Lecture 6 Classification and regression trees : CART

Xiang Zhou

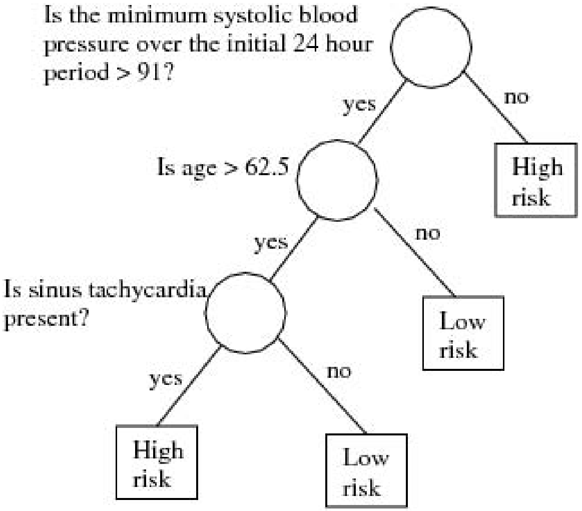
Department of Mathematics
City University of Hong Kong
Based on the slides of Junhui Wang

2019

Tree structure

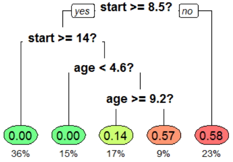
- Tree method is the typical algorithm learning, more close to the viewpoint of computer science. No probabilistic model assumption is needed for the dataset.
- Will focus on classification tree, and extension to regression tree is direct.
- Work as the weak classifier as the candidates for ensemble learning (such as boosting).
- Reference: *Classification and Regression Trees* by L. Breiman, J. Friedman, R. Olshen, and C. Stone, Chapman & Hall, 1984
- A medical example:
  - Predict high risk patients who will not survive at least 30 days on the basis of the initial 24-hour data
  - 19 variables are measured during the first 24 hours. These include blood pressure, age, etc

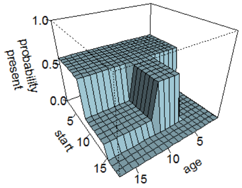
An example of tree structure

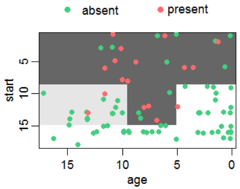


Is the minimum systolic blood pressure over the initial 24 hour period > 91?
yes
no
High risk
Is age > 62.5
yes
no
Low risk
Is sinus tachycardia present?
yes
no
High risk
Low risk

An example of decision tree







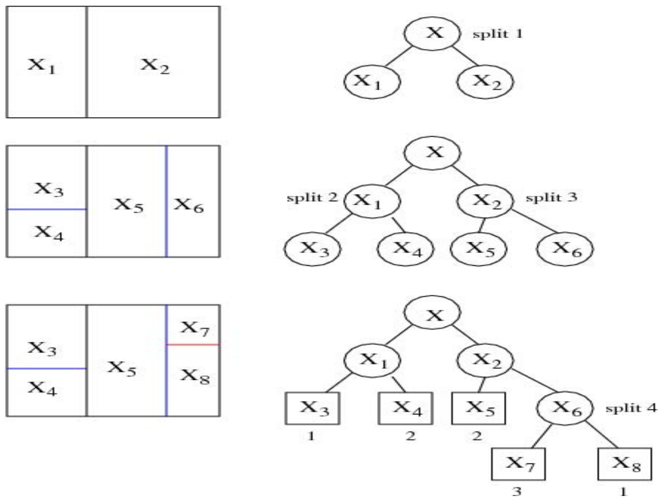
source: wiki

Some notations:

- Denote the feature space by $\mathcal{X}$. The covariate $X = (X_1, X_2, \ldots, X_p)^T \in \mathcal{X} \subset \mathcal{R}^p$, some of which may be categorical.
- $\mathcal{Y} = \{1, \ldots, K\}$
- Classification trees are constructed by repeated splits of subsets of $\mathcal{X}$ into two descendant subsets $\mathcal{X} = \mathcal{X}_+ \cup \mathcal{X}_-$, beginning with $\mathcal{X}$ itself — recursive algorithm!
- Definitions: **node, terminal node (leaf node), parent node, child node, sibling node**
- Two child nodes forms a partition of the region occupied by their parent node
- Every leaf node is assigned with a class in $\mathcal{Y}$.

