# Entropy – General Idea

Definition:

"In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples"

Given a set of examples, S. And a binary categorisation

Where $p_+$ is the proportion of positive "examples"

And $p_-$ is the proportion of negatives

$$Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

# Entropy – General Idea

- Entropy as a measure of Information

Ex. The information content of a message telling the outcome of flipping an honest coin is

I(Coin Toss) = -p(heads) $\log_2$ p(heads) - p(tails) $\log_2$ p(tails)

= - 1/2 $\log_2$(1/2) - 1/2 $\log_2$(1/2)

= 1 bit

If the coin has been rigged to come up heads 75% of the time, the information content will be less or more ?!

# Entropy – General Idea

- Entropy as a measure of Information

Ex. The information content of a message telling the outcome of flipping an honest coin is

I(Coin Toss) = -p(heads) $\log_2$ p(heads) - p(tails) $\log_2$ p(tails)

= - 1/2 $\log_2$(1/2) - 1/2 $\log_2$(1/2)

= 1 bit

If the coin has been rigged to come up heads 75% of the time, the information content will be less or more ?!

I[Coin Toss] = - 3/4 $\log_2$(3/4) - 1/4 $\log_2$(1/4)

= 0.811 bits

Remark: Note for users of old calculators:

May need to use the fact that $\log_2(x) = \ln(x)/\ln(2)$

And also note that, by convention: $0*\log_2(0)$ is taken to be 0

# Entropy – General Idea

- In categorisations $c_1$ to $c_n$

Where $p_n$ is the proportion of examples in $c_n$

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

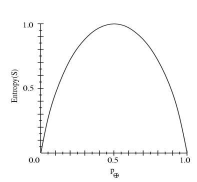
$p_i$ is the probability of class i

Computes the entropy as the proportion of class i in the set.

The higher the entropy the more the information content.

# Entropy – General Idea

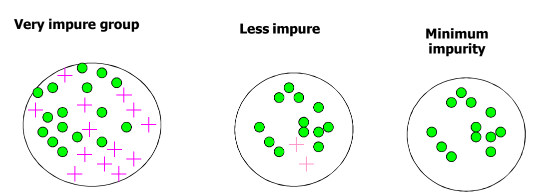
Entropy




- $S$ is a sample of training examples

Entropy(S)
positives p+ approaches 0.5 (very impure),
the Entropy of S converges to 1.0

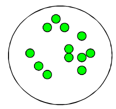
# Entropy – General Idea
# Impurity



# Entropy – General Idea Impurity

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0
  - **not a good training set for learning**

**Minimum impurity**

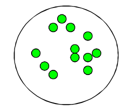




# Entropy – General Idea Impurity

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0
  - **not a good training set for learning**

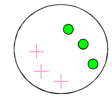
**Minimum impurity**



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

**good training set for learning**

**Maximum impurity**





# Information Gain

We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

i.e.

Information gain tells us how important a given attribute of the feature vectors is.

We will use it to decide the order of attributes in the nodes of a decision tree.



# Calculating Information Gain

Entire population (30 instances)
Fins first its entropy

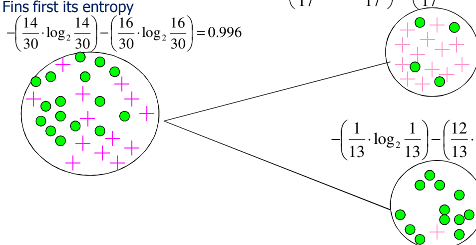
$$-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$$

$$-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$$

Find its entropy 17 instances

$$-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$$

Find its entropy 13 instances



(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

**Information Gain= 0.996 - 0.615 = 0.38**



# Calculating Information Gain

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Which attribute creates the most homogeneous branches?
  - First the entropy of the total dataset is calculated.
  - The dataset is then split on the different attributes.
  - The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split.
  - The resulting entropy is subtracted from the entropy before the split.
- The result is the Information Gain, or decrease in entropy.
- The attribute that yields the largest IG is chosen for the decision node.

