Master of Technology in Knowledge Engineering

Unit 7:

Developing Intelligent Systems for Performing Business Analytics

Introduction & Intelligent Techniques Review

Dr. Zhu Fangming
Institute of Systems Science
National University of Singapore
E-mail: isszfm@nus.edu.sg

Developing Intelligent Systems

- **It is a task of system engineering that needs joint efforts and integrated concepts, ideas, techniques, and approaches from multiple closely related disciplines**
  - » **Knowledge engineering**
    - ♦ Knowledge acquisition & representation, Modeling, Verification and validation, …
  - » **Business analytics**
    - ♦ Data analytics, Predictive modelling, Decision & optimization, …
  - » **Software engineering**
    - ♦ System design, Implementation, Testing, Maintenance, …

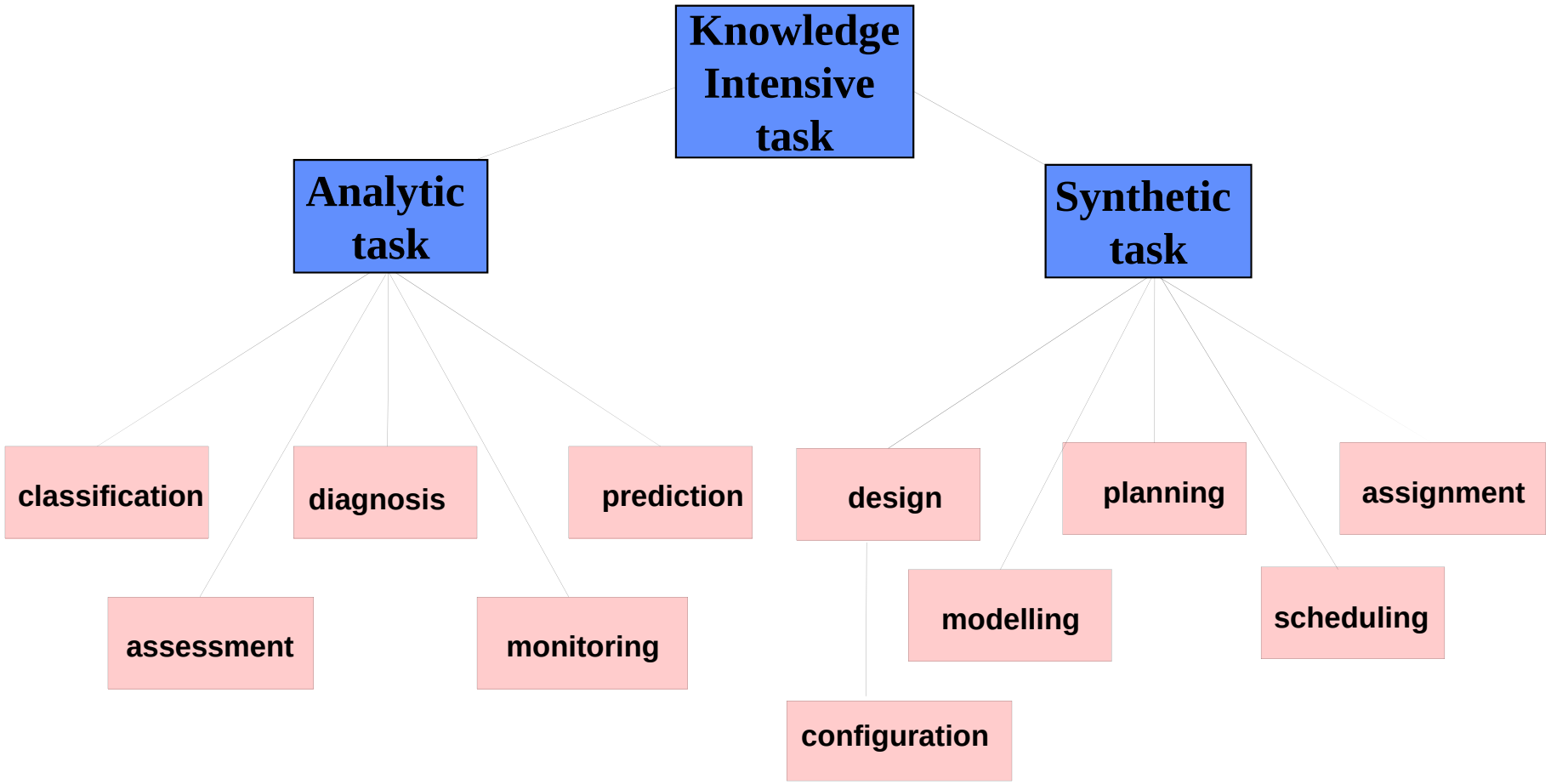
and also other related areas, such as

- » **decision science, risk management, ...**

Types of Tasks for Intelligent Systems

- **Analytic tasks**
  - » System to be analysed pre-exists, but usually not completely "known"
  - » Input: some data about the system (e.g. patient symptoms)
  - » Output: some characterization of the system (e.g. cause of illness)
- **Synthetic tasks**
  - » System does not yet exist
  - » Input: requirements about system to be constructed
  - » Output: constructed system description

Task Hierarchy



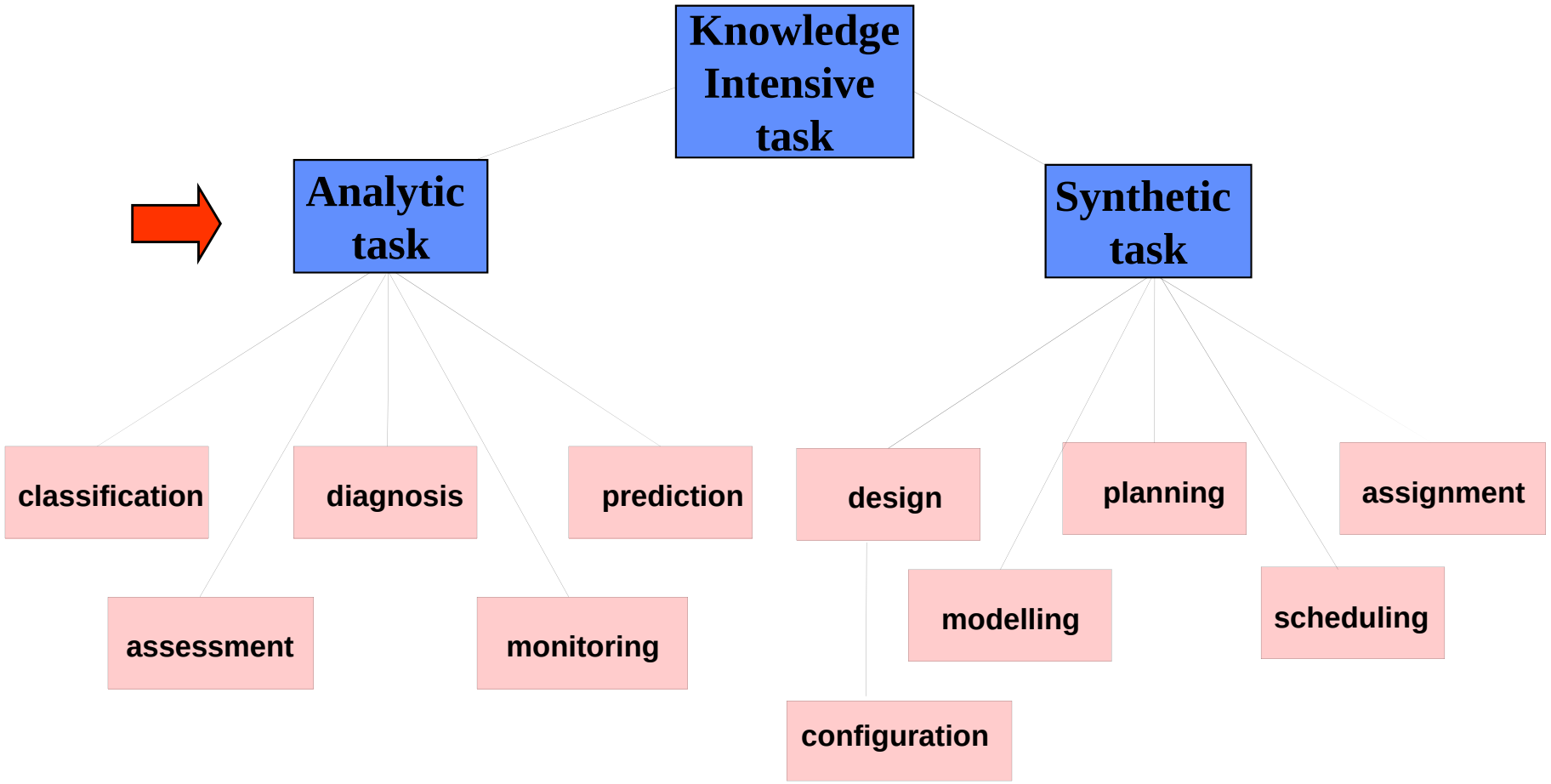
Problem Solving

- **Analytic Tasks**
  - Identification, Classification, Prediction, Clustering/Grouping, …
- **Techniques**
  - **Heuristic rules**
  - **Decision trees**
  - **Fuzzy Systems**
  - **Neural Nets**
  - **Rule Induction**
  - **CBR**
  - **…..**

Problem Solving…

- **Synthetic Tasks**
  - » Planning, Scheduling, Optimisation, Design, …
- **Techniques**
  - » **Brute Force Search**
  - » **Heuristic Search**
  - » **GA's**
  - » **Constraint Propagation**
  - » **Neural Networks (some recurrent architectures)**
  - » **Simulated Annealing**
  - » **Mathematical Modelling**
  - » **….**

