CHAPTER : FIVE
Learning Agents

Introduction

- Artificial Intelligence (AI) is concerned with programming computers to perform tasks that are presently done better by humans.
- AI is about human behavior, the discovery of techniques that will allow computers to learn from humans.
- One of the most often heard criticisms of AI is that machines cannot be called Intelligent until they are able to learn to do new things and adapt to new situations, rather than simply doing as they are told to do.
- There can be little question that the ability to adapt to new surroundings and to solve new problems is an important characteristic of intelligent entities.

What is Learning?

- In the psychology literature, learning is considered one of the keys to human intelligence.
- *what learning is?* Learning is:
  - Memorizing something
  - Knowing facts through observation and exploration
  - Improving motor and/or cognitive skills through practice
- The idea behind learning is that percepts should not only be used for acting now, but also for improving the agent's ability to act in the future.
  - Learning is essential for unknown environments, i.e. when designer lacks omniscience. It enables to organize new knowledge into general, effective representations
  - Learning modifies the agent's decision making mechanisms to improve performance

Cont…….Learning

- Learning can be described as normally a relatively permanent change that occurs in behavior as a result of experience.
- Learning occurs in various regimes, for example: -
- It is possible to learn to open a lock as a result of trial and error.
- It is possible to learn how to use a word processor as a result of following particular instructions.
- There's no decisive definition of learning, but here are some that do justice: -
- "***Learning denotes changes in a system that ... enables a system to do the same task more efficiently the next time.***" Herbert Simon
- "***Learning is constructing or modifying representations of what is being experienced.***" Ryszard Michalski
- "***Learning is making useful changes in our minds.***" Marvin Minsky
- The idea behind learning is that percepts should be used not only for acting, but also for improving the agent's ability to act in the future.
- Learning takes place as the agent observes its interactions with the world and its own decision-making processes.

Machine Learning

- The goal of machine learning is to build computer systems that can learn from their experience and adapt to their environments.
- Obviously, learning is an important aspect or component of intelligence.

Why Machine Learning?

- One response to the idea of AI is to say that computers cannot think because they only do what their programmers tell them to do.
- However, it is not always easy to tell what a particular program will do, but given the same inputs and conditions it will always produce the same outputs.
- If the program gets something right once, it will always get it right and if it makes a mistake once it will always make the same mistake every time it runs.
- Also, humans are able to notice similarities between things and therefore can generate new ideas about the world we live in.
- So, machine learning is a prerequisite for any mature program of artificial intelligence.

Three Phases of Machine Learning

- Machine Learning follows three phases, these are: -

<u>Phase I: Training</u>

- A training set of examples of correct behavior is analyzed and some representation of the newly learnt knowledge is stored.
- This is often some form of rules.
- ***Phase II: Validation***
- The rules are checked and if necessary, additional training is given. Sometimes additional test data are used.
- But instead of using a human to validate the rules, some other automatic knowledge-based component may be used.
- The role of tester is often called the critic.
- ***Phase III: Application***
- The rules are used in responding to some new situations.

Learning Techniques

Rote Learning

- In this kind of learning there is no prior knowledge.
- When a computer stores a piece of data, it is performing an elementary form of learning.
- Examples of correct behavior are stored and when a new situation arises it is matched with the learnt examples.
- The values are stored so that they are not re-computed later. One of the earliest game-playing programs is Checkers Program (Samuel, 1963). This program learned to play checkers well enough to beat its creator/designer.

Deductive Learning

- Deductive learning works on existing facts and knowledge and deduces new knowledge from the old.
- This is best illustrated in the following example: -
- **Example**: Assume, **A = B** and **B = C**, then we can deduce with much confidence that **C = A**.

- Deductive learning does not generate “new” knowledge at all, it simply memorizes the logical consequences of what is known already.
- This implies that virtually all mathematical research would not be classified as learning “new” things.
- However, regardless of whether this is termed as new knowledge or not, it certainly makes the reasoning system more efficient.

Inductive Learning (Learning from Observations)

- Inductive learning takes examples and generalizes rather than starting with existing knowledge.
- For example, having seen many cats, all of which have tails, one might conclude that all cats have tails.
- There is scope of error in inductive reasoning, but still it is a useful technique that has been used as the basis of several successful systems.

Learning: Symbol-Based

- Our world is a world of symbols.
- We use symbolic interpretations to understand the world around us.
- For instance, if we saw a ship and were to tell a friend about its size, we will not say that we saw a 254 meters long ship, instead we'd say that we saw a "huge" ship about the size of "Our Building".
- And our friend would understand the relationship between the size of the ship and its hugeness with the analogies of the symbolic information associated with the two words used: "huge" and "Our Building".
- Similarly, the techniques we are to learn now use symbols to represent knowledge and information. Let us consider a small example to help us see where we're headed.
- **Example**: What if we were to learn the concept of a ***GOOD STUDENT***.
- We would need to define, first of all some attributes of a student, on the basis of which we could tell apart the good student from the average.

- Then we would require some examples of good students and average students.
- To keep the problem simple, we can label all the students who are “not good” (Average, Below Average, Satisfactory, Bad) as ***NOT GOOD STUDENT***.
- Let’s say we choose two attributes to define a student, ***Grade*** and ***Class Participation***. Both attributes can have either of the two values, ***High*** or ***Low***.
- Now from the above Student concept, the learner program might look like: -
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (High)**
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (Low)**
- Student (NOT GOOD STUDENT): **Grade (Low) $\wedge$ Class Participation (High)**
- Student (NOT GOOD STUDENT): **Grade (Low) $\wedge$ Class Participation (Low)**

- As you can see the system is composed of symbolic information, based on which the learner can even generalize that a student is a GOOD STUDENT if his/her grade is **High**, even if the class participation is **Low**.
- Student (GOOD STUDENT): **Grade (High) $\wedge$ Class Participation (?)**
- This is the final rule that the learner has learnt from the enumerated examples.

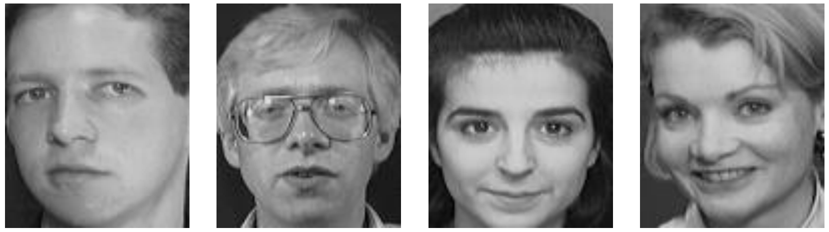
- Here the "**?**" means that the attribute class participation can have any value, as long as the grade is High.

Face Recognition

Learning: it is training (adaptation) from data set

•Training examples of a person –

Test images



Feature extraction

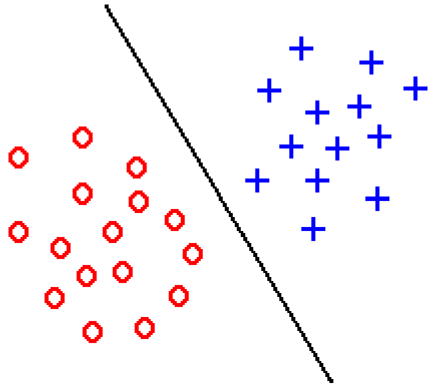
x = Hair length y =0 x = Hair length y = 30 cn

Task: to extract features which are good for classification.

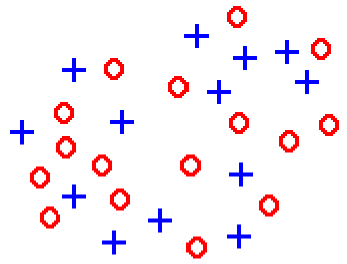
Good features:

•Objects from the same class have similar feature values.

• Objects from different classes have different values.



