nrow(x); p<-ncol(x);
x[,p]<-as.factor(x[,p]);

set.seed(111);
idx2<-sample(1:n,floor(n/2), replace=FALSE);
xtrain<-x[idx2,1:(p-1)]; y<-x[idx2,p];
xtest<-x[-idx2,1:(p-1)]; ytest<-x[-idx2,p];
```

# R code on the German credit data (continued)

```
set.seed(111);
MTRY<-floor(2*sqrt(p-1)); NTREE<-50;
nc<-2; cutoff<-rep(1/nc,nc);
myrf<-randomForest(xtrain,y,xtest=xtest,ytest=ytest,
          ntree=NTREE, mtry=MTRY,
          replace=TRUE,classwt=NULL,cutoff,strata,
          sampsize=nrow(xtrain),
          nodesize=if(!is.null(y) && !is.factor(y)) 5 else 1,
          importance=FALSE, localImp=FALSE,nPerm=1,
          proximity=FALSE,oob.prox=FALSE,
          norm.votes=TRUE,do.trace=FALSE,
          keep.forest=!is.null(y) && is.null(xtest),corr.bias=FA
          keep.onbag=FALSE);
```

# Test on the German credit data

```
> print(myrf);
                     Number of trees: 50
No. of variables tried at each split: 9

        OOB estimate of  error rate: 27.4%
Confusion matrix:
    1  2 class.error
1 313 46   0.1281337
2  91 50   0.6453901
                Test set error rate: 23.4%
Confusion matrix:
    1  2 class.error
1 309 32  0.09384164
2  85 74  0.53459119
```

- Accuracy = (309+74)/(309+74+32+85) = 0.7660.