Applied Data Mining

Rockhurst University

Decision Trees

Section 1

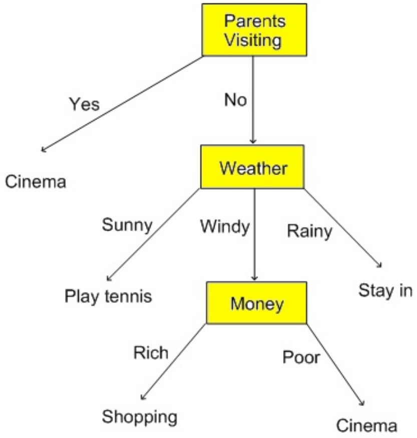
Decision trees

Remember: we want to classify and/or predict things

- We use **supervised learning** to classify or predict things - we know the target variable.
- Naïve Bayes and K Nearest Neighbors are two such approaches
- Tonight: decision trees

Making decisions

Think about how you make decisions. What will you do this weekend?



Decision classification

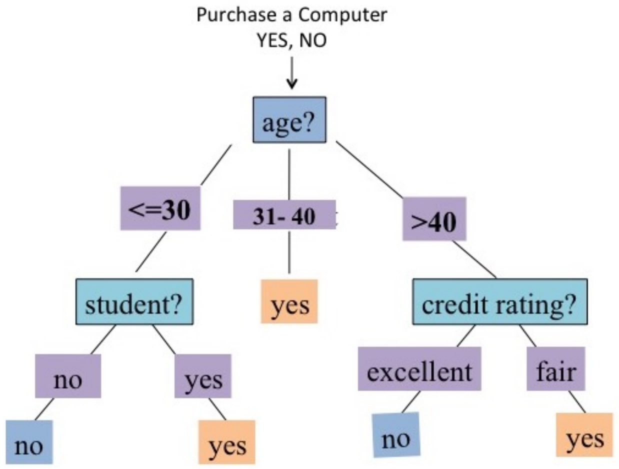
- If you have 4 options for what to do this weekend, you use other information to help decide which one.
- This is is a classification problem where your target variable is "Weekend activity".
- You are using information on your parents' whereabouts, the weather, and your financial situation to figure out to which "class" your "Weekend activity" variable belongs.
- Or maybe you are predicting what your friend will do this weekend based on the data you have about his or her parents, the weather, and their money.

We can do the same thing with data for other activities. For example, how do we know when people might buy a computer?

Decision tree example: data

	Age	Income	Student	Credit rating	Buys computer
1	Under30	high	no	fair	no
2	Under30	high	no	excellent	no
3	30to40	high	no	fair	yes
4	Over40	medium	no	fair	yes
5	Over40	low	yes	fair	yes
6	Over40	low	yes	excellent	no
7	30to40	low	yes	excellent	yes
8	Under30	medium	no	fair	no
9	Under30	low	yes	fair	yes
10	Over40	medium	yes	fair	yes
11	Under30	medium	yes	excellent	yes
12	30to40	medium	no	excellent	yes
13	30to40	high	yes	fair	yes
14	Over40	medium	no	excellent	no

Draw a tree


Purchase a Computer
YES, NO
age?
<=30
31- 40
>40
student?
yes
credit rating?
no
yes
excellent
fair
no
yes
no
yes

Steps in a decision tree

Picture looks simple. How does a decision tree actually work?

Basic algorithm (a greedy algorithm)

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root
- Attributes are categorical or are treated that way
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Attribute selection

How does the tree "decide" how to split into branches? A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).

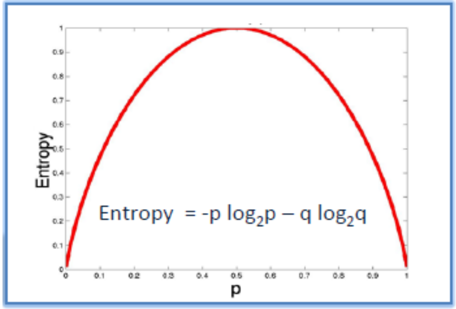
Some algorithms uses **entropy** to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

Entropy: $E(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$

Where p_i is the probability of a particular attribute and c is the number of values that the attribute can take.

