NUMERICAL EXAMPLES OF DECISION TREES

Shingchern D. You

Motivation

- The textbook just gives a bunch of equations
- Not easy to appreciate how to do it
- We illustrate some examples
- First example: Play or not play ball (binary decision)

Example

- Source: https://cis.temple.edu/~giorgio/cis587/readings/id3-c45.html

ATTRIBUTE	POSSIBLE VALUES
outlook	sunny, overcast, rain
temperature	continuous
humidity	continuous
windy	true, false

Example

- OUTLOOK | TEMPERATURE | HUMIDITY | WINDY | PLAY
- ===
- sunny | 85 | 85 | false | Don't Play
- sunny | 80 | 90 | true | Don't Play
- overcast| 83 | 78 | false | Play
- rain | 70 | 96 | false | Play
- rain | 68 | 80 | false | Play
- rain | 65 | 70 | true | Don't Play
- overcast| 64 | 65 | true | Play
- sunny | 72 | 95 | false | Don't Play
- sunny | 69 | 70 | false | Play
- rain | 75 | 80 | false | Play
- sunny | 75 | 70 | true | Play
- overcast| 72 | 90 | true | Play
- overcast| 81 | 75 | false | Play
- rain | 71 | 80 | true | Don't Play

Example

- Arbitrarily set the following

Temperature $\geq$ 80: hot

Temperature between 70 and 79: sweet

Temperature < 70: cold

Humidity$\geq$ 76: high

Humidity <76: low

Example: ID3 algorithm

- From the table, we have 5 No, and 9 Yes
- Entropy of the set:
- $\mathrm{En}(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$ where p_i is probability of each member in *S*
- In our case, S={play, not play}
- Therefore,

$$\mathrm{En(S)} = -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14} = 0.94$$

Example: ID3 algorithm

- Gain for an attribute *T*:

$$G(S,T) = \mathrm{En}(S) - \sum_{j=1}^{n} p_{T_j}\,\mathrm{En}(T_j)$$

where p_{T_j} is probability of value j in attribute *T*, T_j={play, no play} given the value j to compute probability

- The second term is sometimes called $\mathrm{Info}(S,T)$

Example: ID3 algorithm

- Compute attribute T = outlook
- $\sum_{j=\{\text{sunny,overcast,rain}\}} p_{T_j} \operatorname{En}(T_j)$
- $p_{T_{\text{sunny}}} = \frac{5}{14}$
- $\operatorname{En}(T_{\text{sunny}}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$ (recall in "sunny" case, 2 "play" and 3 "no play")

Example: ID3 algorithm

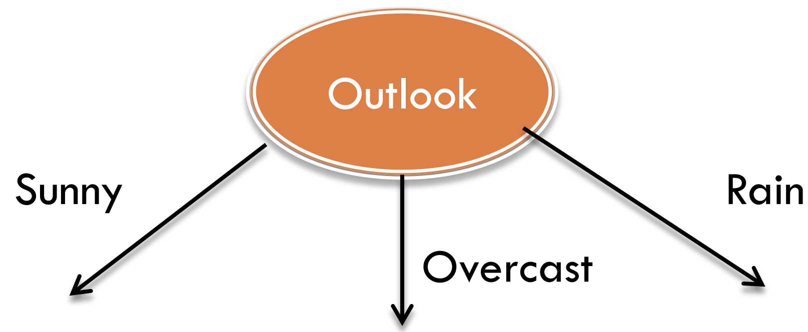
- $p_{T_{\text{overcast}}} = \frac{4}{14}$
- $\text{En}(T_{\text{overcast}}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0$ (in "overcast" case, 4 "play" and 0 "no play")
- We define $0\log_2 0 = 0$ to avoid problems
- $p_{T_{\text{rain}}} = \frac{5}{14}$
- $\text{En}(T_{\text{rain}}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$ (in "rain" case, 3 "play" and 2 "no play")

Example: ID3 algorithm

- Overall, we have G(S, outlook) = 0.246
- G(S, temperature) = 0.0289
- G(S, windy) = 0.048
- G(S, humidity) = ?? (exercise)
- Assuming that G(S, overcast) is greater than others, we use this attribute firs to generate the tree