Task Hierarchy

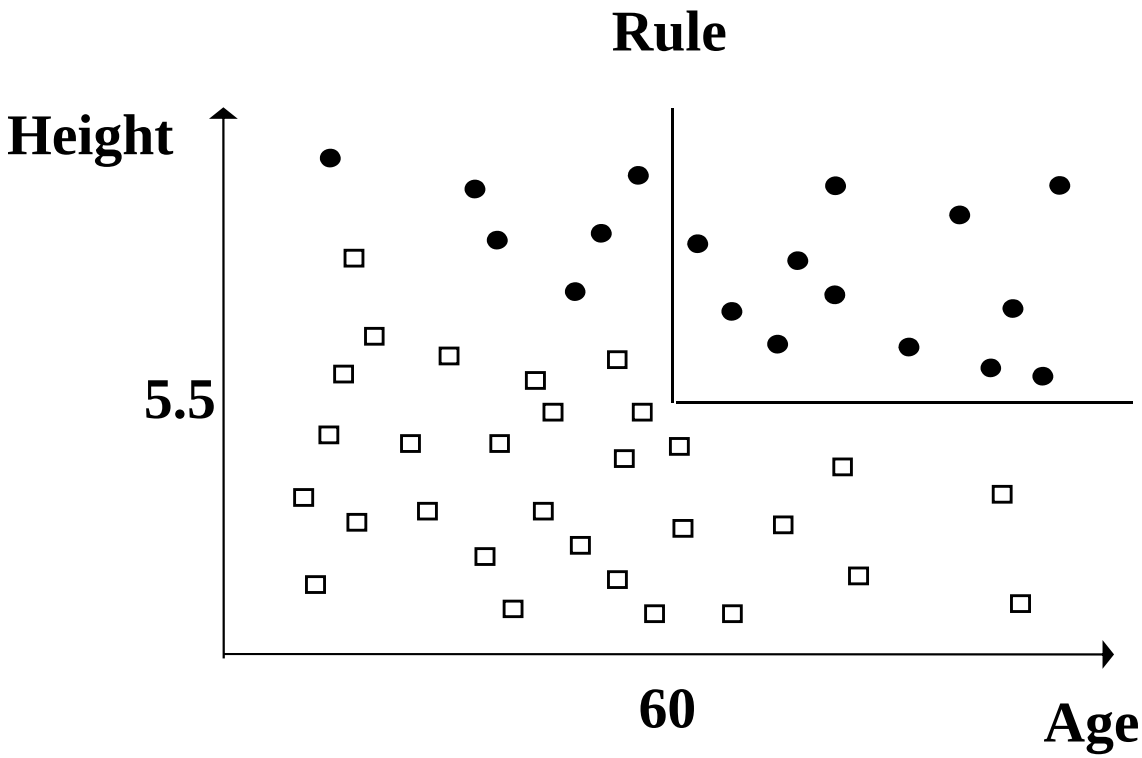


Classification

- Situation/Pattern → Identity/Decision/Class
- Classifiers can be thought of as decision surfaces in pattern space
- Trees and Rules define the surface explicitly
- Other techniques (e.g. K-NN, NN's) define the surface implicitly (via weights etc)

Classification Using Rules & Trees

- **What is the rule shown?**



Risk of Stoop :

Severe risk	●
Low risk	□

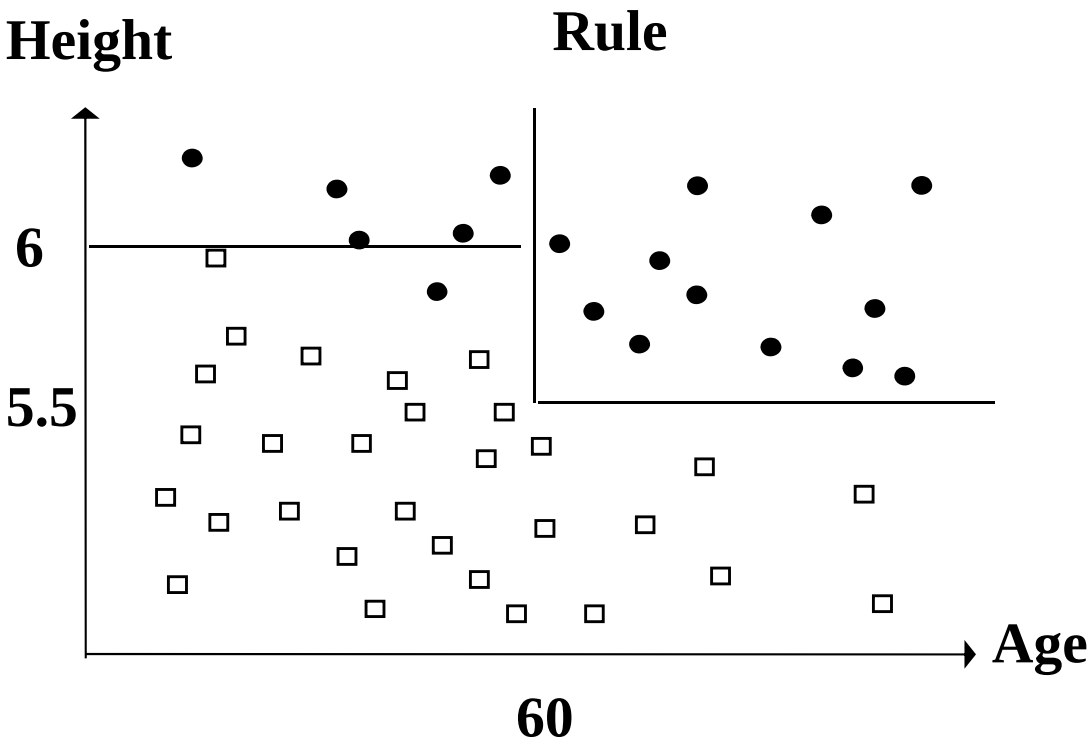
Classification Using Rules & Trees

- **A better ruleset**

If age >=60 and height >= 5.5ft
then risk = high

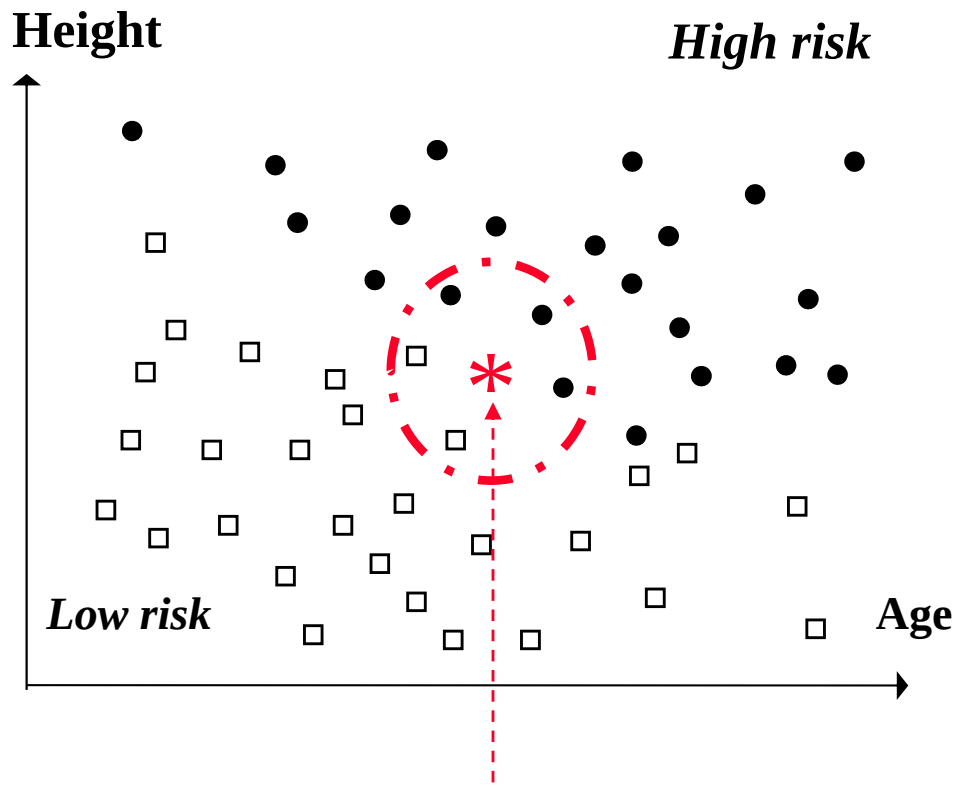
If age < 60 and height >= 6ft
then risk = high

Default = low risk



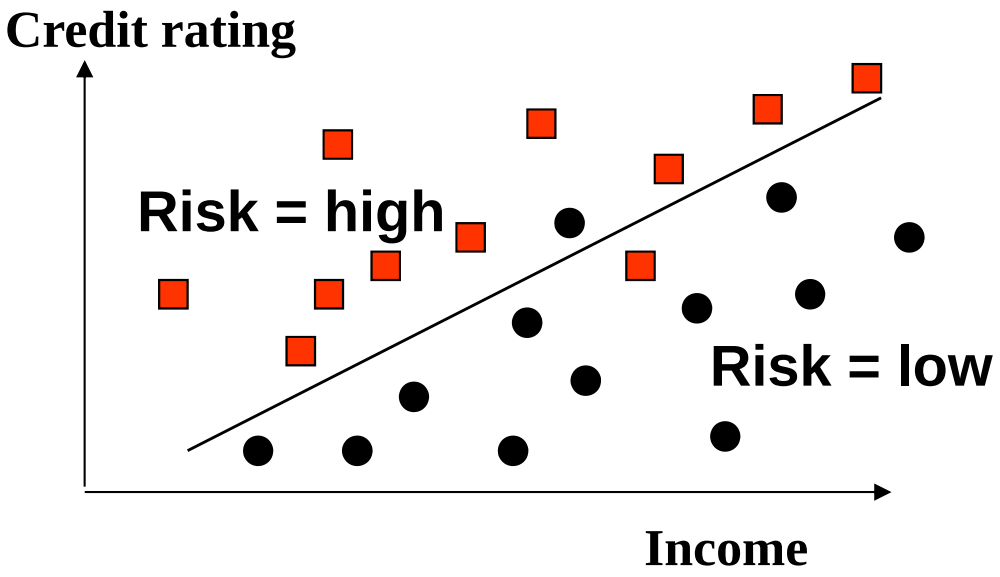
Statistical Classification

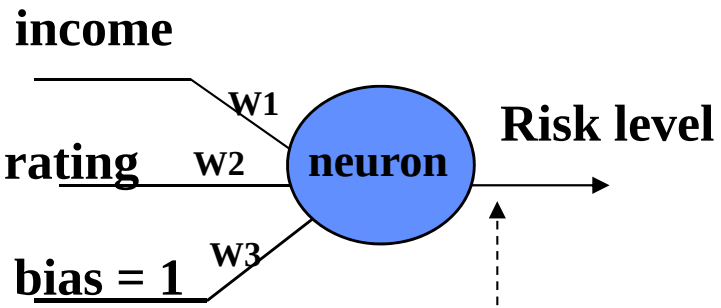
- **Uses the "distances" between data items**
- **E.g. K- Nearest Neighbour Classifier**
  - » **Assign a new pattern to the most represented class in the K nearest neighbours (e.g. K = 5)**
  - » **Non-linear decision surfaces**
  - » **Can be computationally intensive**
  - » **Distance measure is important**