Entropy picture

Entropy is a probabilistic measure of uncertainty or ignorance and **information** is a measure of the reduction of uncertainty.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Entropy calculations

Where do we get the probabilities?

Buy	
Yes	**No**
9	5

$$\begin{aligned} Entropy(Buy) =& Entropy(9/14, 5/14) \\ =& Entropy(0.643, 0.357) \\ =& -(0.643 log_2 * 0.643) - (0.357 log_2 * 0.357) \\ =& 0.41 + 0.53 \\ =& 0.94 \end{aligned}$$

This tells us the entropy of our target - buying a computer.

Joint probabilites

How do we know which other feature of our data helps determine the state of the target variable?

We need to know the entropy for two attributes occurring together.

Entropy: $E(T,X) = \sum_{c \in X} P(c)E(c)$

Or the probability times the entropy for each class of X.

For example, what is the entropy of buying a computer and income, E(Buy, Income)?

According to the formula it is

P(High)E(High) + P(Medium)E(Medium) + P(Low)E(Low)

More calcuations

So we need more numbers:

	Buy	
Income	**Yes**	**No**
High	2	2
Medium	4	2
Low	3	1

$$\begin{aligned} &P(High)E(High)+ \quad P(Medium)E(Medium)+ \quad P(Low)E(Low) = \\ &(4/14)*E(2,2)+ \quad (6/14)*E(4,2)+ \quad (4/14)*E(3,1) = \\ &(0.286)*(1)+ \quad (0.429)*(0.918)+ \quad (0.286)*(0.811) = \\ &\mathbf{0.911} \end{aligned}$$

Information gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute.

Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

$Gain(T, X) = Entropy(T) - Entropy(T, X)$

So we have:

$$\begin{aligned} Gain(Buy, Income) =& Entropy(Buy) - Entropy(Buy, Income) \\ =& (0.94) - (0.911) \\ =& \mathbf{0.029} \end{aligned} \tag{1}$$

Largest gain wins

Do these calculations for all the features.

$$\begin{aligned}
Gain(Buy, Age) =& Entropy(Buy) - Entropy(Buy, Age) && = 0.94 - 0.693 = \mathbf{0.247} \\
Gain(Buy, Student) =& Entropy(Buy) - Entropy(Buy, Student) && = 0.94 - 0.786 = \mathbf{0.152} \\
Gain(Buy, Credit) =& Entropy(Buy) - Entropy(Buy, Credit) && = 0.94 - 0.895 = \mathbf{0.048} \\
Gain(Buy, Income) =& Entropy(Buy) - Entropy(Buy, Income) && = 0.94 - 0.911 = \mathbf{0.029}
\end{aligned}$$

Age provides the highest information gain - it will be the first split.

And repeat

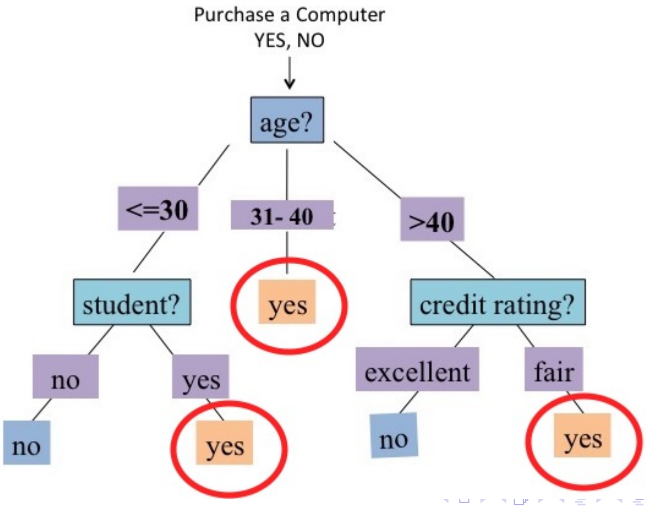
- Now the tree is split into branches by age. Since there are more than two classes to the age variable and trees only bifurcate, the split will have two classes on one branch. The algorithm continues these calculations along each branch separately to determine the next split for that branch.
- Calculations are independent: if the next split for Under30 is Student, it doesn't have to be the case that the next split for Over40 has to be on Student.
- A branch with entropy > 0 can be split further.
- A branch with entropy $= 0$ is a **leaf node**. The 30to40 branch is a leaf: everyone in that branch buys a computer.