“Good” features

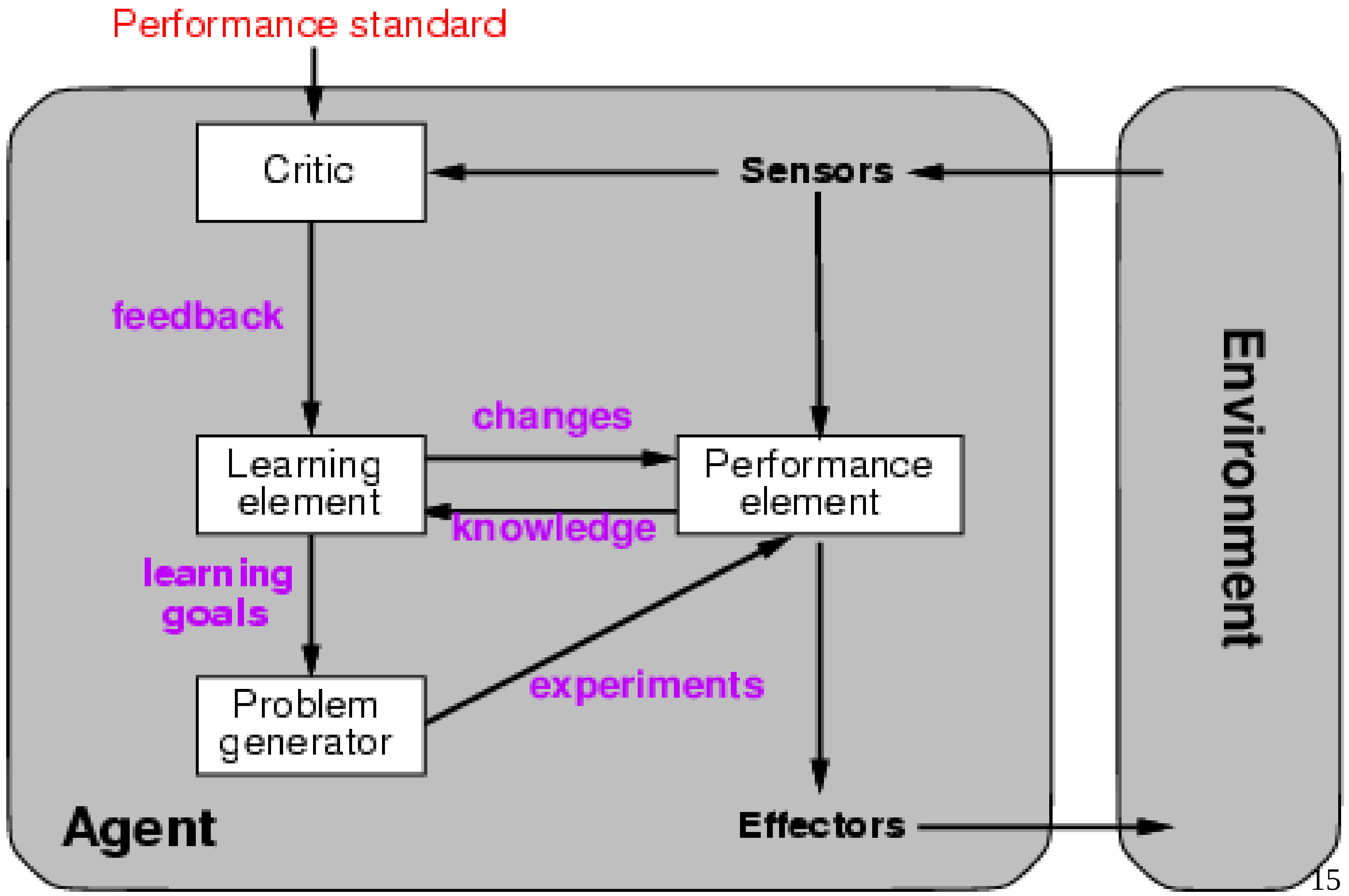


“Bad” features

The Basic Learning Model

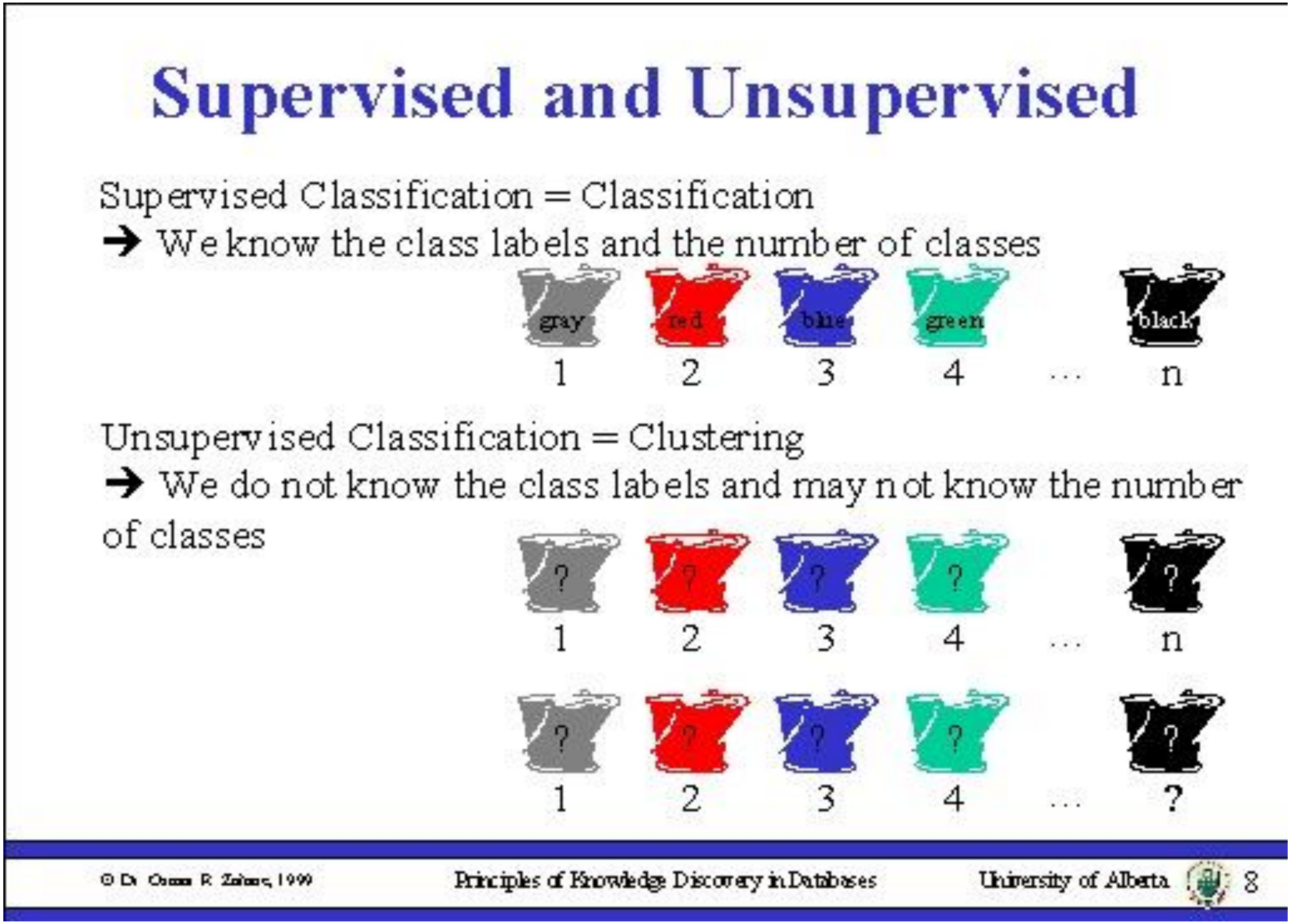
- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P,
  - if its performance at tasks T, as measured by P, improves with experience E.
- Learning agents consist of four main components:
  - **learning element --** the part of the agent responsible for improving its performance
  - **performance element --** the part that chooses the actions to take
  - **critic** – provides feedback for the learning element how the agent is doing with respect to a performance standard
  - **problem generator --** suggests actions that could lead to new, informative experiences (suboptimal from the point of view of the performance element, but designed to improve that element)

Learning Agents



Types of learning

- **Supervised learning:** occurs where a set of input/output pairs are explicitly presented to the agent by a teacher
  - The teacher provides a category label for each pattern in a training set, then the learning algorithm finds a rule that does a good job of predicting the output associated with a new input.
- **Unsupervised learning:** Learning when there is no information about what the correct outputs are.
  - In unsupervised learning or clustering there is no explicit teacher, the system forms clusters or natural groupings of the input patterns.



Supervised and Unsupervised

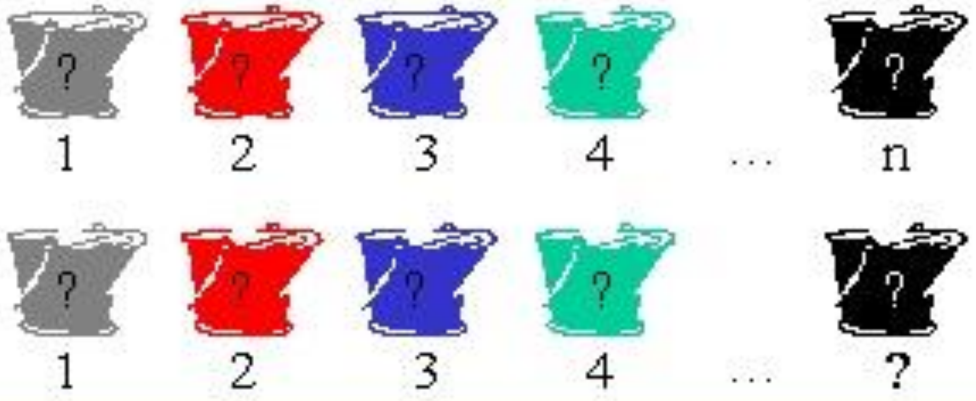
Supervised Classification = Classification

➔ We know the class labels and the number of classes

gray
red
blue
green
black
1
2
3
4
…
n

Unsupervised Classification = Clustering

➔ We do not know the class labels and may not know the number of classes



The Learning Problem

- Given <x,f(x)> pairs, infer f

x	f(x)
1	1
2	4
3	9
4	16
5	?

Given a finite sample, it is often impossible to guess the true function f.

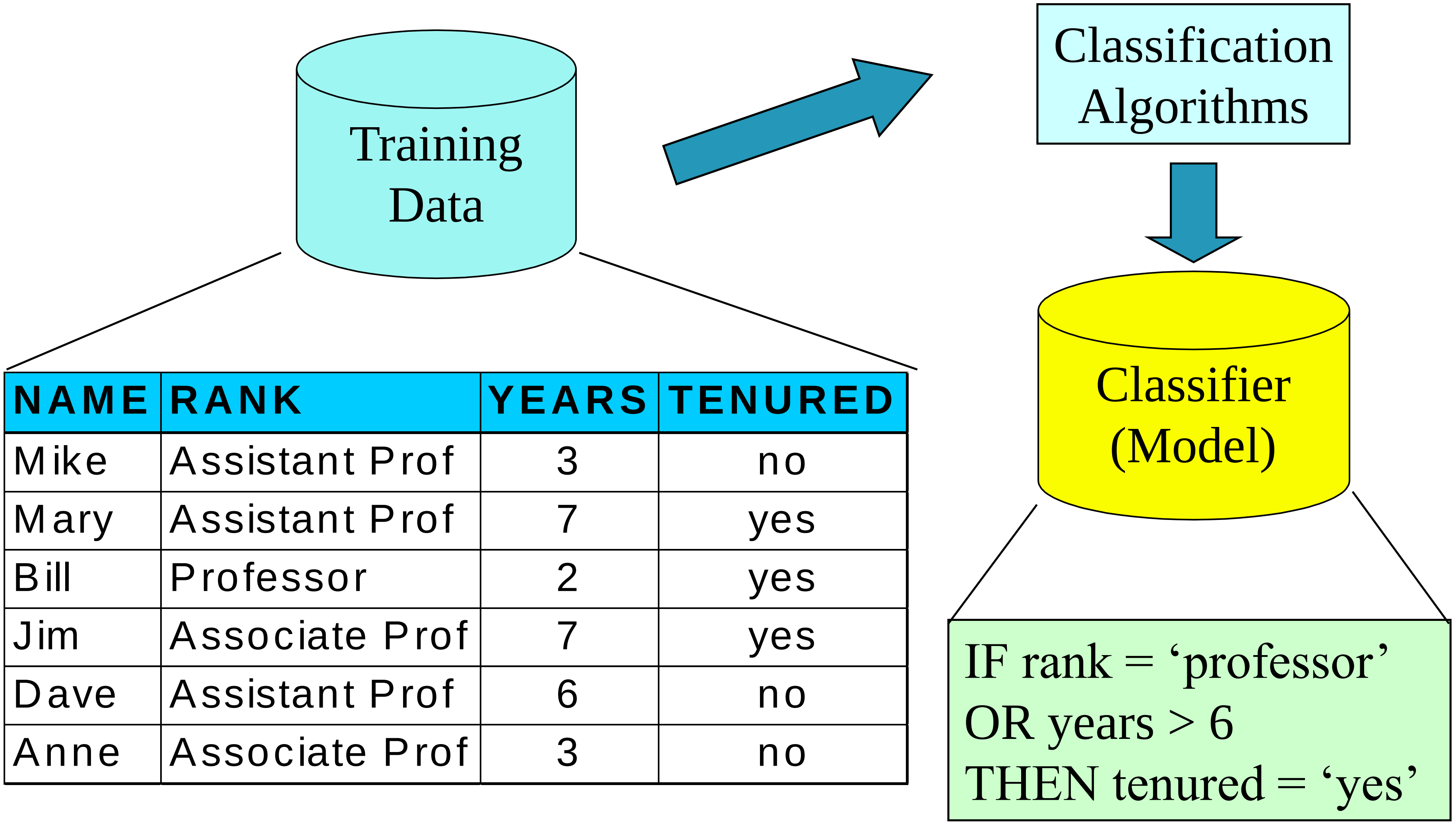
Approach: Find some pattern (called a ***hypothesis***) in the training examples, and assume that the pattern will hold for future examples too.

