CSCE 633: Machine Learning

Lecture 16: Tree-Based Methods

Texas A&M University

9-30-19

Last Time

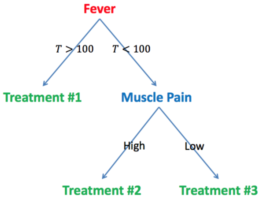
- SVMs

Goals of this lecture

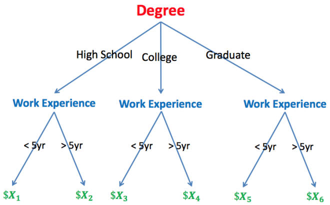
- Decision Trees
- Random Forest

Many decisions are tree-like structures

Medical treatment



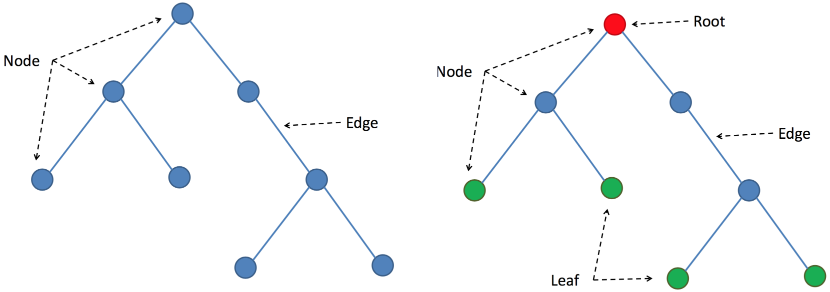
Salary in a company



Decision Trees

What is a decision tree

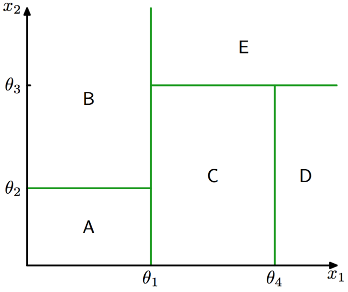
A hierarchical data structure implementing the divide-and-conquer strategy for decision making

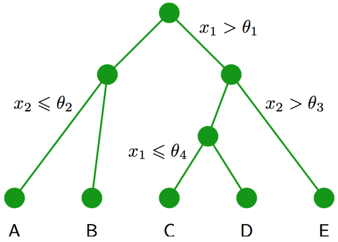


Can be used for both classification & regression

Decision Trees

A decision tree partitions the feature space





Three things to learn

- The tree structure (i.e. attributes and #branches for splitting)
- The threshold values (i.e. θ_i)
- The values of the leaves (i.e. $A, B, \ldots$)

Example

We want to find a regression on baseball player salaries

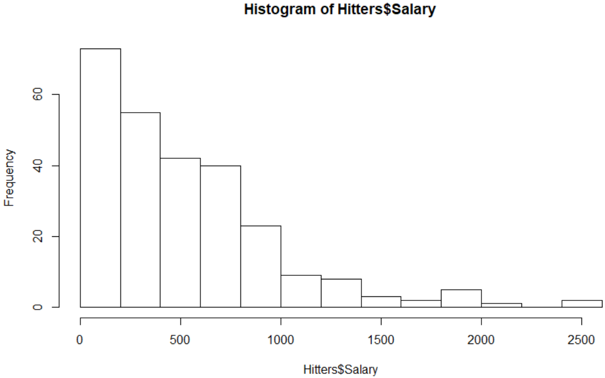
```
     AtBat            Hits          HmRun           Runs            RBI             Walks            Years
 Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 1.000
 1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25   1st Qu.: 28.00   1st Qu.: 22.00   1st Qu.: 4.000
 Median :379.5   Median : 96   Median : 8.00   Median : 48.00   Median : 44.00   Median : 35.00   Median : 6.000
 Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91   Mean   : 48.03   Mean   : 38.74   Mean   : 7.444
 3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00   3rd Qu.: 64.75   3rd Qu.: 53.00   3rd Qu.:11.000
 Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00   Max.   :121.00   Max.   :105.00   Max.   :24.000

     CAtBat            CHits            CHmRun           CRuns             CRBI             CWalks         League
 Min.   :   19.0   Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00   Min.   :   0.00   A:175
 1st Qu.:  816.8   1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75   1st Qu.:  67.25   N:147
 Median : 1928.0   Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50   Median : 170.50
 Mean   : 2648.7   Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12   Mean   : 260.24
 3rd Qu.: 3924.2   3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25   3rd Qu.: 339.25
 Max.   :14053.0   Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00   Max.   :1566.00

 Division    PutOuts          Assists          Errors          Salary        NewLeague
 E:157    Min.   :   0.0   Min.   :  0.0   Min.   : 0.00   Min.   :  67.5   A:176
 W:165    1st Qu.: 109.2   1st Qu.:  7.0   1st Qu.: 3.00   1st Qu.: 190.0   N:146
          Median : 212.0   Median : 39.5   Median : 6.00   Median : 425.0
          Mean   : 288.9   Mean   :106.9   Mean   : 8.04   Mean   : 535.9
          3rd Qu.: 325.0   3rd Qu.:166.0   3rd Qu.:11.00   3rd Qu.: 750.0
          Max.   :1378.0   Max.   :492.0   Max.   :32.00   Max.   :2460.0
                                                           NA's   :59
```

