4th Class

2016-2017

Intelligent Systems

الأنظمة الذكية

أستاذ المادة : أ.د. أحمد طارق

Intelligent Systems

Prof. Dr. Ahmed T. Saadeq

Computer Science Department

University of Technology

Baghdad, Iraq

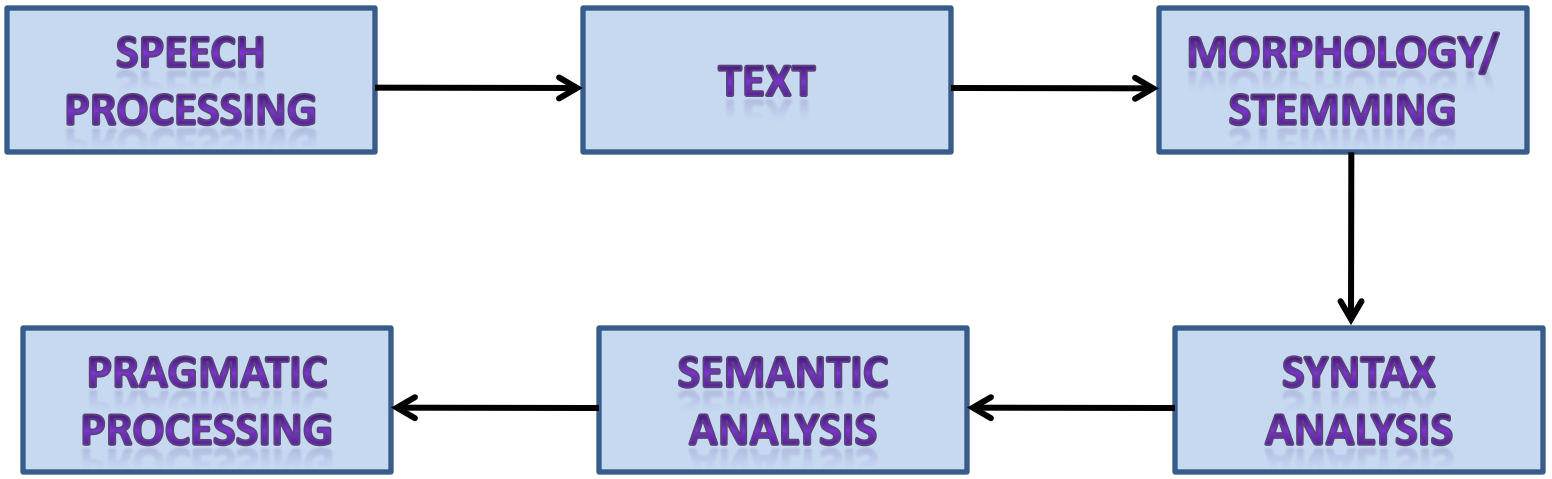
Natural Language Processing(NLP)

Deal with human natural language to process the grams, words, sentences, paragraphs and corpus.

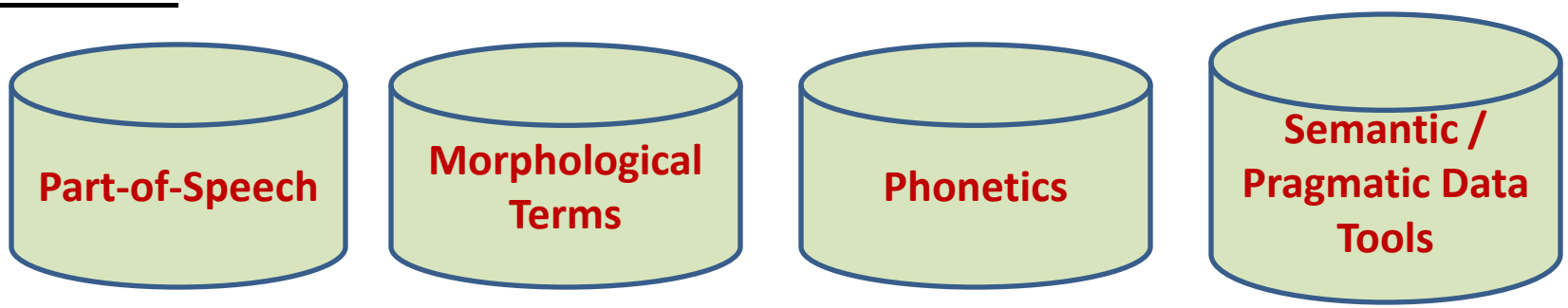
The NL means Arabic, English, etc. language.

NLP have several stages , the following figure illustrate the main stages of NLP.

NLP main stages



Datasets



Part-of-Speech (PoS)

means type of words in NL, one of the best dataset (PoS) is (Penn Treebank)

Example	Description	Part	Example	Description	Part
+, %, &	Symbol	SYM	and, but, or	Coordinating Conj	CC
To	to	TO	one, two	Cardinal Number	CD
Ah, oops	Interjection	UH	a, an, the	Determiner	DT
Eat	Base Verb	VB	there	Existential There	EX
Ate	Past Verb	VBD	copa america	Foreign Word	FW
Eating	Gerund Verb	VBG	of, on, by, in	Preposition	IN
Eaten	P.P. Verb	VBN	yellow, long	Adjective	JJ
Eat	Non-‘s Verb	VBP	bigger	Adjective Comparative	JJR
Eats	‘s Verb	VBZ	wildest	Adjective Superlative	JJS
Which, that	Determiner Sign	WDT	1, 2, 3	List Item Maker	LS
What, who	Pronoun Sign	WP	can, should	Modal	MD
Whose	Possessive WP	WP$	horse	Singular Noun	NN
How, where	Adverb Sign	WRP	horses	Plural Noun	NNS
@	Dollar	$	IBM	Singular Proper Noun	NNP
#	Pound	#	Carolinas	Plural Proper Noun	NNPS
‘ or “	Left Quotation	“	all, both	Predetermine	PDT
’ or ”	Right Quotation	”	‘s	Possessive Ending	POS
[({ <	Left Bracket	(	I, you, he	Personal Pronoun	PRP
]) } >	Right Bracket	)	your, one’s	Possessive Pronoun	PRP$
,	Comma	,	soon, never	Adverb	RB
.	Point	.	faster	Comparative Adverb	RBR
: ; … - —	Special Sent.	:	fastest	Superlative Adverb	RBS
			up, off	Particle	RP

Example

She promised to back the bill

she:	PRP
promised:	VBN, VBD
to:	TO
back:	VB, JJ, RB, NN
the:	DT
bill:	NN, VB

Morphology / Stemming

Means of Morphology in the NL, studied parts added to the words and how to put these parts in words. That is, it means studying how to know the origin of the word through the clearance of the additions, and find out what would happen if these additions were added to the word.

Stemming is extract keyword (meaningful) from the word, for example, if we have the word (going), in Morphology output will be (the original word: (go) and added is (ing), while in Stemming output is a meaningful word which (go).

Types of added:

connect

Prefix : reconnect

Suffix : connected

Prefix – Suffix : reconnected

Sample of Stemming code:

- IF a word ends in "ies", but not "eies" or "aies" THEN "ies" → "y"
- ELSE IF a word ends in "es", but not "aes", "ees" or "oes" THEN "es"→ "e"
- ELSE IF a word ends in "s", but not "us" or "ss" THEN "s"→ NULL

Prof. Dr. Ahmed Tariq Saadeq

Most Common Prefix & Suffix in English Language

Prefix	Meaning	Key Word
anti-	against	antifreeze
de-	opposite	defrost
dis-*	not, opposite of	disagree
en-, em-	cause to	encode, embrace
fore-	before	forecast
in-, im-	in	infield
in-, im-, il-, ir-*	not	injustice, impossible
inter-	between	interact
mid-	middle	midway
mis-	wrongly	misfire
non-	not	nonsense
over-	over	overlook
pre-	before	prefix
re-*	again	return
semi-	half	semicircle
sub-	under	submarine
super-	above	superstar
trans-	across	transport
un-*	not	unfriendly
under-	under	undersea

Suffix	Meaning	Key Word
-able, -ible	can be done	comfortable
-al, -ial	having characteristics of	personal
-ed*	past-tense verbs	hopped
-en	made of	wooden
-er	comparative	higher
-er,	one who	worker, actor
-est	comparative	biggest
-ful	full of	careful
-ic	having characteristics of	linguistic
-ing*	verb form/ present participle	running
-ion, -tion, -ation, ition	act, process	occasion, attraction
-ity, -ty	state of	infinity
-ive, -ative, -itive	adjective form of a noun	plaintive
-less	without	fearless
-ly*	characteristic of	quickly
-ment	action or process	enjoyment
-ness	state of, condition of	kindness
-ous, -eous, -ious	possessing the qualities of	joyous
-s, -es*	more than one	books, boxes
-y	characterized by	happy

Syntax Analysis

Means check the sentence is grammatically correct or no.

There are several techniques to check the NL sentences such as Context Free Grammar (CFG) and Augmented Transition Network (ATN).

Context Free Grammar for Syntax Analysis

Step 1: Find P-o-S for the sentence words.

Step 2: Build the syntax tree.

Step 3: Build CFG.

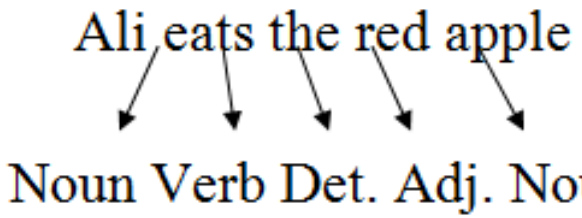
Step 4: Convert CFG to HLL program.

Step 5: Check the outputs program.