Data sets preparation for learning

- **Training set**
  - Used in supervised learning, a training set is a set of problem instances (described as a set of properties and their values), together with a classification of the instance.
- **Test set**
  - A set of instances and their classifications used to test the accuracy of a learned hypothesis.

- Learning—A Two-Step Process
  - **Model construction:** The training set is used to create the model. The model is represented as classification rules, decision trees, or mathematical formulae
  - **Model usage:** the test set is used to see how well it works for classifying future or unknown objects

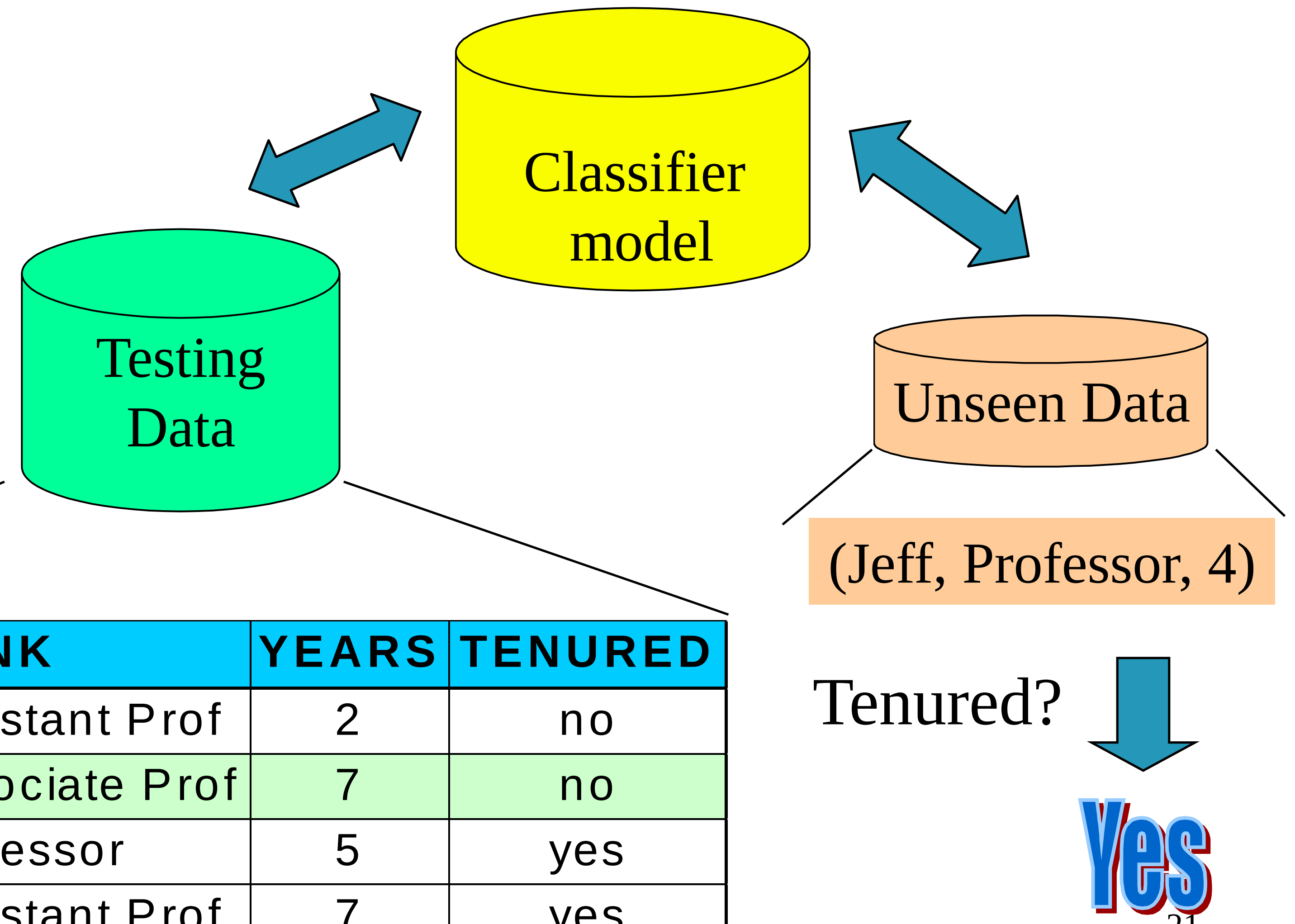
Step 1: Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Step 2: Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Learning methods

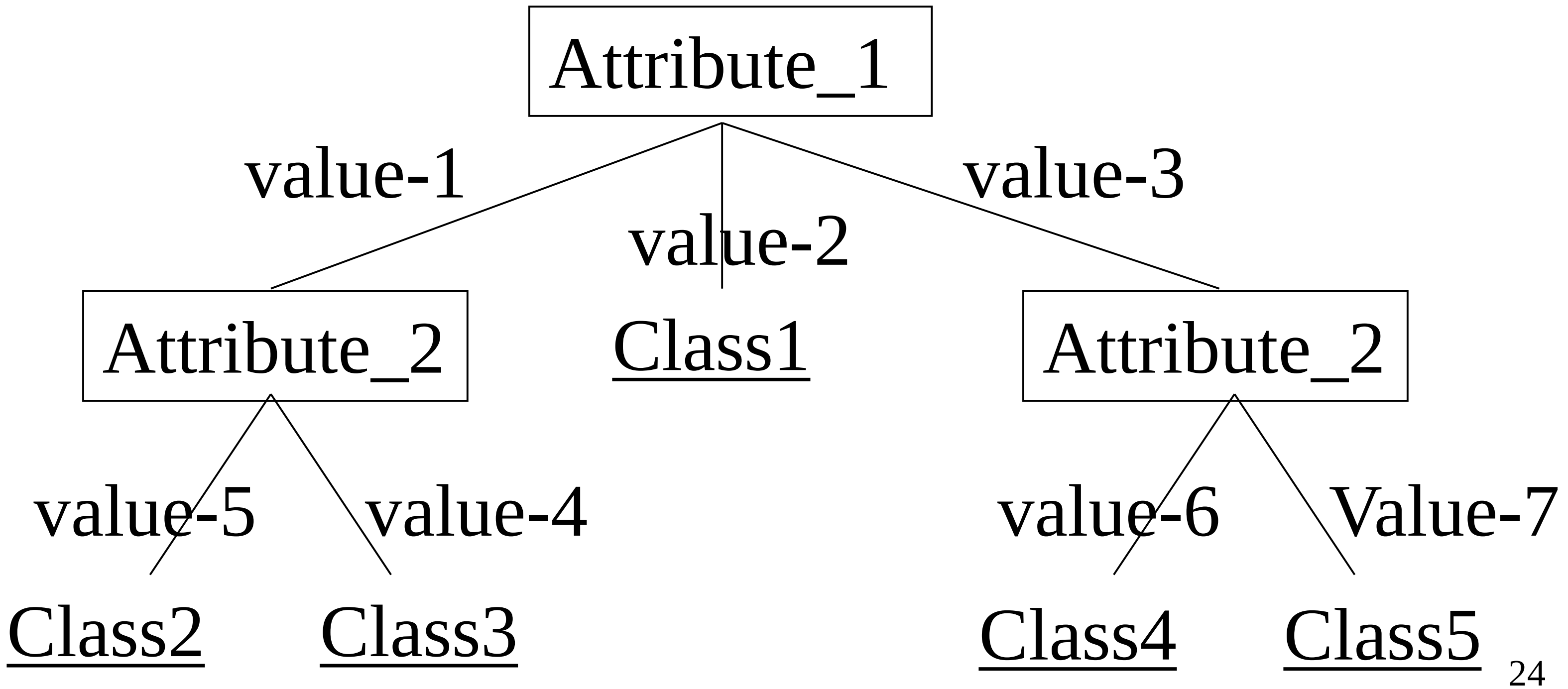
- There are various learning methods. Popular learning techniques include the following.
  - Decision tree : divide decision space into piecewise constant regions.
  - Neural networks: partition by non-linear boundaries
  - Bayesian network: a probabilistic model
  - Regression: (linear or any other polynomial)
  - Support vector machine
  - Expectation maximization algorithm