# Calculating Information Gain

Given a set of examples S and an attribute A

- Let $p_i$ be the probability that an arbitrary leaf in S belongs to class $C_i$, estimated by $|C_i|/|S|$
- Information needed (after using A to split S into v partitions) to classify S:

$$Info_A(S)=\sum_{i=1}^{v}\frac{|C_i|}{|S|}\times I(C_i)$$

- Expected information (entropy) needed to classify a leaf in S:

$$Info(S)=-\sum_{i=1}^{m}p_i\log_2(p_i)$$

- Information gained by branching on attribute A

$$Gain(A)=Info(S)-Info_A(S)$$

- The information is measured in bits.

# Credit Risk Example

Let us consider our credit risk data. There are three feature values in 14 classes.

6 classes have high risk, 3 have moderate risk, 5 have low risk. Assuming *uniform* distribution, their probabilities are as follows:

| NO. | RISK | CREDIT HISTORY | DEBT | COLLATERAL | INCOME |
|---|---|---|---|---|---|
| 1. | high | bad | high | none | \$0 to \$15k |
| 2. | high | unknown | high | none | \$15 to \$35k |
| 3. | moderate | unknown | low | none | \$15 to \$35k |
| 4. | high | unknown | low | none | \$0 to \$15k |
| 5. | low | unknown | low | none | over \$35k |
| 6. | low | unknown | low | adequate | over \$35k |
| 7. | high | bad | low | none | \$0 to \$15k |
| 8. | moderate | bad | low | adequate | over \$35k |
| 9. | low | good | low | none | over \$35k |
| 10. | low | good | high | adequate | over \$35k |
| 11. | high | good | high | none | \$0 to \$15k |
| 12. | moderate | good | high | none | \$15 to \$35k |
| 13. | low | good | high | none | over \$35k |
| 14. | high | bad | high | none | \$15 to \$35k |

high $\frac{6}{14}$, moderate $\frac{3}{14}$ ,low $\frac{5}{14}$

Information contained in this partition:

*Info(S)* = -(6/14) $\log_2$ (6/14) -

(3/14) $\log_2$ (3/14) - (5/14)$\log_2$ (5/14)

≈ 1.531 bits

# Expected Info.

Let property A(Income) be at the root, and let $C_1$, ..., $C_n$ be the partitions of the examples on this feature.

Information needed to build a tree for partition $C_i$ is $I(C_i)$.

Expected information needed to build the whole tree is a *weighted average* of $I(C_i)$.

Let |S| be the cardinality of set S.

Let $\{C_i\}$ be the set of all partitions.

Expected information needed to complete the tree with root A

$$Info_A(S) = \sum_{i=1}^{n} \frac{|C_i|}{|S|} \times I(C_i)$$

# Expected Info.

In our data, there are three partitions based on income:

All examples have high risk:

$I(C_1) = -1 \log_2 1 = 0.0$.

$C_1 = \{1, 4, 7, 11\}$, $|C_1| = 4$, $I(C_1) = 0.0$

Two examples have high risk, two have moderate:

$I(C_2) = -1/2 \log_2 1/2 - 1/2 \log_2 1/2 = 1.0$.

$C_2 = \{2, 3, 12, 14\}$, $|C_2| = 4$, $I(C_2) = 1.0$

$I(C_3) = -1/6 \log_2 1/6 - 5/6 \log_2 5/6 \approx 0.65$.

$C_3 = \{5, 6, 8, 9, 10, 13\}$, $|C_3| = 6$, $I(C_3) \approx 0.65$

The expected information to complete the tree using income as the root feature is this:

4/14 * 0.0 + 4/14 * 1.0 + 6/14 * 0.65 ≈ 0.564 bits

**i.e. $Info_A$ (S)= 0.564**

# The gain of a property A

Now the information gain from selecting feature P for tree-building, given a set of classes C.

$$Gain(A) = Info(S) - Info_A(S)$$

For our sample data and for P = income, we get

*Gain(A)*= 1.531 - 0.564 bits = 0.967 bits.