When does it stop?

Conditions for stopping partitioning:

- All samples for a given node belong to the same class. (Entropy is 0 — complete purity or no ignorance about which class observations belong to.)
- There are no remaining attributes for further partitioning — majority voting is employed for classifying the leaf
- There are no samples left

Note the nodes



Information gain issues

This isn't a perfect measure: attributes with a large number of classes cause problems:

- Subsets are more likely to be pure if there is a large number of values.
- Information gain is biased towards choosing those attributes.
- This may result in *overfitting*

Other measures: Gain ratio

Gain ratio is a modification of information gain that reduces its bias.

- Gain ratio takes the number and size of branches into account when choosing and attribute.
- It scales the gain by the *intrinsic information* or *split information*.
- If the split information is high then the partitions are more or less the same size.
- If the split information is low then a few partitions have most of the observations.

Formulas:

$$SplitInfo(T,X) = -\sum_{j=1} \frac{|X_j|}{|X|} * log_2(\frac{|X_j|}{X})$$

$$GainRatio(X) = \frac{Gain(X)}{SplitInfo(X)}$$

Gain ratio for income

Let's look at our computer example again and see how the gain ratio compares to the information gain measure for the "income"

The attribute *income* has 3 partitions: low (4 cases), medium (6 cases) , and high (4 cases).

$$SplitInfo(Buy, Income) = -\frac{4}{14} * log_2(\frac{4}{14}) - \frac{6}{14} * log_2(\frac{6}{14}) - \frac{4}{14} * log_2(\frac{4}{14}) = 1.557$$

Now compare:

$$Gain(Buy, Income) = 0.029$$

$$GainRatio(Buy, Income) = \frac{0.029}{1.557} = 0.019$$

Gain ratio splits

Which variable would we split on first if we used gain ratio instead of information gain?

$$\begin{aligned} GainRatio(Buy, Age) &= \frac{0.247}{1.577} = \mathbf{0.156} \\ GainRatio(Buy, Student) &= \frac{0.152}{1} = \mathbf{0.152} \\ GainRatio(Buy, Credit) &= \frac{0.048}{0.985} = \mathbf{0.049} \\ GainRatio(Buy, Income) &= \frac{0.029}{1.557} = \mathbf{0.019} \end{aligned}$$

It would still be age but it is much closer now.

Other measures: Gini Index

Instead of looking at information gain, some algorithms decide by looking at the purity of the node resulting from a split.

The **Gini index** measures the impurity for a partition X as:

$$Gini(T) = 1 - \sum_{j=1}^{m} p_j^2$$

Where m is the number of classes and p_j is the probability that an observation belongs to a particular class.

Gini index is “backwards” from the others: when nodes are completely pure, Gini has the lowest value of 0.

So a high Gini index isn’t good (like high information gain).

Using Gini to determine splits

When using the Gini index, splits are determined by which attribute makes the nodes the “purest” so the change in the Gini index is the increase in purity rather than an increase in information.

