Decision trees

- Tree-based methods for regression and classification.
- These methods involve segmenting or partitioning the predictor space into a number of simple regions(rectangles).
- In each region, a simple model is fit (constant or average of the observations in that regions).
- To make prediction of a new observation, you can use the average of the observations that are already in that region.
- Introduce bagging, random forests, and boosting.

Decision trees

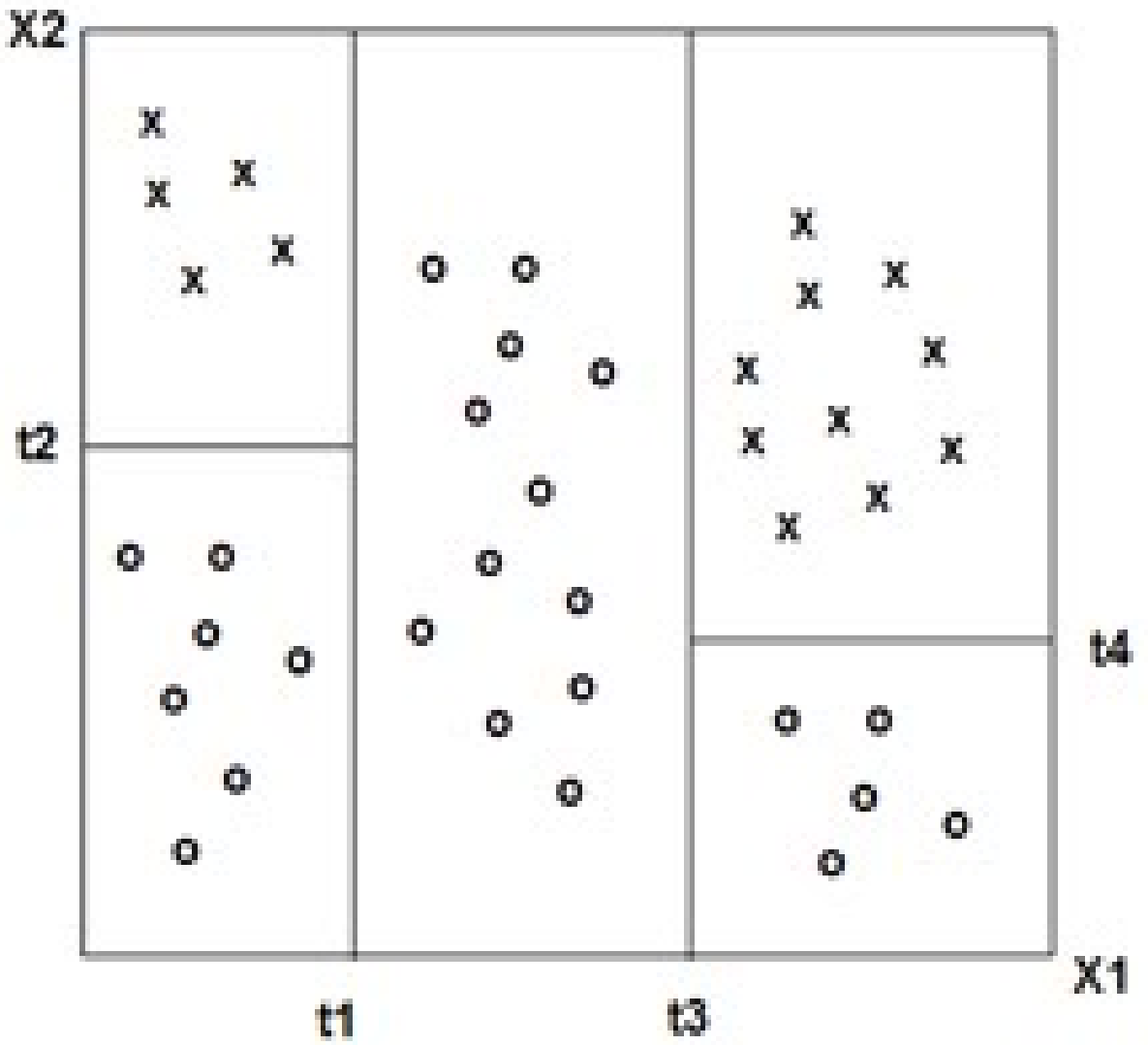

Figure: http://econometricsense.blogspot.com/

Decision trees

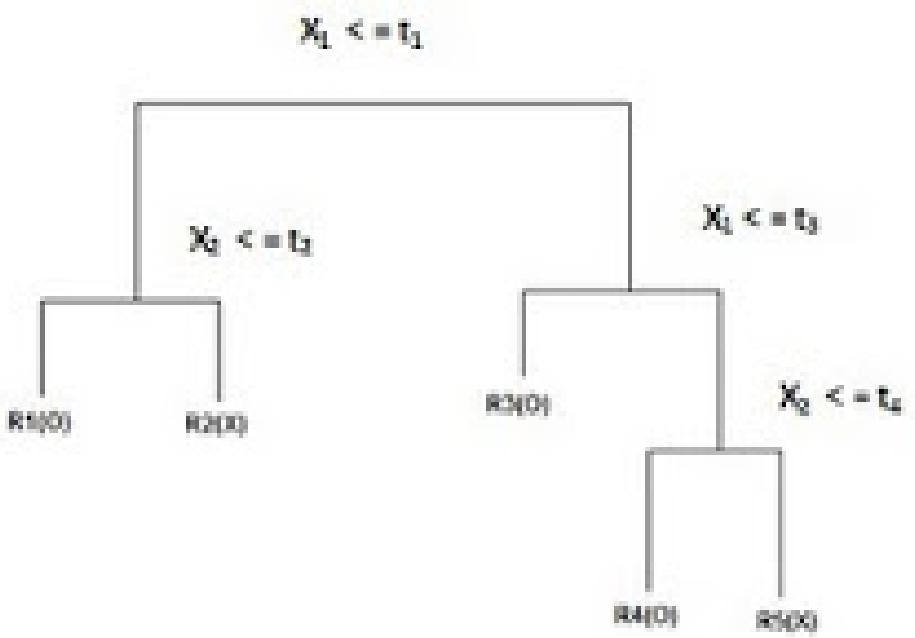

Figure: http://econometricsense.blogspot.com/

Decision trees

- Consider regression problem: continuous response variable Y and inputs X_1 and X_2.
- Recursive binary partition: The predictor space first split into two regions, for example R_1 and R_2. The response in each region is modeled by the averge.
- After that one or both of these regions are split into two or more regions.
- The process is continued until some stopping criteria. (i.e regions can't have less than certain number of observations).
- Which predictor to split and where to split?

Decision trees

- Suppose after the procedure we obtain m regions: $R_1, R_2, \cdots, R_m$. In each region, the response variable Y is modeled by a constant c_m.
- Let $I_m(X)$ be an indicator function:

$$I_m(X) = \begin{cases} 1 & X \in R_m \\ 0 & X \notin R_m \end{cases}$$

