

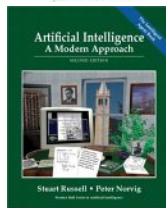
Advanced Artificial Intelligence

729-b

References of this Lec.



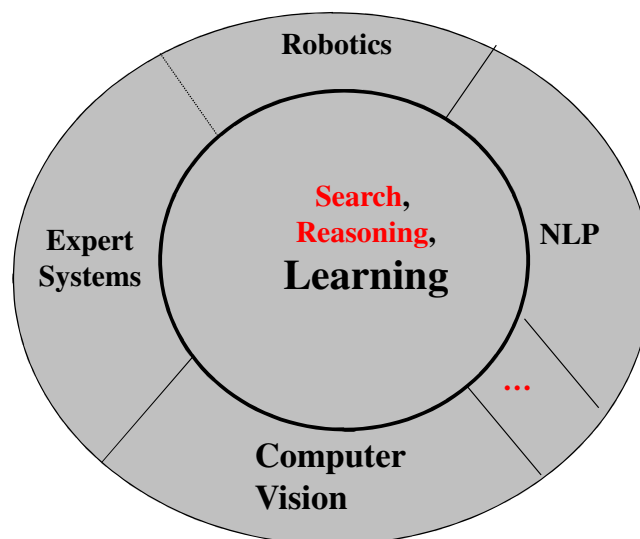
Artificial Intelligence :
Structures and Strategies for Complex Problem Solving
Fifth Edition by George F. Luger
(2005, Addison Wesley)



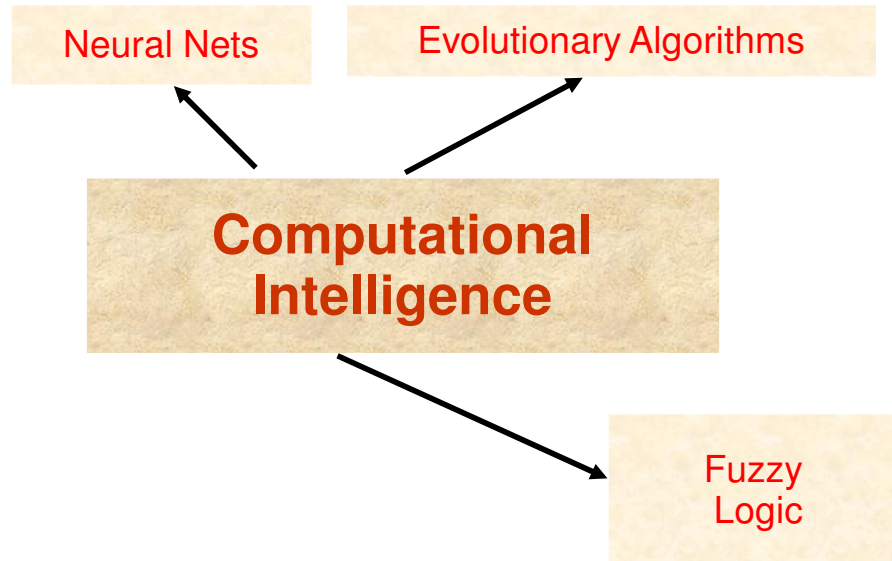
Artificial Intelligence :
A Modern Approach
Second Edition by Stuart J. Russell & Peter Norvig
(2003, Prentice Hall)

1

Recall



"Soft Computing"



Machine Learning (ML)

- ❑ Machine Learning is the study of how to build computer systems that adapt and improve with experience. It is a subfield of Artificial Intelligence and intersects with cognitive science, information theory, and probability theory, ...etc
- ❑ Intelligent agents must be able to manage through the course of their interactions with the world, as well as through the experience of their own internal states and processes.
- ❑ Simon's definition (1983) describes learning as allowing the system to "perform better the second time." : "Learning is any process by which a system improves performance from experience."