- A node [1] is denoted by t
- Its left child node is denoted by t_L and right by t_R
- The collection of all the nodes is denoted by T
- The collection of all the leaf nodes by $\widetilde{T} \subset T$
- A split is denoted by s, and the set of splits is denoted by $\mathcal{S}$

[1] a subset of $\mathcal{X}$, usually in a rectangular form



X_i in the figure refers to a subset of the input space $\mathcal{X}$. The collection of all leaf nodes, $\widetilde{T}$, is a partition of the input space $\mathcal{X}$.

Key elements of a tree

- The construction of a tree involves the following three elements:
  1. when to split? The decisions when to declare a node terminal or to continue splitting it
  2. how to split? The selection of the splits
  3. Classifier on leaf ? The assignment of each terminal node to a class
- In particular,
  1. (how to split ?) A set of binary questions used as splits ("`if` ... else ... ")
  2. (which specific split is preferred ?) A goodness of split criterion $\phi(s, t)$ that can be evaluated for any split s of any node t
  3. (when to stop split?) A stop-splitting rule
  4. (classifier?) A rule for assigning every terminal node to a class

Standard set of questions

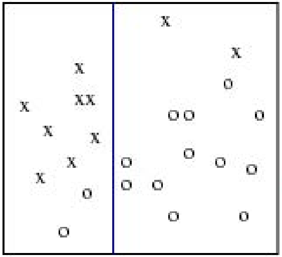
- Each split depends on the value of only a unique variable (one component/feature of the input vector X)
- For each variable X_j, $\mathcal{S}$ includes all questions of the form

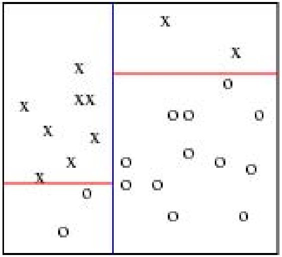
$$\{\text{Is } X_j \in A?\}$$

  - If X_j is continuous or ordinal, $A = (-\infty, c]$
  - If X_j is categorical, A is a subset of categories
- The splits for all p variables constitute the standard set of questions

Goodness of splits

- The goodness of split is measured by an **impurity** function defined for each node
- Intuitively, we want each leaf node to be “pure”, that is, one class dominates and then assignment to this dominant class would have a minimal misclassification rate:





The node impurity function

Define the empirical distribution inside the node t:

$$p_{tk} = |t|^{-1} \sum_{x_i \in t} I(y_i = k) \approx \Pr(Y = k | X \in t), \quad k = 1, \dots, K$$

"Purity" means p_{tk} is independent of k. The possible **impurity functions** [2] $i(t)$ of the node t are

- Misclassification error: $1 - \max_{1 \le k \le K} p_{tk}$
- Gini index: $\sum_k p_{tk}(1 - p_{tk}) = 1 - \sum_k p_{tk}^2$
- (Cross) entropy: $-\sum_k p_{tk} \log p_{tk}$

[2] a **larger** value means more impure, less pure.

For the two classes, write the binary distribution vector $p_t = (p, 1-p)$, then

- Misclassification error: $1 - \max(p, 1-p)$
- Gini index: $2p(1-p)$
- (Cross) entropy: $-p \log p - (1-p) \log(1-p)$

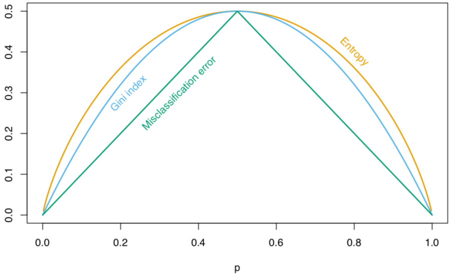


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* (0.5, 0.5).