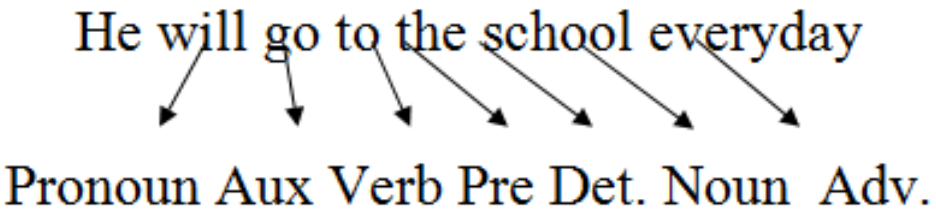
Example 1:

Ali eats the red apple

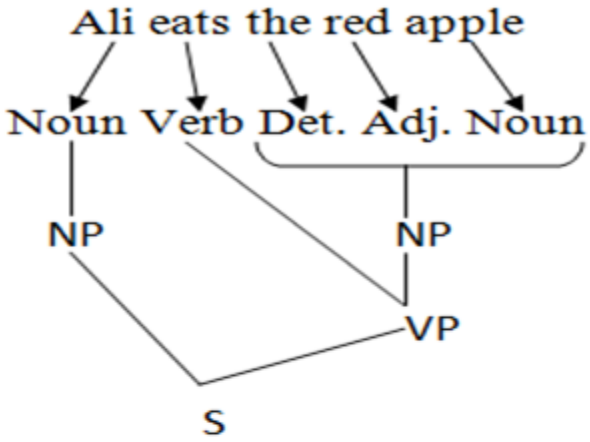
He will go to the school everyday

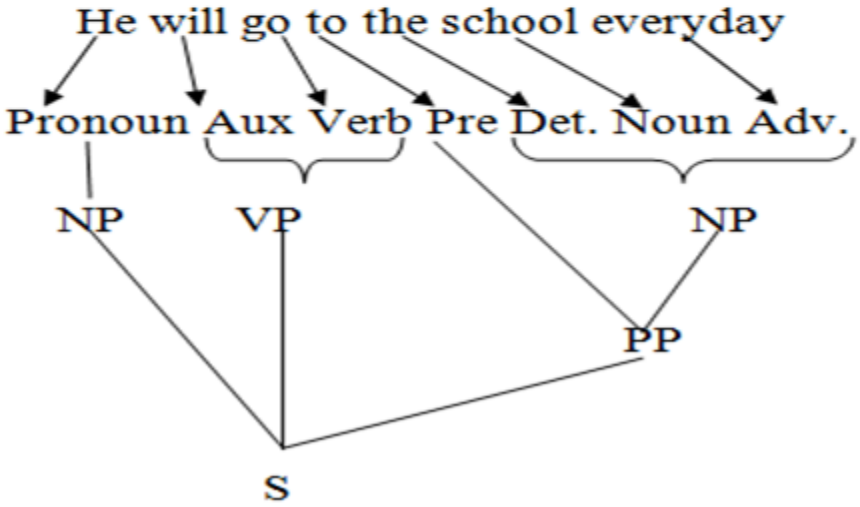


P-o-S



Search Tree





CFG

S ⟶ NP VP

NP ⟶ Noun | Det Adj Noun

VP ⟶ Verb NP

Noun ⟶ ali | apple

Det ⟶ the

Adj ⟶ red

Verb ⟶ eats

S ⟶ NP VP PP

NP ⟶ Pronoun | Det Noun Adv

VP ⟶ Aux Verb

PP ⟶ Pre NP

Noun ⟶ he | school

Det ⟶ the

Adv ⟶ everyday

Verb ⟶ go

Aux ⟶ will

Pre ⟶ to

Note:- NP : Noun Phrase, VP : Verb Phrase, PP : Preposition Phrase

The two sentence

S ⟶ NP VP | NP VP PP

NP ⟶ Noun | Det Adj Noun | Pronoun | Det Noun Adv

VP ⟶ Verb NP | Aux Verb

PP ⟶ Pre NP

Noun ⟶ ali | apple | school

Pronoun ⟶ he

Det ⟶ the

Adj ⟶ red

Verb ⟶ eats | go

Adv ⟶ everyday

Aux ⟶ will

Pre ⟶ to

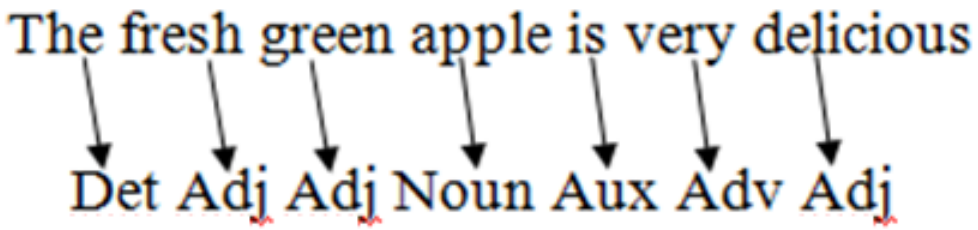
Prolog Program

Prof. Dr. Ahmed Tariq Saadeq

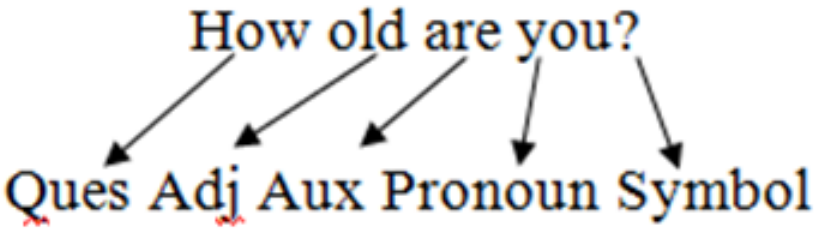
```
Domains
        s=string
        sl=s*
Predicates
        Run.
        Split(s,sl)
        Sen(sl)
        NP(sl)
        VP(sl)
        PP(sl)
        Noun(sl)
        Verb(sl)
        Aux(sl)
        Pronoun(sl)
        Det(sl)
        Adv(sl)
        Adj(sl)
        Pre(sl)
 Clauses
Run:-readln(S),split(S,X),Sen(X),write("correct…"),!.
Run:-write("wrong…").

Sen(X):-append(X1,X2,X),NP(X1),VP(X2),!.
Sen(X):-append(X1,Y,X),append(X2,X3,Y),NP(X1),VP(X2),PP(X3),!.
NP(X):-Noun(X),!.
NP(X):- append(X1,Y,X),append(X2,X3,Y),Det(X1),Adj(X2),Noun(X3),!.
NP(X):- Pronoun(X),!.
NP(X):- append(X1,Y,X),append(X2,X3,Y),Det(X1),Noun(X2),Adv(X3),!.
VP(X):- append(X1,X2,X),Verb(X1),NP(X2),!.
VP(X):- append(X1,X2,X),Aux(X1),Verb(X2),!.
PP(X):- append(X1,X2,X),Pre(X1),NP(X2),!.

Noun(["ali"]).
Noun(["apple"]).
Noun(["school"]).
Pronoun(["he"]).
Det(["the"]).
Adj(["red"]).
Adv(["everyday"]).
Aux(["will"]).
Pre(["to"]).
Verb(["eats"]).
Verb(["go"]).

split("", [ ]).
split(S,[H|T]) :- fronttoken(S,H,W), split(W,T).

append([], X, X).
append([H|T], X, [H|T1]):- append(T, X, T1).
```

Example 2:

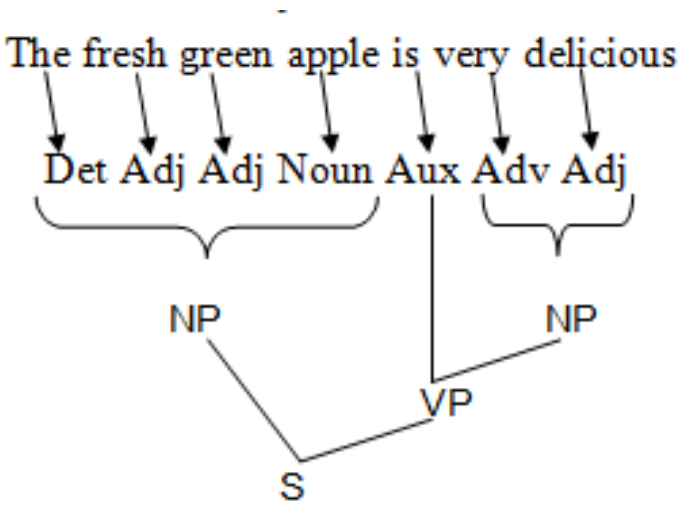
The fresh green apple is very delicious
How old are you?

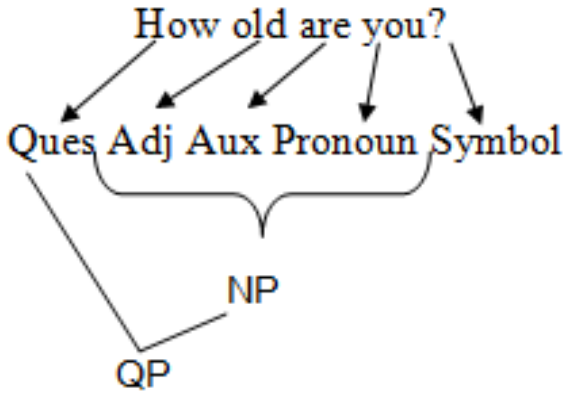


P-o-S



Search Tree





CFG

S ⟶ NP VP

NP ⟶ Det Adj Adj Noun | Adv Adj

VP ⟶ Aux NP

Noun ⟶ apple

Det ⟶ the

Aux ⟶ is

Adj ⟶ fresh | green | delicious

Adv ⟶ very

S ⟶ QP Symbol

QP ⟶ Ques NP

NP ⟶ Adj Aux Pronoun

Ques ⟶ how

Aux ⟶ is

Pronoun ⟶ you

Adj ⟶ old

Symbol ⟶ ?

The two sentence

S ⟶ NP VP | QP Symbol

NP ⟶ Det Adj Adj Noun | Adv Adj | Adj Aux Pronoun

VP ⟶ Aux NP

QP ⟶ Ques NP

Noun ⟶ apple

Det ⟶ the

Aux ⟶ is

Adj ⟶ fresh | green | delicious | old

Adv ⟶ very

Ques ⟶ how

Pronoun ⟶ you

Symbol ⟶ ?

Prolog Program

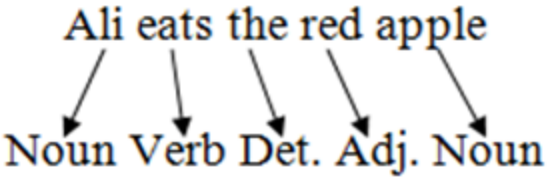
```
Sen(X):-append(X1,X2,X),NP(X1),VP(X2),!.
Sen(X):-append(X1,X2,X),QP(X1),Symbol(X2),!.
NP(X):- append(X1,Y,X), append(X2,Z,Y), append(X3,X4,Z), Det(X1), Adj(X2),
Adj(X3), Noun(X4),!.
NP(X):- append(X1,X2,X),Adv(X1),Adj(X2),!.
NP(X):- append(X1,Y,X),append(X2,X3,Y),Adj(X1),Aux(X2),Pronoun(X3),!.
VP(X):- append(X1,X2,X),Aux(X1),NP(X2),!.
QP(X):- append(X1,X2,X),Ques(X1),NP(X2),!.

Noun(["apple"]).
Pronoun(["you"]).
Det(["the"]).
Adj(["fresh"]).
Adj(["old"]).
Adj(["green"]).
Adj(["delicious"]).
Adv(["very"]).
Aux(["is"]).
Ques(["how"]).
Symbol(["?"]).
```

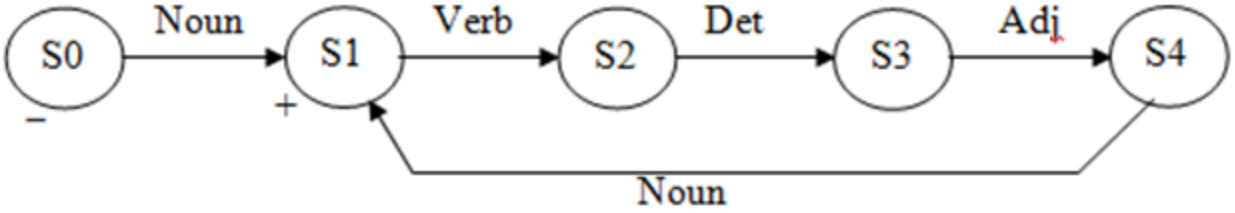
Augmented Transition Networks for Syntax Analysis

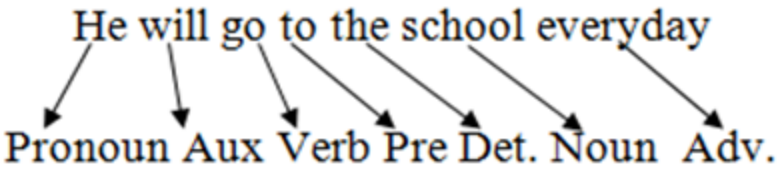
ATN is a mathematical model for modeling and representation the operations and programs.

