ECON484 Machine Learning

6. Decision Trees (Part 2, Terminology, Tree Generation, Pruning and Rule Extraction)

Lecturer:
Bora GÜNGÖREN
bora.gungoren@atilim.edu.tr

Decision Trees

- A **decision tree** is a hierarchical data structure, designed to systematically divide the decision problem into sub-problems.
  - It is a useful approach for both classification and regression problems, but in practice it is used in classification more than regression.
- We will first show and discuss basic methods for building a decision tree from a training set with labels. Then we will discuss performance issues and robustness.

Decision Trees

- By recursively splitting our problem space we will end up with a number of regions.
  - What if we develop individual models for each of these regions? These models would be valid for those regions.
- This idea is very similar to splines. But there is a difference.
  - In the spline approach we do not care much about the small issue of locating the region which should be applied to an input. This is because splines are usually used for lower-dimension data sets.
  - But in practice, locating the exact region that matches a particular input is a search problem with a high complexity.
    - How to **represent** a region? A center of mass? A border?
    - How to measure **distance** from one data point to all possible regions?
    - Having to calculate **all** the distances.

Decision Trees

- Several methods utilizing branch and bound algorithms have been proposed to overcome this problem.
    - The core idea is always to **evaluate distance (or membership) to a much smaller number of regions** before locating the matching region.
- So a decision tree is better defined as a hierarchical model for supervised learning which works by **executing a sequence of recursive evaluations to locate the correct region** that matches an input.
    - A decision tree consists of nodes that represents these evaluations as decision making problems $\mathbf{f_m(x)}$ for each node m.
    - The outcome after evaluating each problem gives us the direction to take in the decision tree.
    - The process continues until we with a leaf node, which represents the region.
    - Therefore we end up with **O(log(N))** evaluations.

Decision Trees

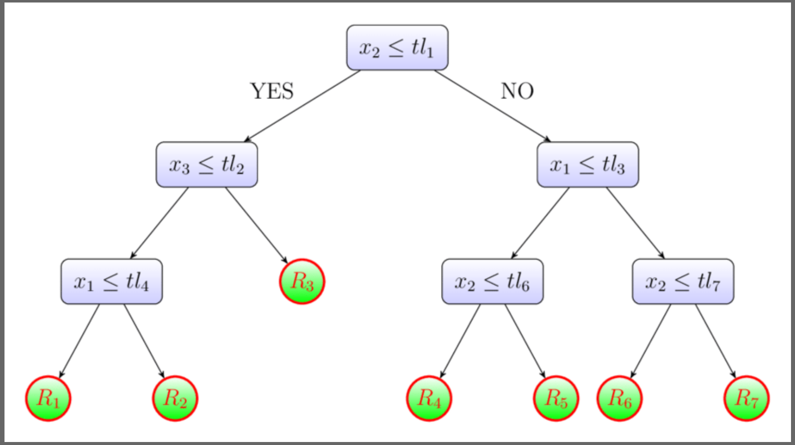
- For a classification problem, the leaf nodes already contain the label for the assigned class.
    - This means the non-leaf nodes could be better defined as **decision nodes** and the leaf nodes better defined as **outcome nodes**.
    - So we traverse the tree through decision nodes until we arrive at an outcome node.
- For a regression problem, this is interestingly similar.
    - For a regression problem, the outcome is a **pre-calculated value for the region represented**.
        - For example, **estimate the price** for the next used car sale based on its brand, model, age, mileage, etc.
        - With a decision tree we split the training set into **regions representing similar cars**.
        - The outcome could be any estimate based on the sale prices of the cars in the training set. A simple average of sale prices of cars, but only those in the region could be used.
        - Therefore instead of a label we arrive at a numerical value, which should be considered a fixed value when we run the decision tree against the particular input.
        - We could be updating that value (ie. continuous/continual learning), but **at the moment we run the model we are working with a fixed value** (that may or may not have been updated recently).
- Therefore design and implementation of a decision tree is not very different for classification and estimation problems.

