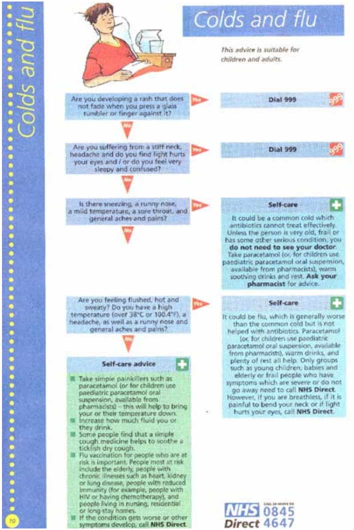
Statistical Machine Learning
Hilary Term 2019

Pier Francesco Palamara
Department of Statistics
University of Oxford

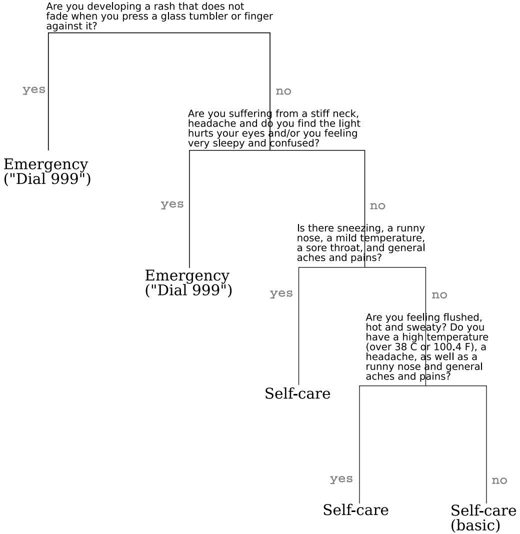
Slide credits and other course material can be found at:
http://www.stats.ox.ac.uk/~palamara/SML19.html