Example: ID3 algorithm

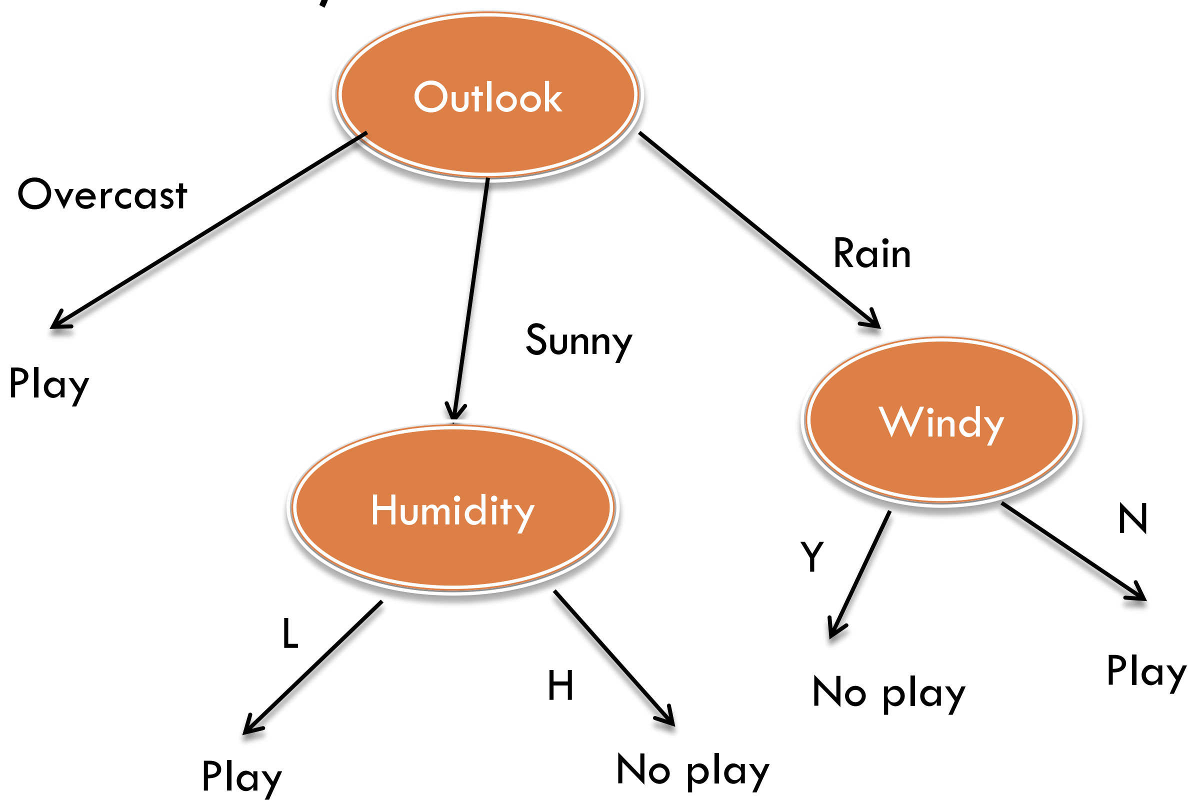
- The figure is like this



- Repeat the same procedure for each subtree with gain for temperature, humidity and windy, until (almost) all samples belonging to correct classes

Example: ID3 algorithm

- The complete tree look like this (note: This figure may not be correct)



Example: C4.5

- ID3 is overly sensitive to features with large numbers of values. Extreme case: all values in an attribute are distinct, then 2nd term in $G(S,T)$ is zero (recall $\log 1 = 0$), so $G(S,T)$ is max
- C4.5 uses a gain ratio function, defined as

$$\mathrm{GR}(S,T) = \frac{G(S,T)}{\mathrm{Split}(S,T)}$$

where $\mathrm{Split}(S,T) = \mathrm{En}(T)$ with probability calculation considering elements on T only