- The regression model is:

$$\hat{f}(\hat{X}) = \sum_{i=1}^{m} c_m I_m(X).$$

Decision trees

- How to build a regression tree?
- Data: N observations, p inputs (predictors), a response:$(x_i, y_i), x_i \in \mathcal{R}^{1\times p}, y_i \in \mathcal{R}$.
- Suppose we have m regions: $R_1, R_2, \cdots, R_m$, the responses in each region modeled bya constant c_m

$$f(x) = \sum_{i=1}^{m} c_m I_m(x).$$

- Choose our criterion for minimization: least square loss $\sum_i (y_i - f(x_i))^2$, we can see that:

$$\hat{c}_m = mean(y_i | x_i \in R_m).$$

Decison trees

- For large data sets, finding the best binary partition in terms of least square minimization is infeasible.
- Greedy approach: Considering a variable (predictor) for splitting j, and a split point s. Define:

$$R_1(j,s) = \{X|X_j \leq s\}, R_2(j,s) = \{X|X_j > s\},$$

- To find j and s, we minimize:

$$\min_{j,s}\{\min_{c1} \sum_{i \in R_1(j,s)} (yi - c1)^2 + \min_{c2} \sum_{i \in R_2(j,s)} (y_i - c_2)^2\}$$

- For $\min_{c1} \sum_{i \in R_1(j,s)} (yi - c1)^2$: The minimizer is $\hat{c1} = mean(y_i|x_i \in R_1(j,s)$

Decison trees

- For each variable j, the split point s can be determined by scanning through all inputs of predictor j.
- The splitting process is repeated on each of the two regions.
- A large tree will fit the training data very well, but does not do well on testing data.
- A small tree might neglect important structure.