Decision tree

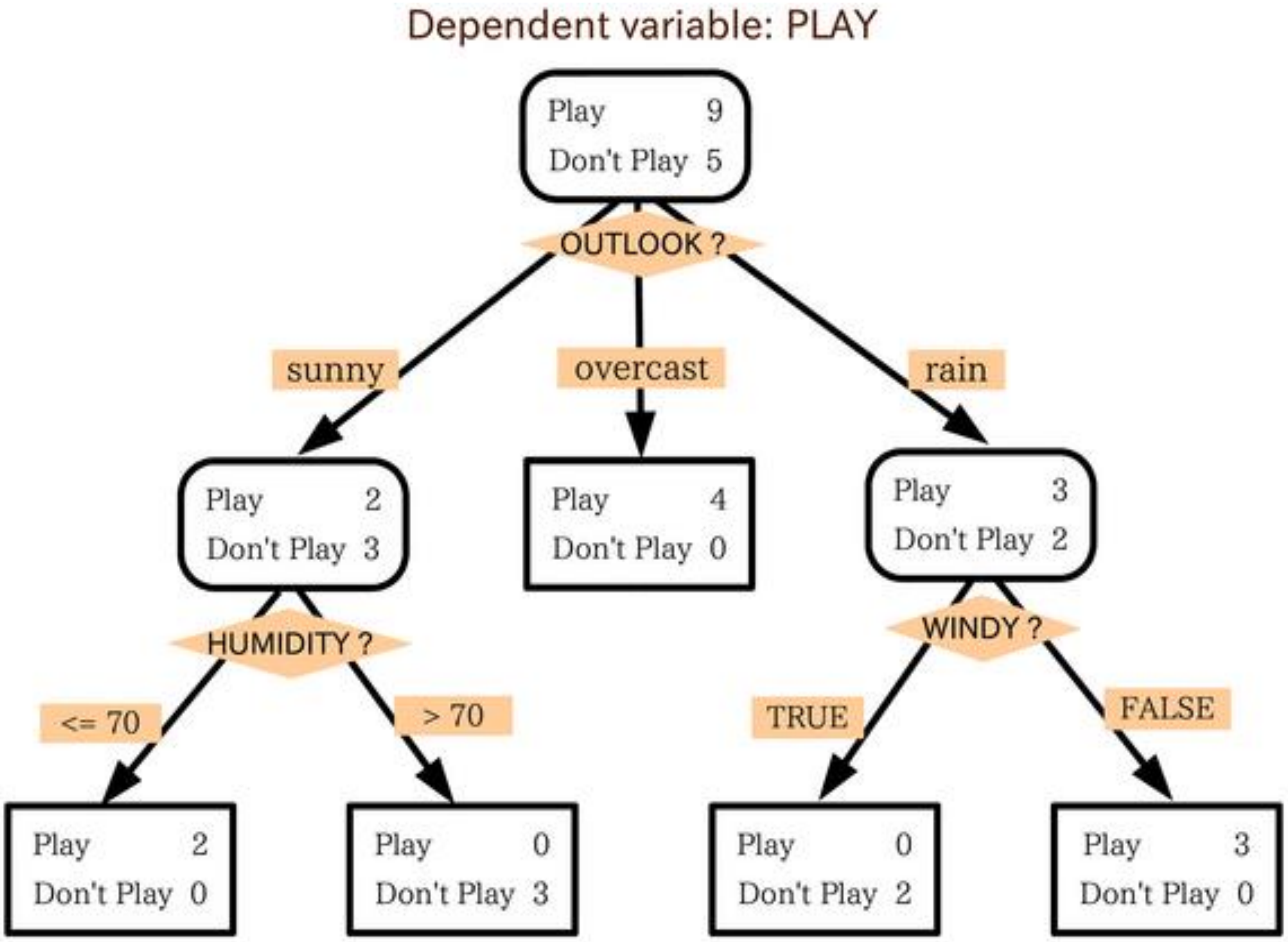
- *Decision tree* performs classification by constructing a tree based on training instances with leaves having class labels.
  - The tree is traversed for each test instance to find a leaf, and the class of the leaf is the predicted class. This is a directed knowledge discovery in the sense that there is a specific field whose value we want to predict.
- Widely used learning method. It has been applied to:
  - classify medical patients based on the disease,
  - equipment malfunction by cause,
  - loan applicant by likelihood of payment.
- Easy to interpret: can be re-represented as if-then-else rules
- Does not require any prior knowledge of data distribution, works well on noisy data.

Decision Trees

- Tree where internal nodes are simple decision rules on one or more attributes and leaf nodes are predicted class labels; i.e. a Boolean classifier for the input instance.
  - ✓ Given an instance of an object or situation, which is specified by a set of properties, the tree returns a "yes" or "no" decision about that instance.



Decision trees



Example 1: The problem of "Sunburn"

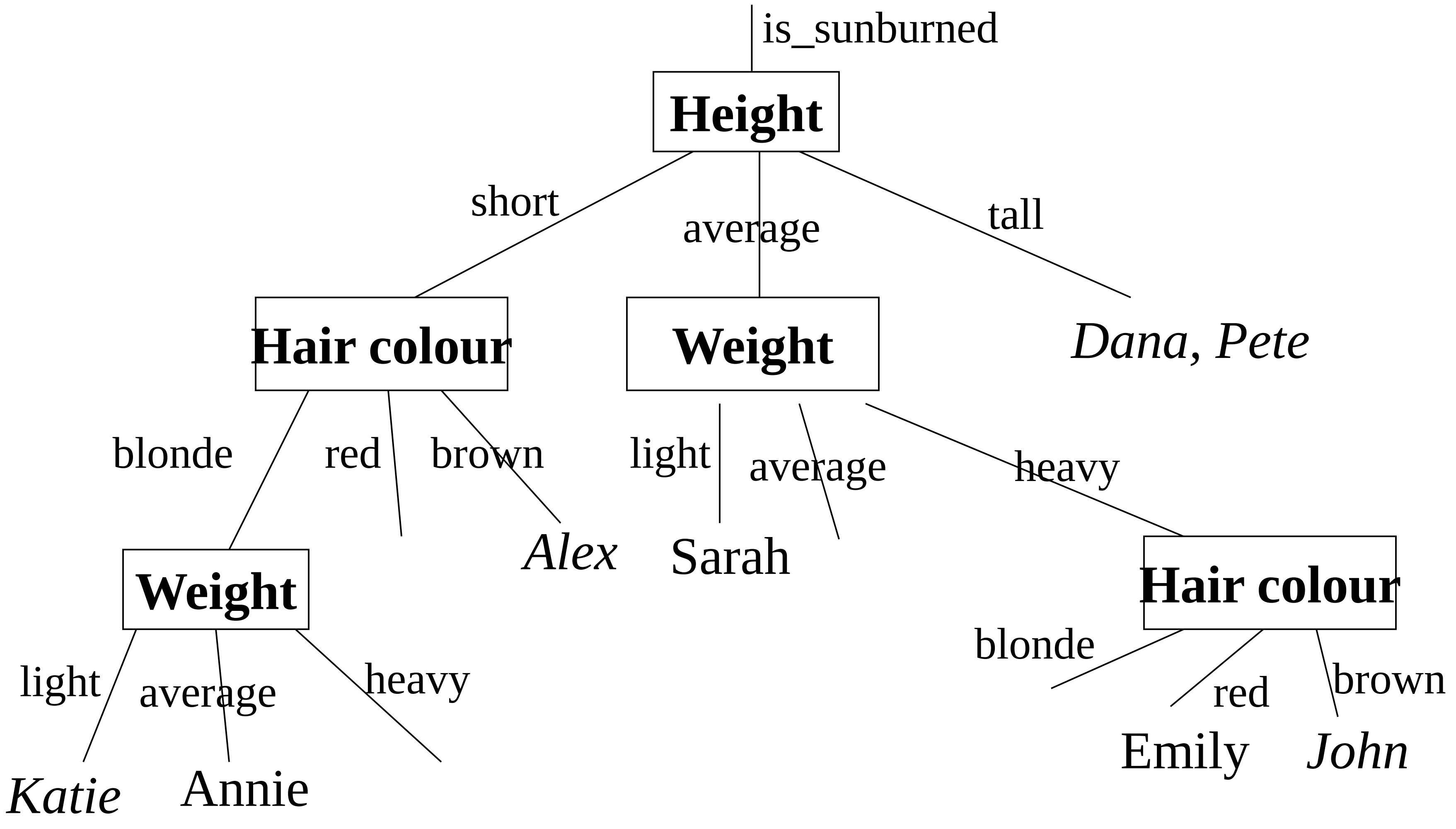
- You want to predict whether another person is likely to get sunburned if he is back to the beach. How can you do this?
- Data Collected: predict based on the observed properties of the people

Name	Hair	Height	Weight	Lotion	class
Sarah	Blonde	Average	Light	No	Sunburned
Dana	Blonde	Tall	Average	Yes	*None*
Alex	Brown	Short	Average	Yes	*None*
Annie	Blonde	Short	Average	No	Sunburned
Emily	Red	Average	Heavy	No	Sunburned
Pete	Brown	Tall	Heavy	No	*None*
John	Brown	Average	Heavy	No	*None*
Kate	Blonde	Short	Light	Yes	*None*

- You might hope that the new person would be an exact match to one of the examples in the table. What are the chances of this?
- Actually there are 3 x 3 x 3x 2 = 54 possible combinations of attributes. Eight examples so chance of an exact match for a randomly chosen example is 8/54=0.15
- Not good. In general there would be many more factors or attributes and each might take many values.

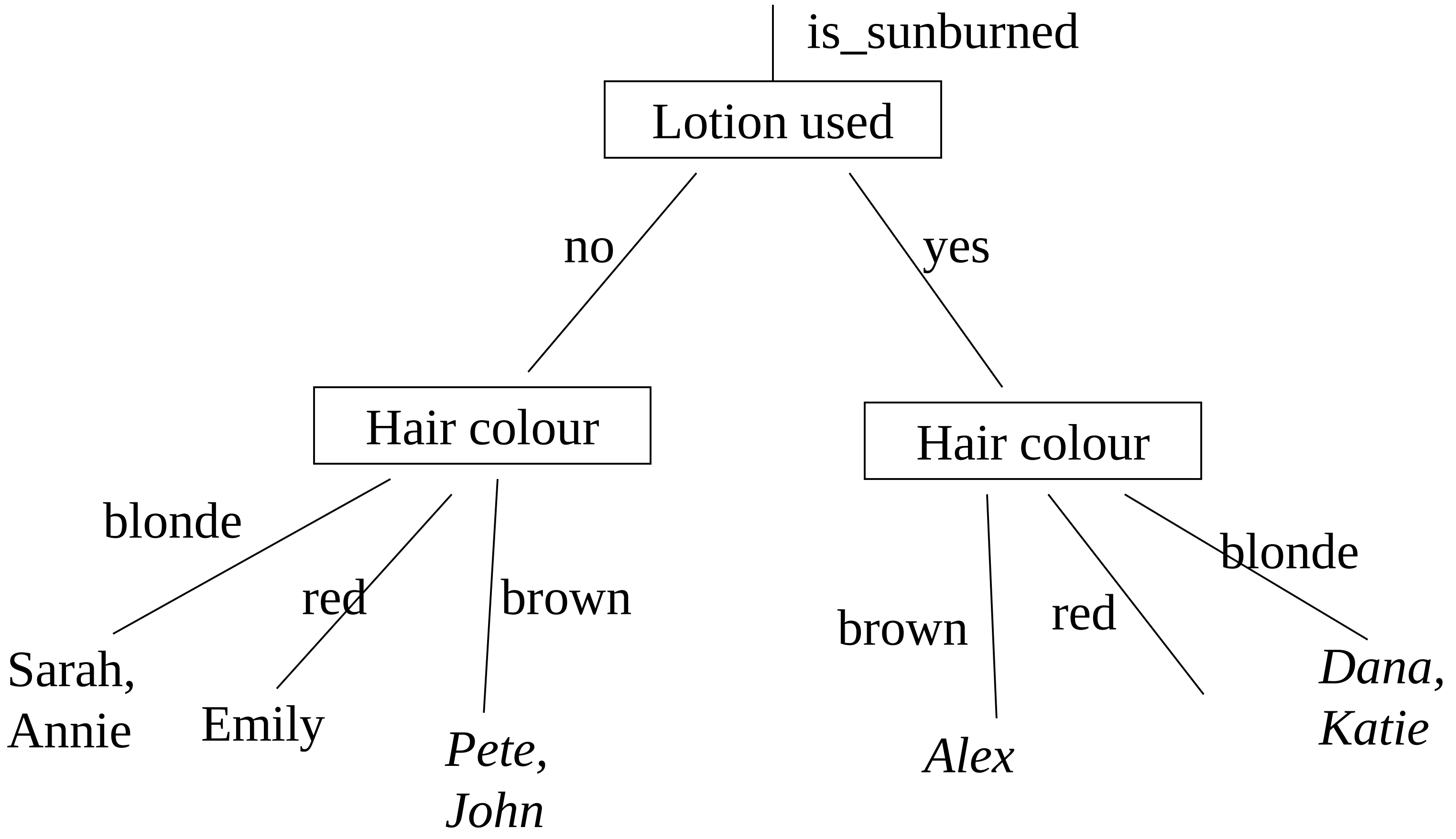
Decision Tree 1

- This is one way of looking at the data, starting with 'Height' property. The decision tree correctly CLASSIFIES all the data!