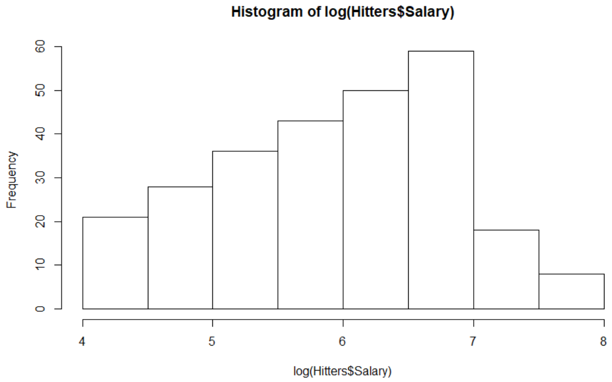
Example

We want to find a regression on baseball player salaries

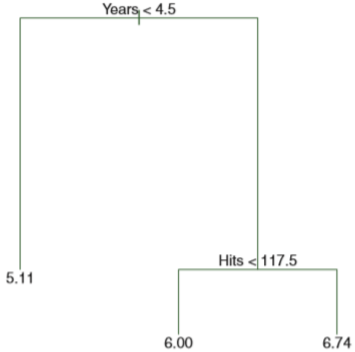


Example

We want to find a regression on baseball player salaries

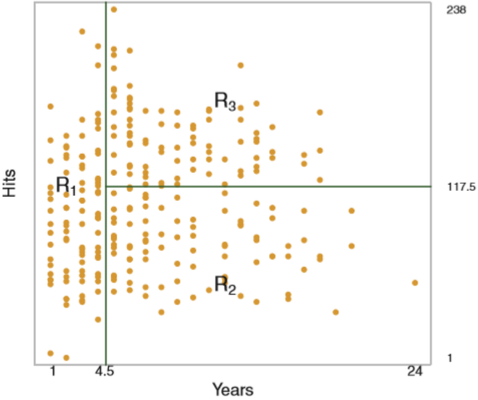


Create a basic tree



make a prediction of e raised to the regression value What is the most important variable?

This partitions our data space



These regions are known as leaves or terminal nodes

Prediction via Stratification

- Divide the predictor space, $X_1, X_2, \cdots, X_p$ into J distinct and non-overlapping regions $R_1, R_2, \cdots, R_J$
- For every observation that falls into the region R_j we make the same prediction, which is simply the mean of the response values for the training observations in R_j

Goal: find boxes that minimize the *RSS* given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

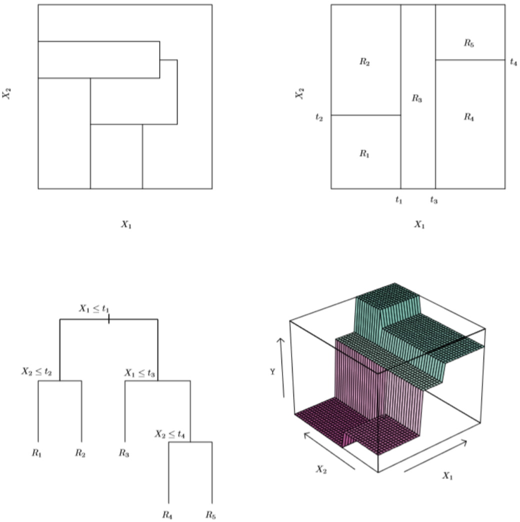
Prediction

- This is done in a top-down, greedy approach called recursive binary splitting
- At each step of the tree, we make a best split decision
- At each cut point s that splits a region into two partitions $R_1(j,s) = \{X|X_j < s\}$ and $R_2(j,s) = \{X|X_j \geq s\}$ that leads to the greatest minimization in RSS