Decision Trees

- Continuing with the example of used car pricing.
- The European car classification (commonly referred to as segments, check https://bit.ly/3KUjLnL for details.)
  - No exact formula to assign a car to a segment.
  - Weight and size seem to be major factors, but overlaps exist.
  - Engine power, fuel economy, and chassis-type are also considered.
  - An old international standard ISO 3833-1977 was designed to do this, but the car types and production technologies have evolved over time so that the standard is not relevant.
    - For example weight of a modern large sedan could be less than an apparently smaller but older car due to improved materials and concerns over fuel consumption in the design process.
  - The core idea is comparison to some reference car models from major brands. If a car looks very similar to those reference models, then it is assigned to the same segment.

Decision Trees

- How regions could be used?
  - When you want to price a 2012 VW Golf at 150.000 kms, it is easy to assign it to a region defined as “several 2012 VW Golf sales in the past month”
  - When you want to price a 2012 Cherry A3 Chance at 150.000 kms, you cannot have such a large sample, so you might end up in a larger region such as “2012 Asian compact cars sales in the past month” which would probably include similar Kia, Nissan and Hyundai models.
- Note that not all regions should be at the same level in the tree.
  - We define each region by a sequence of decisions that makes you arrive at the region.
  - The **number of decisions** need not be fixed.
  - However in many cases, **we observe a minimum number of decisions** to make before arriving at a region.



Decision Trees

- From a theoretical point of view each decision in a particular sequence defines **a discriminant which divides the d-dimensional space**.
  - In textbook examples we over-simplify the problems, and usually end up with decisions that involve less and less number of variables as we progress, in effect reducing the dimensionality as we make more decisions.
  - This is observed in many practical problems as well.
  - This is not necessarily true in all cases.
  - Each decision tree method has its own method to define the **shape of a discriminant** (ie. the geometry of the region borders).

Decision Trees

- The advantage of the decision tree method is not about the shape of the discriminant, but rather **fast localization** (ie. smaller number of decisions to make).
    - What produces this advantage? **Hierarchical placement** of decisions. This means, which decisions should be made before the others.
    - So we need to solve a sub-problem, given **the shape of discriminant as a design choice** we already made, how to order the decisions so we need to make less decisions.
        - The most basic design choice about the shape of discriminant is whether to use one variable in each decision (ie. if x1 > 0.5) or uses multiple variables (ie. if x1 > 0.5 and x2 > 0.5).
        - A **univariate decision tree** that uses a single variable discriminant creates a simple binary split in the decision space, whereas a **multivariate decision tree** can end up with more splits.
- Another advantage of this method is that we can **extract "if then" type rules** from the sequence of decisions.
    - This is called the **interpretability of a decision tree**.
- Decision trees need not be very accurate methods. They involve a decent amount of error.
    - Their advantage lies in speed and interpretability.

Decision Trees

- A simple univariate decision tree uses a single value to make the decision.
- If a categorical variable is used, then each category is split separately, creating an **n-ary split**.
- If a numeric threshold is chosen as the decision
  - The format is set as $f_m(x) : x_j \geq w_{m0}$
  - The space is divided into two parts; “left” part defined as $L_m = \{x \mid x_j \geq w_{m0}\}$ and “right” part defined as $R_m = \{x \mid x_j < w_{m0}\}$. This is obviously called a **binary split**.
    - A single binary split on a variable creates two infinite regions.
    - Successive binary splits on the same variable can be used to create finite regions.
- When training data which is already labeled is presented, **we already know the number of leaf nodes** (ie. one leaf node for each label).
  - So our algorithm in constructing the decision tree should **minimize the number of decision nodes** in the tree.
  - Algorithms in creating such a tree are greedy.

Decision Trees

- There are some common properties of these algorithms
  - We begin as considering the whole training set as an un-split space at the root.
  - We prefer categorical variables over continuous variables because of the obvious split.
    - If possible, as a data preprocessing step we should **discretize continuous variables into meaningful categories**.
    - However, sometimes part of our problem is to **discover the thresholds for discretization**.
  - Records are distributed recursively on the basis of attribute values.
  - The usefulness of a particular split is evaluated using some **statistical approach**.