February 22, 2019

Many decisions are tree-structured

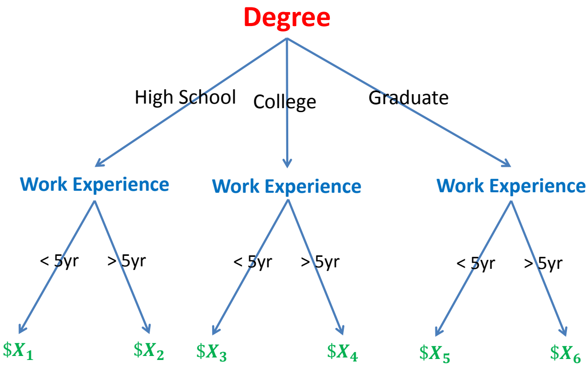


Many decisions are tree-structured



Many decisions are tree-structured

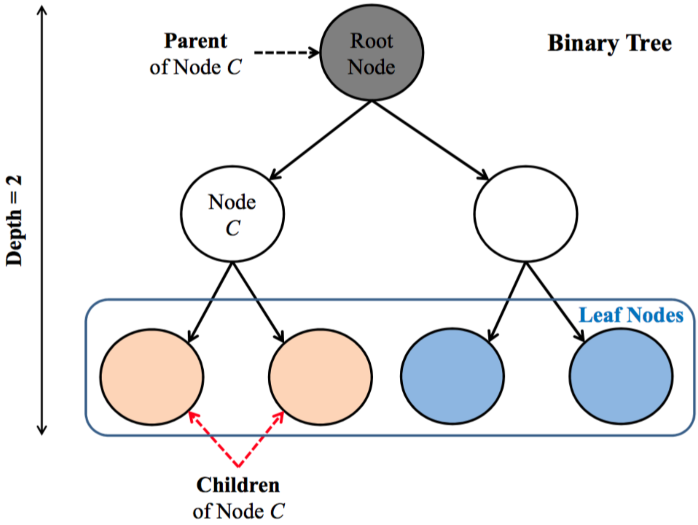
- Employee salary



Terminology

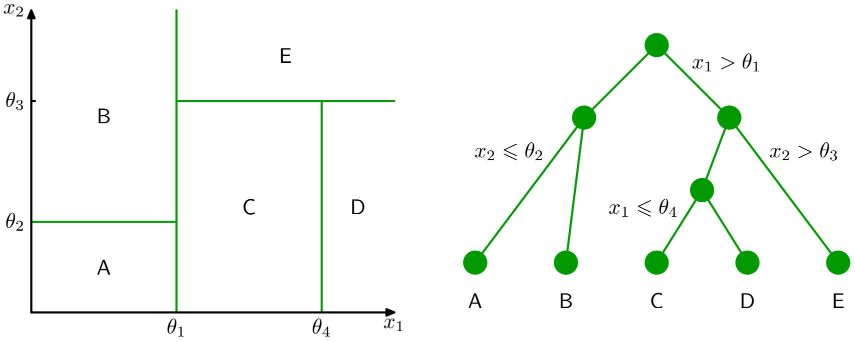
- **Parent** of a node c is the immediate predecessor node.
- **Children** of a node c are the immediate successors of c, equivalently nodes which have c as a parent.
- **Branch** are the edges/arrows connecting the nodes.
- **Root** node is the top node of the tree; the only node without parents.
- **Leaf** nodes are nodes which do not have children.
- **Stumps** are trees with just the root node and two leaf nodes.
- A $K-$**ary** tree is a tree where each node (except for leaf nodes) has K children. Usually working with binary trees $(K = 2)$.
- **Depth** of a tree is the maximal length of a path from the root node to a leaf node.

Terminology

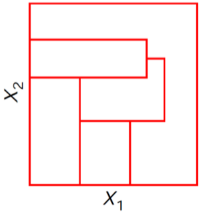


A tree partitions the feature space

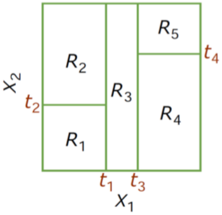
- A Decision Tree is a hierarchically organized structure, with each node splitting the data space into pieces based on value of a feature.
  - Equivalent to a partition of $\mathcal{R}_d$ into K disjoint feature regions $\{\mathcal{R}_j, \ldots, \mathcal{R}_j\}$, where each $\mathcal{R}_j \subset \mathbb{R}^p$
  - On each feature region $\mathcal{R}_j$, the same decision/prediction is made for all $x \in \mathcal{R}_j$.



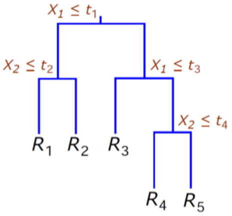
Partitions and regression trees



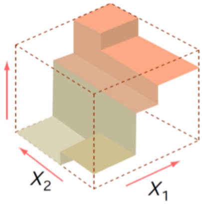
(a) General partition that cannot be obtained from recursive binary splitting.



(b) Partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data.



(c) Tree corresponding to the partition in the top right panel.

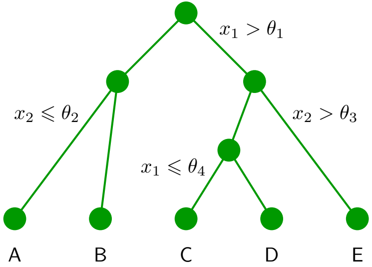


(d) A perspective plot of the prediction surface.

Learning a tree model

Three things to learn:

1. The structure of the tree.
2. The threshold values (θ_i).
3. The values for the leaves ($A, B, \ldots$).



Classification Tree

Classification Tree:

- Given the dataset $D = (x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}, y_i \in Y = \{1, \ldots, m\}$.
- minimize the misclassification error in each leaf
- the estimated probability of each class k in region $\mathcal{R}_j$ is simply:

$$\beta_{jk} = \frac{\sum_i \mathbb{I}(y_i = k) \cdot \mathbb{I}(x_i \in \mathcal{R}_j)}{\sum_i \mathbb{I}(x_i \in \mathcal{R}_j)}$$

- This is the frequency in which label k occurs in the leaf $\mathcal{R}_j$. (These estimates can be regularized.)

Example: A tree model for deciding where to eat

Decide whether to wait for a table at a restaurant, based on the following attributes (Example from Russell and Norvig, AIMA)

- Alternate: is there an alternative restaurant nearby?
- Bar: is there a comfortable bar area to wait in?
- Fri/Sat: is today Friday or Saturday?
- Hungry: are we hungry?
- Patrons: number of people in the restaurant (None, Some, Full)
- Price: price range ($, $$, $$$)
- Raining: is it raining outside?
- Reservation: have we made a reservation?
- Type: kind of restaurant (French, Italian, Thai, Burger)
- Wait Estimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Example: A tree model for deciding where to eat