Process is very similar: compare overall Gini index with the Gini that results from using information from one attribute.

$$\Delta Gini(T) = Gini(T) - Gini(T, X)$$

Gini calculations for computer example

Compute the Gini for the overall purchase class: $Gini(Buy)$. Remember we have 9 "Yes" and 5 "No"

$Gini(Buy) = 1 - ((\frac{9}{14})^2 + (\frac{5}{14})^2) = 0.459$

Now let's check income:

$$\begin{aligned}
Gini(Income_{High}) &= 1 - ((\frac{2}{4})^2 + (\frac{2}{4})^2) = && 0.5 \\
Gini(Income_{Medium}) &= 1 - ((\frac{4}{6})^2 + (\frac{2}{6})^2) = && 0.444 \\
Gini(Income_{Low}) &= 1 - ((\frac{3}{4})^2 + (\frac{1}{4})^2) = && 0.375
\end{aligned}$$

$Gini(Income) = \frac{4}{14} * Gini(Income_{High}) + \frac{6}{14} * Gini(Income_{Medium}) + \frac{4}{14} * Gini(Income_{Low}) = 0.44$

Gini index splits

What happens if we use Gini instead?

$$\begin{aligned}
\Delta Gini(Age) &= 0.459 - 0.342 && = \mathbf{0.116} \\
\Delta Gini(Student) &= 0.459 - 0.367 && = \mathbf{0.092} \\
\Delta Gini(Credit) &= 0.459 - 0.429 && = \mathbf{0.031} \\
\Delta Gini(Income) &= 0.459 - 0.44 && = \mathbf{0.019}
\end{aligned}$$

Age still wins.

Comparing measures

The three measures, in general, return good results but

- **Information gain**: biased towards multivalued attributes
- **Gain ratio**: tends to prefer unbalanced splits in which one partition is much smaller than the others
- **Gini index**: biased to multivalued attributes; has difficulty when number of classes is large; tends to favor tests that result in equal-sized partitions and purity in both partitions

Other measures

Information gain, gain ratio, and Gini index are the most common measures but there are others:

- C5.0 old but popular algorithm. Evolution of ID3 and C4.5. Uses entropy
- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than information gain and Gini index in certain cases
- G-statistic: has a close approximation to χ^2distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred): Requires the fewest number of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- CART: finds multivariate splits based on a linear combination of attributes

Overfitting

We mentioned that the bias in the information gain measure might lead to *overfitting*. What is that?

- Too many branches, some may reflect anomalies due to noise or outliers
- Poor accuracy for unseen samples

In general a model may overfit the data if it works *really well* on the training data but not well at all on data it has never encountered.

Essentially, it fits the current sample nearly perfectly but if that sample doesn't look like the rest of the data, it won't do well with new data.

Pruning the tree

One way to avoid to overfitting with decision trees is to **prune** the tree. You can choose to do this at the beginning or the end of the tree construction.

- **Prepruning**: This approach halts tree construction early. Even if there is another split that could be done, do not do it if this would result in the goodness measure falling below a threshold It can be difficult to choose an appropriate threshold
- **Postpruning**: Remove branches from a “fully grown” tree — get a sequence of progressively pruned trees. You need to use a set of data different from the training data to decide which is the “best pruned tree”

Automated pruning

One method is to decide how to prune based on a particular calculation: the *cost complexity* (CC) is defined as:

$$CC(k) = RSS_k + \lambda k$$

Where:

- λ is a smoothing parameter
- k is the number of terminal nodes
- RSS is the residual sum of squares

The goal is to find the λ that minimizes the cost complexity.

Automated pruning in R

The package *Rpart* calculates a complexity parameter: $CP = \frac{\lambda}{RSS_1}$

where RSS_1 is the root node. The default stopping point is where $CP = 0.01$

The complexity parameter essential checks to see if the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue.

We'll look at the CP table in the markdown.

Wrap up

- Decision trees are very popular classification algorithms.
- There are a variety of measures you can use to determine splits which have pros and cons
- Overfitting can be addressed by pruning the tree
- Like KNN and Naïve Bayes, prediction isn't perfect

Sources

- http://study.com/academy/lesson/what-is-a-decision-tree-examples-advantages-role-in-managem html
- http://www.saedsayad.com/decision_tree.htm