Exercise

Assume $p = (p_1, \ldots, p_K)$ *is the distribution of a random variable :* $p_k = \Pr(Y = k)$. *Solving the following optimization problem subject to the constraint* $\sum_k p_k = 1$ *and* $p_k \in [0, 1]$ *for all* k*:*

1. *(entropy)* $\max_p($ *or* $\min_p)$ $-\sum_k p_k \log p_k$
2. *(Gini)* $\max_p($ *or* $\min_p)$ $\sum_k p_k(1 - p_k)$

Exercise

To see the above form of the misclassification error, consider the Bayes classifier: $x \in t \to k^*(x) = \operatorname{argmax}_k p_k$*. Show that the misclassification rate of this classifier* $|t|^{-1} \sum_{x_i \in t} I(y_i \neq k^*(x_i))$ *equals* $1 - \max_k p_k$*. (we left out* t *in* p_{tk}*)*

Exercise

To see the Gini index, consider the Gibbs classifier: assign $x \to G(x) = k$ *with probability* p_k*. Show that the training error rate of this Gibbs classifier*

$$|t|^{-1} \sum_{x_i \in t} I(y_i \neq G(x_i))$$

equals $\sum_{k \neq k'} p_k p_{k'} = \sum_k p_k(1 - p_k)$*.*

Goodness of split

- A split s means that t is partitioned into $t = t_L \cup t_R$.
- Its **goodness** is measured by

$$\boxed{\phi(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)},$$

  where $p_L = |t_L|/|t|$ and $p_R = |t_R|/|t|$ are proportions of the samples in node t that go to the left child t_L and the right child of t_R, respectively.
- Search the optimal split s with the largest value of goodness from all possible split $\mathcal{S}$, i.e.,

$$\underset{s \in \mathcal{S}}{\operatorname{argmin}} \quad \{p_L i(t_L) + p_R i(t_R)\}$$

Exercise

Consider a node has two classes with 400 observations in each class, denoted by $(400+, 400-)$. *Compute the goodness of split corresponding to the above three impurity functions for the following two kinds of split*

1. *one child node contains* $(300+, 100-)$ *and the other contains* $(100+, 300-)$.
2. *one child node contains* $(200+, 400-)$ *and the other contains* $(200+, 0-)$.

explain your result.

Exercise

Show that the goodness $\phi(s, t)$ is always non-negative for the misclassification error impurity function. i.e., the mis-classficiation error always increases after splitting. What about the other two choices of impurity functions.

Stopping criteria

- A simple criteria: stop splitting a node t when

$$\max_{s \in \mathcal{S}} \frac{|t|}{n} \phi(s, t) < \beta,$$

  where β is a pre-specified threshold
- The idea is to stop splitting when the node is sufficiently pure or sufficiently small
- The above stopping criteria may not work well; other more sophisticated stopping rules have been proposed

Class assignment rule

- A class assignment rule assigns a class $k = \{1, \ldots, K\}$ to every terminal node $t \in \widetilde{T}$
- The class assigned to node $t \in \widetilde{T}$ is denoted by $\kappa(t)$
- For 0-1 loss, the class assignment rule is the Bayes rule

$$\kappa(t) = \underset{k}{\operatorname{argmax}}\ p_{tk}$$

the dominant class inside the leaf node.

Extension to regression

- $\mathcal{S}$ remains the same
- Impurity function is replaced by the variance : $i(t) = \sum_{i \in t} (y_i - \text{avg}(y|t))^2$
- Goodness of fit means the reduction of the variance: $\phi(s,t) = i(t) - i(t_L) - i(t_R)$
- Stopping rule remains the same
- Response assignment rule: $\kappa(t) = \text{avg}(y|t)$

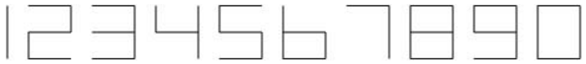
Personal Remark

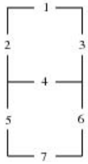
The tree splitting method is quite similar to one of the most important adaptive computational methods in traditional the numerical methods for PDE: adaptive mesh refinement (AMR)
Indeed, the implement of AMR uses the tree data structure for fast processing.

The impurity function in tree classification is like the monitor function (the simplest form is the magnitude of the solution's gradient) in AMR.
There are lots of similarities of adaptivity ideas in machine learning and computational maths, which is now under the fast advancement in research works.