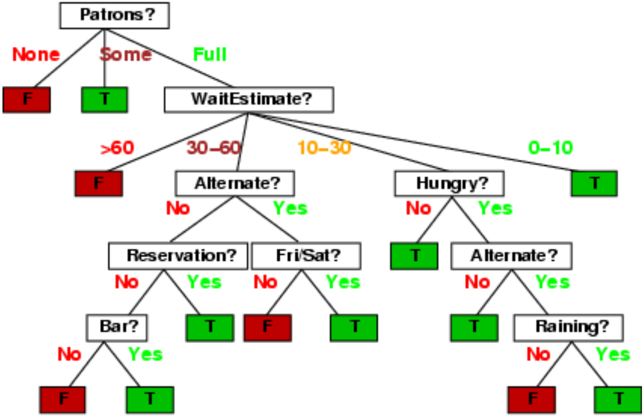
Choosing a restaurant

(Example from Russell & Norvig, AIMA)

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Classification of examples is positive (T) or negative (F)

A possible decision tree



Is this the best decision tree?

Decision tree training/learning

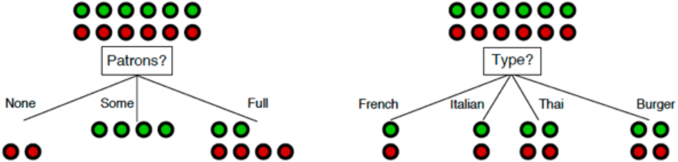
For simplicity assume both features and outcome are binary (take YES/NO values).

Algorithm 1 DecisionTreeTrain ($data, features$)

```
guess ← the most frequent label in data
if all labels in data are the same then
    return LEAF (guess)
else
    f ← the "best" feature ∈ features
    NO ← the subset of data on which f = NO
    YES ← the subset of data on which f = YES
    left ← DecisionTreeTrain (NO, features − {f})
    right ← DecisionTreeTrain (YES, features − {f})
    return NODE(f, left, right)
end if
```

First decision: at the root of the tree

Which attribute to split?



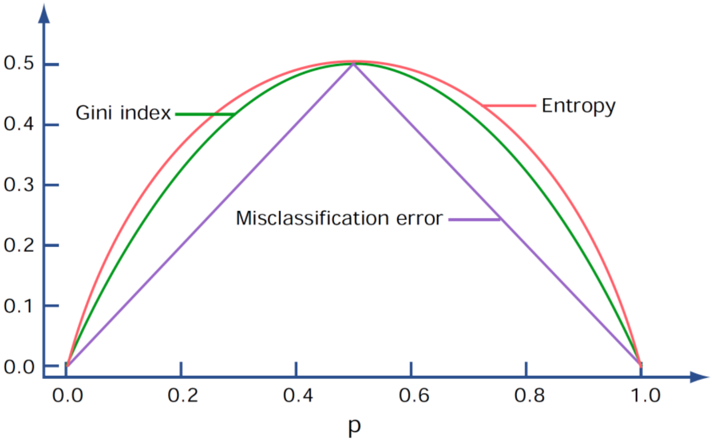
Patrons? is a better choice—gives **information** about the classification

Idea: use information gain to choose
which attribute to split

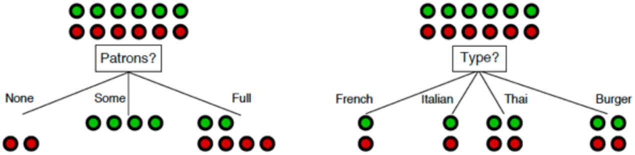
Information gain

- Basic idea: **Gaining information** reduces uncertainty
- Given a random variable X with K different values, $(a_1, \ldots, a_K)$, we can use different measures of "purity" of a node:
  - **Entropy** (measured in bits, max$= 1$):
  $$H[X] = -\sum_{k=1}^{K} P(X = a_k) \times \log_2 P(X = a_k)$$
  - **Misclassification error** (max$= 0.5$): if c is the most common class label
  $$1 - P(X = c)$$
  - **GINI Index** (max$= 0.5$):
  $$\sum_{k=1}^{K} P(X = a_k)(1 - P(X = a_k))$$
  - E.g. compare splits $[(300, 100), (100, 300)]$ and $[(200, 400), (200, 0)]$, taking average of scores for nodes produced (but note different max values). which node will each measure prefer, and would you agree?
- **C4.5** Tree algorithm: Classification uses entropy to measure uncertainty.
- **CART** (class. and regression tree) algorithm: Classification uses Gini.

Different measures of uncertainty



Which attribute to split?