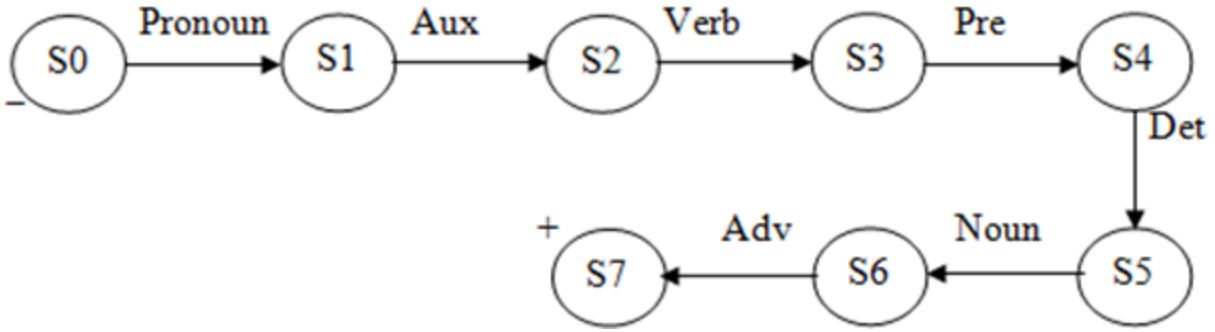
ATN is another model for Finite States Automata (FSA).

Example 1:


Ali eats the red apple
Noun Verb Det. Adj. Noun


S0
Noun
S1
Verb
S2
Det
S3
Adj
S4
–
+
Noun


He will go to the school everyday
Pronoun Aux Verb Pre Det. Noun Adv.

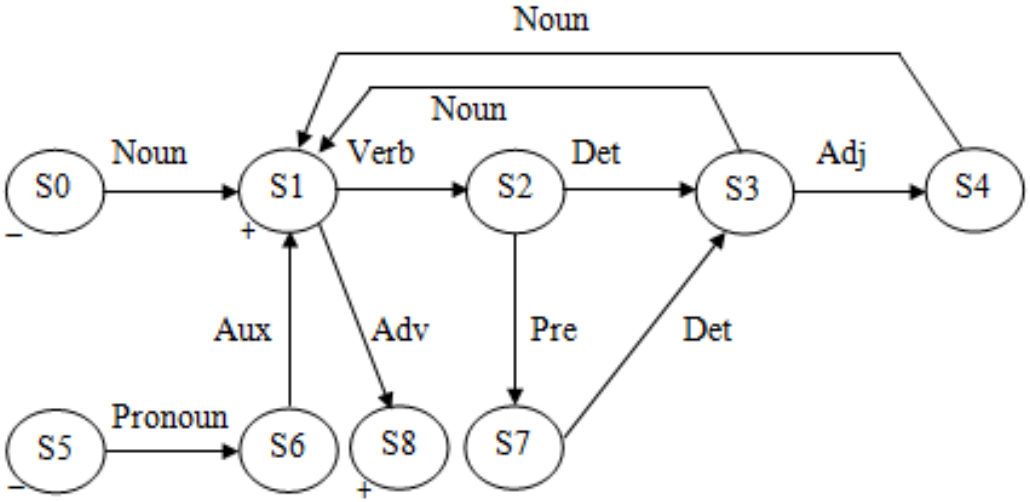

S0
Pronoun
S1
Aux
S2
Verb
S3
Pre
S4
Det
S5
Noun
S6
Adv
S7
+
–

The two sentences:

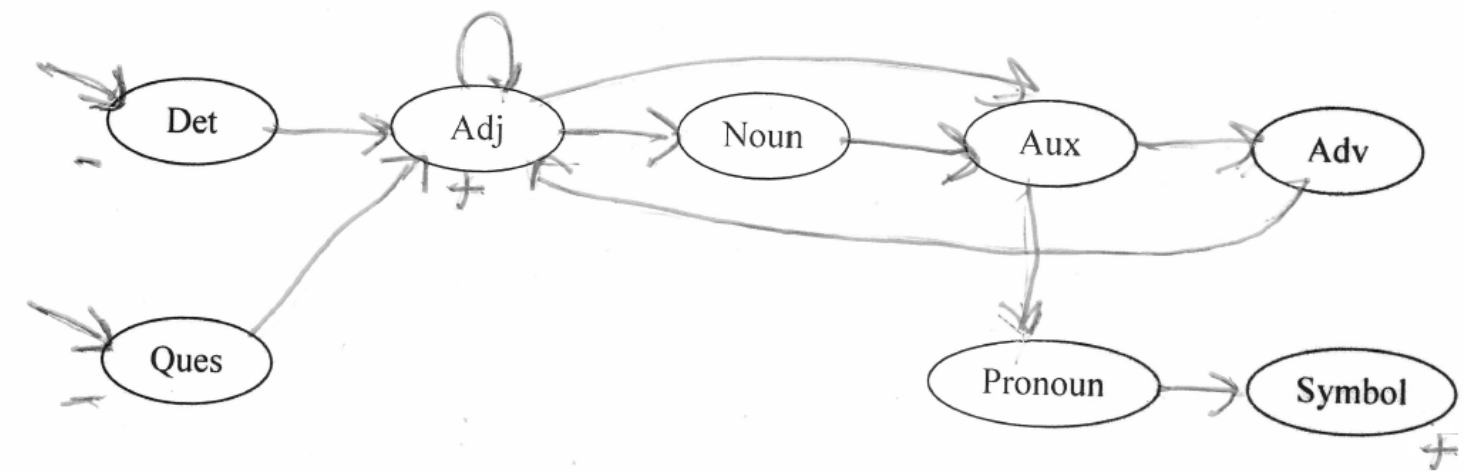
Ali eats the red apple

He will go to the school everyday

ATN



FSA



Prolog Program

Clauses

```
Run:-readln(S),split(S,X),Sen(X),write("correct…"),!.
Run:-write("wrong…").
Sen(X):-Det_ST(X),!.
Sen(X):-Ques_ST(X),!.
Det_ST([H|T]):-Det(H), Adj_ST(T),!.
Ques_ST([H|T]):-Ques(H), Adj_ST(T),!.
Adj_ST([H|T]):-Adj(H), Noun_ST(T),!
Adj_ST([H|T]):-Adj(H), Adj_ST(T),!.
Adj_ST([H|T]):-Adj(H), Aux_ST(T),!.
Adj_ST([H|T]):-Adj(H), T= [ ],!.
Noun_ST([H|T]):-Noun(H), Aux_ST(T),!.
Aux_ST([H|T]):-Aux(H), Adv_ST(T),!.
Aux_ST([H|T]):-Aux(H), Pro_ST(T),!.
Pro_ST([H|T]):-Pronoun(H), Sym_ST(T),!.
Adv_ST([H|T]):-Adv(H), Adj_ST(T),!.
Sym_ST([H|T]):-Symbol(H), T=[ ],!.
```

```
Noun("apple").
Pronoun("you").
Det("the").
Adj("fresh").
Adj("old").
Adj("green").
Adj("delicious").
Adv("very").
Aux("is").
Ques("how").
Symbol("?").

split("", [ ]).
split(S,[H|T]) :- fronttoken(S,H,W), split(W,T).
```

Semantic Analysis

Meaning that the analysis of the stage of dealing with the meanings of words in sentences and the interdependence of these words among them in terms of intended meaning of the sentences. In other words, this stage literally scan in the sentence on the one hand and examine the extent of interdependence of the words of that sentence among them.

There are several techniques treats semantic analysis such as:

- Part-of-Speech Based.
- Semantic Grammars Based.
- Role-Themes Based.

Semantic Analysis Verification Using Semantic Grammars

When there is a need to design a specific application and a specific range system, this method is optimal analysis of meaning it is through certain rules established in accordance with the requirements and the possibilities contained and as we know, these rules do not accept the mistake, and you know it as the rules with specific specifications (Custom-Tailored Grammar) belonging to a given application itself.

Let us take an example of a clear and important and The daily application in human life, a (flight), what is required design rules include a moral analysis of natural language sentences that deal with the flights.