Decision Trees

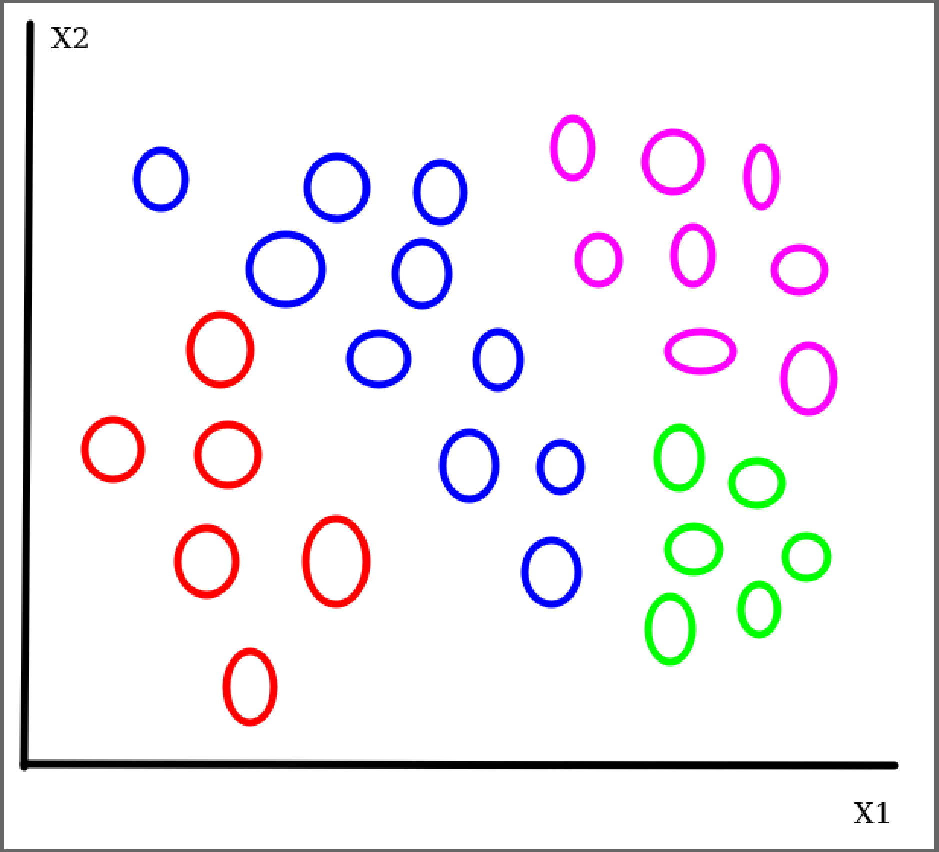
- The basic statistical approach to evaluate a split is the **impurity measure**.
  - A split is **pure**, if after the split, each branch contains data points belonging to the same class. A split is **impure** if any branch contains data points belonging to more than one class.
  - Note that each node is splitting a different number of training data points.
    - If node m is splitting a space containing N_m data points, then each of them belong to their particular classes (because this is a training data set with labels).
    - Let N^i_m data points in this space belong to class C_i such that $\Sigma N^i_m = N_m$.
    - Then **an estimate for the probability of any given data point in this node belonging to class C_i** can be easily formulated as $P(C_i|x,m) = p^i_m = N^i_m / N_m$.
      - If all data points belong to a single class then one of the probabilities will be 1 and all other probabilities will be 0.
      - We can therefore define purity as **having probability estimates being either 0 or 1**.
      - How to **measure degrees of impurity** (regardless of the number of classes)?
      - **Entropy** at a particular node is a common measure (Quinlan 1986): $I_m = - \Sigma \; p^i_m \log_2(p^i_m)$ (and we assume 0 log 0 is 0). The best entropy value is 0.
      - Entropy is an information theoretic measure, which is interpreted as the additional amount of information needed to explain our classification.
        - A pure split would not need any additional information, so the entropy would be 0.

Decision Trees

- For binary classification, there are only two probabilities p and 1-p. So the calculation of entropy is simplified as **$-p\log_2 p - (1-p)\log_2(1-p)$**.
- Is entropy the only method? No.
  - Gini Index (Breiman et.al., 1984)
    - 2p(1-p)
  - Misclassification error
    - 1- max (p, 1-p)
  - Both can be generalized to more than 2 classes.
- In any case, if we see that node m is impure, it should be **split into further nodes**.
  - We try to split so that we minimize impurity.
  - That's where **the decision to choose which variable to use in the splitting decision** comes in.