Minimize:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Partitioning: Another Example



- A big tree might overfit

Pruning: Avoiding Overfitting

- A big tree might overfit
- Limiting the depth up front might not result in a huge reduction in RSS, missing a key split

Pruning: Avoiding Overfitting

- A big tree might overfit
- Limiting the depth up front might not result in a huge reduction in RSS, missing a key split
- Best to create a very large tree T_0 then prune it down to obtain an optimal subtree.

Cost Complexity Pruning

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

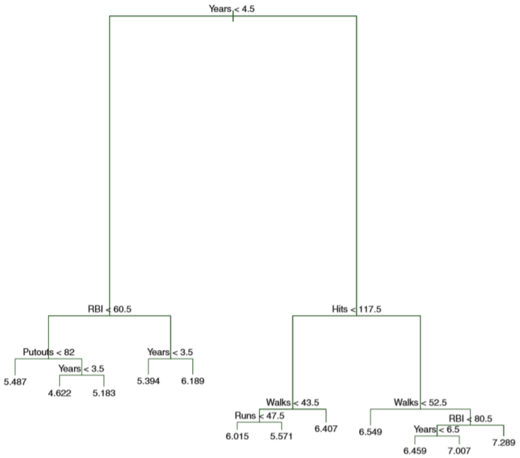
   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

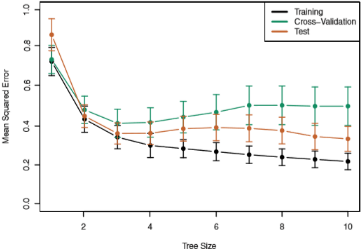
For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \qquad (8.4)$$

Cost Complexity Pruning



Cost Complexity Pruning



Classification Trees

- Same for classification trees as regression trees - only pick most commonly occurring class
- Instead of RSS we look at classification error rate.

$$E = 1 - max_k(\hat{p}_{mk})$$

, where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class. Turns out this is insufficient for tree growing

Gini Index and Entropy

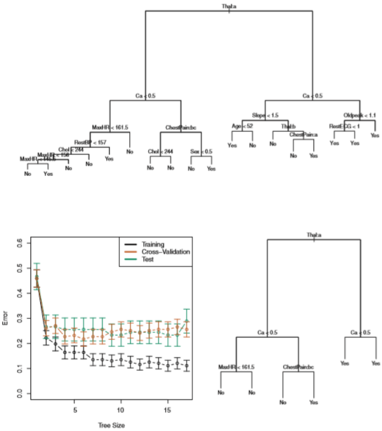
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

, which measures the total variance across K classes. This is a measure of node purity.

$$H = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

, Entropy which takes a value near 0 if all the $\hat{p}$ are near zero or one - smaller value if node is pure

Classification and Pruning



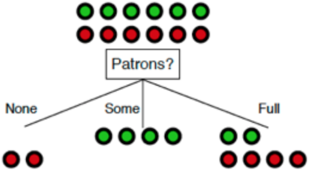
Example

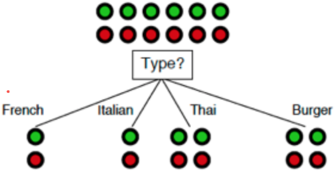
We want to find a decision process for choosing a restaurant

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*

Decision Trees

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





If we split the train samples with respect to the attribute "Patron", we will gain more information regarding the outcome.

How do we measure information gain?

- Intuitively, information gain tells us how important a given attribute is for predicting the outcome
- We will use it to decide the ordering of attributes in the nodes of a decision tree (i.e. tree structure)
- Main idea: Gaining information reduces uncertainty
- From information theory, we have a measure of uncertainty $\rightarrow$ entropy

Entropy for discrete distribution

Let X be a discrete random variable with $\{x_1, \ldots, x_N\}$ outcomes, each occurring with probability $p(x_1), \ldots, p(x_N)$.