Neural Networks

- Build decision surfaces (for classification)
- Perform curve fitting (for prediction, approximation)
- A simple *perceptron* can learn where to place a linear decision boundary between two classes

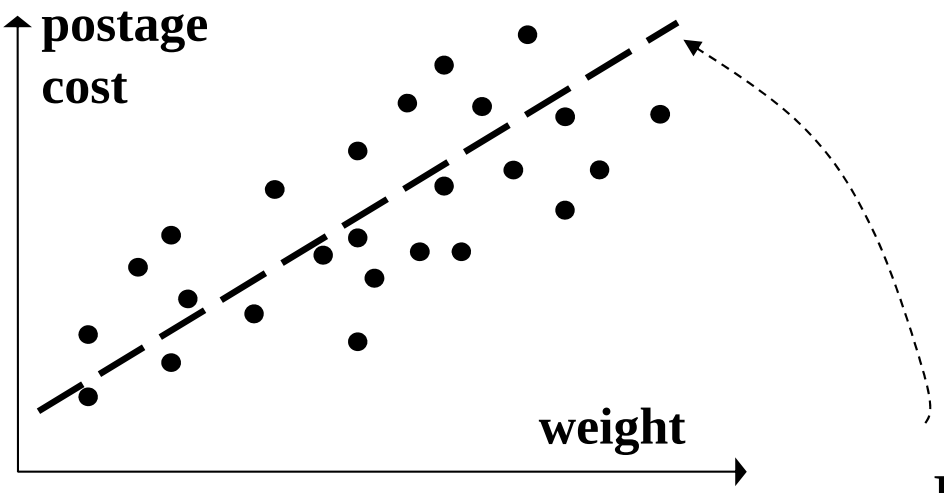


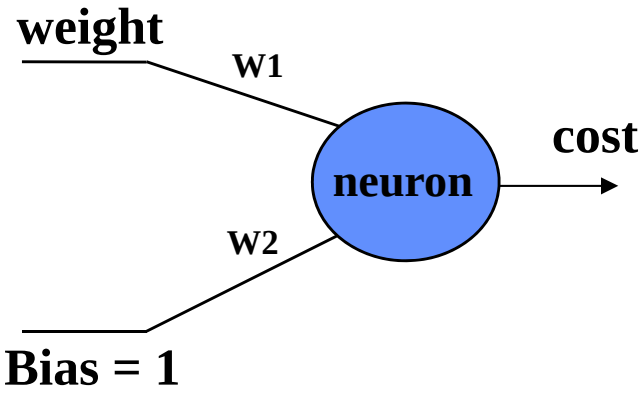


A high output value indicates the input pattern lies above the line and hence is high risk.
A low value indicates the input pattern lies below the line and hence is low risk

Neural Networks

- E.g. predict postage cost of a parcel based on weight



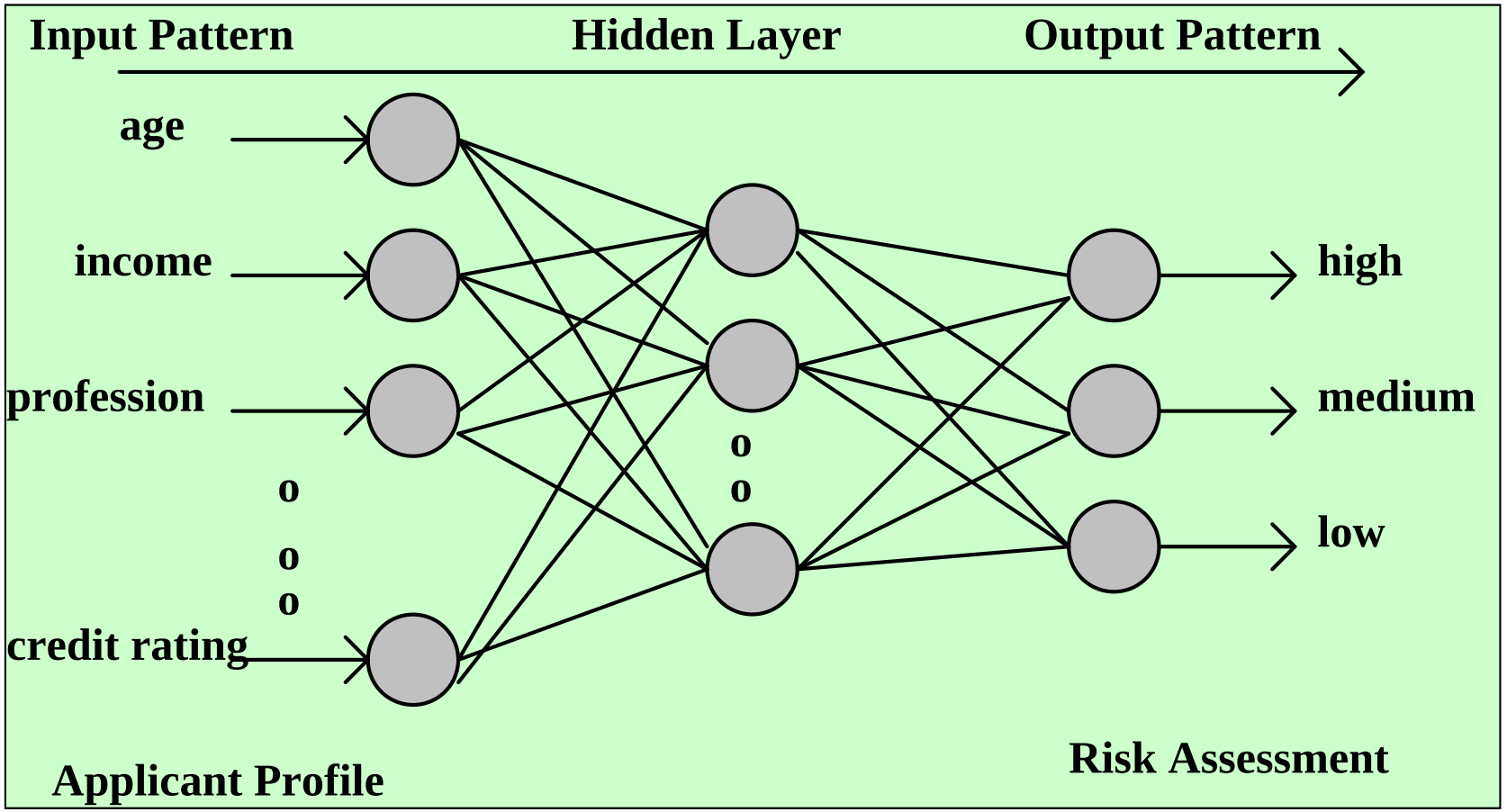


Equation of the regression line modeled by the NN:

Cost = w1*weight + w2

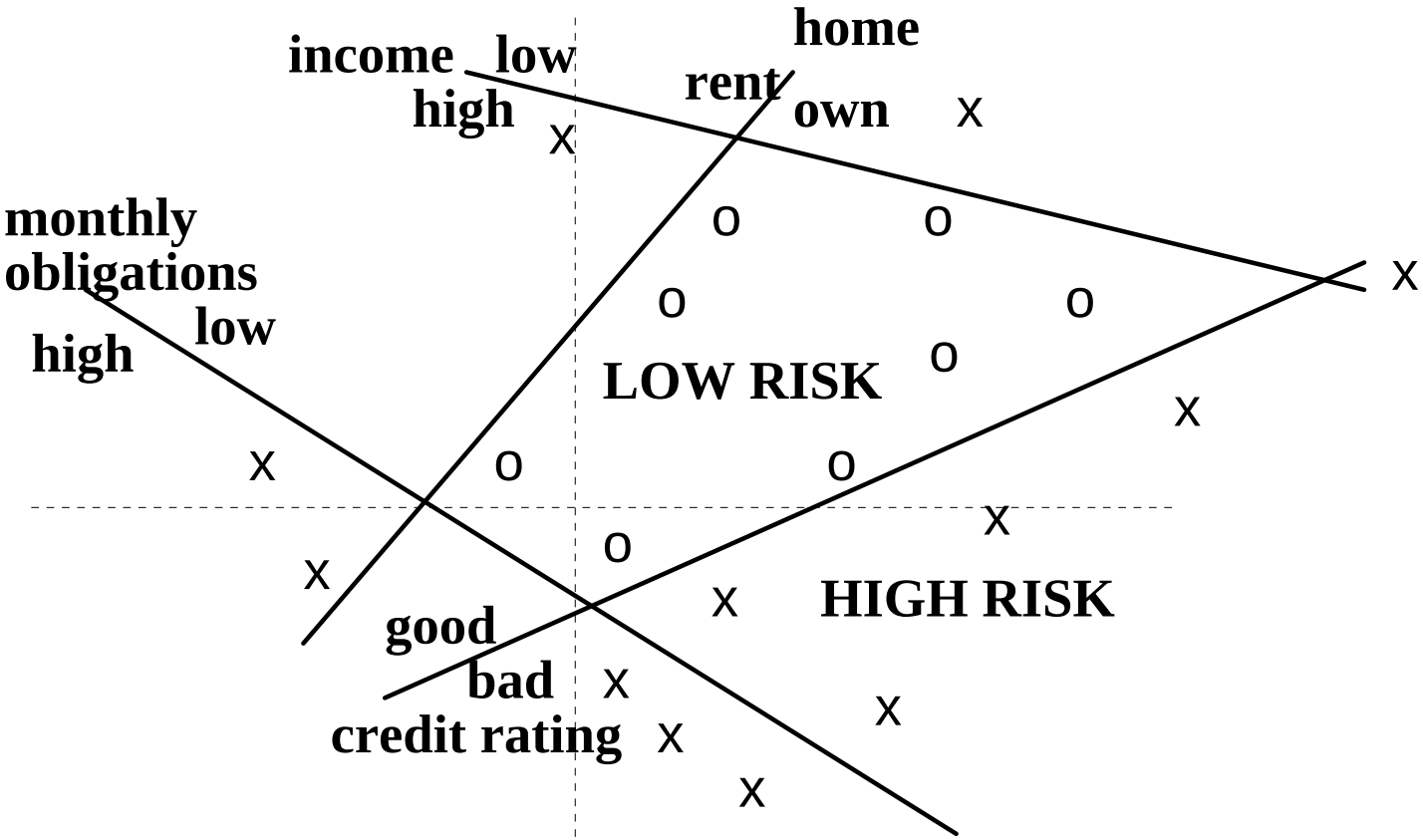
Multi Layer Perceptrons

- Use more nodes & layers to handle more classes and represent more complex decision surfaces