Assume the following sentences:

- the flight to Baghdad
- the 7 o'clock flight
- the first flight out
- flight 321 to Baghdad

The following grammars represent the semantic grammars for the above sentences

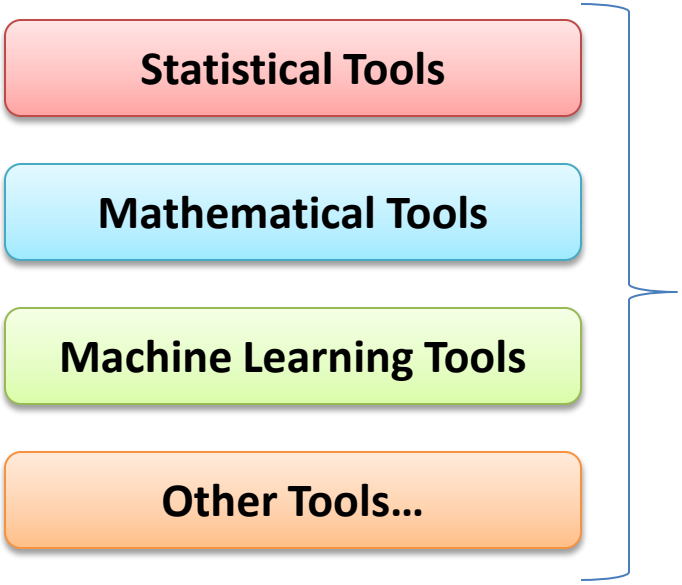
rules			Example
NP	$\longrightarrow$	Det CNP	(the flight)
CNP	$\longrightarrow$	Noun	(flight)
CNP	$\longrightarrow$	CNP PP	(flight to Baghdad)
CNP	$\longrightarrow$	Noun Part	(flight out)
CNP	$\longrightarrow$	PMOD CNP	(7 o'clock flight)
NP	$\longrightarrow$	Noun Num	(flight 321)

To include these cases in the rules of the general sense, we need to add the parameters and characteristics of those words and formulas to determine the grammatical form and morale. Certainly if the domain specific and clear will be the analysis phase grammatical and moral easy to implement. Turning now to our application, which deals with the flights, after updating the rules will become the following formula:

FLY-NP ⟶ Det FLY-CNP	(the flight)
FLY-CNP ⟶ FLY-Noun	(flight)
FLY-CNP ⟶ FLY-CNP FLY-Dest	(flight to Baghdad)
FLY-CNP ⟶ FLY-CNP FLY-Source	(flight from Babylon)
FLY-CNP ⟶ FLY-Noun FLY-Part	(flight out)
FLY-CNP ⟶ FLY-PMOD FLY-CNP	(7 o'clock flight)
FLY-NP ⟶ FLY-Noun NUM	(flight 321)
CITY-NP ⟶ CITY-NAME	(Babylon)
CITY-NP ⟶ Det CITY-CNP	(the city)
CITY-CNP ⟶ CITY-Noun	(city)
CITY-CNP ⟶ CITY-MOD CITY-CNPCITY-MOD ARG	(nearest city to Baghdad)
FLY-DEST ⟶ to CITY-NP	
FLY-DEST ⟶ for CITY-NP	
TIME-QUERY ⟶ When does FLY-NP FLY-VP	

Text Mining

Text Mining is understood as a process of automatically extracting meaningful, useful, previously unknown and ultimately comprehensible information from textual document repositories.



Text Mining
=
Data Mining Tools (applied to text data)
+
basic linguistics

Text Mining tools

- Linguistic analysis
  - Thesauri, dictionaries, grammar analysers etc.

Language Tools

- Machine Translation
- Automatic Feature Extraction
- Keywords Extraction
- Automatic Summarization

Single Document Tools

- Document Categorization
- Document Clustering
- Document Similarity
- Information Retrieval
- Visualization Mmethods

Multiple Document Tools

Feature Extraction

Not all words are equally important

Stop-words

Other Words

- Technical multiword terminology
- Abbreviations
- Relations
- Names
- Numbers

Data bases

Databases

Knowledge discovery in databases

MineIT

Microsoft

Discovering important terms

- Finding lexical affinities
- Dictionary-based methods
- Grammar based heuristics

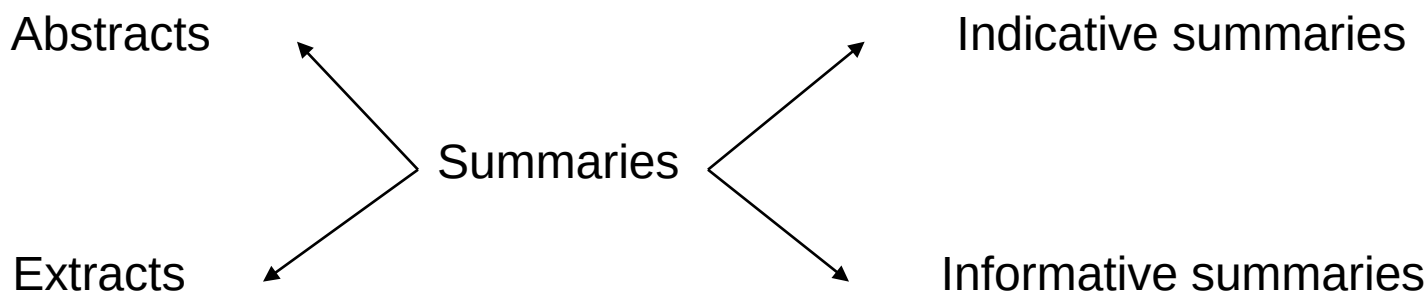
Knowledge discovery in		databases
Knowledge discovery in	large	databases
Knowledge discovery in	big	databases

Prof. Dr. Ahmed Tariq Saadeq

Samples of English Language Stop-words

a	can	everyone	him	may	of	perhaps	that	toward	what
an	could	except	his	me	off	rather	than	un	where
about	cry	few	how	more	often	same	the	under	which
after	detail	fill	if	most	once	seem	they	until	while
again	do	for	in	much	only	serious	then	up	whose
all	done	former	indeed	my	onto	several	thereby	us	why
along	due	from	is	neither	or	she	these	very	with
also	during	get	it	never	other	should	they	very	without
and	each	give	its	next	our	so	this	via	would
as	either	go	last	no	out	some	those	was	yet
be	else	have	least	nor	over	someone	through	we	you
below	enough	hence	less	not	part	since	thus	well	your
both	every	here	many	now	please	therefore	top	were	yours

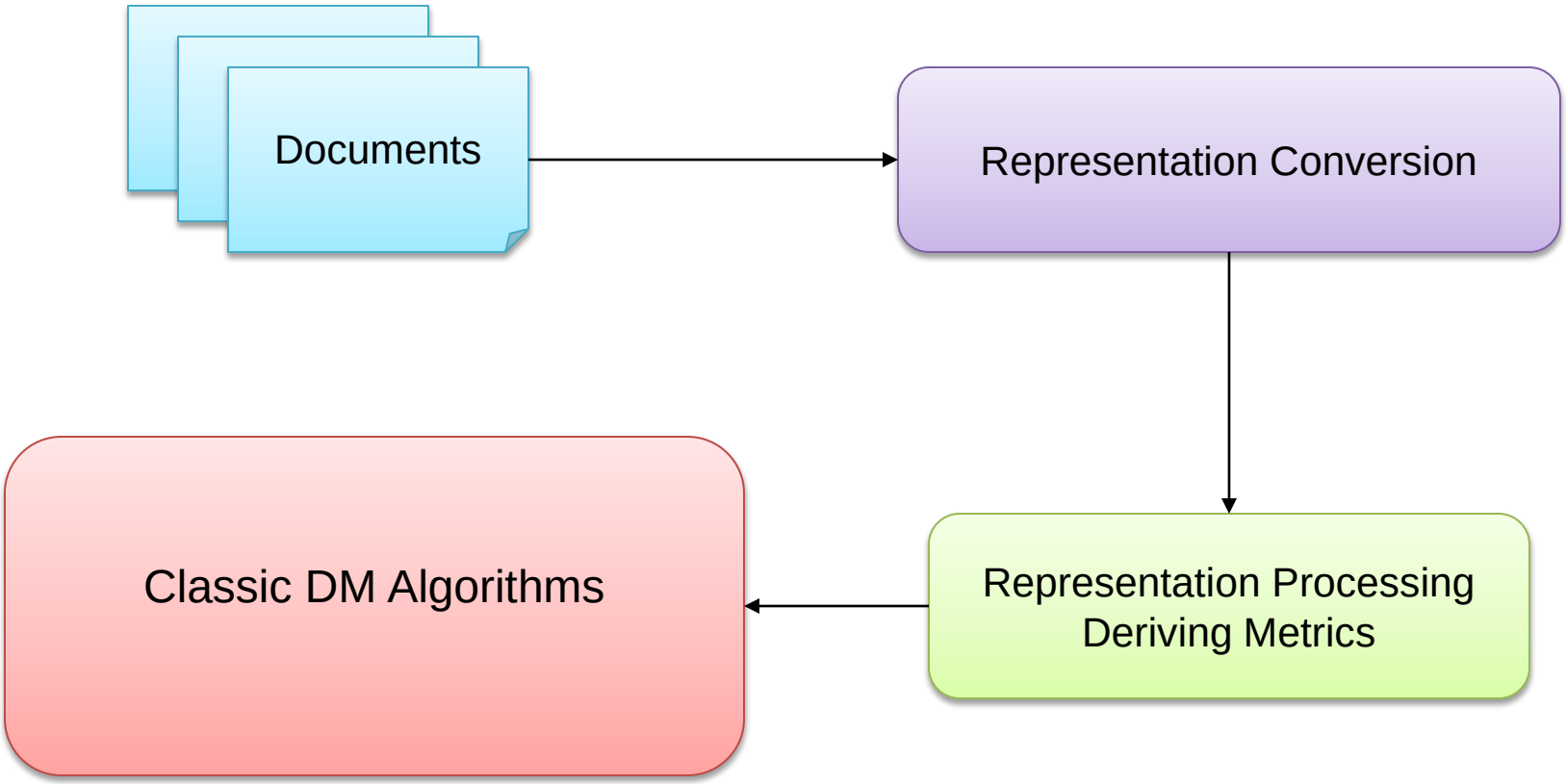
Document (Text) Summarization



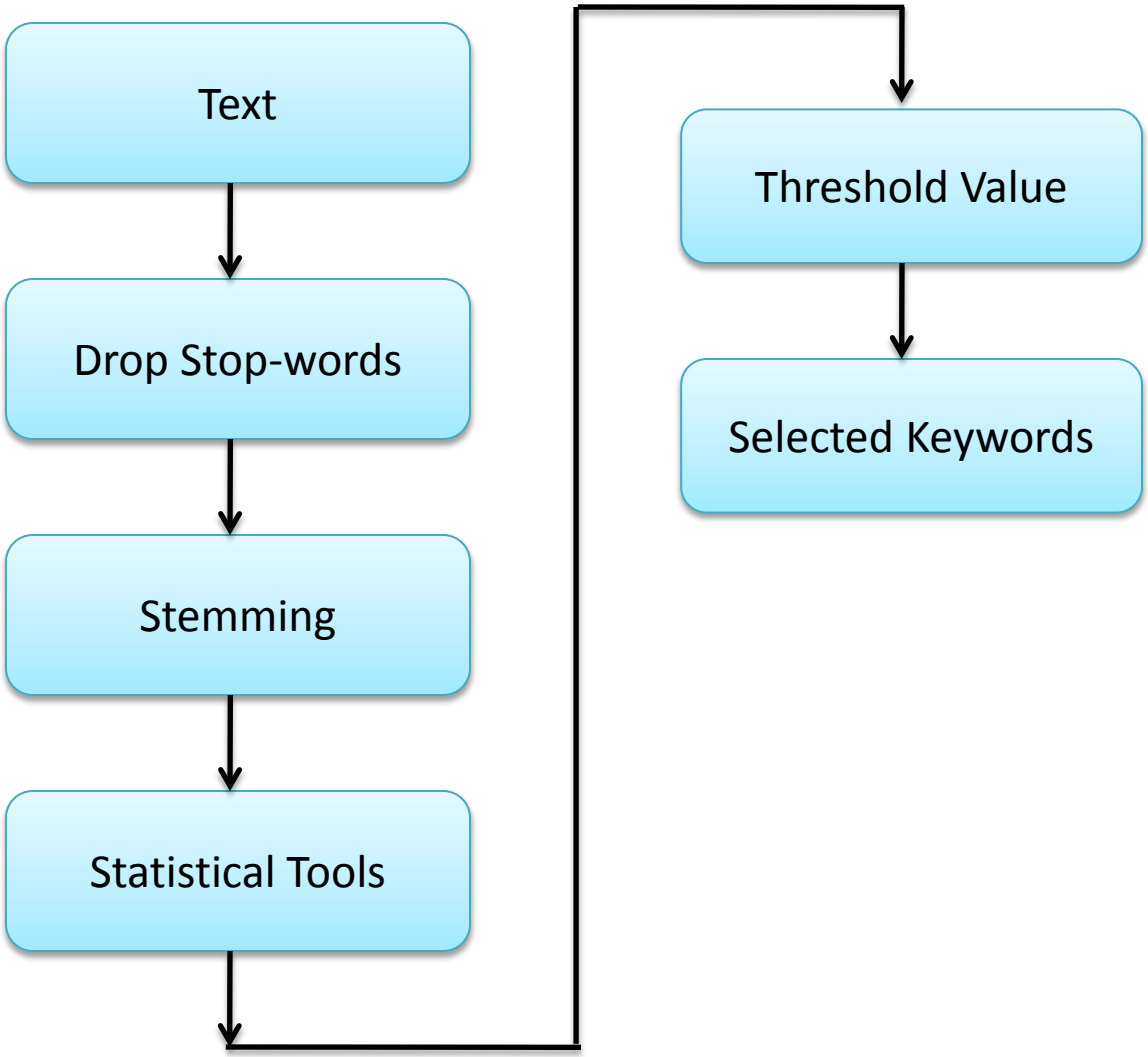
Summary creation methods:

- statistical analysis of sentence and word frequency + dictionary analysis (i.e. „abstract", „conclusion" words etc.).
- text representation methods – grammatical analysis of sentences.
- document structure analysis (question-answer patterns, formatting, vocabulary shifts etc.)

Text Categorization/Clustering/Similarity System



Application: Keywords Extraction



Example:

What is the culture?

- Culture refers to the cumulative deposit of knowledge, experience, beliefs, values, attitudes, meanings, hierarchies, religion, notions of time, roles, spatial relations, concepts of the universe, and material objects and possessions acquired by a group of people in the course of generations through individual and group striving.
- Culture is the systems of knowledge shared by a relatively large group of people.
- Culture is communication, communication is culture.
- Culture in its broadest sense is cultivated behavior; that is the totality of a person's learned, accumulated experience which is socially transmitted, or more briefly, behavior through social learning.
- A culture is a way of life of a group of people--the behaviors, beliefs, values, and symbols that they accept, generally without thinking about them, and that are passed along by communication and imitation from one generation to the next.
- Culture is symbolic communication. Some of its symbols include a group's skills, knowledge, attitudes, values, and motives. The meanings of the symbols are learned and deliberately perpetuated in a society through its institutions.
- Culture consists of patterns, explicit and implicit, of and for behavior acquired and transmitted by symbols, constituting the distinctive achievement of human groups, including their embodiments in artifacts; the essential core of culture consists of traditional ideas and especially their attached values; culture systems may, on the one hand, be considered as products of action, on the other hand, as conditioning influences upon further action.
- Culture is the sum of total of the learned behavior of a group of people that are generally considered to be the tradition of that people and are transmitted from generation to generation.
- Culture is a collective programming of the mind that distinguishes the members of one group or category of people from another.

Stemming And Frequency

Frequency	Word
12	culture , culture , culture , culture , culture , culture , culture , culture , culture , culture , culture, culture
8	group , group , group , group , group, group , groups , group's
5	behavior , behavior , behavior , behavior , behaviors
5	symbolic , symbols , symbols , symbols , symbols
4	communication , communication , communication , communication
4	generation , generation , generation , generations
4	learned , learned , learned , learning
4	values , values , values , values
3	knowledge , knowledge , knowledge
3	transmitted , transmitted , transmitted
2	acquired , acquired
2	action , action
2	attitudes , attitudes
2	beliefs , beliefs
2	experience , experience
2	meanings , meanings
2	social , socially
2	systems , systems
2	tradition , traditional
1	accumulated, achievement, artifacts, attached, broadest, category, collective, conditioning, constituting, core, cultivated, cumulative, deliberately, deposit, distinctive, distinguishes, embodiments, essential, explicit, generally, hierarchies, human, ideas, imitation, implicit, individual, influences, institutions, large, material, mind, ,motives, notions, objects, passed, patterns, people, perpetuated, possessions, products, programming, relations, relatively, religion, roles, shared, skills, spatial, striving, thinking, time, universe

-Drop the stop-words

- Culture cumulative deposit knowledge, experience, beliefs, values, attitudes, meanings, hierarchies, religion, notions time, roles, spatial relations, universe, material objects possessions acquired group generations individual group striving.
- Culture systems knowledge shared relatively large group.
- Culture communication, communication culture.
- Culture broadest cultivated behavior; learned, accumulated experience socially transmitted, behavior social learning.
- culture group behaviors, beliefs, values, symbols accept, generally thinking passed communication imitation generation.
- Culture symbolic communication. symbols group's skills, knowledge, attitudes, values, motives. meanings symbols learned deliberately perpetuated institutions.
- Culture patterns, explicit implicit, behavior acquired transmitted symbols, constituting distinctive achievement human groups, embodiments artifacts; essential core culture traditional ideas attached values; culture systems products action, conditioning influences action.
- Culture learned behavior group tradition transmitted generation generation.
- Culture collective programming mind distinguishes group category people

Simple Statistical Tools, Threshold & Output

Total number of words = 122

Words	Frequency	Weight
culture	12	0.098
group	8	0.065
behaviors , symbols	5	0.041
communication, generation, learned, values	4	0.032
knowledge, transmitted	3	0.024
acquired, action, attitudes, beliefs, experience, meanings, social, systems, tradition	2	0.016
accumulated, achievement, artifacts, attached, broadest, category, collective, conditioning, constituting, core, cultivated, cumulative, deliberately, deposit, distinctive, distinguishes, embodiments, essential, explicit, generally, hierarchies, human, ideas, imitation, implicit, individual, influences, institutions, large, material, mind, ,motives, notions, objects, passed, patterns, people, perpetuated, possessions, products, programming, relations, relatively, religion, roles, shared, skills, spatial, striving, thinking, time, universe	1	0.008

No. of all Words = 122

Threshold = 4 keywords

i.e. the output is the highest 4 word scores.

The keywords are:

Culture, group, behaviors, & symbols

Expert Systems

The expertise possessed by man in any discipline or field of disciplines and areas of life does not come easily but is the result of painstaking effort the work of a large and sometimes interfere with the gift of man and his nature to be creative in his field, but we must exploit this experience in the development of this specialization and of people who work in it from hand to hand another store this experience to utilize them in the future in the development of future generations development. The human experience is an asset dear and can not be easily compromised, therefore, nations and civilizations progress and evolve as a result of the accumulation of experience with and benefit from the experiences of others. Hence the idea of expert systems, they are systems that solve complex problems, the way you solve the human expert, and provide advice and interpretation as provided by the human expert in his field.

Prof. Dr. Ahmed Tariq Saadeq

Expert System Components

- **Knowledge Base**
- **Search**
- **Heuristic**
- **Inference Engine**
- **User Interface**
- **Explanation**

Expert System Components

1. **Knowledge Base** : It is the basis of an expert system, since it is human experience, information and data concerning the description and problem-solving that are meant to solve an expert system and to provide advice regarding its own.
2. **Search** : Apply a method on the knowledge base to get to the proper solution(s) to the situation in the space problem is concerned.
3. **Heuristic** : What is meant by any assistance or additional information, or mathematical values additional help in the research stage to reach the right solution.
4. **Inference Engine** : It is responsible for logical base of knowledge and provide logical explanations necessary for the beneficiary conclusion mechanism (if it so wishes) and through a variety of ways (we will talk about later).
5. **User Interface** : namely, the way dialogue with an expert system with its beneficiaries, Sometimes a conversation is through the sentences or texts of a natural language (Arabic or English .... etc), the user asks and answers expert system
6. **Explanation** : its task to provide the logical explanation for the results, path of advises and persuasion the users that the results is true.

It is better to separate between Knowledge Base & Inference Engine !!!

In expert systems, in general, it is important to sustain the separation between the Knowledge Base and the Inference Engine for several reasons: -

1. This separation makes it possible to represent the natural patterns of knowledge more. For example, if the base (if - then) (If-Then) are closer to the way in which people describe their skills in solving problems better than the other way.
2. Because the knowledge base is separate from the control structures of programs, the expert system designers can focus on the collection and organization of special knowledge by better than focusing on details in other applications problems.
3. Ideally, the separation of knowledge and control allows for alterations to put new knowledge base without the affected parts for control of knowledge.
4. The separation of knowledge and control elements of the program allows for the control of the same software interface to be used in a variety of systems.

When we can use the Expert Systems?

Expert systems ensure a significant investment of money and human effort, an attempt to resolve the problem that is very complex, very bad concept, except, it is not suitable for the available technology that could lead to price rises and critical failures. Researchers have a level of evidence or to set that were appropriate to resolve the problem and expert system as follows: -

1. The need to resolve justify the cost and effort to build an expert system: many expert systems built in areas such as medicine, engineering, mineral exploration, business administration and defense, where there is great potential benefit in terms of time and quality of solutions and preserving the lives of humans.
2. human experiences are not available in all cases whenever you need it: Geology For example, there is a need for expertise in mining and drilling sites remotely, as well as earth scientists and other engineers have found themselves traveling long distances to visit the places on site with the resulting expense and lost time . But using expert systems in place in distant places, many of the problems can be solved without the need to increase the sites. The same situation in the medical field, there are many skilled doctors can benefit from their experience through the design of expert systems.
3. problem as possible be resolved using symbolic thinking: solutions to the problem should not require physical skill or cognitive sensory skill. Robotics and visual system at the present time evolving in this direction.

When we can use the Expert Systems?

4. The problem area is structured well, and does not require thinking with common sense: high-tech fields have the benefits of being good studies of the formation of good, because ferries are well and have knowledge of the areas of particular concepts and clear methods.
5. problems that are not solved using conventional computational theories: that the expert system technology may not be used unnecessarily. If the problem was resolved reasonably satisfactory and there is no need for the use of expert systems.
6. The presence of the obvious expertise and collaborating: expert knowledge used in the system comes from the experience and judgment on the work of the people in the area. It is important that these experiences be able to participate and to give them the required data and information.
7. The problem is in terms of volume and appropriate term: For example, a program that tries to get the expertise of the doctor just can not be appropriate: the program that provides medical advice on the use of the results and tests for diagnostic devices with a range of regular diagnoses could be more Relevance.