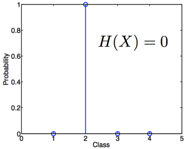
The information content of outcome x_i is inversely proportional to its probability, $h(x_i) = \log \frac{1}{p(x_i)}$

The entropy of the random variable X is the average information content of the outcomes:

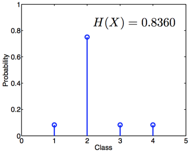
$$H(X) = \sum p(x_i) \log\left(\frac{1}{p(x_i)}\right) = -\sum p(x_i) \log(p(x_i))$$

Example

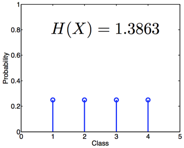
no uncertainty



some uncertainty



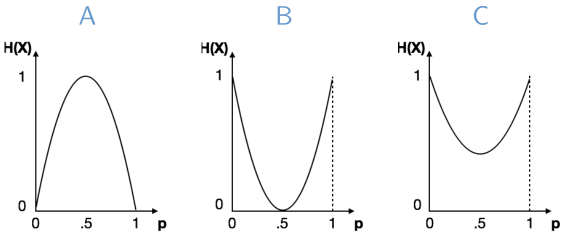
high uncertainty



[Watch this: https://www.khanacademy.org/computing/computer-science/informationtheory/

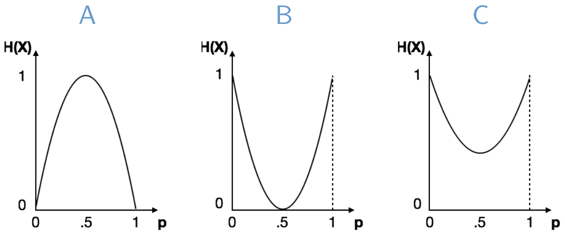
Question:

Suppose $X \sim Bernoulli(p)$ with $X \in \{0, 1\}$, i.e. coin toss with probability p of getting heads and $1 - p$ of getting tails. What would be a correct plot for the entropy $H(X)$ in relation to the probability of getting heads?



Question:

Suppose $X \sim Bernoulli(p)$ with $X \in \{0, 1\}$, i.e. coin toss with probability p of getting heads and $1 - p$ of getting tails. What would be a correct plot for the entropy $H(X)$ in relation to the probability of getting heads?



The correct answer is A

- Minimum entropy $\rightarrow$ no uncertainty about X, i.e. $p = 0$ (only tails) or $p = 1$ (only heads)
- Maximum entropy $\rightarrow$ complete uncertainty about X, i.e. $p = 0.5$ (tails and heads with equal probability)

Entropy for continuous distribution

Let X be a continuous random variable with $x \in \Omega$. Its entropy is defined as follows:

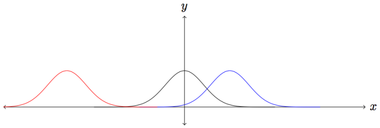
$$H(X) = -\int_{x \in \Omega} p(x) \log(p(x)) dx$$

Example

If $X \sim \mathcal{N}(\mu, \sigma^2)$ its entropy is $\boxed{H(X) = \frac{1}{2}(1 + \log(2\pi\sigma^2))}$.

The entropy depends on the variance of the Gaussian.

i.e. higher variance $\rightarrow$ higher uncertainty, and vice-versa.



Gaussians with the same σ, therefore same entropy.

Conditional Entropy

We want to quantify how much uncertainty the realization of a random variable X has if the outcome of another random variable Y is known. The conditional entropy is defined as:

$$
\begin{aligned}
H(X|Y) &= \sum_{m=1}^{M} p_Y(y_m) H_{X|Y=y_m}(X) \\
&= \sum_{m=1}^{M} p_Y(y_m) \left(- \sum_{n=1}^{N} p_{X|Y}(x_n|y_m) \log(p_{X|Y}(x_n|y_m)) \right) \\
&= - \sum_{m=1}^{M} \sum_{n=1}^{N} p_Y(y_m) p_{X|Y}(x_n|y_m) \log(p_{X|Y}(x_n|y_m))
\end{aligned}
$$

Decision Trees

Example: Choosing a restaurant

Measuring the conditional entropy on each of the "Patrons" attributes

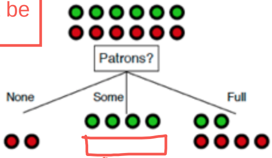
For "None" branch

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

For "Some" branch

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

Shouldn't be 0???