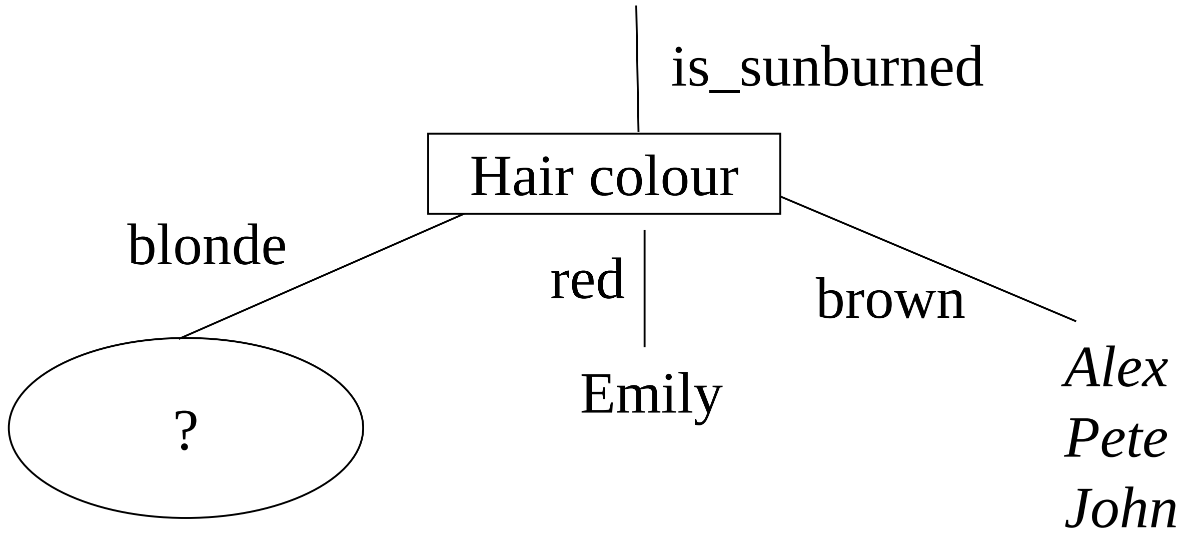
Decision Tree 2

- Is there any attributes you don’t like to use for predicting whether people will suffer from sunburn? Of course it isn’t reasonable to use height.



- This decision tree is a lot better than the first one and it doesn’t involve any of the irrelevant attributes. Unfortunately though it uses the hair colour attribute twice which doesn’t seem so efficient.

The best decision tree?



Sunburned = Sarah, Annie,

None = *Dana*, *Katie*

- Once we have finished with hair colour we then need to calculate the remaining branches of the decision tree.
- Which attributes is better to classify the remaining ?

Once we have finished with hair colour we then need to calculate the remaining branches of the decision tree.

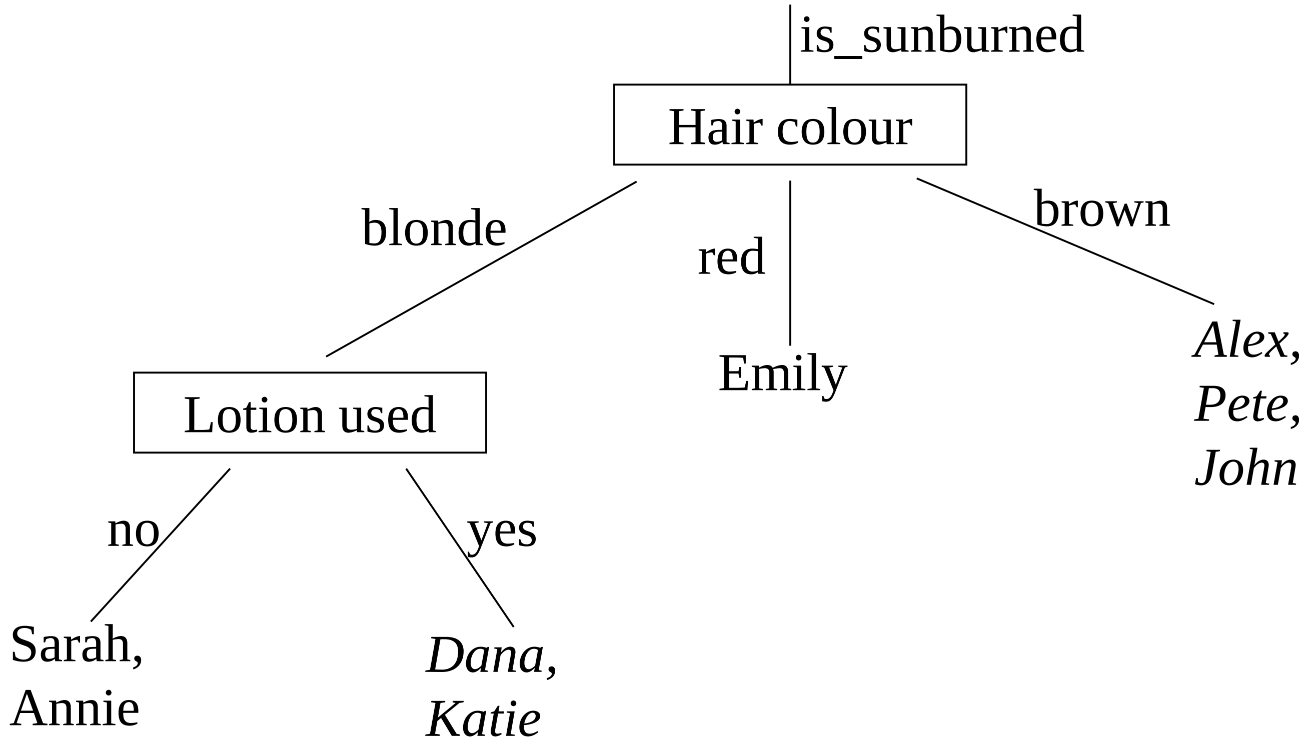
The examples corresponding to that branch are now the total set and no reference is made to the whole set of examples.

One just applies the same procedure as before with JUST those examples (i.e. the blondes).

This is left as an exercise for you on the exercise sheet.

The best Decision Tree

- This is the simplest and optimal one possible and it makes a lot of sense.
- It classifies 4 of the people on just the hair colour alone.



Sunburn sufferers are ...

- You can view Decision Tree as an IF-THEN_ELSE statement which tells us whether someone will suffer from sunburn.

```
 If (Hair-Colour="red") then
                 return (sunburned = yes)
else if (hair-colour="blonde" and lotion-used="No") then
                 return (sunburned = yes)
else
                 return (false)
```

- Is there anything you don’t like about this this program for predicting whether people will suffer from sunburn?
- It is all perfectly reasonable?
- Of course it isn’t reasonable at all. We KNOW that height is IRRELEVANT for determining whether someone will suffer from sunburn.
- Well, what I mean is that there are much MORE relevant decision variables to use.
- Now I want you to draw a new decision diagram that makes a bit more sense. I ‘ll give you five minutes.

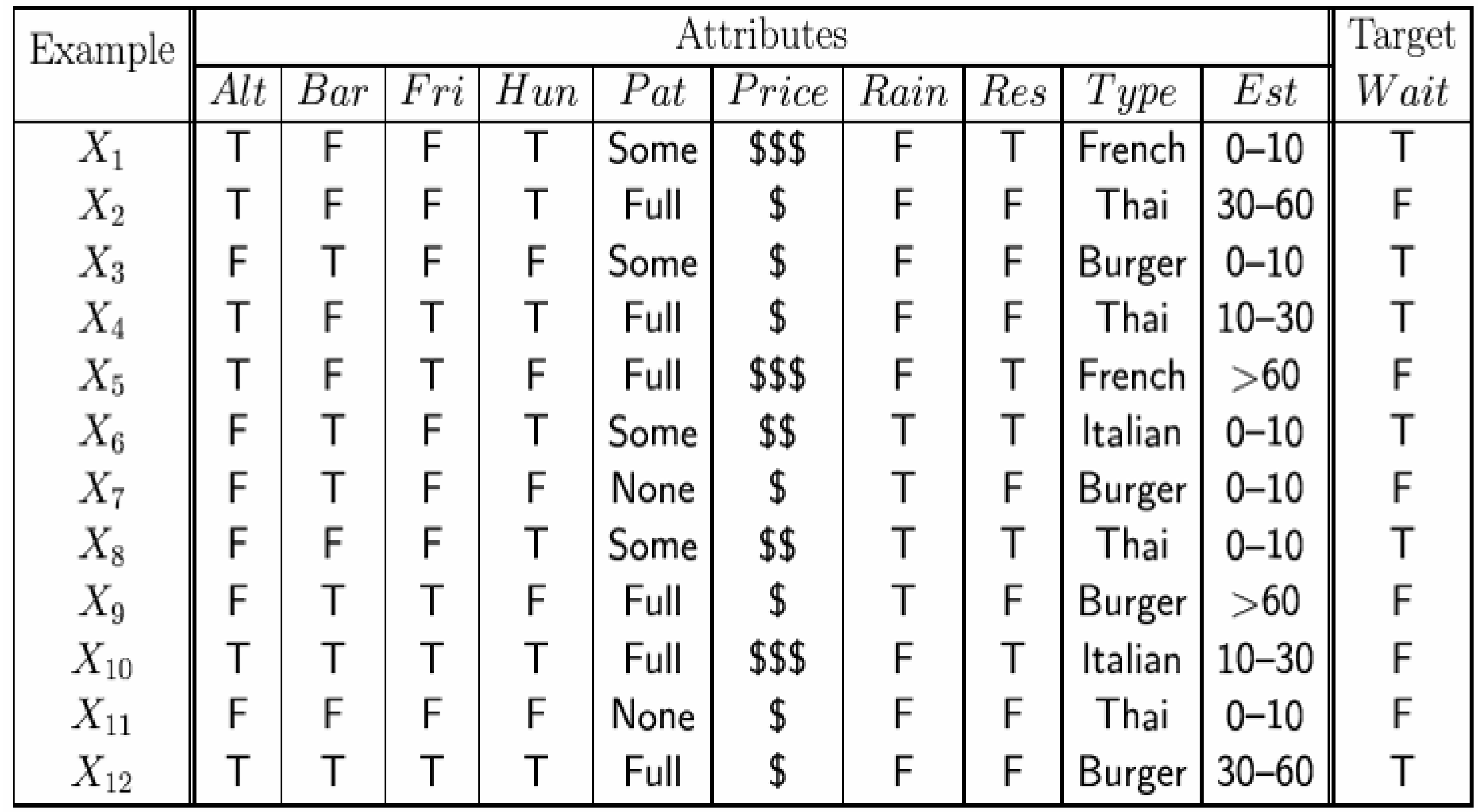
Example: Wait for a Table at a Restaurant