Expert System Types

- **Rule-Based Expert Systems.**
- **Case-Based Expert Systems.**
- **Model-Based Expert Systems.**

Rule-Based Expert Systems

Expert systems based on rules occupies the largest space among other species, as the vast majority of expert systems rely on this particular type applications. These systems represent a special knowledge of solving - problem-mediated rules (If-Then), one of the oldest techniques to represent the field of knowledge in expert system, one of the natural methods and the remaining wide use in the practical and experimental fields of expert systems. Architectural expert systems based on rules based on the productive system models (Production System) to represent the solutions to the problem.

If (Conditions or Events) **Then** (Actions or Results)

Why Rule-Based E.S. ?

The preference which won him these types of expert systems namely, the approved rules is due to several reasons :

1. The rules style often reflects expert human framework to resolve particularly the one hand and described a good problem.
2. The rules are a natural model for the structuring of the problem and how to solve them. It features easily added to the system and easily modified afterwards that the need arises.
3. Ease of implementation of logical conclusion the process of using these rules in expert systems is one of the most important reasons for the prevalence of use.
4. Using rules is a more coherent system and the strength of the hand to provide explanations for the beneficiary, especially explanatory (why would you want that?) & (How do you know that?).

Rule-Based E.S. Driving

Backward Driving

Goal ⟶ **Data**

Forward Driving

Data ⟶ **Goal**

Useful for

- **Classification**
- **Diagnosis**
- **Planning**
- **Desining**

Rule-Based E.S. Programs

Case Study : Animals Classification

Backward Chaining

```
guess_animal :- identify(X), write("Your animal is a(n) ",X),!.

identify(giraffe) :- it_is(ungulate), confirm(has, long_neck), confirm(has, long_legs),
                     confirm(has, dark_spots), !.
identify(zebra) :- it_is(ungulate), confirm(has, black_strips),!.
identify(cheetah) :- it_is(mammal), it-is(carnivorous), confirm(has, tawny_color),
                     confirm(has, black_spots),!.
identify(tiger) :- it_is(mammal), it-is(carnivorous), confirm(has, tawny_color), confirm(has, black_strips),!.
identify(eagle) :- it_is(bird), confirm(does, fly), it-is(carnivorous), confirm(has, use_as_national_symbol),!.
identify(ostrich) :- it_is(bird), not(confirm(does, fly)), confirm(has, long_neck), confirm(has, long_legs),!.
identify(penguin) :- it_is(bird), not(confirm(does, fly)), confirm(does, swim),
                     confirm(has, black_and_white_color),!.
identify(blue_whale) :-  it_is(mammal), not(it-is(carnivorous)), confirm(does, swim),
                     confirm(has, huge_size),!.
identify(octopus) :- not(it_is(mammal), it_is(carnivorous), confirm(does, swim), confirm(has, tentacles),!.
identify(sardine) :- it_is(fish), confirm(has, small_size), confirm(has, use_in_sandwiches),!.

identify(unknown).
```

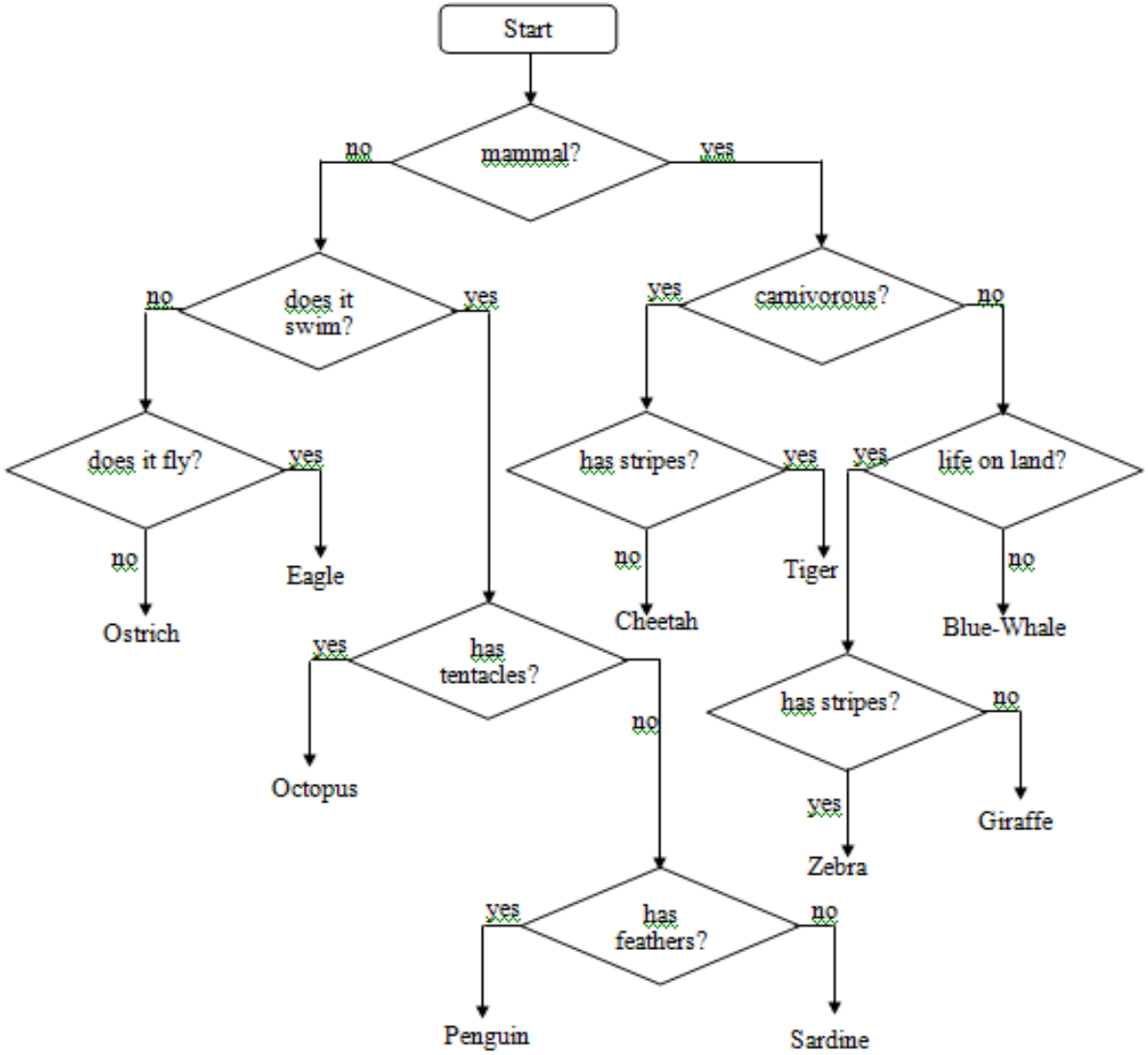
```
it-is(bird):- confirm(has, feathers), confirm(does, lay_eggs),!
it-is(fish):- confirm(does, swim), confirm(has, fins),!.
it-is(mammal):- confirm(has, hair),!.
it-is(mammal):- confirm(does, give_milk),!.
it-is(ungulate):- it-is(mammal), confirm(has, hooves), confirm(does, chew_cud),!.
it-is(carnivorous):- confirm(has, pointed_teeth),!.
it-is(carnivorous):- confirm(does, eat_meat),!.
```

```
confirm(X,Y):- db_confirm(X,Y),!.
confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).
denied(X,Y):- db-denied(X,Y),!.

check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y, Reply).

remember(X, Y, yes):- asserta(db_confirm(X, Y)).
remember(X, y, no):- assereta(db_denied(X, Y)), fail.
```

Rule-Based E.S. Programs

Case Study : Animals Classification

Forward Chaining



```
guess_animal :-
find_animal, have_found(X), write("Your animal is a(n) ",X),nl,!.

find_animal:- test1(X), test2(X,Y), test3(X,Y,Z), test4(X,Y,Z,_),!.
find_animal.

test1(m):- it_is(mammal),!.
test1(n).

test2(m,c):- it_is(carnivorous),!.
test2(m,n).
test2(n,w):- confirm(does, swim),!.
test2(n,n).

test3(m,c,s):- confirm(has, strips), asserta(have_found(tiger)),!.
test3(m,c,n):- asserta(have_found(cheetah)),!.
test3(m,n,l):- not(confirm(does, swim)), not(confirm(does, fly)),!.
test3(m,n,n):- asserta(have_found(blue_whale)),!.
test3(n,n,f):- confirm(does, fly), asserta(have_found(eagle)),!.
test3(n,n,n):- asserta(have_found(ostrich)),!.
test3(n,w,t):- cofirm(has, tentacles), asserta(have_found(octopus)),!.
test3(n,w,n).
```

```
test4(m,n,l,s):- confirm(has, strips), asserta(have_found(zebra)),!.
test4(m,n,l,n):- asserta(have_found(giraffe)),!.
test4(n,w,n,f):- confirm(has, feathers), asserta(have_found(penguin)),!.
test4(n,w,n,n):- asserta(have_found(sardine)),!.

it-is(bird):- confirm(has, feathers), confirm(does, lay_eggs),!.
it-is(fish):- confirm(does, swim), confirm(has, fins),!.
it-is(mammal):-  confirm(has, hair),!.
it-is(mammal):-  confirm(does, give_milk),!.
it-is(ungulate):-  it-is(mammal), confirm(has, hooves), confirm(does, chew_cud),!.
it-is(carnivorous):- confirm(has, pointed_teeth),!.
it-is(carnivorous):- confirm(does, eat_meat),!.


confirm(X,Y):- db_confirm(X,Y),!.
confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).
denied(X,Y):- db-denied(X,Y),!.

check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y, Reply).

remember(X, Y, yes):- asserta(db_confirm(X, Y)).
remember(X, y, no):- assereta(db_denied(X, Y)), fail.
```

Uncertainty Inference in E.S.

Prof. Dr. Ahmed Tariq Saadeq

If (Evidence) Then (Implication)

ct(e) ct(i)

A value (ct (e)) means the value of uncertainty for evidence or condition and value (ct (i)) means the value of uncertainty for output or modulated. For example, if we assume the following rule:

If (it is raining) Then (it is winter)

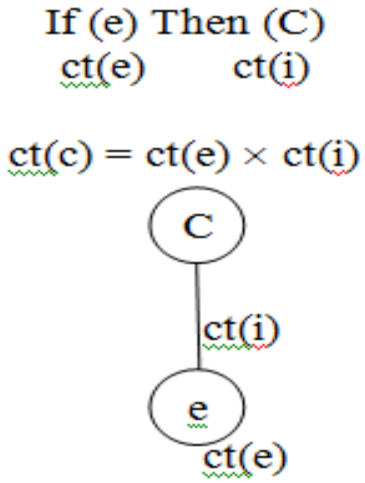
0.6 0.25

ct(conclusion) = ct(c) = ct(e) × ct(i)

= 0.6 * 0.25 = 0.15

There are several cases for rule types :

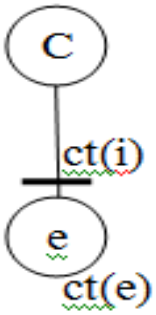
1- Single rule.