Task of ML

- ❑ **Prediction:** To predict the desired output for a given input based on previous input/output pairs. E.g., to predict the value of a stock given other inputs like interest rates etc.
- ❑ **Categorization:** To classify an object into one of several categories based on features of the object. E.g., a spam email based on subject
- ❑ **Clustering:** To organize a group of objects into homogeneous segments. E.g., a satellite image analysis system which groups land areas into forest, urban...etc, for better utilization of natural resources.
- ❑ **Planning:** To generate an optimal sequence of actions to solve a particular problem. E.g., an Automated Vehicle which plans its path to avoid obstacles

Classification

Assign object/event to one of a given finite set of categories:

Medical diagnosis
Credit card applications or transactions
Fraud detection in e-commerce
Worm detection in network packets
Spam filtering in email
Recommended articles in a newspaper
Recommended books, movies, music, or jokes
Financial investments
Spoken words
Handwritten letters

Planning

Performing actions in an environment in order to achieve a goal:

- Playing checkers, chess, or backgammon
- Driving a car or a jeep
- Flying a plane, helicopter, or rocket
- Controlling an elevator
- Controlling a character in a video game
- Controlling a mobile robot

7

Models of Learning

- ❑ Classical AI deals mainly with deductive reasoning, learning represents inductive reasoning.
 - **Deductive reasoning** arrives at answers to queries relating to a particular situation starting from a set of general axioms,
 - **inductive reasoning** arrives at general axioms from a set of particular instances.
- ❑ Classical AI often suffers from the knowledge acquisition problem in real life applications where obtaining and updating the knowledge base is costly and prone to errors.
- ❑ Machine learning serves to solve the knowledge acquisition bottleneck by obtaining the result from data by induction.

Inductive Learning

Simplest way to describe it: learn a **function** (table-tree) from examples

- Ignores prior knowledge
- Assumes a deterministic, observable environment
- Assumes examples are given
- Assumes the "agent" wants to learn the function (for a reason so and so)

Why Machine Learning

❑ Machine learning is important in several real life problem because of the following reasons:

- Some tasks cannot be defined well except by example
- Working environment may not be known at design time
- Explicit knowledge encoding may be difficult and not available
- Environments change over time

❑ learning is widely used in a number of application areas such as:

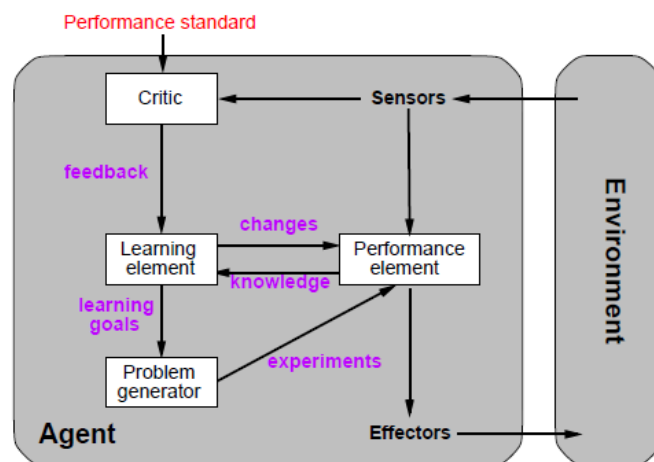
- Data mining and knowledge discovery
- Speech/image/video (pattern) recognition
- Adaptive control
- Autonomous vehicles/robots
- Decision support systems
- Bioinformatics and WWW

Models of Learning

❑ **Induction**, which is learning a generalization from a set of examples, is one of the most fundamental learning tasks.

Different Approaches:

- Symbolic approach
 - Neural Nets
 - genetic and evolutionary learning.
- ❑ Strongest models of learning we have, may be seen in the human and animal systems that have evolved towards equilibration with the world. This approach to learning through adaptation is reflected in **genetic algorithms**, **genetic programming**
- ❑ In the real world this information is often not immediately available AI needs to be able to **learn from experience**



Different kinds of learning

Supervised learning:

Someone gives us examples and the right answer for those examples

We have to predict the right answer for unseen examples

Unsupervised learning:

We see examples but get no feedback

We need to find patterns in the data

Reinforcement learning:

We take actions and get rewards

Have to learn how to get high rewards

Active learning: (?!)

Classification

To lend money to people. We have to predict whether they will pay you back or not. People have various (say, binary) features:

do we know their Address? do they have a Criminal record? high Income?
Educated? Old? Unemployed?