Decison trees

- Strategy: Building a large tree untill regions have a small number of observations, then *treepruning*.
- Define: terminal node of a tree as the lowest nodes of the tree (final regions). Internal node: nodes of tree at which the splitting process is done.
- Let $|T|$ be the number of terminal node of tree T, N_m be number of observations in R_m.
- $\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$, $Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$.

Decison trees

- Define cost complexity critertion:

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

- For each value of α, find a subtree T_α of T to minimize $C_\alpha(T)$,
- Tuning parameter $\alpha \geq 0$ tradeoff between treesize and the fit to the data. Large values of α result in smaller trees and vice versa.
- For each α there is a unique smallest tree T_α that minimize $C_\alpha(T)$.

Decison trees

- To T_alpha, successively collapse each of the internal node, we obtain a tree. Stop when we have a single node tree.
- When an internal node is collapse, it leads to increase of $\sum_{m=1}^{|T|} N_m Q_m(T)$ (fitness of model to the data). Choose the internal node that has least increase.
- Along this process, we obtain several subtrees, T_α is among these trees.
- Best value of α is chosen via cross-validation.

Decison trees

- What about classification? Data (x_i, y_i), $y_i = 1, 2, \cdots, K$.
- In a node m, corresponding region R_m with N_m observations:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k).$$

- A new observation in R_m is classified to class k with highest probability in that region:k(m).
- $Q_m(T) = \frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk}$

Decison trees

- Decision trees are easy to interpret.
- Decision trees for regression and classification are highly instable. A small change in the data can result in a very different tree.
- Trees do not have the same level of predictive accuracy as other methods.
- To construct more powerful predictive models using decision trees, use bagging, random forests, boosting.

Decison trees

- Bagging: Boosting aggregation, is a general procedure for reducing variance of a learning method. (can be use for regression or classification methods).
- Generate B separate training sets and average them to obtain a low-variance learning model:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

- New training sets can be generated from the original training data using uniform sampling with replacement.

Decision trees

- Build B regression trees (or classification). Each has high variance but low bias.
- The final model is the average of B trees.
- Typically when a new training set is drawn, some observations will be left out as out-of-the-bag observations.
- Overall MSE on OOB observations can be used as an estimate for test error. (so no need to do cross-validation).

Decision trees

- Random forests: provides an improvement of over bagging.
- Build B regression (classification) trees over B bagging training data sets like bagging.
- At each step of building a tree, you can consider only $m < n$ number of nodes (instead of all of them) p, for splitting.
- These m predictors are chosen at random.
- The final model is aggregated over B tree models similar to bagging.

Decision trees

- Algorithm for random forests: Repeat B times. First draw a bootstrap sample from training data.
- Grow a random forest tree on the bootstrap data:
  1. When a node is considered for spliting, only choose among m randomly chosen predictors out of the original p predictors. (in practice, $m = \sqrt{p}$)
  2. Chooese the best split and split the node.
- Output B trees $T_1, T_2, \cdots, T_B$.
- To make a prediction for a new observation, take the average of the predictions of these B trees (similar to bagging).

Decision trees

- Boosting tree: Build trees on the same training data set, but sequentially. (no bootstrap data)
- Build the first tree on training data but only up to a certain number of node splits.
- The next tree is build on the residuals of the first tree.
- The model is updated based on the newly built tree.

Decision trees

- Algorithm: Start with $\hat{f}(x) = 0$, $r_i = y_i$, r_i are the residuals.
- For $b = 1, 2, \cdots, B$, build a tree $\hat{f}_b$ with d splits (d+1 terminal nodes) on training data (X,r).
- Update: $\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}_b(x)$,$r_i = r_i - \lambda \hat{f}_b(x_i)$.
- Final model: $\hat{f}(x) = \sum_{b=1}^{B} \hat{f}_b(x)$

Decision trees

- Demonstration:
- Fit regression tree on car data set.
- Fit classification tree on heart disease data set.
- Fit bagged trees on heart data set.
- Fit random forest on heart data set.
- Bagged trees on Corporate Credit rating data (Demonstration provided by Matlab).