Patrons? is a better choice—gives **information** about the classification

Patron vs. Type?

By choosing Patron, we end up with a partition (3 branches) with smaller entropy, ie, smaller uncertainty (0.45 bit)

By choosing Type, we end up with uncertainty of 1 bit.

Thus, we choose Patron over Type.

Uncertainty if we go with "Patron"

For "None" branch

$$-\left(\frac{0}{0+2}\log\frac{0}{0+2}+\frac{2}{0+2}\log\frac{2}{0+2}\right)=0$$

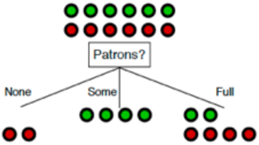
For "Some" branch

$$-\left(\frac{4}{4+0}\log\frac{4}{4+0}+\frac{4}{4+0}\log\frac{4}{4+0}\right)=0$$

For "Full" branch

$$-\left(\frac{2}{2+4}\log\frac{2}{2+4}+\frac{4}{2+4}\log\frac{4}{2+4}\right)\approx 0.9$$

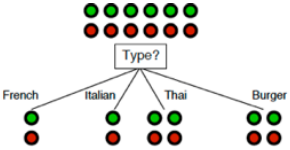
For choosing "Patrons"



weighted average of each branch: this quantity is called conditional entropy

$$\frac{2}{12}*0+\frac{4}{12}*0+\frac{6}{12}*0.9=0.45$$

Conditional entropy for Type



For "French" branch

$$-\left(\frac{1}{1+1}\log\frac{1}{1+1}+\frac{1}{1+1}\log\frac{1}{1+1}\right)=1$$

For "Italian" branch

$$-\left(\frac{1}{1+1}\log\frac{1}{1+1}+\frac{1}{1+1}\log\frac{1}{1+1}\right)=1$$

For "Thai" and "Burger" branches

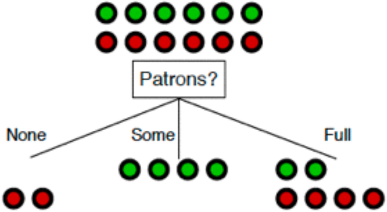
$$-\left(\frac{2}{2+2}\log\frac{2}{2+2}+\frac{2}{2+2}\log\frac{2}{2+2}\right)=1$$

For choosing "Type"

weighted average of each branch:

$$\frac{2}{12}*1+\frac{2}{12}*1+\frac{4}{12}*1+\frac{4}{12}*1=1$$

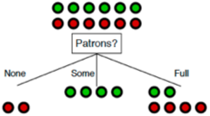
Do we split on “Non” or “Some”?



No, we do not

The decision is deterministic, as seen from the training data

next split?

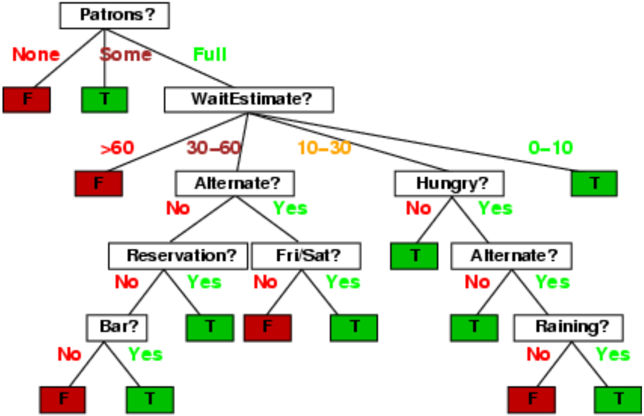


We will look only at the 6 instances with Patrons == Full

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Classification of examples is positive (T) or negative (F)

Greedily, we build



An Algorithm for Classification Trees

Assume binary classification for simplicity ($y_i \in \{0, 1\}$), and numerical features (see Section 9.2.4 in ESL for categorical).