We see examples: (Y = paid back, N = not)

+a, -c, +i, +e, +o, +u: Y

-a, +c, -i, +e, -o, -u: N

+a, -c, +i, -e, -o, -u: Y

-a, -c, +i, +e, -o, -u: Y

-a, +c, +i, -e, -o, -u: N

-a, -c, +i, -e, -o, +u: Y

+a, -c, -i, -e, +o, -u: N

+a, +c, +i, -e, +o, -u: N

Next person is +a, -c, +i, -e, +o, -u. Will we get paid back?

Classification...

We want some hypothesis h that predicts whether we will be paid back

+a, -c, +i, +e, +o, +u: Y
-a, +c, -i, +e, -o, -u: N
+a, -c, +i, -e, -o, -u: Y
-a, -c, +i, +e, -o, -u: Y
-a, +c, +i, -e, -o, -u: N
-a, -c, +i, -e, -o, +u: Y
+a, -c, -i, -e, +o, -u: N
+a, +c, +i, -e, +o, -u: N

Lots of possible hypotheses: will be paid back if...

Income is high (*wrong on 2 occasions in training data*)

Income is high and no Criminal record (*always right in training data*)

(Address is known AND ((NOT Old) OR Unemployed)) OR ((NOT Address is known) AND (NOT Criminal Record)) (*always right in training data*)

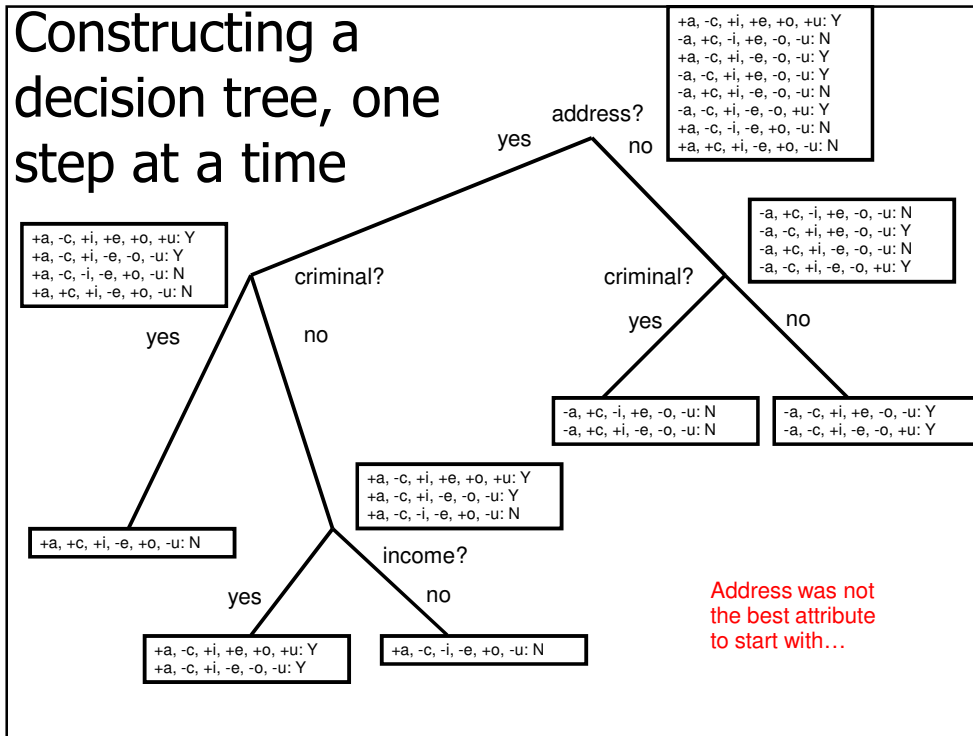
Which one seems best? Anything better?