For "Full" branch

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

Measuring the conditional entropy on Patrons

$H(Outcome|Patron) = \frac{2}{12} \times 0 + \frac{4}{12} \times 0 + \frac{6}{12} \times 0.9 = 0.45$

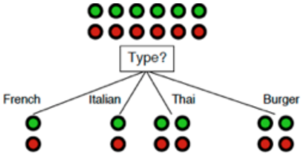
"How uncertain is the Outcome with respect to attribute Patrons"

Example: Choosing a restaurant

Measuring the conditional entropy on each of the "Type" attributes

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"

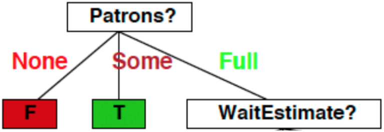
Measuring the conditional entropy on Type

$H(Outcome|Type) = \frac{2}{12} \times 1 + \frac{2}{12} \times 1 + \frac{4}{12} \times 1 + \frac{4}{12} \times 1 = 1$

"How uncertain is the Outcome with respect to attribute Type"

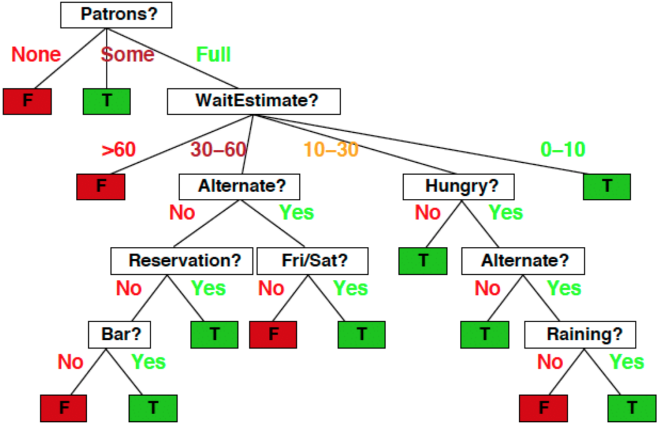
Example: Choosing a restaurant

- $H(Outcome|Patron) < H(Outcome|Type)$,
  $H(Outcome|Patron) < H(Outcome|Price)$, ...
- The entropy of the Outcome conditioned on Patron is the largest
- So the first split is performed with respect to Patron
- We do not split the "None" and "Some" nodes, since their decision is deterministic from the train data
- Next split? We will look only at the 6 instances assigned to the node "Full"



Example: Choosing a restaurant

Greedily we build the tree and looks like this



Decision Trees: Algorithm Outline

GenerateTree($\mathcal{X}$) (Input $\mathcal{X}$: training samples)

1. $i := SplitAttribute(\mathcal{X})$ (find attribute with lowest uncertainty)
2. For each branch of $\mathbf{x_i}$
   - 2a Find $\mathcal{X}_i$ falling in branch
   - 2b GenerateTree($\mathcal{X}_i$)

SplitAttribute($\mathcal{X}$) (Input $\mathcal{X}$: training samples)

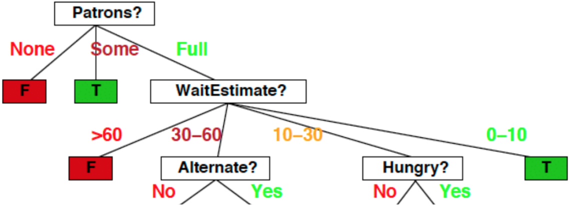
1. $MinEnt := MAX$
2. For all attributes X_i, $i = 1, \ldots, D$
   - 2a Compute $H(Y|\mathcal{X}_i)$ (entropy of attribute X_i)
   - 2b If $MinEnt > H(Y|\mathcal{X}_i)$ (current attribute X_i has the lowest entropy so far)
     - 2b.i $MinEnt := H(Y|\mathcal{X}_i)$
     - 2b.ii $SplitAttr := i$
3. Return $SplitAttr$

Should we continue to split until every training sample is classified correctly?