Multi Layer Perceptrons

- Complex "decision" regions need multiple neurons to compute



Creditworthyness Decision Region

NN-Pros and Cons

Advantages

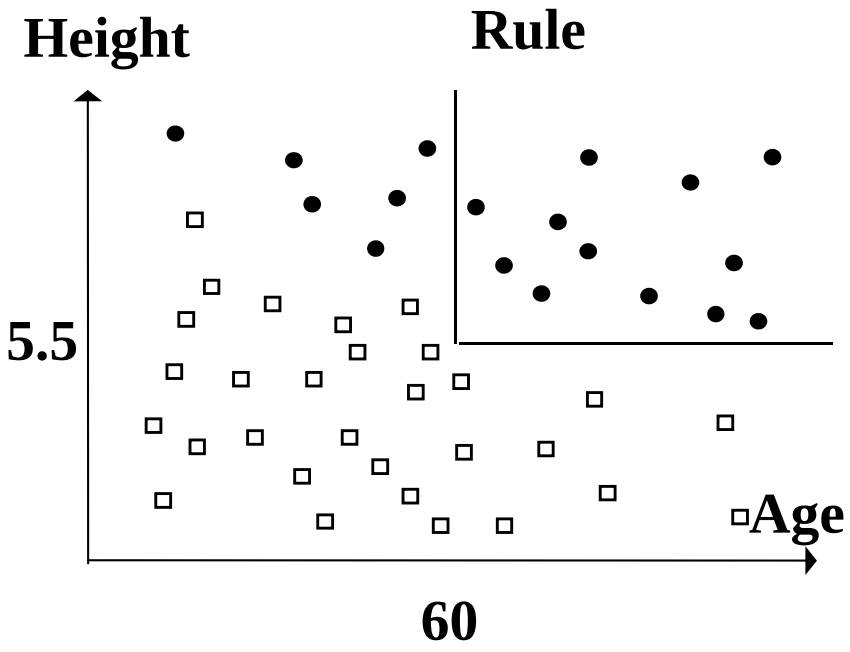
- Build from data alone (many learning algorithms exist to learn the weights)
- Can model complex non-linear functions & decision surfaces

Disadvantages

- Data alone can give problems if new situation is not covered by data
- Black Box solution, no good if the objective is to gain an "understanding" of the data
- Hard to determine the best NN architecture for a particular problem, number of layers & nodes, training parameters etc
- Construction is often done by trial & error - build many NN's, retrain many times, and then pick best (But can use other techniques to help pick the best parameters & architecture)

Fuzzy Expert Systems

- **Crisp rules are often not suited to map human expertise**



Risk of Stoop:

Severe risk	●
Low risk	□

Crisp Rule

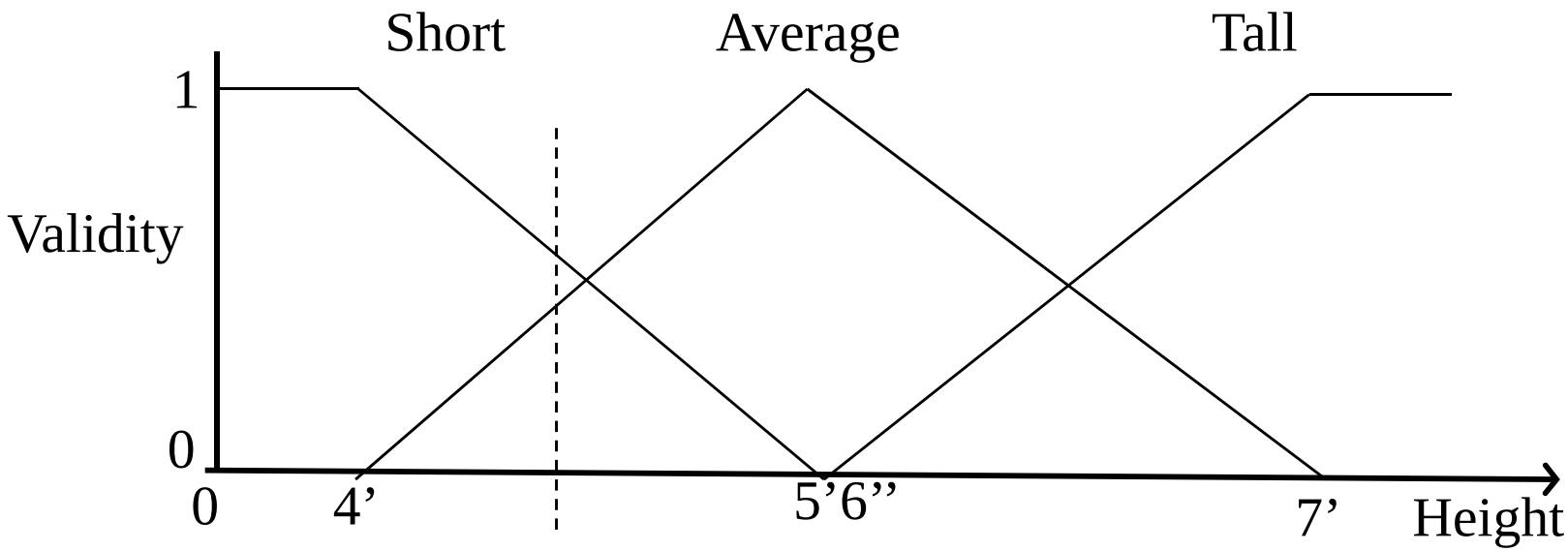
If persons age >= 60 and height >= 5.5ft then risk is high

Better rule

??

Membership Functions

- **Specify a grade (validity) of membership for each linguistic value, e.g.:**



- **Person who is 4ft 7inch is :**

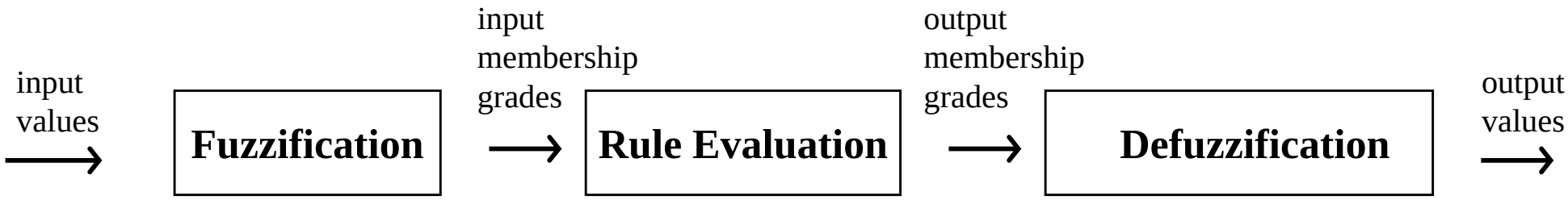
  short with validity 0.61

  average with validity 0.39

  tall with validity 0

Fuzzy Systems: Pros & Cons

- **Many successful applications in control**
  - » **e.g. washing machines, quay crane, etc**
- **Increasing use for business problems**
- **Issues:**
  - » **Creating the rule base**
  - » **Devising correct membership functions**
  - » **Updating/modifying the rule base**



Rule Learning Systems

Can obtain rules (crisp or fuzzy) either…

(1) Derive directly from experts (knowledge acquisition)

(2) Learn from historical data

(3) Combination of (1) & (2)

Learning Fuzzy Rules

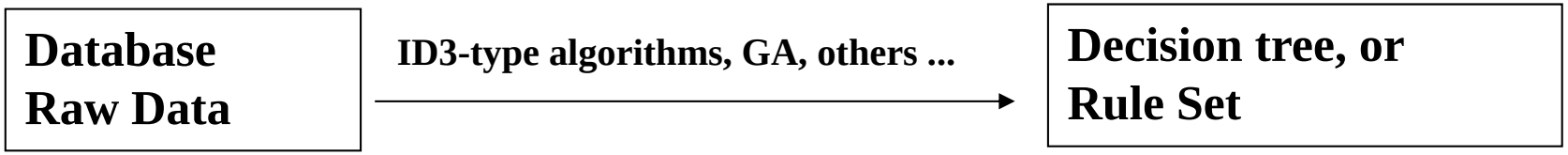
Can use training examples to learn which rules are most important, e.g.