Example: C4.5

- For example, $\mathrm{Split}(S,T)|_{T=\mathrm{outlook}}$ is computed as (sunny 5, overcast 4, rain 5)

$$-\frac{5}{14}*\log_2\left(\frac{5}{14}\right)-\frac{4}{14}*\log_2\left(\frac{4}{14}\right)-\frac{5}{14}*\log_2\left(\frac{5}{14}\right)$$
$$=\ 1.577$$

Thus, $\mathrm{GR}(S,T)|_{T=\mathrm{outlook}}=\frac{0.246}{1.577}=0.156$

$$\mathrm{GR}(S,T)\Big|_{T=\mathrm{windy}}=\frac{0.048}{0.985}=0.049$$

Example: C4.5

- Dealing with continuous numbers (like the attribute of humidity)
- Step 1: list all sorted values in training set. EX: Humidity in the working problem

{65, 70, 70, 70, 75, 78, 80, 80, 80, 85, 90, 90, 95, 96}

- Step 2: remove redundancy. EX:

{65, 70, 75, 78, 80, 85, 90, 95, 96}

Example: C4.5

- Step 3: Let the humidity be partitioned as $\{H <= H_0\}$ and $\{H > H_0\}$, where H_0 is a number in the list in step 2
- Step 4: Compute all Gains based on all possible H_0
- Step 5: Pick H_0 with max Gain

Example: C4.5

- Numerical example
- $H_0 = 65$

$H <= H_0$: # play = 1, # no play = 0

$H > H_0$: # play = 8, # no play = 5

Use the gain equation, we find Gain = 0.048

- Continuing the computation, we find

Max Gain = 0.102 at $H_0 = 80$

(http://saiconference.com/Downloads/SpecialIssueNo10/Paper_3-A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf)

Tree Pruning

- Generating a decision to function best with a given of training data set often creates a tree that **over-fits the data** and is too sensitive on the sample noise
- Such decision trees do not perform well with new unseen samples
- Therefore, we may want to prune the tree
- Pruning means removing some nodes (and branches) of the tree
- Usually from leaf to root

Tree Pruning

- Prepruning: Limit the number of nodes during tree generation
- Postpruning: Cut nodes after tree generated
- Pessimistic pruning tech: a type of postpruning, used in C4.5
- Pessimistic pruning checks the error rate in training set

Tree Pruning

- Pessimistic pruning estimates error based on binomial distribution. With some calculations, classification error is (pessimistically) estimated as

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N} \right)$$

where *z* is from standard normal table (confidence level of 75 %, z = 0.69 default value), *N* is # of instances in a pruned leaf, and *f* is error instances

Handling Unknown Attributes

- C4.5 can handle attribute (feature) missing in some training samples
- Though powerful, this situation is not typical (recall that we “generate” training sets by ourselves)
- Details about this part is omitted. You may refer to references to know how to do it

Random Forest

- What is random forest
  - A whole bunch of decision tress (that is why it gets its name)
  - Decision is made by voting (**ensemble** decision)
- Why use many trees
  - Less likely overfitting (pruning not necessary)
  - Usually performs better

Random Forest

- Suppose N sample points with each one on M dimensional space
- Algorithm to generate K trees
  - For $i = 1 .. K$
  - Sample with replacement N out of N to form training set of tree i
  - Randomly select a subset of M features (say, m) to build a tree
  - Repeat until all trees are built

Random Forest

- Sample with replacement N out of N? Does it mean "select all"?
- NO. Recall that same points may be chosen repeatedly (repeated samples will be discarded)
- Usually around (1/3)*N* out of bag (OOB) sample points
- Sample with replace *N* out of *N* sometimes called "bagging," standing for Bootstrap aggregating
- OOB samples can be used as validation

Random Forest

- Combining many classifiers is covered in CH 17
- Though powerful, random forest has many hyper-parameters to determine
- Such as values of m and K?
- A rule of thumb is $m = \lfloor \sqrt{M} \rfloor$, K is determined by experiments (by using validation dataset)
- Another setting: $K \geq M$, $m = \left\lfloor \frac{K}{2} \right\rfloor + 1$