2- Negation single rule.

If not(e) Then (C)
ct(e) ct(i)

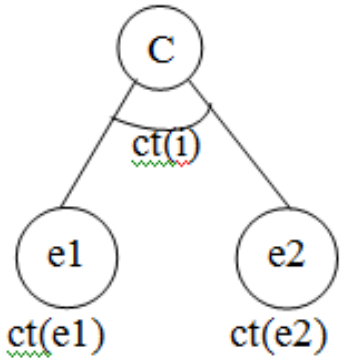
$ct(c) = -ct(e) \times ct(i)$



3- AND Logical rule.

If (e1 and e2) Then (C)

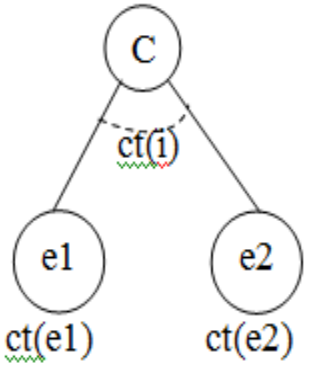
$ct(c) = Min(ct(e1), ct(e2)) \times ct(i)$



4- OR Logical rule.

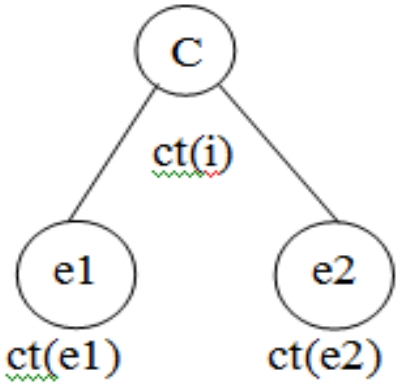
If (e1 and e2) Then (C)

$ct(c) = Max(ct(e1), ct(e2)) \times ct(i)$



5- Multi-rule Inference.

If (e1) Then (C)
If (e2) Then (C)



(+, +) $ct(total) = ct1 + ct2 - ct1 \times ct2$

(-, -) $ct(total) = ct1 + ct2 + ct1 \times ct2$

(+, -) $ct(total) = (ct1 + ct2) / (1 - Min(Abs(ct1), Abs(ct2)))$

Note:-

r : means Reversible rule.

n : means Non-Reversible rule.

The rule will be drop when it is (Negative & Non-Reversible).

Example:

Assume you have the following tree rules, find the value of C5.

<u>Solution :</u>

Find C1value by apply rule no. 1

$ct(C1) = 0.8 \times 0.9 = 0.72$

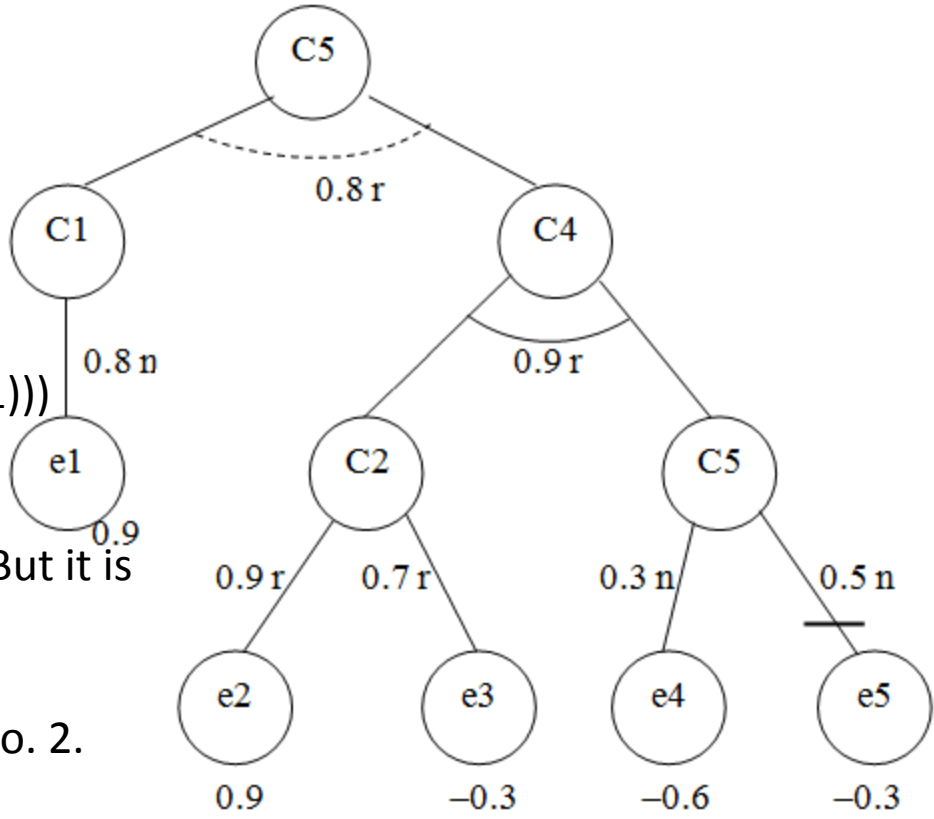
Find C2 value by apply rules no. 5

$ct(c2_1) = 0.9 \times 0.9 = 0.81$

$ct(c2_2) = -0.3 \times 0.7 = -0.21$

$ct(C2) = 0.81 + (-0.21) / (1 - \text{Min}(\text{Abs}(0.81), \text{Abs}(-0.21)))$

$= 0.74$



Find one branch of C3 value by apply rule no. 2. But it is Negative & Non-Reversible, so, it is dropped.

$ct(C3_1) = -0.6 \times 0.3 = -0.18$

Find the other branch of C3 value by apply rule no. 2.

$ct(C3_2) = -(-0.3) \times 0.5 = 0.15$

Find C4 value by apply rule no. 3.

$ct(C4) = 0.9 \times \text{Min}(0.74, 0.15) = 0.9 \times 0.15 = 0.13$

Find C5 value by apply rule no. 4.

$ct(C5) = 0.8 \times \text{Max}(0.72, 0.13) = 0.9 \times 0.72 = 0.58$

i.e. the uncertainty of C5 = 0.58

Example:

Compute the uncertainty values for the following rules:

If you have flu & vulnerable age then see doctor (0.8)

If you have strep throat then see doctor (0.9)

If you have cold then get rest & take aspirin (0.5)

If you have flu & not vulnerable age then get rest & take aspirin (0.4)

If you have fever & muscle ache then you have fever (0.8)

If you have muscle aches & sniffles & no fever then you have cold (0.8)

If you have blotches of throat & you have fever then you have strep throat (0.8)

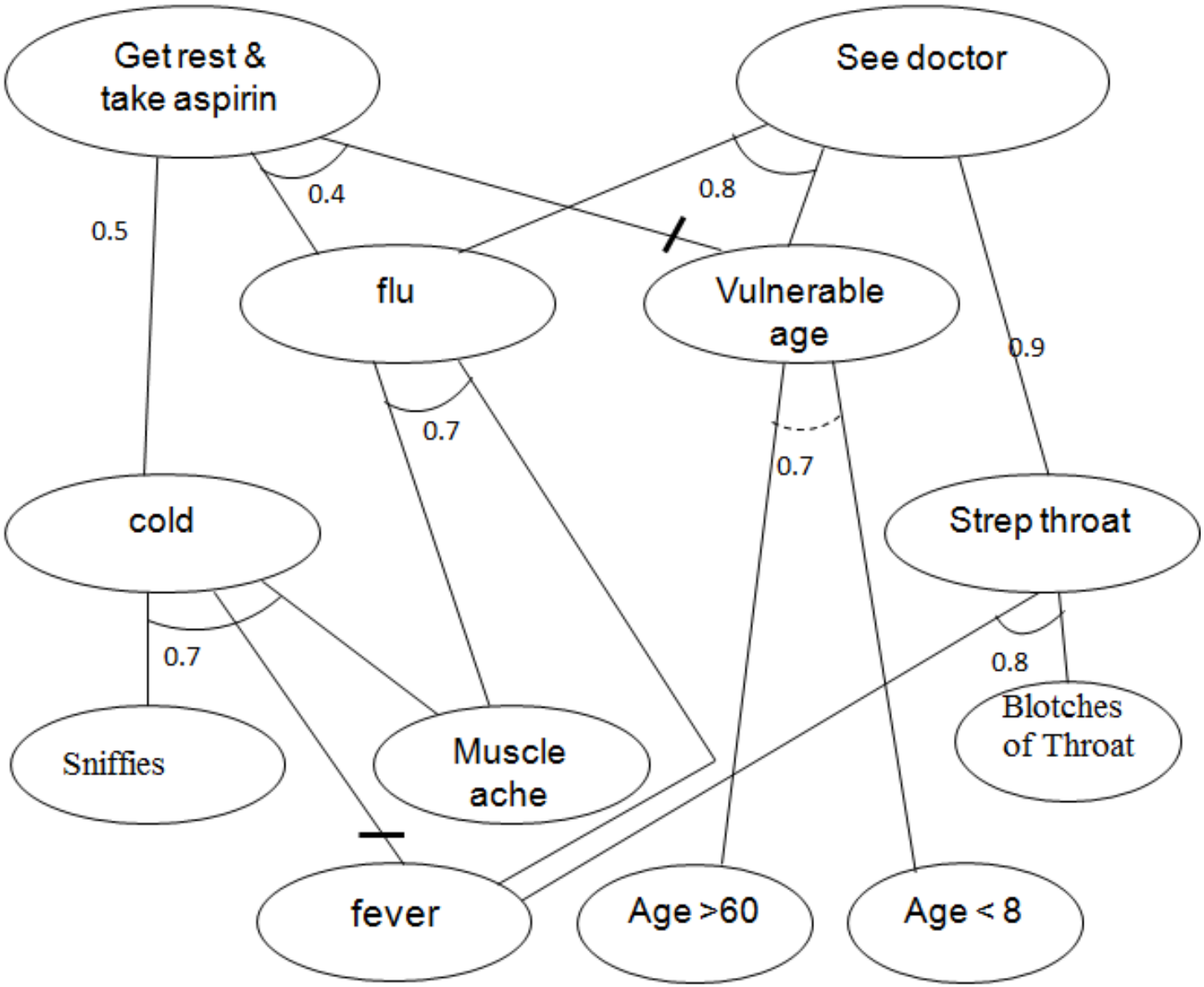
If you age less than 8 years & greater than 60 years then your age is vulnerable (0.8)

Assume that the uncertainty value as below:

fever (0.8), muscle aches (0.7), sniffles (0.7), blotches of throat (0.9), age less than 8 years (0.8) and age greater than 60 years (0.9).