Example: Digit recognition example

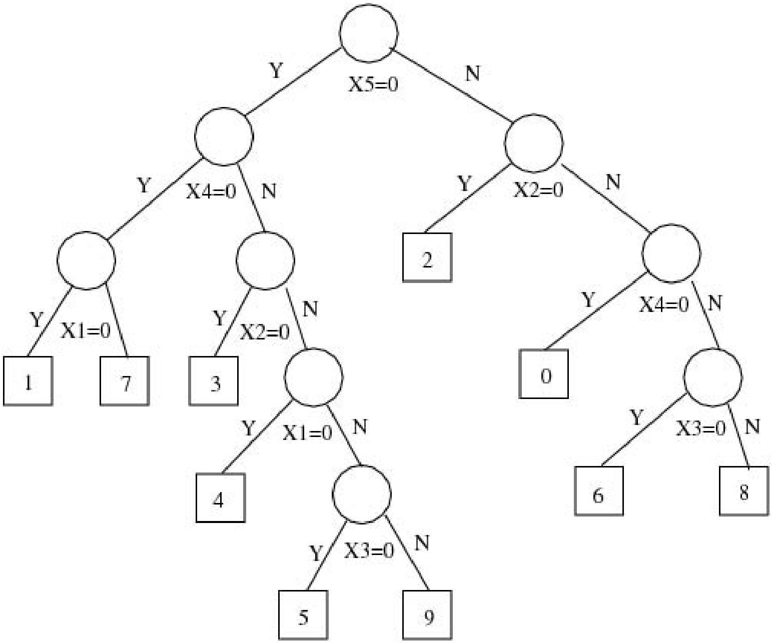
- The 10 digits are shown by different on-off combinations of seven horizontal and vertical lights
- Each digit can be represented by a 7-dimensional vector of zeros and ones. The i-th sample is $x_i = (x_{i1}, \dots, x_{i7})$. If $x_{ij} = 1$, the j-th light is on, and vice versa





Digit	$x_{\cdot 1}$	$x_{\cdot 2}$	$x_{\cdot 3}$	$x_{\cdot 4}$	$x_{\cdot 5}$	$x_{\cdot 6}$	$x_{\cdot 7}$
1	0	0	1	0	0	1	0
2	1	0	1	1	1	0	1
3	1	0	1	1	0	1	1
4	0	1	1	1	0	1	0
5	1	1	0	1	0	1	1
6	1	1	0	1	1	1	1
7	1	0	1	0	0	1	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1

- The data for the example are generated by a malfunctioning calculator
- Each of the seven lights has probability 0.1 of being in the wrong state independently
- The training set contains 200 samples generated according to the specified distribution


X5=0
Y
N
X4=0
Y
N
X2=0
Y
N
X1=0
Y
1
7
X2=0
Y
N
3
2
X4=0
Y
N
0
X1=0
Y
N
4
X3=0
Y
N
6
8
X3=0
Y
N
5
9

Right-sized trees

- Denote the misclassification error of a tree T by $R(T)$, then

$$R(T) = \sum_{t \in \tilde{T}} p(t) i(t),$$

  where $p(t) = |t|/n$ is the proportion of observations in node t, and $i(t)$ is defined by the misclassification error

- $R(T)$ is biased downward, because

$$p(t)i(t) \geq p(t_L)i(t_L) + p(t_R)i(t_R),$$

  thus the larger the tree is, the smaller the misclassification error (overfitting!)

Digit recognition example: cont'd

No. Terminal Nodes	$R(T)$	$R^{ts}(T)$
71	.00	.42
63	.00	.40
58	.03	.39
40	.10	.32
34	.12	.32
19	.29	.31
10	.29	.30
9	.32	.34
7	.41	.47
6	.46	.54
5	.53	.61
2	.75	.82
1	.86	.91

Second column: training error; Third column: test error.

Pruning

- Grow a very large tree T_{max}
  1. Until all terminal nodes are pure (contain only one class) or contain only identical measurement vectors
  2. When the number of data in each terminal node is no greater than a certain threshold
  3. As long as the tree is sufficiently large

- A "selective" pruning procedure is needed
  1. The pruning is optimal in a certain sense
  2. The search for different ways of pruning should be of manageable computational load

Minimal cost pruning

For any subtree T of T_{max}, define its cost function as

$$R_\lambda(T) = R(T) + \lambda|\widetilde{T}|,$$

where λ is a tuning parameter. For each λ, solve

$$T(\lambda) = \operatorname*{argmin}_T R_\lambda(T)$$

Remarks:

- To optimally determine λ, a model selection criterion such as cross validation can be employed
- To search for the pruning branch, many technique has been proposed, such as the weakest-link cutting

Advantages of the tree-structured approach

- Handles both categorical and ordered variables in a simple and natural way
- Automatic stepwise variable selection and complexity reduction
- It provides an estimate of conditional class probability
- It is invariant under all monotone transformations of individual ordered variables
- Robust to outliers and misclassified points in the training set
- **Easy to interpret**