1. Determine set of possible rules.
2. Compute the number of training examples covered by each possible rule
3. Select rules with highest frequency
4. To avoid contradictory rules only select one rule for each input condition combination

Age	Height	Stoop	DoS
Young	Short	Zero	1.0
Young	Short	Small	0.0
Young	Short	Severe	0.0
Young	Average	Zero	0.9
Young	Average	Small	0.1
Young	Average	Severe	0.0
Young	Tall	Zero	0.84
Young	Tall	Small	0.13
Young	Tall	Severe	0.03
Middle	Tall	Severe	0.03
Old	Tall	Severe	0.03

Rule Induction

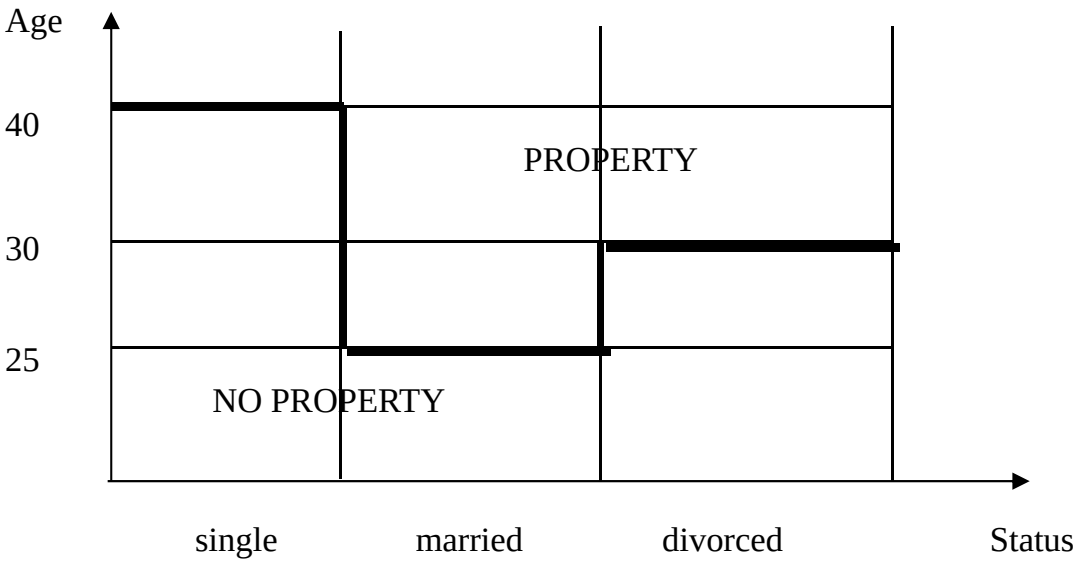
- **Inductive learning ~ learning from examples. One category of machine learning**



- **Inductive learning is essentially a search task:**
  - » **Pick one variable as outcome**
  - » **Search for a combination of tests of input variables that best predict the outcome over the training data**
- **Alternative to NNs for classification problems**
  - » **Unlike NNs, they can be visualised, are made explicit**
  - » **Good for knowledge discovery, data mining**

Rule Induction

- **A rule set or rule tree describes a piecewise linear discriminant surface ~ the union of boxes! E.g.**



If Age < 25 Then NOPROPERTY

If Age >= 40 Then PROPERTY

If Status = single & Age < 40

Then NOPROPERTY

If Status = married & Age >= 25

Then PROPERTY

If Status = divorced & Age >= 30

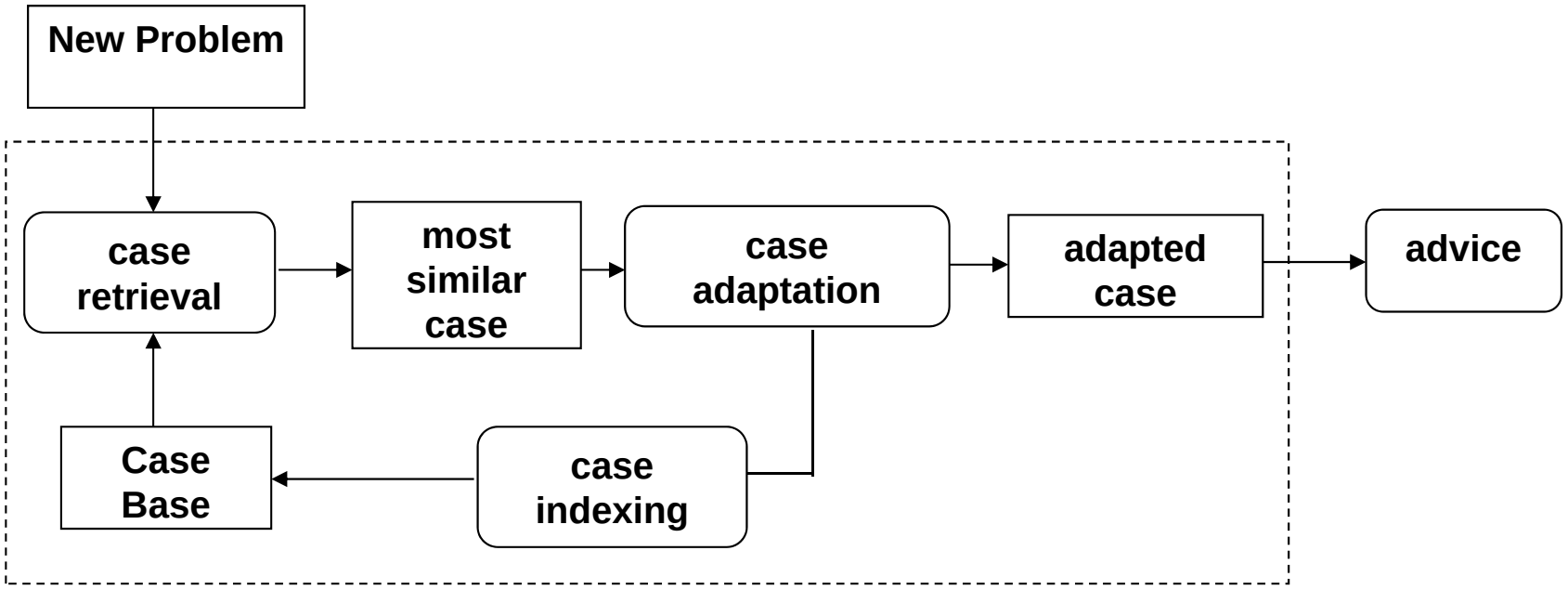
Then PROPERTY

Rule Induction vs NN

Rule Induction	NNs
Classification tasks for ID3, C5.0; Both classification and prediction for CART	Classification and prediction
Trained from examples	Trained from examples
Training can be slow	Training can be very slow
Rules easy to understand	Black box
Data preprocessing is a must	Data preprocessing is a must
Can handle non-numeric variables without coding	Must code non-numeric variables into numbers
Builds piecewise linear surfaces	Can build highly complex surfaces (models)
Current methods not good for incremental learning	Good for incremental learning
Can be over-trained	Can be over-trained

Case-Based Reasoning

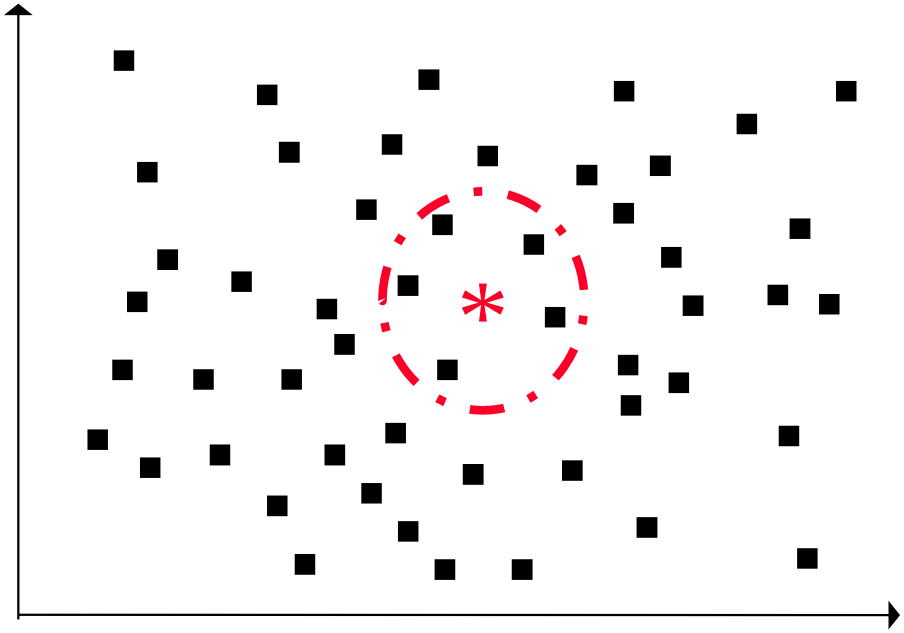
- Requires no rules, knowledge is stored as past cases (easy to update)
- A case is a unit of closely related information, e.g. patient with a specific disease, fault in a product, court case, business record, problem/solution pair
- Most common applications are “help-desk”



Case-Based Reasoning

- CBR systems are essentially Nearest Neighbour classifiers (K=1)

- Past cases are points in pattern space
- CBR system computes the distances from the "problem case" to all past cases
- Nearest case to the "problem" case is presented to the user as the "answer"
- Can also show the nearest K cases and their outcomes



Case Matching

Current Case

Auto-dialer	Doc feeder	Plain paper fax	Built-in answer	Shared phone line	Fax bought	Similarity
yes	?	yes	no	?	?	-

Past Cases (sample)

yes	no	no	yes	yes	Fax300	1
no	no	yes	no	no	Fax100	2
no	yes	no	yes	no	Fax750	0

Case Matching

- **The matching computation is tool specific, an example shows basic principles**