Decision Trees

- Let’s try to demonstrate how we do the splits.
- Initially we have all four colors in the root node. We try to split using some available lines.
- You should create a table with the following columns
  - Current node contains which colors.
  - N_{Red}, N_{Blue}, N_{Purple}, N_{Green}
  - p_{Red}, p_{Blue}, p_{Purple}, p_{Green}
  - Entropy in the node
  - Total Entropy



Decision Trees

- The splitting process (by hand, ie Spreadheet).
  - In the first split involving root with all colors, we evaluated 4 different configurations, based on different alternatives to split (or not to split).
  - In the second split involving child with 3 color, we evaluated 3 different configurations, based on different alternatives to split (or not to split).
  - In the third split involving child with 2 colors, we evaluated 3 different configurations, based on different alternatives to split (or not to split).
- At the end, we constructed a univariate decision tree with 3 decision nodes, and 4 leaf nodes.
  - **Question.** Does the **order of these decisions matter** in order to minimize entropy?
  - **Question.** Would using Gini instead of Entropy make much difference in this example?
  - **Question.** When you split into two nodes where same colored items (same class) are split between nodes, **can you get minimum entropy?**
  - **Question.** With N colors and two nodes to split, there will be N-1 options to consider in our splitting process. **Can you prove this?**
  - **Question.** Can you optimize the splitting process, so that you will **calculate entropies for a smaller number of options?**

Decision Trees

- Here is a recursive generic algorithm for tree generation:
    - While total entropy > threshold
        - Split node
        - **Repeat for child nodes**
- Here is a generic algorithm for node splitting (numeric)
    - Calculate total entropy
    - Create empty left node, Create empty right node, Move all colors to left node
    - while total entropy > threshold
        - **Discover mathematical rule** so that **one additional arbitrary color** passes to left node
        - Move that color to left node
        - Recalculate total entropy (left, right, sum them up)
- Here is a generic algorithm for node splitting (categorical)
    - Create child nodes as many as the categories
    - Move one category to each child node
    - Recalculate total entropy

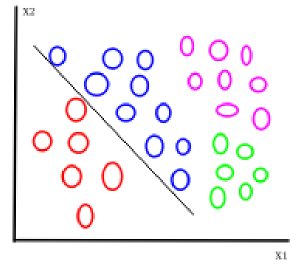
Decision Trees

- So how do we find the mathematical rule?
  - In **univariate trees**, the rule format is really simple.
    - We **search** for the threshold which is basically a boundary value for the items.
    - Example:
      - X1 values for Red = {1, 2,3, 4, 4, 5, 7} and Blue ={9, 10, 10, 11,12, 13}
      - Find the minimum value for X1 for Red, try if this threshold works
        - $X1 \geq 1$ does not split
      - Find the minimum value for X1 for Blue, try if this threshold works
        - $X1 \geq 9$ does split correctly
      - This means with N classes we need to try N thresholds (worst case).
    - With N classes and D dimensions (X_1 to X_D), we might need to try ND thresholds (worst case).

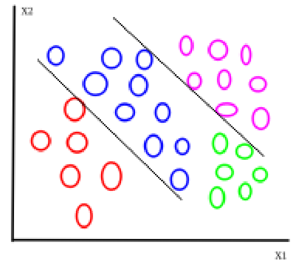
Decision Trees

- So how do we find the mathematical rule?
  - In **multivariate trees**
    - In multivariate trees, there will be $2^d \binom{N_m}{d}$ possible hyperplanes!
    -
    - We can define a border with an equation. This equation ends with “= threshold”. A relationship that ends with “≥ threshold” is defining one side of the border.
    - Usually a linear combination such as $w_1x_1 + w_2x_2 + \dots + w_Dx_D \geq w_0$ will work.
    - When a linear combination does not work we can try a “sphere” defined with the center and radius formula $\|x - c_m\| \leq r_m$ (Devroye, et.al. 1996).