Issues

- Learning an optimal decision tree is NP-complete. Categorical covariate with q classes produces $2^q - 1$ possible splits
  - Simplification is possible for binary classification with Gini index or cross entropy and regression with squared error loss
  - But it is unclear for multiclass classification (digit recognition example)
- Cost-sensitive (non-standard) classification with unequal misclassification costs
- Splits based on combined covariates
- Lack of smoothness (blockwise constant)
- High variance, constructed tree is very sensitive to sample
  Trees can be very non-robust.A small change in the training data can result in a large change in the tree

From Tree to Ensemble of trees

Some techniques, often called ensemble methods, construct more than one decision tree:

1. **Boosted trees**: Incrementally building an ensemble by training each new instance to emphasize the training instances previously mis-modeled. A typical example is AdaBoost. These can be used for regression-type and classification-type problems.
   We shall have one chapter focusing on boosting algorithms.
2. **Bootstrap aggregated (or bagged) decision trees**: an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.
   - A **random forest** classifier is a specific type of bootstrap aggregating

Bagging (bootstrap aggregation)

Bagging is a technique to reduce the variance of an estimated prediction function.

1. Draw B bootstrap samples [3] $(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})$; $b = 1, \ldots, B$
2. For each bootstrap sample, fit the tree model $\hat{f}^{*b}(x)$
3. The bagging estimate is

$$\hat{f}_{bag}(x) = \begin{cases} B^{-1} \sum_{b=1}^{B} \hat{f}^{*b}(x), & \text{for regression/classification;} \\ \text{mode}(\hat{f}^{*1}(x), \ldots, \hat{f}^{*B}(x)), & \text{for classification} \end{cases}$$

[3] these samples are randomly drawn from the original dataset $\mathcal{D} = \{(x_i, y_i)\}$ with replacement

Why it works: some intuitions

- For regression, pretend we draw bootstrap samples from the true distribution rather than the training set $\mathcal{D}$ (so independent),

$$E_{X,Y}(Y - \hat{f}(X))^2 \geq E(Y - \hat{f}_{bag}(X))^2,$$

as one could see that $\hat{f}_{bag}(x) = E_{\mathcal{T}}(\hat{f}(x))$

- For classification, $\hat{f}_{bag}(x) \neq E_{\mathcal{T}}(\hat{f}(x))$ and hence that the above inequality does not hold
  - "wisdom of crowds": the collective knowledge of a diverse and independent body of people exceeds the knowledge of any single individual
  - Two heads are better than one

Random forest

Random forest is similar to bagging, but different as it averages *de-correlated* trees by introducing *additional randomness* in each tree $\hat{f}^{*,b}(x)$ by using random subsets of features

1. Draw bootstrap samples $(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})$
2. For each bootstrap sample, grow a tree by repeating:
    a. randomly select a subset of the p covariates
    b. pick the best split among the chosen subset
    c. split the node
3. The random forest estimate is constructed similarly as in bagging (average for regression, and majority vote for classification)

Why it works: some intuitions

- In bagging, the bootstrap trees are correlated, and correlation limits the benefit of averaging
- Averaging B random variables with variance σ^2 and correlation ρ yields the variance[4]

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 = \rho\sigma^2(1 - \frac{1}{B}) + \frac{\sigma^2}{B}$$

- In random forest, the correlation is reduced by selection of covariates in growing trees
- The price it has to pay is the increment of bias and variance, which, fortunately, is usually small
- Breiman 2001 paper

[4] the proof is an exercise

Random Forest: pro and con

Advantage:

1. flexible, nonlinear decision boundary
2. easy to implement; favourite of CS programmers
3. easy to use and interpret
4. it can be used for both regression and classification tasks
5. not seen to overfit easily

Limits:

1. a large number of trees can make the algorithm to slow and ineffective for real-time predictions.

   Committee make any decision very slowly.
2. High dimension and small/unbalanced samples?

Additional readings and resource

Breiman, L. (2001). Random forests. *Mach. Learn.*, **45**, 5-32.

Biau, G., Devroye, L. and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *J. Mach. Learn. Res.*, **9**, 2039-2057.

Random Forests: some methodological insights, 2008

An Implementation and Explanation of the Random Forest in Python

Wikipedia of Random Forest