Occam's Razor

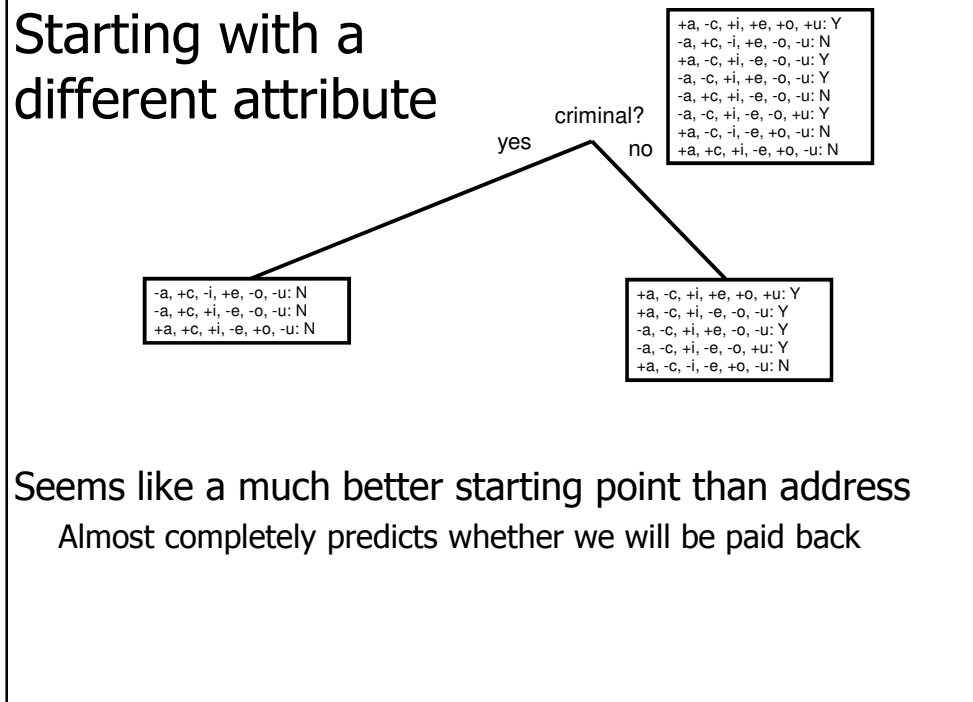
(William o. Occam in 1324)

- preferring simplicity and avoiding unnecessary assumptions.
- we should always accept the simplest answer that correctly fits our data.
- *simpler hypotheses tend to generalize to better data*
- Intuition: given limited training data, it is likely that there is some **complicated** hypothesis that is not actually good but that happens to perform well on the training data
- it is less likely that there is a **simple** hypothesis that is not actually good but that happens to perform well on the training data

Constructing a decision tree, one step at a time



Starting with a different attribute



Decision Tree Learning

- ❑ Decision tree is a class of learning models that are more robust to noise as well as more powerful as compared to concept learning.
- ❑ Decision tree can be seen as rules for performing a categorisation
E.g., “will we be paid back?”
- ❑ We’re learning from examples
Not turning thought processes into decision trees

Decision Tree Learning

- ❑ A decision-tree learning algorithm approximates a target concept using a tree representation, where each internal node corresponds to an attribute, and every terminal node corresponds to a class.
- ❑ Attributes describe examples (background knowledge)
Each attribute takes only a finite set of values
- ❑ They classify instances or examples by starting at the root of the tree and moving through it until a leaf node.

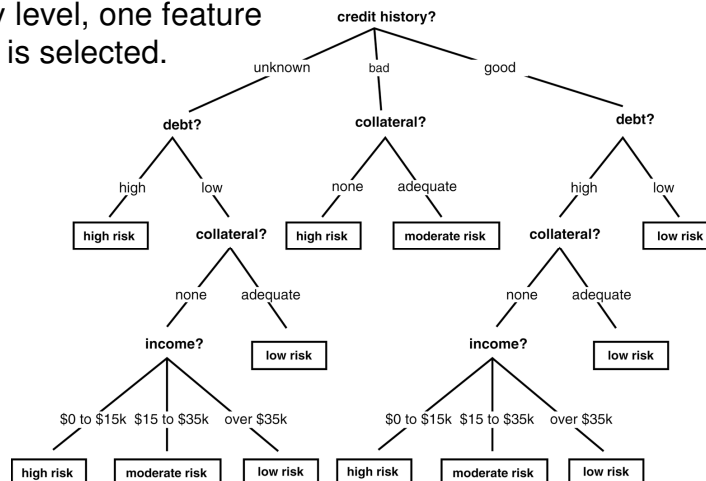
Decision Tree

Example (from the book “Luger”): The problem of estimating credit risk by considering four features of a potential creditor. Such data can be derived from the history of credit applications.