Decision Trees

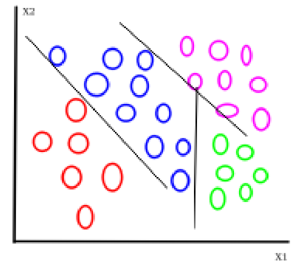


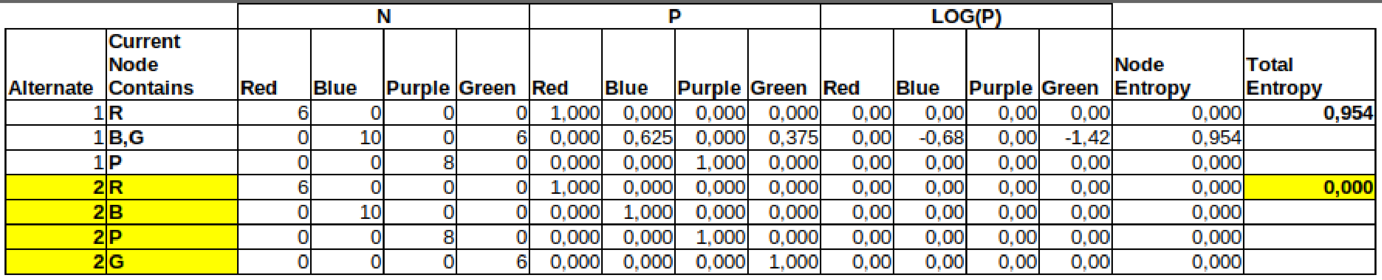
		N				P				LOG(P)					
Alternate	Current Node Contains	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Node Entropy	Total Entropy
1	R, B, P, G	6	10	8	6	0,200	0,333	0,267	0,200	-2,32	-1,58	-1,91	-2,32	1,966	1,966
2	**R**	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	**1,555**
2	**B, P, G**	0	10	8	6	0,000	0,417	0,333	0,250	0,00	-1,26	-1,58	-2,00	1,555	
3	R,B	6	10	0	0	0,375	0,625	0,000	0,000	-1,42	-0,68	0,00	0,00	0,954	1,940
3	P,G	0	0	8	6	0,000	0,000	0,571	0,429	0,00	0,00	-0,81	-1,22	0,985	
4	R,B,P	6	10	2	0	0,333	0,556	0,111	0,000	-1,58	-0,85	-3,17	0,00	1,352	2,352
4	P,G	0	0	6	6	0,000	0,000	0,500	0,500	0,00	0,00	-1,00	-1,00	1,000	



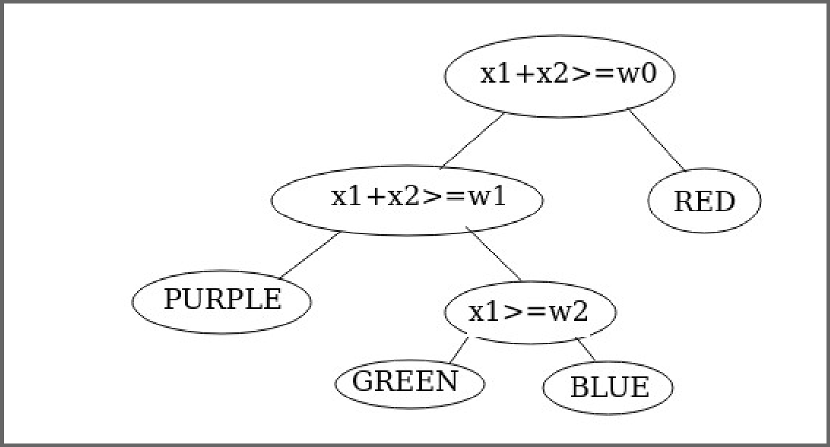
Alternate	Current Node Contains	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Node Entropy	Total Entropy
1	R	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	1,555
1	B, P, G	0	10	8	6	0,000	0,417	0,333	0,250	0,00	-1,26	-1,58	-2,00	1,555	
2	**R**	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	**0,954**
2	**B,G**	0	10	0	6	0,000	0,625	0,000	0,375	0,00	-0,68	0,00	-1,42	0,954	
2	**P**	0	0	8	0	0,000	0,000	1,000	0,000	0,00	0,00	0,00	0,00	0,000	
3	R	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	1,650
3	B,P	0	10	2	0	0,000	0,833	0,167	0,000	0,00	-0,26	-2,58	0,00	0,650	
3	P,G	0	0	6	6	0,000	0,000	0,500	0,500	0,00	0,00	-1,00	-1,00	1,000	