# The gain of a property A

Our analysis will be complete, and our choice clear, after we have similarly considered the remaining three features. The values are as follows:

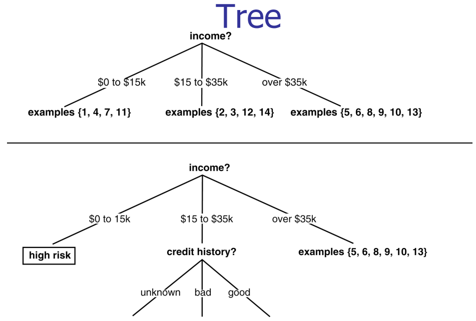
*Gain*(COLLATERAL) ≈ 0.756 bits,

*Gain*(DEBT) ≈ 0.581 bits,

*Gain*(CREDIT HISTORY) ≈ 0.266 bits.

That is, we should choose INCOME as the criterion in the root of the best decision tree that we can construct. And continue recursively…

# Recursively Creating the Decision Tree



## The ID3 Algorithm

Given a set of examples, S

- Described by a set of attributes $A_i$
- Categorised into categories $c_j$

1. Choose the root node to be attribute A
   - Such that A scores highest for information gain
     - Relative to S, i.e., gain(S,A) is the highest over all attributes
2. For each value v that A can take
   - Draw a branch and label each with corresponding v

## The ID3 Algorithm

For each branch you've just drawn (for value v)

- If $S_v$ only contains examples in category c
  - Then put that category as a leaf node in the tree

Remove A from attributes which can be put into nodes

Replace S with $S_v$

Find new attribute A scoring best for Gain(S, A)

Start again at part 2

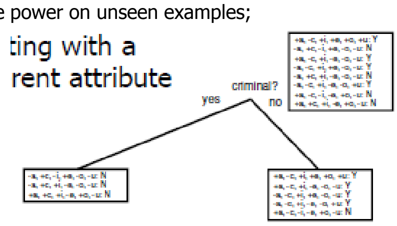
Remark: This is a greedy algorithm: (a form of hill climbing.)

## Overfitting the DT

The depth of the tree is related to the generalization capability of the tree. If not carefully chosen it may lead to overfitting.

A tree **overfits** the data if we let it grow deep enough so that it begins to capture "adeviation" in the data that harm the predictive power on unseen examples;

Recall




## Overfitting the DT

There are two main solutions to overfitting in a decision tree:

1) Stop the tree early before it begins to overfit the data

→ In practice this solution is hard to implement because it is not clear what is a stopping point.

2) Grow the tree until the algorithm stops even if the overfitting problem shows ,Then prune the tree.

→ This method has found great popularity in the machine learning community

## Decision Tree Pruning

common decision tree pruning algorithm depends on :

1- Considering all internal nodes in the tree

2- For each node, check if removing it (along with the subtree) and assigning most common class to it does not harm the accuracy of the data.

## Practical issues in DT

Practical issues while building a decision tree:

1) Choosing a node (using the info gain concept)

2) How deep should the tree be?

3) How do we handle continuous attributes (So far we discussed only discrete)?

4) What happens when attribute values are missing?

5) How do we improve the computational efficiency

## Advantages of using ID3

- Understandable prediction rules are created from the training data.
- Builds the fastest tree.
- Builds a short tree.
- Only need to test enough attributes until all data is classified.
- Finding leaf nodes enables test data to be pruned, reducing number of tests.
- Whole dataset is searched to create tree.

## Disadvantages of using ID3

- Data may be over-fitted or over-classified, if a small sample is tested.
- Only one attribute at a time is tested for making a decision.
- Classifying continuous data may be computationally expensive, as many trees must be generated to see where to break the continuum.