- We should be very careful about the depth of the tree
- Eventually, we can get all training examples right
  - Is this what we want?
- The maximum depth of the tree is a hyperparameter

Example: Choosing a restaurant

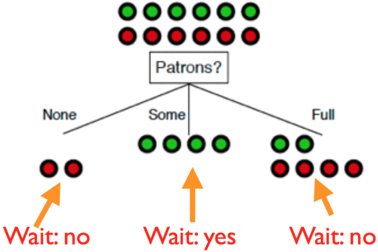
We should prune some of the leaves of the tree to get a smaller depth



- If we stop here, not all training samples are classified correctly
- How do we classify a new instance?
    - We label the leaves of this smaller tree with the label of the majority of training samples

Example: Choosing a restaurant

If we wanted to prune at this first node, we would take the following decisions



- Pre-Pruning
  - Stop growing the tree earlier, before it perfectly classifies the training set
  - Use a min entropy parameter θ_I
- Post-Pruning
  - Grow the tree full until no training error
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node
    - Class label of leaf node is determined from majority class of instances in the sub-tree

Decision Trees: Algorithm Outline with Pre-Pruning

GenerateTree($\mathcal{X}$) (Input $\mathcal{X}$: training samples)

1 If $Entropy(\mathcal{X}) < \theta_I$ (very small uncertainty)

  1a Create leaf labelled by majority class in $\mathcal{X}$

2 Else

  2a $i := SplitAttribute(\mathcal{X})$ (find attribute with lowest uncertainty)

  2b For each branch of $\mathbf{x_i}$

    2b.i Find $\mathcal{X}_i$ falling in branch

    2b.ii GenerateTree($\mathcal{X}_i$)

2-class problem

$\hat{p}$, $1 - \hat{p}$: frequency of class 0 and 1

- Entropy:
  $\phi(\hat{p}) =$
  $-\hat{p} \log \hat{p} - (1 - \hat{p}) \log(1 - \hat{p})$
- Gini index:
  $\phi(\hat{p}) = 2\hat{p}(1 - \hat{p})$
- Misclassification error:
  $\phi(\hat{p}) = 1 - \max(\hat{p}, 1 - \hat{p})$

C-class problem

$\hat{p}_1$,...,$\hat{p}_C$: frequency of class $1, \ldots, C$

- Entropy:
  $\phi(\hat{p}_1, \ldots, \hat{p}_C) =$
  $-\sum_c \hat{p}_c \log \hat{p}_c$
- Gini index:
  $\phi(\hat{p}_1, \ldots, \hat{p}_C) =$
  $\sum_c \hat{p}_c(1 - \hat{p}_c)$
- Misclassification error:
  $\phi(\hat{p}_1, \ldots, \hat{p}_C) = 1 - \max_c(\hat{p}_c)$

Regression Trees

- Similar to classification trees with some differences
- Split criterion
  - Mean square error between predicted and actual value of samples that have reached current node
- Leaf node value
  - Mean (or medium) of samples that have reached the node
  - Linear regression estimate on samples that have reached the node
  - Leaf node is created (splitting stops) if the current node has "acceptable" error

Decision Trees

Advantages

- The models are transparent: easily interpretable by human (as long as the tree is not too big)
- Data can contain combination of continuous and discrete features
- Decision tress more closely mirror human decision making than do regressions?
- Graphical representation
- Qualitative predictors without dummy variables!

Disadvantages

- Usually not same level of predictive accuracy as other regression and classification approaches
- Non-robust - small change in data can change a large amount of the final estimated tree
- Solutions? Bagging, Random Forest, Boosting

Random Forests

- We grow many classification trees through bagging & randomization
- Bagging (**B**ootstrap **agg**regat**ing**)
  - Generate independently bootstrap datasets from original data
  - Run a decision tree in each one of them
- Randomize over the set of attributes
  - Before growing a bootstrap decision tree
  - When splitting an interior node of the classification tree
- No pruning (small trees)
- For each sample, each tree "votes" for a class and we perform majority voting for final decision

Random Forests

Advantages

- Very good performance in practice
- Runs efficiently on large data bases
- Runs efficiently on large feature sets
- Gives estimates of the most relevant variables for the problem

What have we learnt so far

Decision Trees

- Hierarchical (tree-like) structure to perform classification/regression
- Tree structure determined by splitting criterion
  - Entropy (measure of uncertainty), gini index, etc.
- Pruning
  - Prevent overfitting by limiting the depth of the tree
  - Avoids perfect performance on train set
  - Pre/Post-pruning
- Main advantage: interpretability

Random Forests

- Tree ensemble
- Bagging & Randomization
- Good peformance in practice

Takeaways and Next Time

- Decision Trees
- Random Forest
- Next Time: Random Forest and AdaBoost