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

- Examples described by **attribute values** (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

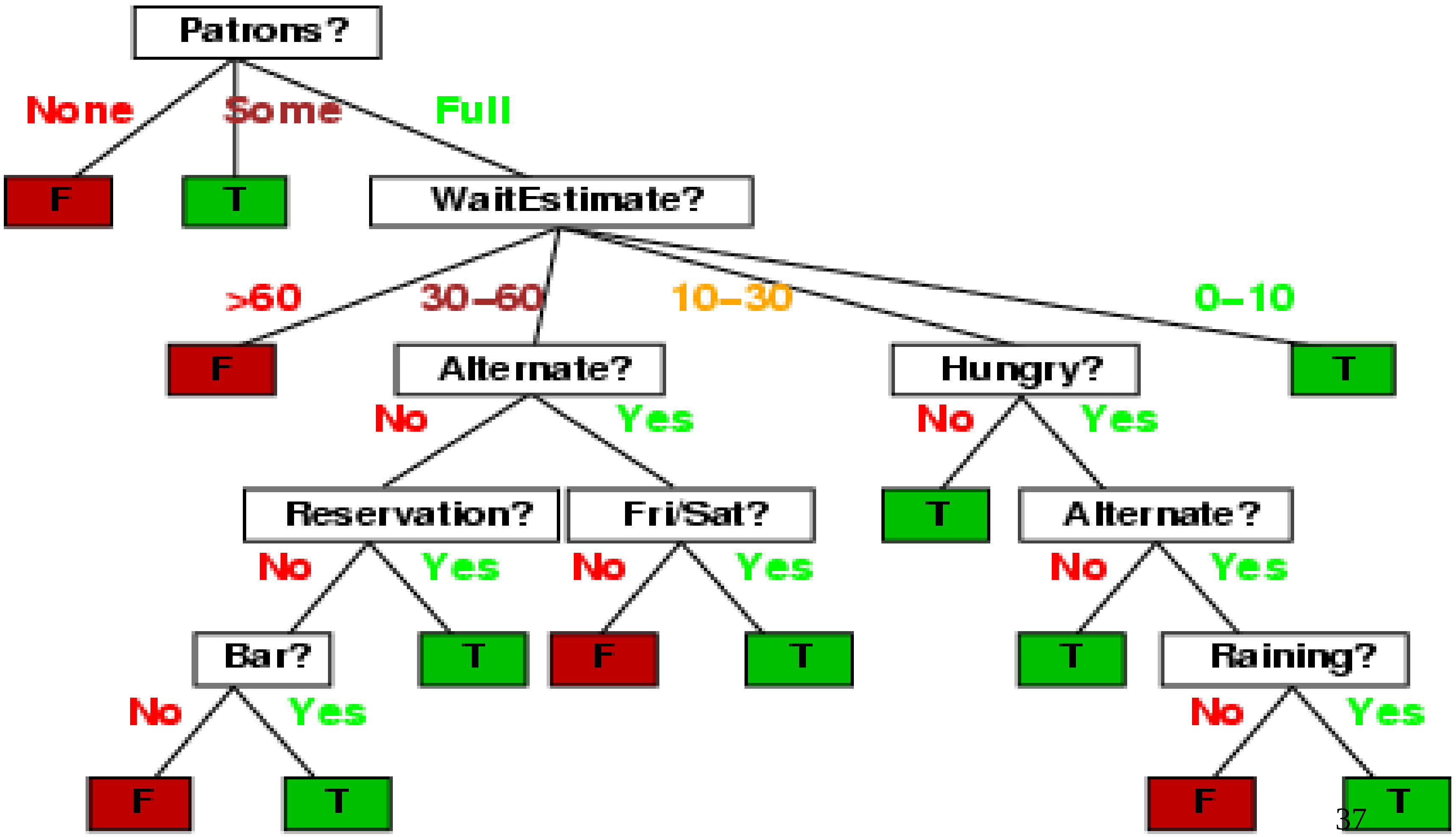


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

- **Classification** of examples is **positiv**e (T) or **negative** (F)

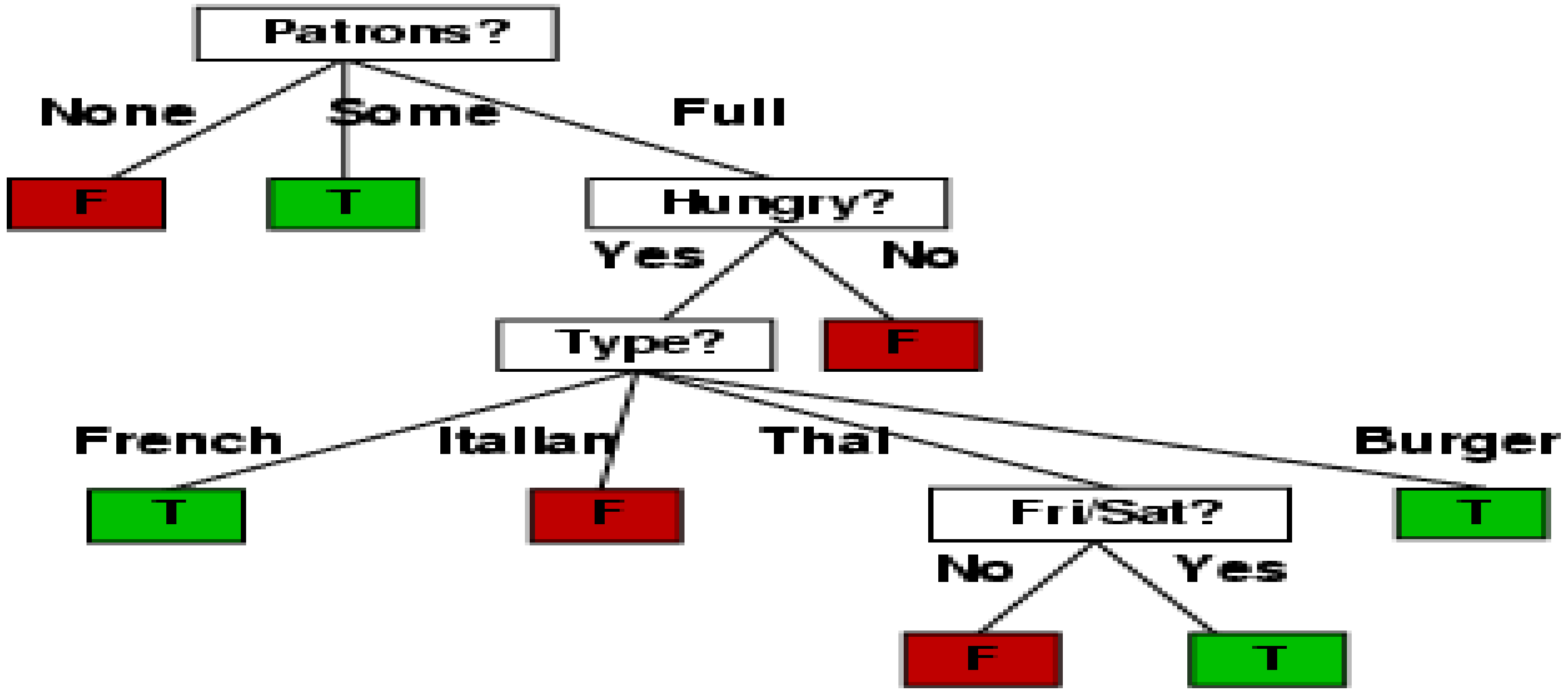
Learning Decision trees 1

- One possible representation for hypotheses
- E.g., here is the "true" tree for deciding whether to wait:



Learning decision trees 2

- Decision tree learned from the 12 examples:

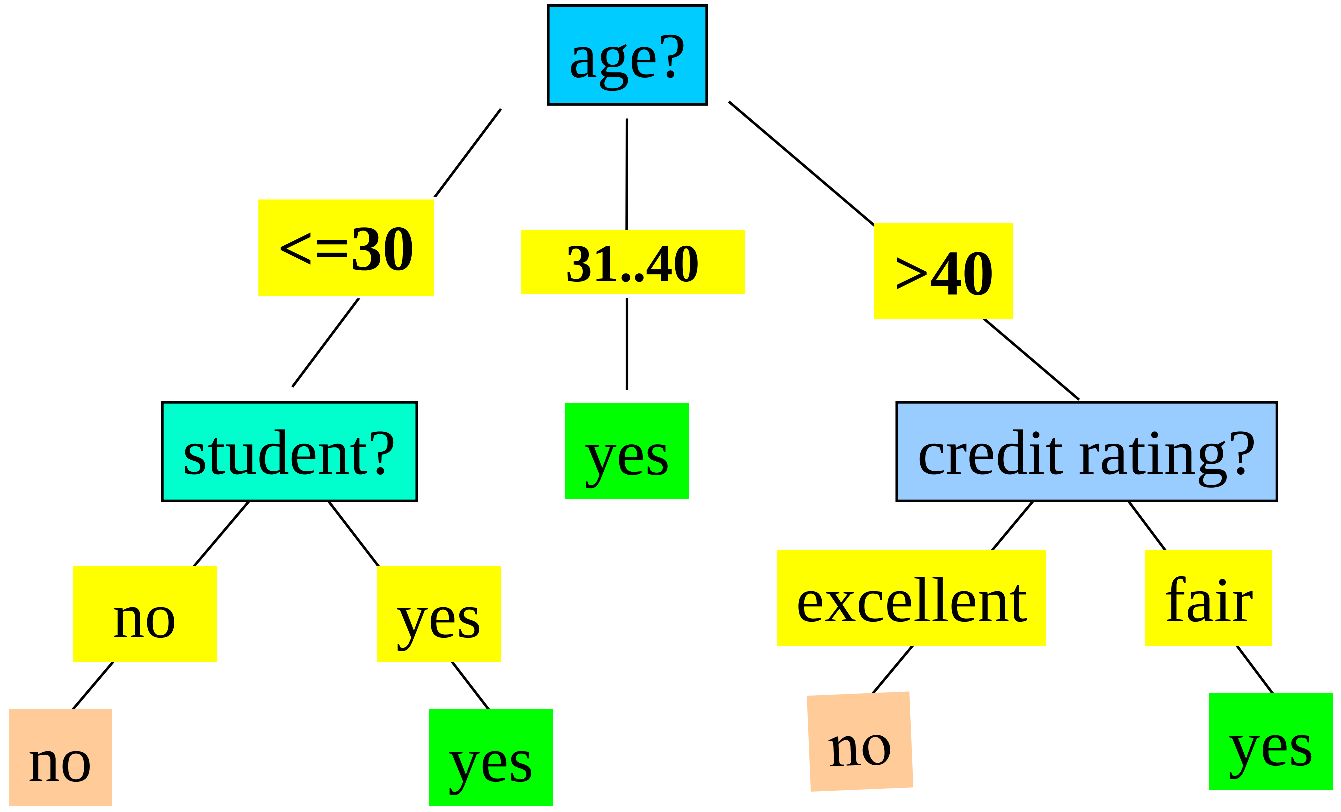


- Substantially simpler than "true" tree---a more complex hypothesis isn't justified by small amount of data

Exercise: Decision Tree for “buy computer or not”. Use the training Dataset given below to construct decision tree

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Pros and Cons of decision trees

- **Pros**

+ Reasonable training time
+ Fast application
+ Easy to interpret
+ Easy to implement
+ Can handle large number of features

- **Cons**

- Cannot handle complicated relationship between features
- simple decision boundaries
- problems with lots of missing data

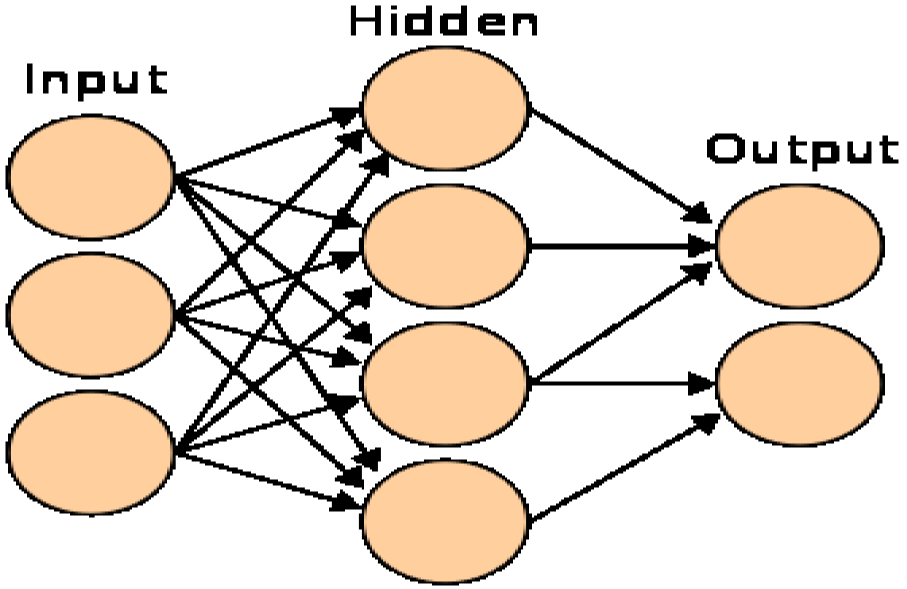
Why decision tree induction in data mining?

- relatively faster learning speed (than other classification methods)
- convertible to simple and easy to understand classification rules
- comparable classification accuracy with other methods

Neural Network

- It is represented as a layered set of interconnected processors. These processor nodes has a relationship with the neurons of the brain. Each node has a weighted connection to several other nodes in adjacent layers. Individual nodes take the input received from connected nodes and use the weights together to compute output values.

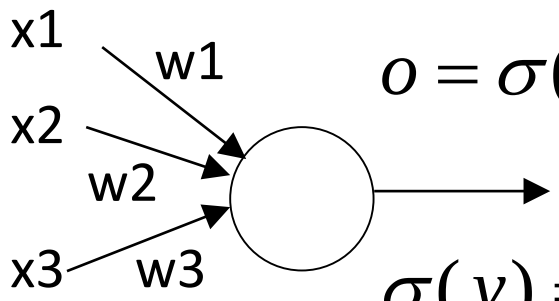
- The inputs are fed simultaneously into the input layer.
- The weighted outputs of these units are fed into hidden layer.
- The weighted outputs of the last hidden layer are inputs to units making up the output layer.



Architecture of Neural network

- Neural networks are used to look for patterns in data, learn these patterns, and then classify new patterns & make forecasts
- A network with the input and output layer only is called single-layered network. Whereas, a multilayer neural network is a generalized one with one or more hidden layer.
  - A network containing two hidden layers is called a three-layer neural network, and so on.

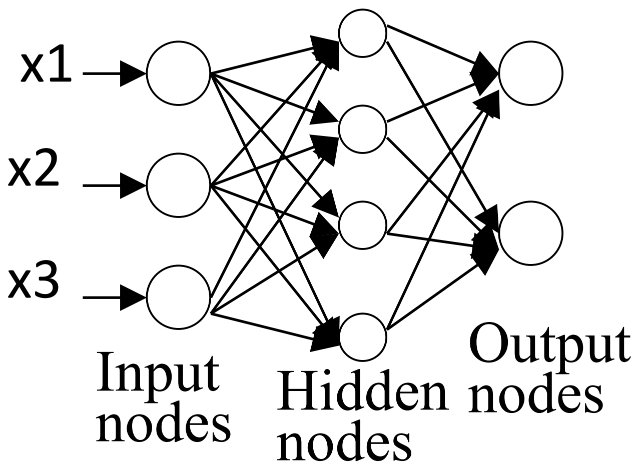
Single layered NN



$$o = \sigma\left(\sum_{i=1}^{n} w_i x_i\right)$$

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

Multilayer NN



Hidden layer: Neuron with Activation

- The neuron is the basic information processing unit of a NN. It consists of:

  1 A set of links, describing the neuron inputs, with weights $W_1, W_2, \ldots, W_m$

  2. An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$\boxed{y = \sum_{j=1}^{m} w_j x_j}$$

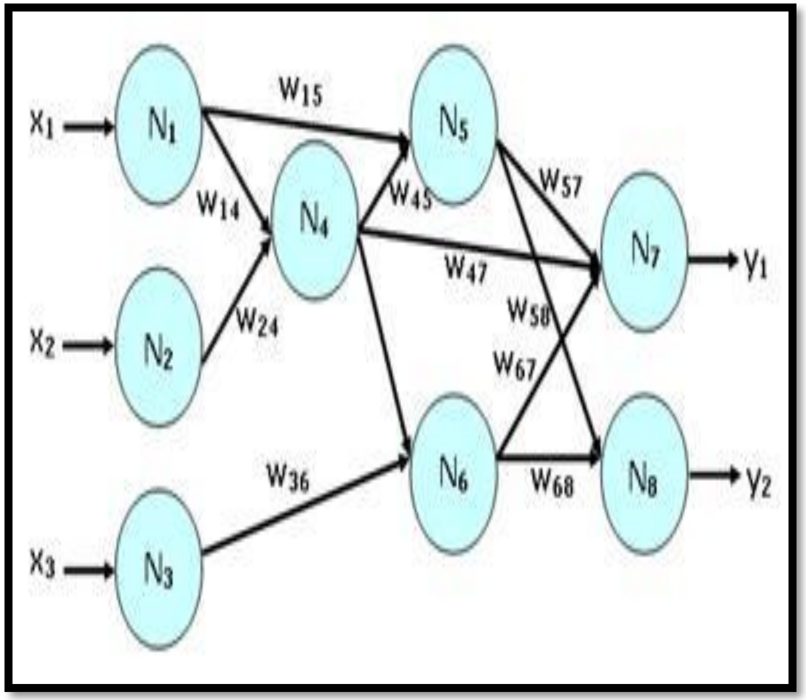
  3. Activation function : for limiting the output behavior of the neuron.

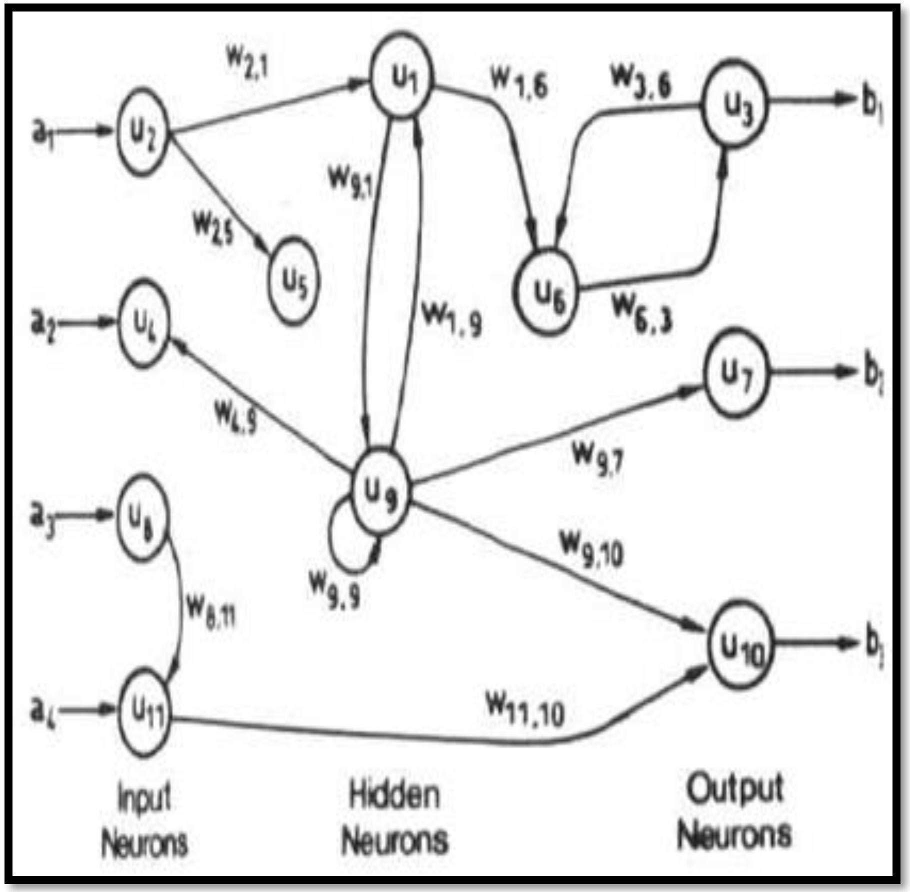
$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

Topologies of neural network

- In a feed forward neural network connections between the units do not form a directed cycle.
  - In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.
- *In recurrent networks* data circulates back and forth until the activation of the units is stabilized
  - Recurrent networks have a feedback loop where data can be fed back into the input at some point before it is fed forward again for further processing and final output.
- In a feed forward NN the data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.