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

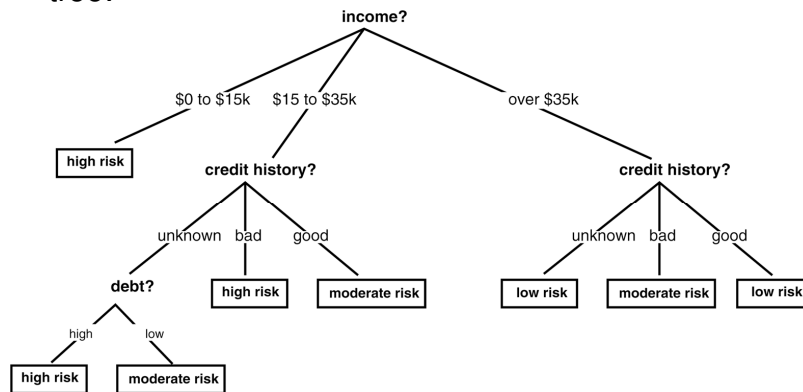
Decision Tree

At every level, one feature value is selected.



Decision Tree

Another possible decision tree:



The ID3 Algorithm

- The major question in decision tree learning Which nodes to put in which positions
- ID3 uses a measure called Information Gain Based on the notion of *entropy*
- Used to choose which node to put in next
- Node with the highest information gain is chosen
- When there are no choices, a leaf node is put on

information gain

measures how well a given attribute separates the training examples according to their target classification

This measure is used to select among the candidate attributes at each step while growing the tree

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

$$\begin{aligned} I(\text{Coin Toss}) &= -p(\text{heads}) \log_2 p(\text{heads}) - p(\text{tails}) \log_2 p(\text{tails}) \\ &= -1/2 \log_2(1/2) - 1/2 \log_2(1/2) \\ &= 1 \text{ bit} \end{aligned}$$

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

$$\begin{aligned} I(\text{Coin Toss}) &= -p(\text{heads}) \log_2 p(\text{heads}) - p(\text{tails}) \log_2 p(\text{tails}) \\ &= -1/2 \log_2(1/2) - 1/2 \log_2(1/2) \\ &= 1 \text{ bit} \end{aligned}$$

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

$$\begin{aligned} I[\text{Coin Toss}] &= -3/4 \log_2(3/4) - 1/4 \log_2(1/4) \\ &= 0.811 \text{ bits} \end{aligned}$$

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: $0 \cdot \log_2(0)$ 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

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

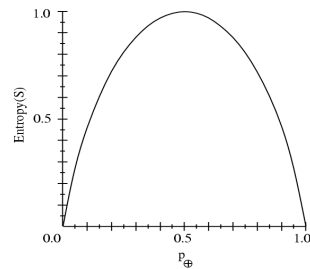
p_i is the probability of class i

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

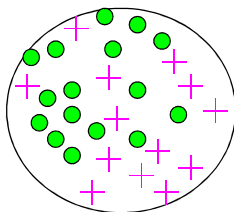


- S is a sample of training examples

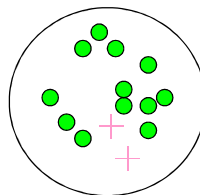
Entropy(S)
positives p_+ approaches 0.5 (very impure),
the Entropy of S converges to 1.0

Entropy – General Idea Impurity

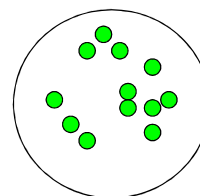
Very impure group



Less impure



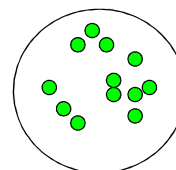
Minimum impurity



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**

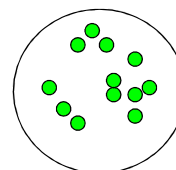


31

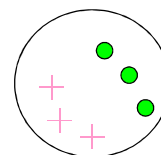
Entropy – General Idea Impurity

- 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**
- 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**

**Minimum
impurity**



**Maximum
impurity**



32

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.