Decision Trees





		N				P				LOG(P)					
Alternate	Current Node Contains	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Red	Blue	Purple	Green	Node Entropy	Total Entropy
1	R	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	**0,954**
1	B,G	0	10	0	6	0,000	0,625	0,000	0,375	0,00	-0,68	0,00	-1,42	0,954	
1	P	0	0	8	0	0,000	0,000	1,000	0,000	0,00	0,00	0,00	0,00	0,000	
2	R	6	0	0	0	1,000	0,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	**0,000**
2	B	0	10	0	0	0,000	1,000	0,000	0,000	0,00	0,00	0,00	0,00	0,000	
2	P	0	0	8	0	0,000	0,000	1,000	0,000	0,00	0,00	0,00	0,00	0,000	
2	G	0	0	0	6	0,000	0,000	0,000	1,000	0,00	0,00	0,00	0,00	0,000	



Decision Trees

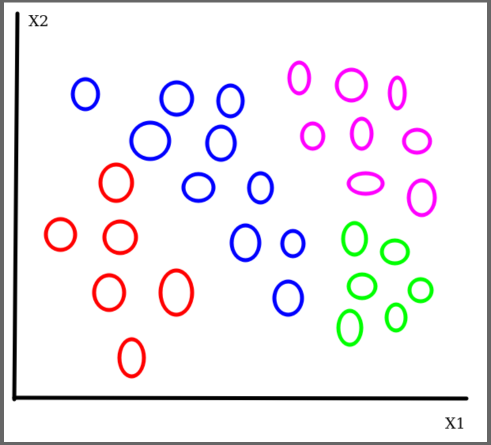
- The most popular algorithms for decision tree generation
  - Classification and Regression Trees (CART) algorithm (Breiman, etal. 1984)
  - ID3 algorithm (Quinlan, 1986) runs faster than CART in most cases.
- Both algorithms are readily available in Python and R.
  - Python sci-kit learn library - https://bit.ly/3vtxJGV - also read https://bit.ly/3Op3BFq and https://bit.ly/3OobPxm for explanations of the algorithms in the library.
  - R has the CART algorithm in Rpart library and the ID3 algorithm in the stnadard data.tree package – see https://bit.ly/3L1EJRW and https://bit.ly/36xR4OO
- Apart from the algorithmic complexity, our common concern of overfitting is still relevant for decision trees.
  - In theory entropy should be zero at the end, so the threshold for our algorithm should be set at zero.
  - In practice, allowing for some entropy by selecting a small positive entropy value as threshold, relaxes the solution and reduces overfitting.