  Case1: age = 31, bp = low, health = good

  Case2: age = 55, bp = high, health = poor

  Distance between text fields is 0 if they are identical else 1

  distance between “low” & “high” = 1

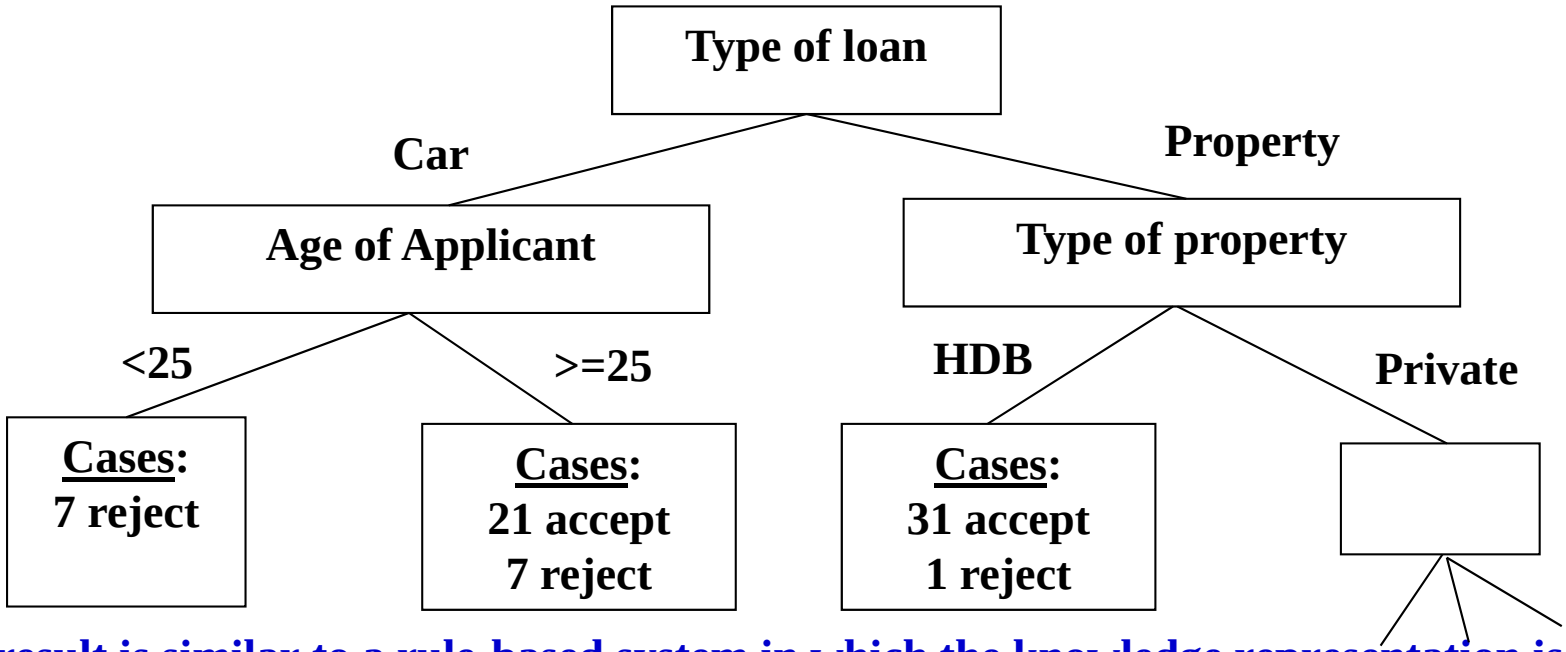
  distance between “low” & “low” = 0

  Distance between numbers is computed in units of standard deviation and scaled to the range 0 to 1

- **Can change relative importance of the fields via user-defined weights:**
  - **dist = W1 * age_distance + W2 * bp_distance**

CBR with Inductive Indexing

- Some CBR tools use a decision tree to index the cases. The decision tree is learnt from the cases using inductive learning techniques. The leaves of the tree represent clusters of similar cases.

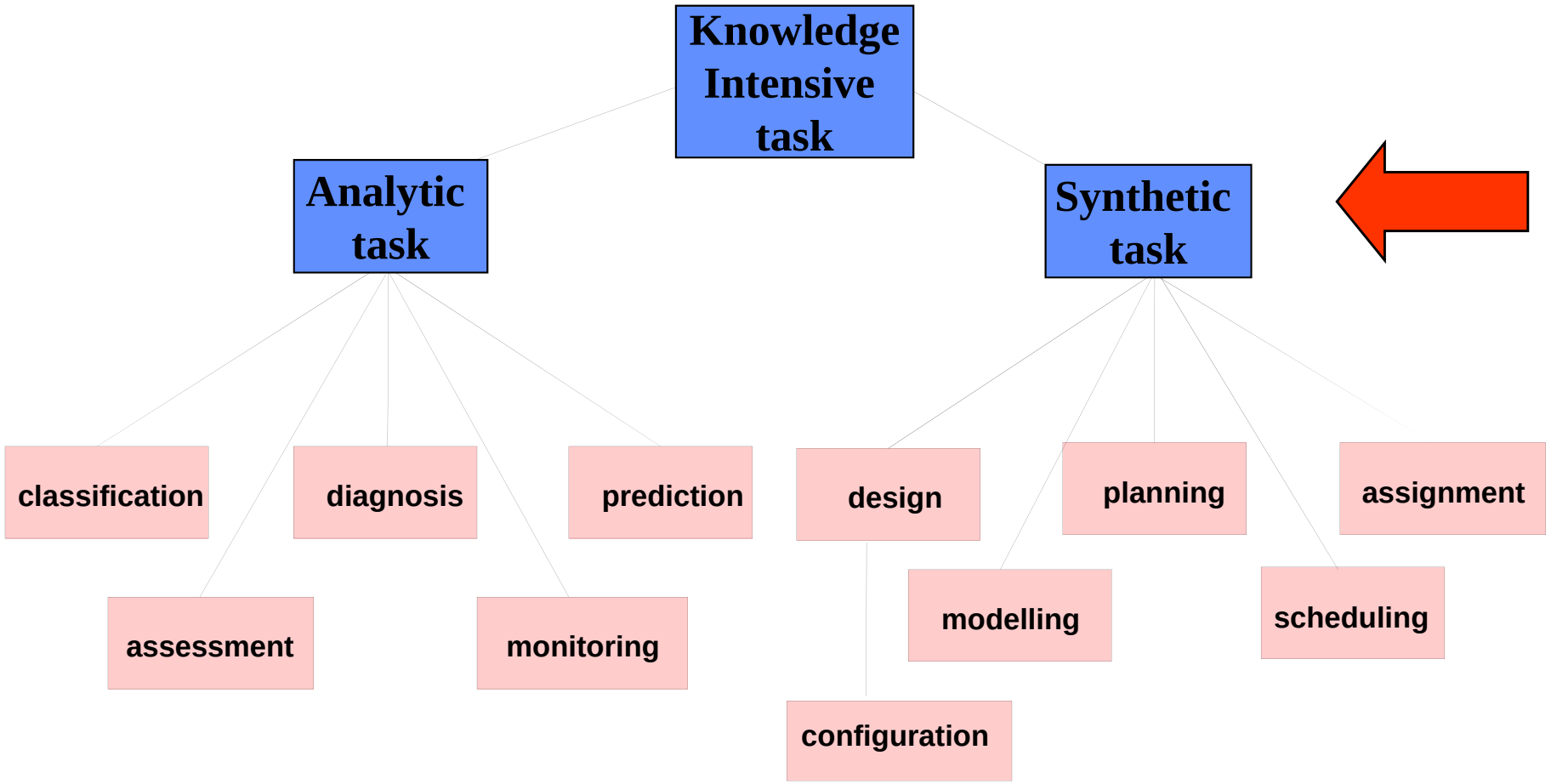


- The result is similar to a rule-based system in which the knowledge representation is a tree
  - » Advantage: case indexing is very fast
  - » Disadvantage: only complete cases can be matched

CBR Pros & Cons

- Advantages
  - » **Good if knowledge is hard to elicit ~ expert supplies cases rather than specifies rules**
  - » **Good if historical data/past cases are available**
  - » **Can handle incomplete knowledge**
  - » **Good if knowledge changes over time - no rule base to update, just add new cases to case base**
- Disadvantages
  - » **Case representation may not suit the knowledge available**
  - » **Lots of cases may be necessary ~ problem if no past historical data exists**
  - » **Determining correct weights can be difficult**

Task Hierarchy

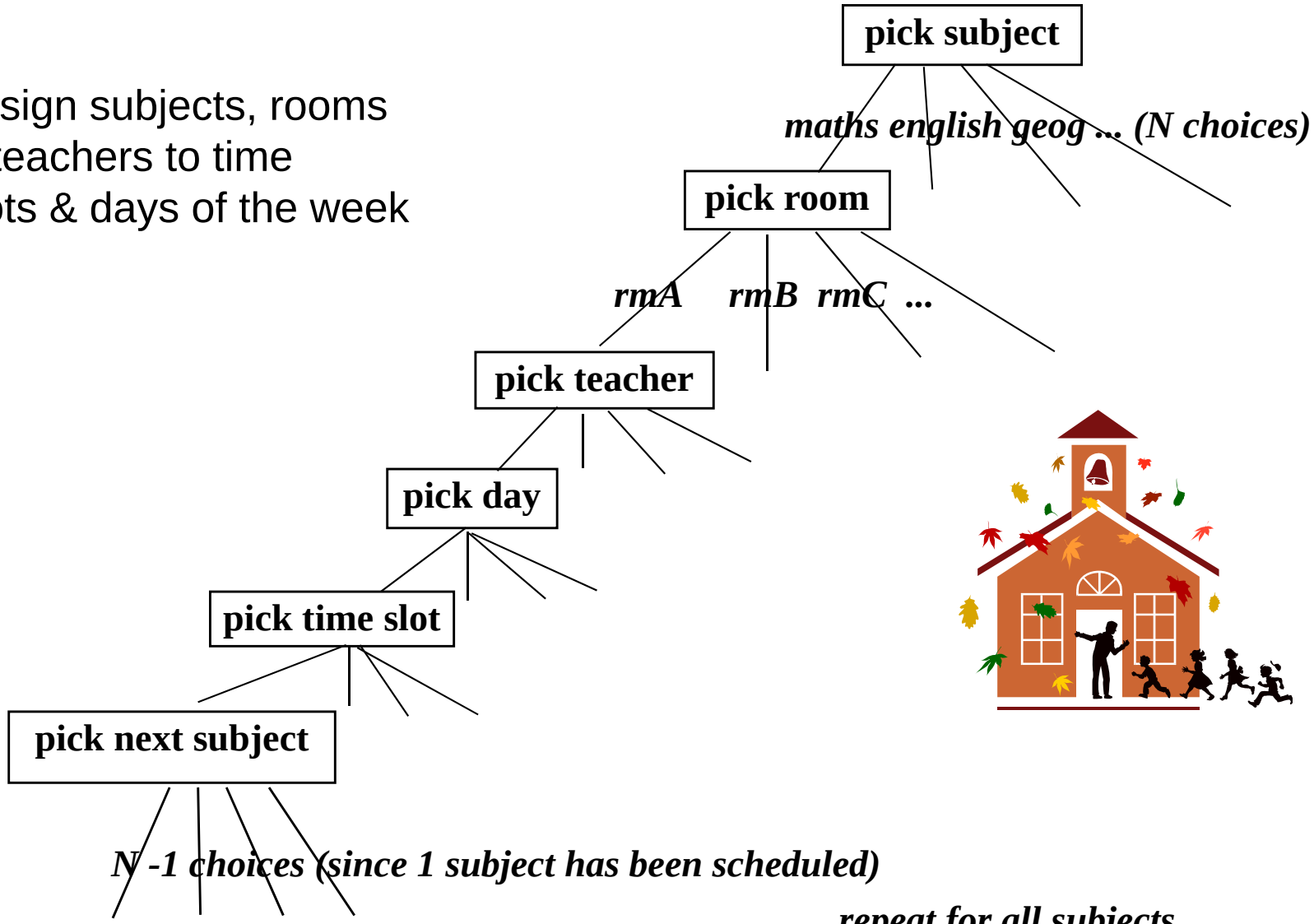


KE Techniques for Synthesis

- Synthesis of a new valid solution is performed by searching through the space of all possible solutions
- Each possible solution is evaluated to see if it is valid and/or the optimum
  - » e.g. a valid design, a valid schedule, an optimal schedule
- Validity of solution involves satisfaction of a set of constraints on the solution variables
- Optimality is measured by a user-defined function which measures the "goodness" of the solution