1. Start with $\mathcal{R}_1 = \mathcal{X} = \mathbb{R}^p$.
2. For each feature $j = 1, \dots, p$, for each value $v \in \mathbb{R}$ that we can split on:
    1. Split data set:
    $$I_< = \{i : x_i^{(j)} < v\} \qquad I_> = \{i : x_i^{(j)} \geq v\}$$
    2. Estimate parameters:
    $$\beta_< = \frac{\sum_{i \in I_<} y_i}{|I_<|} \qquad \beta_> = \frac{\sum_{i \in I_>} y_i}{|I_>|}$$
    3. Compute the **quality of split**, e.g., using entropy (note: we take $0 \log 0 = 0$)
    $$\frac{|I_<|}{|I_<| + |I_>|}\mathsf{B}(\beta_<) + \frac{|I_>|}{|I_<| + |I_>|}\mathsf{B}(\beta_>)$$
    where
    $$\mathsf{B}\left(q\right) = -\left[q \log_2(q) + (1 - q) \log_2(1 - q)\right]$$
3. Choose split, i.e., feature j and value v, with maximum quality.
4. Recurse on both children, with datasets $(x_i, y_i)_{i \in I_<}$ and $(x_i, y_i)_{i \in I_>}$.

Comparing the features with conditional entropy

- Given two random variables X and Y, conditional entropy is

$$H[Y|X] = \sum_k P(X = a_k) \times H[Y|X = a_k]$$

- In the algorithm,
  - X: the attribute to be split (e.g. patrons)
  - Y: the labels (e.g. wait or not)
  - Estimated $P(X = a_k)$ is the weight in the quality calculation
- Relation to **information gain**

$$\text{Gain}[Y, X] = H[Y] - H[Y|X]$$

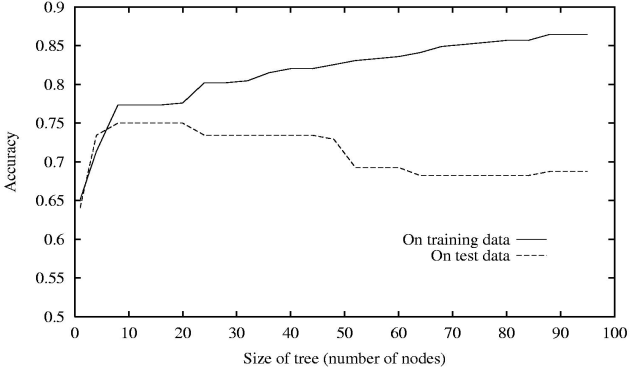
  - When $H[Y]$ is fixed, we need only to compare conditional entropy.
  - Minimizing conditional entropy is equivalent to maximizing information gain.

Patrons vs Type

$$\text{Gain}[Y, \text{Patrons}] = H[Y] - H[Y|\text{Patrons}] = 1 - 0.45 = 0.55$$
$$\text{Gain}[Y, \text{Type}] = H[Y] - H[Y|\text{Type}] \quad = 1 - 1 \quad = 0$$

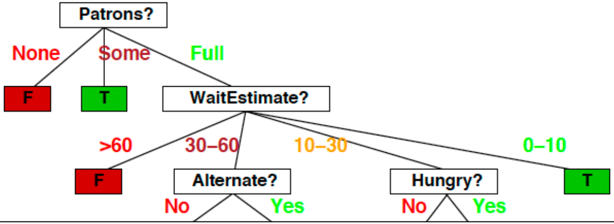
What is the optimal Tree Depth?

- We need to be careful to pick an appropriate tree depth.
- If the tree is too deep, we can overfit.
- If the tree is too shallow, we underfit
- Max depth is a hyper-parameter that should be tuned by the data.
- Alternative strategy is to create a very deep tree, and then to prune it.



Control the size of the tree

We would prune to have a smaller one



If we stop here, not all training sample would be classified correctly.

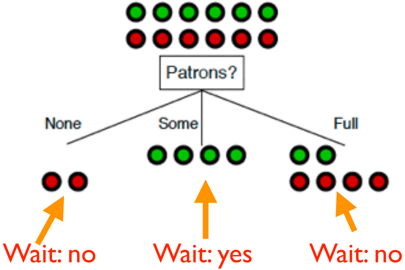
More importantly, how do we classify a new instance?

We label the leaves of this smaller tree with the majority of training samples' labels

Example

Example

We stop after the root (first node)



Computational Considerations

Numerical Features

- We could split on any feature, with any threshold
- However, for a given feature, the only split points we need to consider are the the n values in the training data for this feature.
- If we sort each feature by these n values, we can quickly compute our impurity metric of interest (cross entropy or others), skipping values where labels are unchanged.
  - This takes $O(d\, n \log n)$ time (sorting n elements takes $O(n \log n)$ steps).

Categorical Features

- Assuming q distinct categories, there are $2^{q-1} - 1$ possible binary partitions we can consider.
- However, things simplify in the case of binary classification (or regression, see Section 9.2.4 in ESL for details).