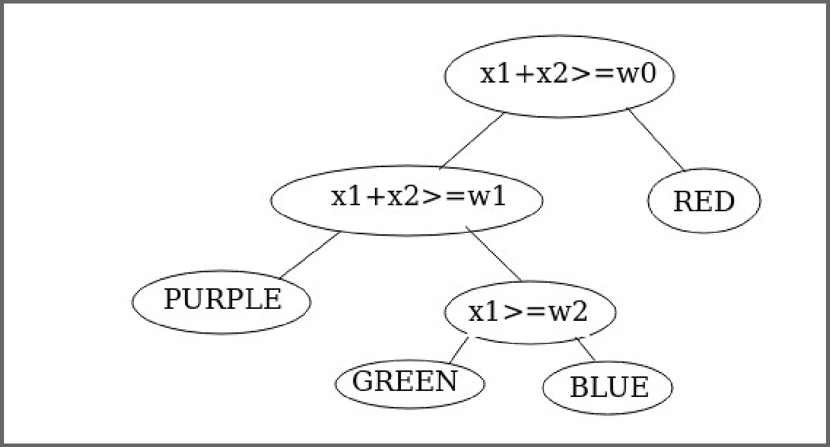
Decision Trees

- Another concern is that a class with very small number of items will not have the statistical stability required.
    - Therefore it is concerned acceptable to combine classes with small populations.
    - The practical method to apply this is to choose **a minimum size for a node**.
        - This is called **pruning**.
        - The size is calculated as a percentage (ie. %2, %5) of the size for the training data set.
    - There are two methods of pruning: pre-pruning and post-pruning.
        - In **pre-pruning**, when splitting a node, if we will end up with child nodes smaller than this minimum size, then we do not split that way.
            - Therefore we will stop tree generation **earlier**.
            - We do not split nodes where we **suspect** there will be overfitting.
        - In **post-pruning**, we first generate the tree with zero entropy.
            - This also means there is zero training error.
            - Then we work bottom-up to join nodes.
            - We select nodes that **will** cause overfitting to join another node.
        - In general pre-pruning is faster, post-pruning ends up with more accurate trees.

Decision Trees

- Once we have a decision tree, we can extract rules easily.
- For each leaf node, we traverse the tree top to bottom.
  - The decisions we make through the tree are joined with logical AND operator.
  - Based on the side, you add NOT operator
  - Example:
    - RED: NOT (x1+x2>=w0)
    - PURPLE: (x1+x2>=w0) AND NOT (x1+x2>=w1)
    - GREEN: (x1+x2>=w0) AND (x1+x2>=w1) AND NOT (x1+x2>=w1) AND NOT (x1 >=w2)
    - BLUE: (x1+x2>=w0) AND (x1+x2>=w1) AND NOT (x1+x2>=w1) AND (x1 >=w2)





Decision Trees

- In some cases, a class may be partially defined.
- For partially defined classes, there will be multiple leaf nodes labeled the same.
- Then we will use the logical OR to combine each rule.
    - Example: Normal = { Students with grades between 40-89 inclusive}, Students to Call = {All others}
        - Here we need to define NOT (Grade >= 40) OR (Grade >=90)

Decision Trees

- The process of extracting these rules is called **knowledge extraction**. It falls into the broad category of **knowledge discovery**.
- Once we have the rules, we can also calculate easily what percentage of the training data set falls under each rule. This is called a **rule support**.
  - With zero entropy, rule supports should match the percentage populations of labels in the training set.
  - However, to avoid over-fitting, we usually allow a small amount of entropy. Therefore in practice, rule supports are different than percentage populations of labels.

Decision Trees

- There is also an alternate approach called **rule induction**, in which we do not need to create the whole to tree to extract a rule.
  - In this approach we extract one rule at a time.
    - Then you remove the data points that are lab and re-run the process. This is called **sequential covering**.
    - You can also run the process to further discover rules within the class.
  - The first rule induction algorithm is called LEM1 (Pawlak, 1982) and easiest to learn in detail. It has academically competed for some time with AQ (Michalski, et.al. 1986).
  - Two algorithms are very popular in this approach.
    - Repeated Incremental Pruning to Produce Error Reduction, Ripper (Cohen, 1995).
    - Irep (Fürnkratz and Widmer, 1994).
  - Again libraries exist for ease of use
    - For Python, a reliable implementation of both is in the wittgenstein library – see https://bit.ly/3EKeHjX for the code and https://bit.ly/3uXekyY for an example.
    - For R, you will need caret and rweka libraries – see https://bit.ly/3KYCTBh
  - S-fold cross validation is very common in validating rule induction results.

Decision Trees

- Note that when developing **an actual decision support software**, there are very efficient ways to determine if a case fits one or more of a large number of rules.
  - Software components for this type of operation are called **rule engines** and have been around for several decades.
- In practical data science projects:
  - Data scientists **extract the rules from the training data**, and create a report (ie. text file with if-then rules.)
  - Software developers (ie. computer engineers) developing the decision support software **get these rules from the report**, and embed them into the rule engine of their choice.
  - This is not theoretically optimal, because you do not discover new rules in run-time (ie. no continuous learning), but it is **efficient from a project management point of view**.
  - The ready availability of this approach is one of the reasons why decision trees are very popular.

Questions?

CONTACT:
bora.gungoren@atilim.edu.tr