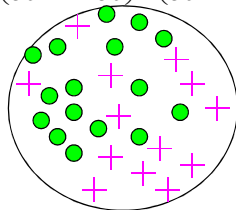
33

Calculating Information Gain

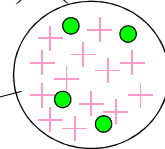
Entire population (30 instances)

Find its entropy

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

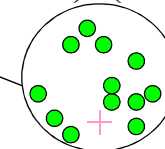


$$-\left(\frac{13}{17} \cdot \log_2 \frac{13}{17}\right) - \left(\frac{4}{17} \cdot \log_2 \frac{4}{17}\right) = 0.787$$



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

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

$$\text{Information Gain} = 0.996 - 0.615 = 0.38$$

34

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:

$$\begin{aligned} Info(S) &= -(6/14) \log_2 (6/14) - \\ &\quad (3/14) \log_2 (3/14) - (5/14) \log_2 (5/14) \\ &\approx 1.531 \text{ bits} \end{aligned}$$

Expected Info.

Let property A(Income) be at the root, and let C_1, \dots, 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:

$$4/14 * 0.0 + 4/14 * 1.0 + 6/14 * 0.65 \approx 0.564 \text{ bits}$$

$$\text{i.e. } Info_A(S) = 0.564$$

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

$$Gain(A) = 1.531 - 0.564 \text{ bits} = 0.967 \text{ bits.}$$

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:

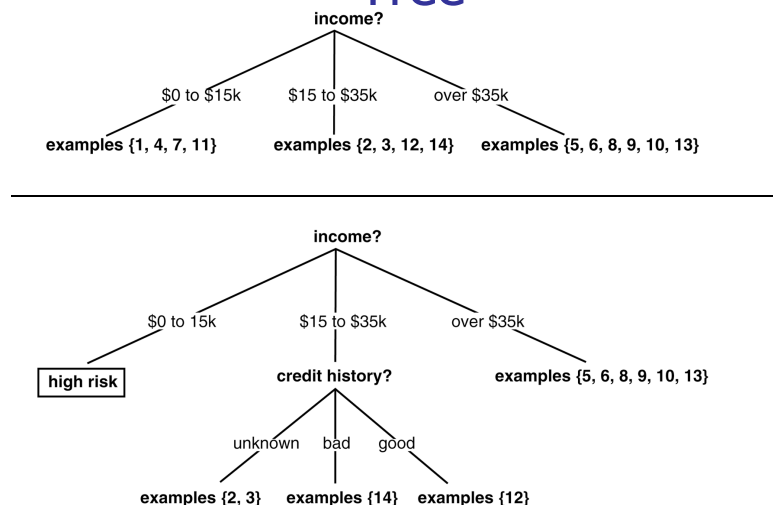
$\text{Gain}(\text{COLLATERAL}) \approx 0.756$ bits,

$\text{Gain}(\text{DEBT}) \approx 0.581$ bits,

$\text{Gain}(\text{CREDIT HISTORY}) \approx 0.266$ 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



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 (IG), 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.

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 information gain

Relative to S , i.e., $\text{gain}(S, A)$ 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 $\text{Gain}(S, A)$

Start again at part 2

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

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.

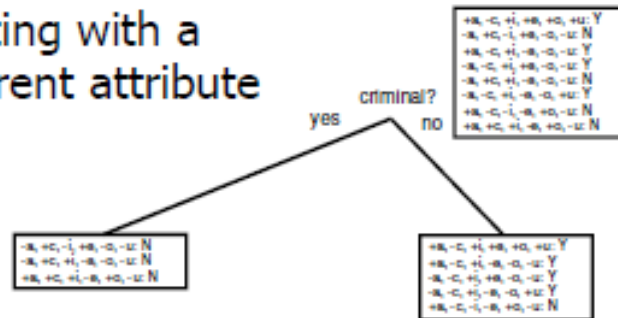
Overfitting the DT

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;