Summary of learning classification trees

Advantages

- Easily interpretable by human (as long as the tree is not too big)
- Computationally efficient
- Handles both numerical and categorical data
- It is parametric thus compact: unlike Nearest Neighborhood Classification, we do not have to carry our training instances around
- Building block for various ensemble methods (more on this later)

Disadvantages

- Heuristic training techniques
- Finding partition of space that minimizes empirical error is NP-hard.
- We resort to greedy approaches with limited theoretical underpinning.
- Unstable: small changes in input data lead to different trees. Mitigated by ensable methods (e.g. random forests, coming up).

Regression Tree

Regression Tree:

- Given the dataset $D = (x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}, y_i \in Y = \{1, \ldots, m\}$.
- minimize the squared loss (may use others!) in each leaf
- the parameterized function is:

$$\hat{f}(x) = \sum_{j=1}^{K} \beta_j \cdot \mathbb{I}(x \in \mathcal{R}_j)$$

- Using squared loss, optimal parameters are:

$$\hat{\beta}_j = \frac{\sum_{i=1}^{n} y_i \cdot \mathbb{I}(x_i \in \mathcal{R}_j)}{\sum_{i=1}^{n} \mathbb{I}(x_i \in \mathcal{R}_j)}$$

i.e. the sample mean.

An Algorithm for Regression Trees

Assume numerical features (see Section 9.2.4 in ESL for categorical).

1. Start with $\mathcal{R}_1 = \mathcal{X} = \mathbb{R}^p$.
2. For each feature $j = 1, \ldots, p$, for each value $v \in \mathbb{R}$ that we can split on:
    1. Split data set:

    $$I_< = \{i : x_i^{(j)} < v\} \qquad I_> = \{i : x_i^{(j)} \geq v\}$$

    2. Estimate parameters:

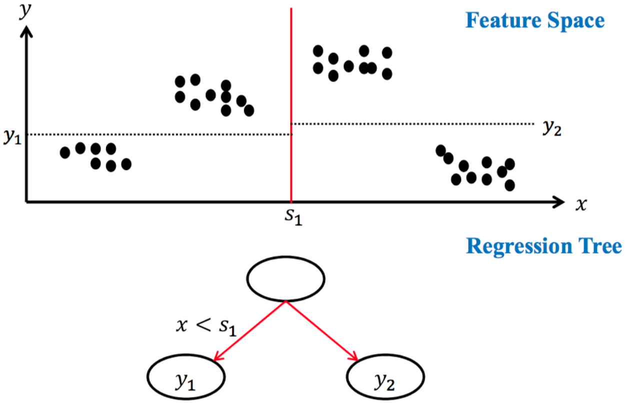
    $$\beta_< = \frac{\sum_{i \in I_<} y_i}{|I_<|} \qquad \beta_> = \frac{\sum_{i \in I_>} y_i}{|I_>|}$$

    3. **Quality of split**: highest quality is achieved for minimum squared loss, which is defined as

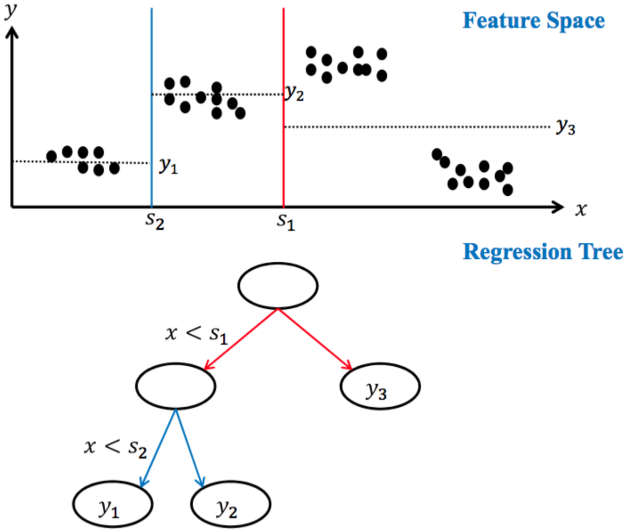
    $$\sum_{i \in I_<} (y_i - \beta_<)^2 + \sum_{i \in I_>} (y_i - \beta_>)^2$$

3. Choose split, i.e., feature j and value v, with maximum quality.
4. Recurse on both children, with datasets $(x_i, y_i)_{i \in I_<}$ and $(x_i, y_i)_{i \in I_>}$.

Example of Regression Trees



Example of Regression Trees



Example of Regression Trees