Search Tree for a School Timetable

Assign subjects, rooms & teachers to time slots & days of the week

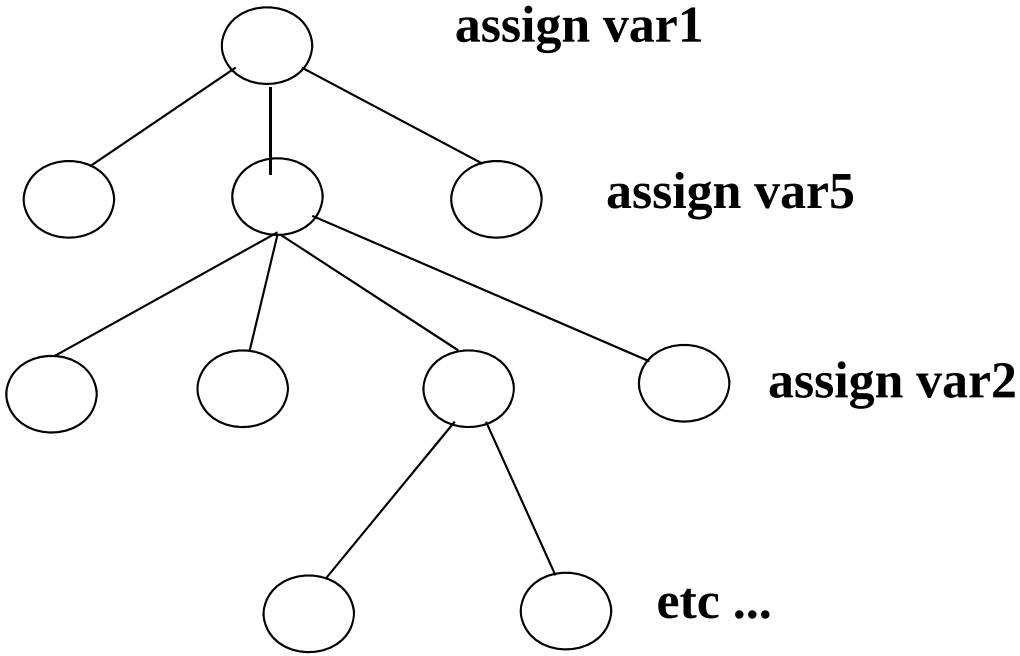


Search

- **Brute Force (Blind) Search**
  - » Evaluate all possible solutions in a systematic way until a valid solution is found
  - » No guarantee of optimality of the first solution - to find a good solution, search for more valid solutions & pick best
  - » Impractical for highly combinatorial complex problems
  - » Algorithms: Depth-first, Breadth-first, etc.
- **Heuristic Search**
  - » Make use of heuristics and human expertise to focus the search in the most promising directions
  - » The first solution found is (hopefully) moderately optimal
  - » Algorithms: Hill climbing, A*, etc.

Hill Climbing Search

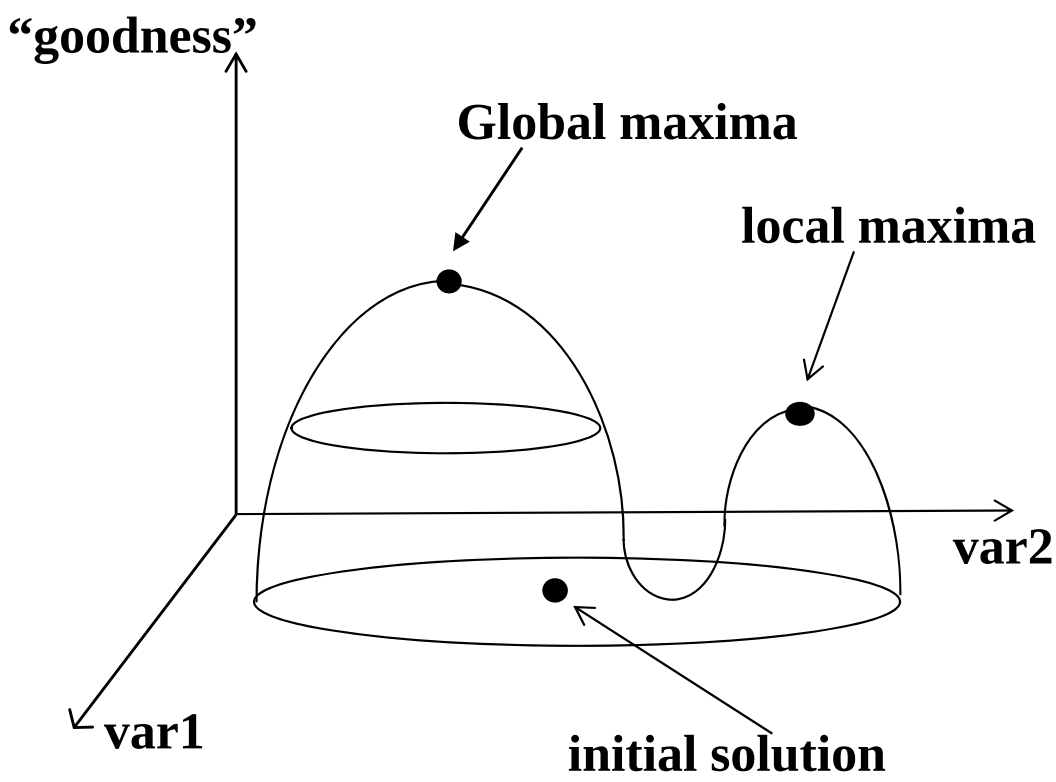
- Each variable assignment is the “best” according to some heuristic. Can assign variables in any order.



- Hopefully the heuristic choices lead the search “up-the hill”.
- In pure hill climbing, there is no turning back!

Search as Hill Climbing

- In pattern space the set of all valid solutions can be thought of as a surface
- Finding the maximum (i.e. biggest peak) is the task of the search process
- Hill climbing takes an initial point on the surface (one valid solution) and then incrementally changes variable assignments to move always “upwards”.
- Problem: can become stuck in a local maxima/ minima



Constraint Propagation

- **Method for reducing the size of the search space.**
- **Domains are defined for the search variables**
  - » **e.g. day = {Mon, Tue, Wed}, age = {18.. 65}**
- **Whenever a variable is assigned a value during search (or has its domain reduced) then examine the effects on the other unassigned variables**
- **Example**
  - » **Assign two work shifts each to John, Fred, Mary.**
- ***Constraints*:**
  - » **Only one person can work on Tuesday**
  - » **Fred cannot work on Wednesday**
  - » **Intermediate search state:**

    Can we assign Mary to Tuesday am ?

	Monday	Tuesday	Wednesday
AM	John		
PM	Fred		

Constraint Programming Languages

- A programming environment with constraint propagation built in.
- User specifies domain variables, constraints and search method, the tool performs the search.
- Specifying variables & constraints in a form of problem modelling, e.g.
  - » Const day = 3, shift = 2
  - » Const Mon = 1, Tue = 2, Wed =3
  - » Const John = 1, Fred = 2, Mary = 3, unavail=0
  - » DVar roster[day, shift] of 0..3
  - » roster[wed, 1] != Fred
  - » roster[wed, 2] != Fred
  - » roster[tue,1] = roster[tue,2]

	Monday	Tuesday	Wednesday
AM	John		
PM	Fred		

Constraint Programming

- **Success can depend greatly on the model you build!**
- **In the previous example, try to represent the two constraints using a schedule modelled as:**
  - » **Const person = 3, shift = 2**
  - » **Const John = 1, Fred = 2, Mary = 3,**
  - » **Dvar roster[person, shift] of 1.. 6**
  - » ***Where* 1 ~ monAM, 2~ monPM, 3 ~ tueAM, 4 ~ tuePM, 5 ~ wedAM, 6~ wedPM**

	John	Fred	Mary
Shift1			
Shift2			

C1: Fred cannot work on Wednesday

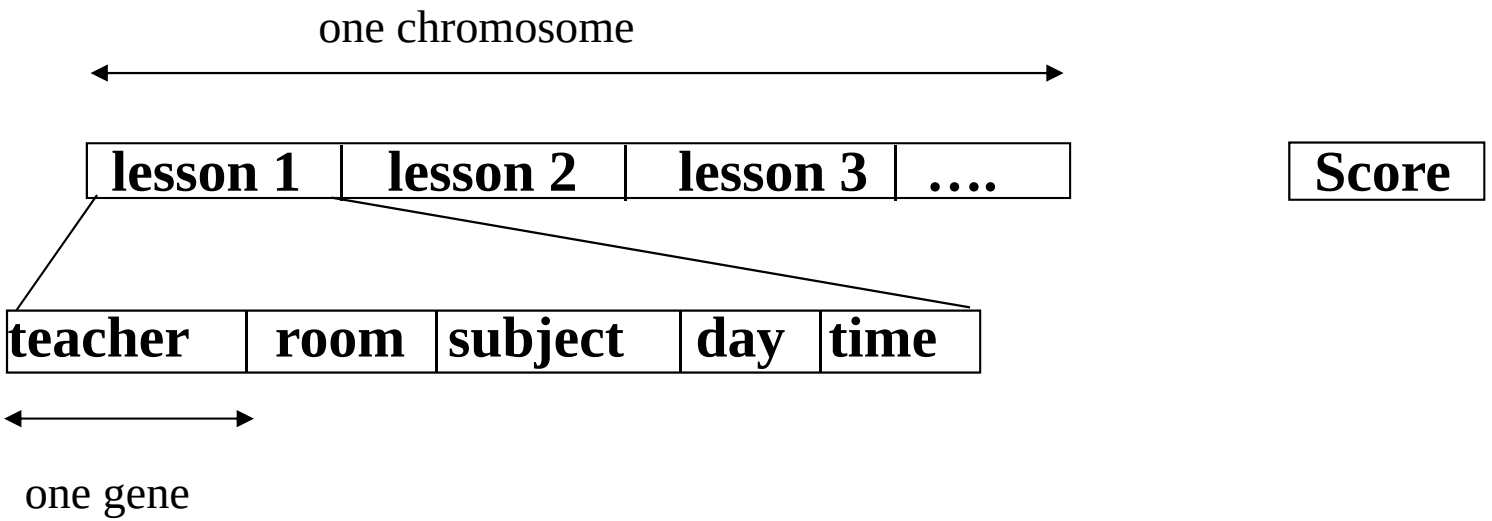
C2: Only one person can work on Tuesday

Genetic Algorithms

- **A form of parallel random search that**
  - » **improves and multiplies the fittest members of a population and eliminates the weakest - uses laws of natural selection**
- **Model problem as**
  - » **a set of chromosome strings - each representing a trial solution.**
  - » **specify an evaluation function to measure the “fitness” of each solution**