Topologies of neural network

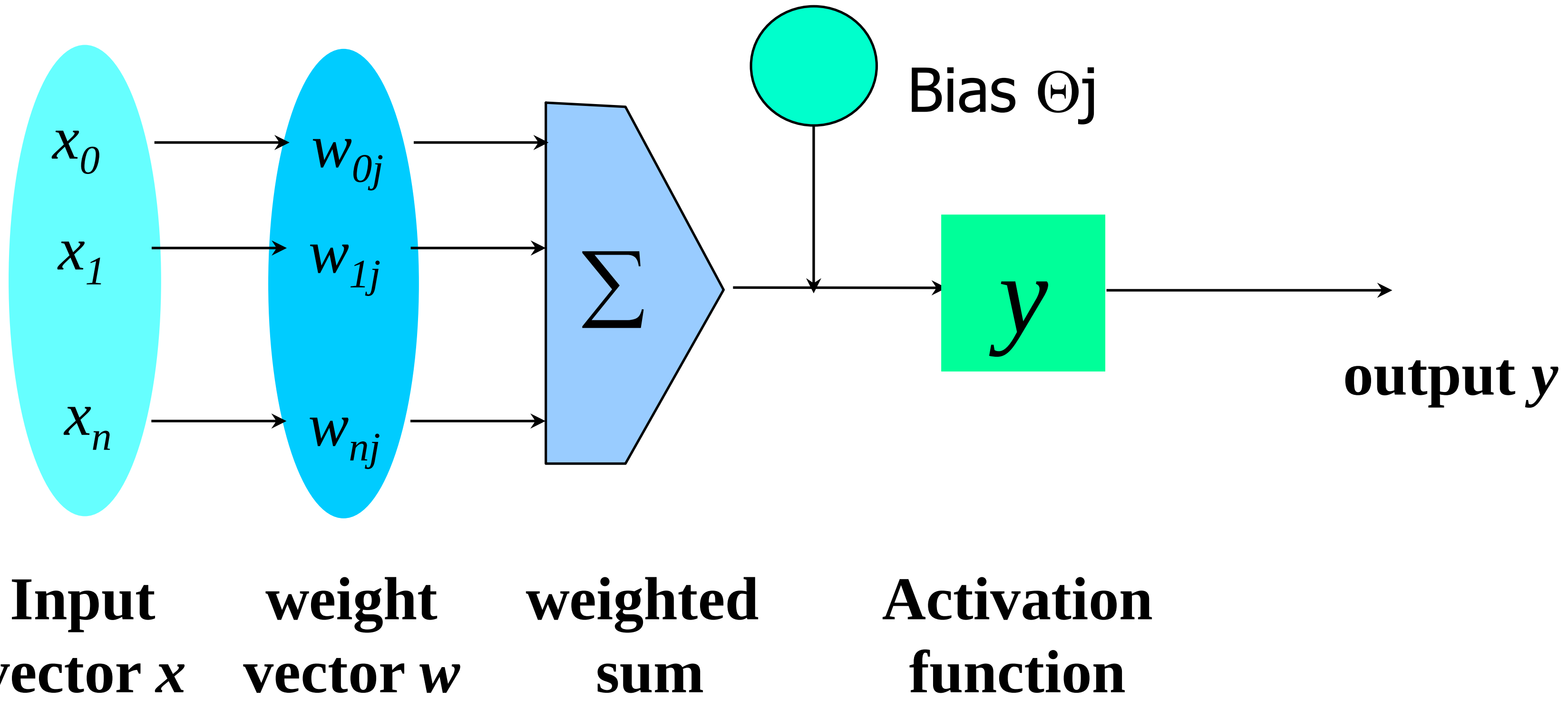




Training the neural network

- *Back Propagation* is the most commonly used method for training multilayer feed forward NN.
  - *Back propagation* learns by iteratively processing a set of training data (samples).
  - For each sample, weights are modified to minimize the error between the desired output and the actual output.
- After propagating an input through the network, the error is calculated and the error is propagated back through the network while the weights are adjusted in order to make the error smaller.

Propagation through Hidden Layer



- The inputs to unit j are outputs from the previous layer. These are multiplied by their corresponding weights in order to form a weighted sum, which is added to the bias associated with unit j.
- A nonlinear activation function f is applied to the net input.

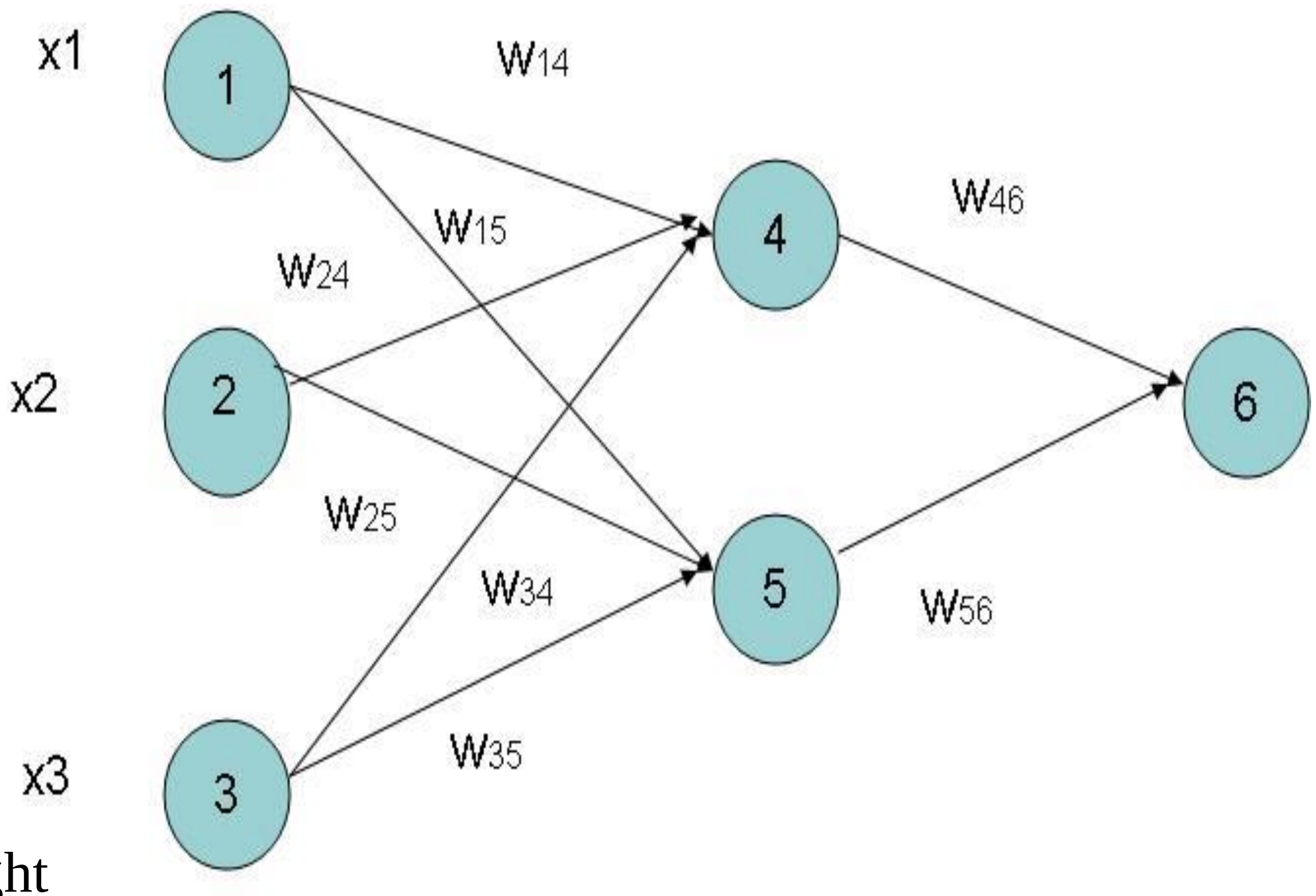
Steps in Back propagation Algorithm

- STEP ONE: initialize the weights and biases.
  - The weights in the network are initialized to random numbers from the interval [-1,1].
  - Each unit has a BIAS associated with it
  - The biases are similarly initialized to random numbers from the interval [-1,1].
- STEP TWO: feed the training sample.
- STEP THREE: Propagate the inputs forward; we compute the net input and output of each unit in the hidden and output layers.
- STEP FOUR: back propagate the error.
- STEP FIVE: update weights and biases to reflect the propagated errors.
- STEP SIX: terminating conditions.

Example of Back propagation

Input = 3, Hidden Neuron = 2, Output =1

Initialize weights :
Random Numbers from -1.0 to 1.0



Initial Input and weight

x1	x2	x3	w14	w15	w24	w25	w34	w35	w46	w56
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.1

Pros and Cons of Neural Network

- Useful for learning complex data like handwriting, speech and image recognition

Pros

+ Can learn more complicated class boundaries

+ Fast application

+ Can handle large number of features

Cons

- Slow training time

- Hard to interpret

- Hard to implement: trial and error for choosing number of nodes

- Neural Network needs long time for training.
- Neural Network has a high tolerance to noisy and incomplete data

- Conclusion: Use neural nets only if decision-trees fail.