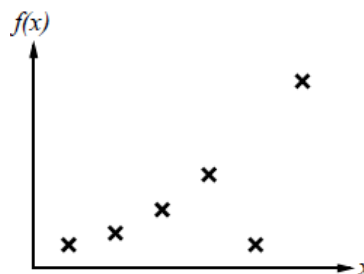
Recall

ing with a
rent attribute



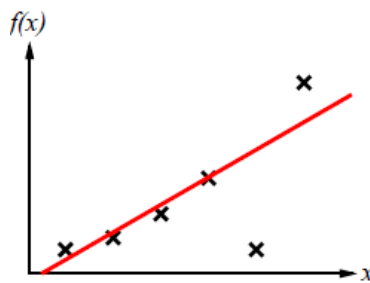
Overfitting in General

Given some points and finding a curve to fit them



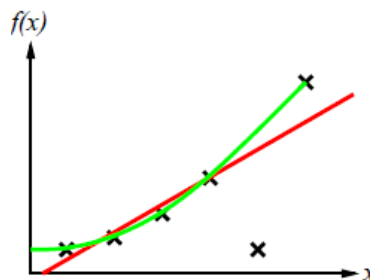
Overfitting in General

Given some points and finding a curve to fit them



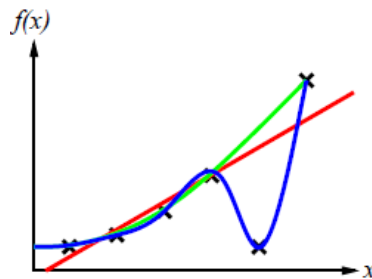
Overfitting in General

Given some points and finding a curve to fit them



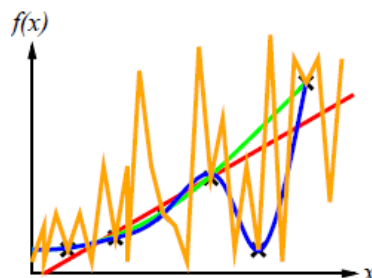
Overfitting in General

Given some points and finding a curve to fit them



Overfitting in General

Given some points and finding a curve to fit them



Ockham's Razor: (Occam))

Maximize a combination of consistency and simplicity

Overfitting the DT

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

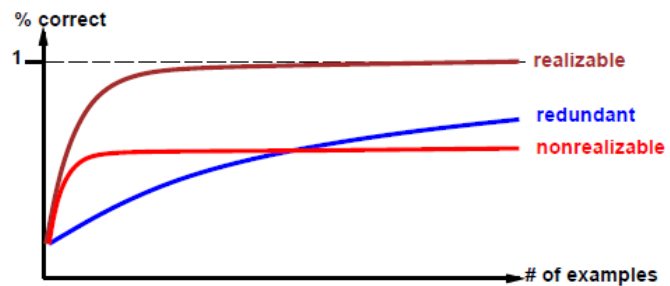
- 1) Stop the tree early before it begins to overfit the data
→ In practice this solution is hard to implement because it is not clear what is a stopping point.
- 2) Grow the tree until the algorithm stops even if the overfitting problem shows ,Then prune the tree.
→ This method has found great popularity 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.

Performance Measure



Practical issues in DT

Practical issues while building a decision tree:

- 1) What happens when attribute values are missing?
- 2) 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)?
- 5) How do we improve the computational efficiency

Missing Data

Missing data: *In many domains, not all the attribute values will be known for every example. The values might have gone unrecorded, or they might be too expensive to obtain. This gives rise to two problems:*

- First, given a complete decision tree, how should one classify an object that is missing one of the test attributes?
- Second, how should one modify the information gain formula when some examples have unknown values for the attribute?

Multi-valued Attributes

Multivalued attributes: *When an attribute has many possible values, the information gain measure gives an inappropriate indication of the attribute's usefulness.*

In the extreme case, we could use an attribute, such as *Name*, that has a different value for every example. Then each subset of examples would be a singleton with a unique classification, so the information gain measure would have its highest value for this attribute. While, the attribute could be irrelevant or useless.

Continuous attributes

Continuous and integer-valued input attributes:

such as Height and Weight, have an infinite set of possible values. Rather than generate infinitely many branches, decision-tree learning algorithms typically find the

split point that gives the highest information gain.

For example, at a given node in the tree, it might be the case that testing on $\text{Weight} > 160$ gives the most information.

Efficient dynamic programming methods exist for finding good split points, but it is still by far the most expensive part of real-world decision tree learning applications.