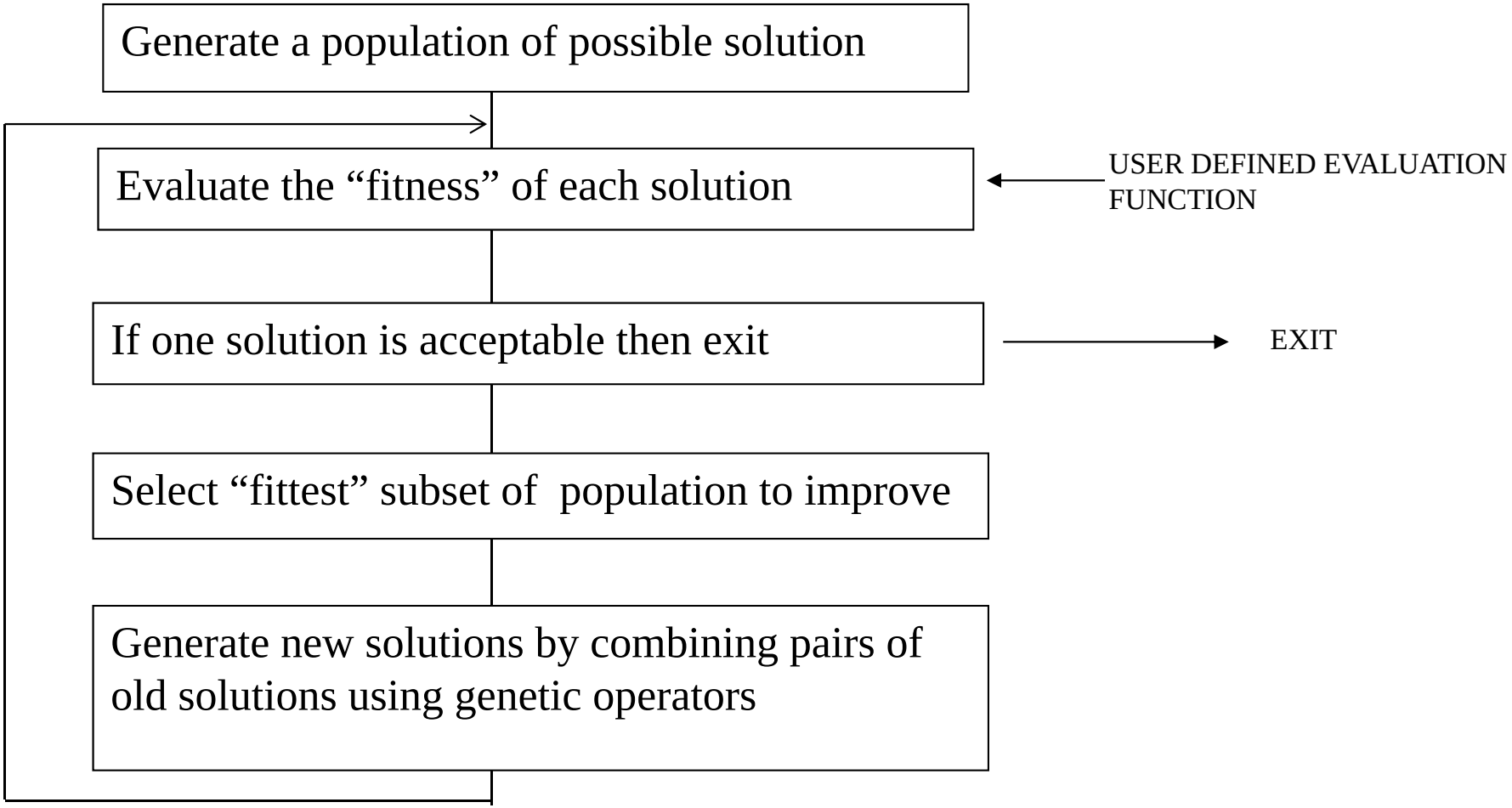
Genetic Algorithms

E.g. Generating a school timetable:



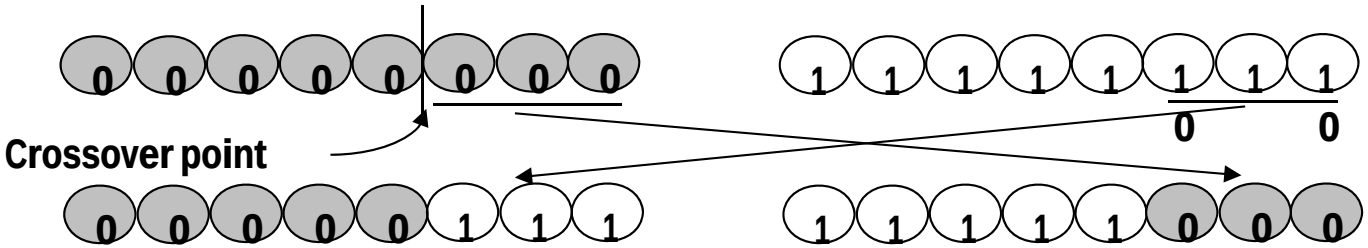
Genetic Algorithms

- **Iterate the following process:**



Genetic Algorithms

- **Crossover**



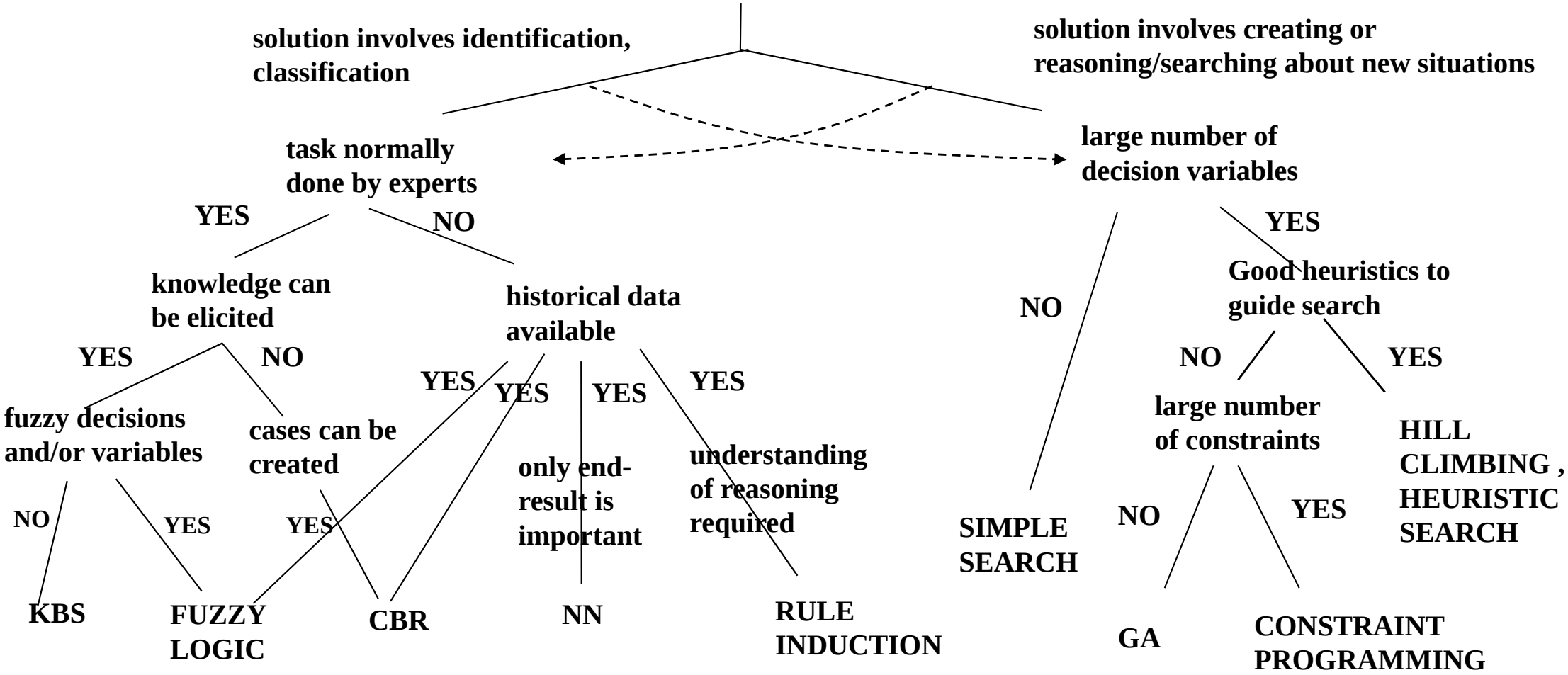
- **Draws mainly only on information already in the population.**
- **Similar to local search so can get stuck in local minima (premature convergence)**

- **Mutation**
  - » **Randomly change parts of the chromosome**
  - » **Introduces completely new info into the pop.**
  - » **Prevents getting stuck in local minima**

GA Pros & Cons

- **Advantages**
  - » Laws of natural selection facilitate very fast search
  - » Generic problem solving approach
- **Problems**
  - » Too many constraints can cause problems
  - » May converge to local optima
  - » Take long time for convergence

Very Rough Guide to Problem Solving



The above is only a very crude approximation. Many real world problems require a mix of techniques….. a hybrid system approach!