<u>Solution</u>

It is best to draw the tree for the above rules as:


Get rest & take aspirin
See doctor
0.4
0.8
0.5
flu
Vulnerable age
0.9
0.7
0.7
cold
Strep throat
0.7
0.8
Sniffies
Muscle ache
Blotches of Throat
fever
Age >60
Age < 8

Compute the uncertainty value of (strep throat),

$$= 0.8 \times \text{Min}(0.8, 0.9) = 0.64$$

Compute the uncertainty value of (vulnerable age),

$$= 0.7 \times \text{Max}(0.8, 0.9) = 0.63$$

Compute the uncertainty value of (flu),

$$= 0.7 \times \text{Min}(0.8, 0.7) = 0.49$$

Compute the uncertainty value of (cold),

$$= 0.7 \times \text{Min}(0.7, 0.7) = 0.49$$

Drop the fever because it is negative & non-reversible.

Compute the uncertainty value of (get rest & take aspirin),

It has 2 branch,

The first $= 0.49 \times 0.5 = 0.245$

The second $= 0.4 \times 0.49 = 0.196$

Drop the age because it is negative & non-reversible.

Compute the uncertainty value of (get rest & take aspirin),

$$= 0.245 + 0.196 - 0.245 \times 0.196 = 0.393$$

Compute the uncertainty value of (see doctor),

It has 2 branch,

The first $= 0.9 \times 0.64 = 0.576$

The second $= 0.8 \times \text{Min}(0.49,0.63) = 0.392$

The total value

$$= 0.576 + 0.392 - 0.576 \times 0.392 = 0.742$$

Prolog Program for Uncertainty Computations

Prof. Dr. Ahmed Tariq Saadeq

```
driver:- hypothesis-node(X), allinfer(X, Ct),  write("The certainty for ", X, "is", Ct), nl, fail.

allinfer(Node):- findall(C1, infer(Node, C1), Ctlist), supercombine(Ctlist, Ct).

infer(Node, Ct):- imp(s, Use, Node1, Sign, Node2, _, _, C1), allinfer(Node2, C2),
find_multiplier(Sign, Mult, dummy, 0), CS = Mult * C2, qualifier(Use, CS, Qmult),
Ct = CS * C1 * Qmult.

infer(Node1, Ct):- imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1),
allinfer(Node2, C2), allinfer(Node3, C3), find_multiplier(SignL, MultL, SignR, MultR),
C2S = MultL * C2, C3S = MultR * C3, min(C2S, C3S, CX), qualifier(Use, CX, Qmult),
Ct = CX * C1 *Qmult.

infer(Node1, Ct):- imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1),
allinfer(Node2, C2), allinfer(Node3, C3), find_multiplier(SignL, MultL, SignR, MultR),
C2S = MultL * C2, C3S = MultR * C3, max(C2S, C3S, CX), qualifier(Use, CX, Qmult),
Ct = CX * C1 *Qmult.

infer(Node1, Ct):- terminal_node(Node1), evidence(Node1, Ct),!.

infer(Node1, Ct):- terminal_node(Node1), write("What is the certainty for node", Node1),
nl, readreal(Ct), asserta(evidence(Node1, Ct)),!.
```

```
find_multiplier(pos, 1, dummy, 0).
find_multiplier(neg, -1, dummy, 0).

find_multiplier(pos, 1, pos, 1).
find_multiplier(pos, 1, neg, -1).
find_multiplier(neg, -1, pos, 1).
find_multiplier(neg, -1, neg, -1).

supercombine([Ct], Ct):-!.
supercombine([C1, C2], Ct):- combine([C1, C2], Ct), !.
supercombine([C1, C2|T], Ct):- combine([C1, C2], C3), append([C3], T, TL),
supercombine(TL, Ct), !.

combine([-1, 1], 0).
combine([1, -1], 0).
combine([C1, C2], Ct):- C1 >= 0, C2>= 0, Ct = C1 + C2 - C1 * C2.
combine([C1, C2], Ct):- C1 < 0, C2< 0, Ct = C1 + C2 + C1 * C2.
combine([C1, C2], Ct):- C1 < 0, C2 >= 0, absvalue(C1, Z1), absvalue(C2, Z2),
min(Z1, Z2, Z3), Ct = (C1 + C2) / (1 – Z3).
combine([C1, C2], Ct):- C2 < 0, C1 >= 0, absvalue(C1, Z1), absvalue(C2, Z2),
min(Z1, Z2, Z3), Ct = (C1 + C2) / (1 – Z3).
```

```
absvalue(X, Y):- X = 0, Y = 0, !.
absvalue(X, Y):- X > 0, Y = X, !.
absvalue(X, Y):- X < 0, Y = -X, !.

qualifier(Use, C, Qmult):- Use = "r", Qmult = 1, !.
qualifier(Use, C, Qmult):- Use = "n", C >= 0, Qmult = 1, !.
qualifier(Use, C, Qmult):- Use = "n", C < 0, Qmult = 0, !.
```

The representation facts as:

Imp(T, M, R, S1, E1, S2, E2, V)

where

T : rule type (s: single, a: and, o: or), M : (r: reversible, n: non-reversible),
R : name of result, S1: sign of first condition, E1: name of first condition,
S2: sign of second condition, E2: name of second condition, V : value of results

For example:

If e1 then C2 r (0.1)
If e1 and e4 then C5 n (0.7)
If e2 or e3 then C1 r (0.4)

as:

Imp(s, r, C2, pos, e1, null, null, 0.1)
Imp(a, n, C5, pos, e1, neg, e4, 0.7)
Imp(o, r, C1, pos, e2, pos, e3, 0.4)

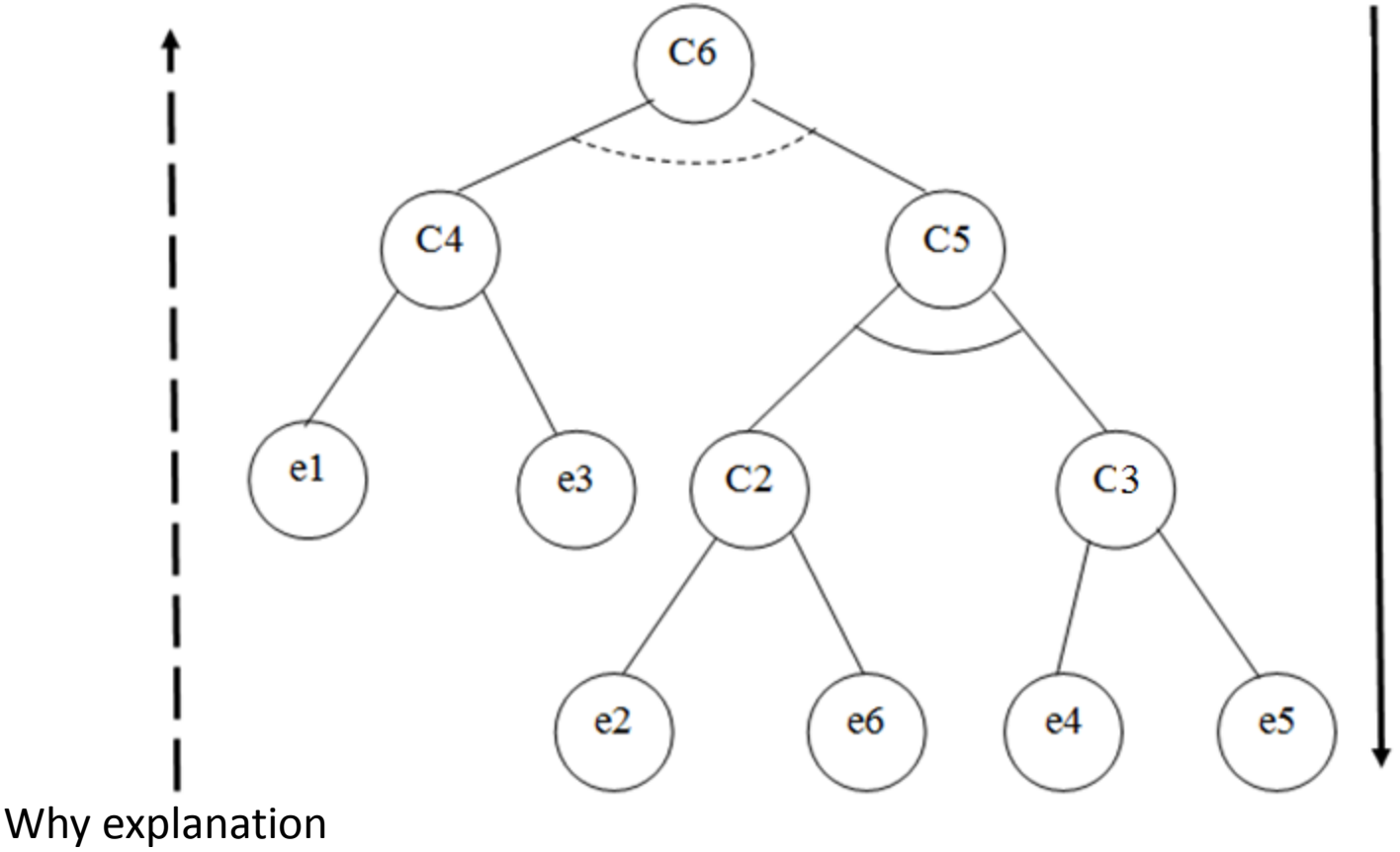
Simple Explanation in Rule-Based E.S.

The explanations is of important requirements in the expert system must be provided to the beneficiaries in order to obtain the reliability of the best, and intended explanation here to provide rationales for the course of events and the questions that will be presented to the user, and also provide a logical explanation of how to get the result or final advice to be provided by an expert system. The most important explanations must provided by the expert system are:

1- The question (why?) (Why?), it is the explanation that the user wants the beneficiary of the functioning of the system in the process of asking questions on the beneficiary, and the purpose of submitting this inquiry to provide explanation about the importance of this question or the current demand in both certainty value or answered. So, this explanation works during the implementation of an expert system, starting from the first condition or event down to the result.

2- The question (How?) (How?), which is the explanation that the beneficiary wants to answer him after receiving the result or the advice of an expert system, which it wants the inquiry system providing how they have been through it to get to this result or any advice he wants conviction the reliability of expert system and method of work to get to the solution. Work is under way in this inquiry after obtaining the result or advice.

How explanation



Why explanation

Prolog Program for E.S. Explanation

```
infer(Node1, Ct):-
imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1),
assserta(dbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)),
assserta(tdbimp(a, Use, Node1, SignL, Node2, SignR, Node3,C1)),
allinfer(Node2, C2), allinfer(Node3, C3), find_multiplier(SignL, MultL, SignR, MultR),
C2S = MultL * C2, C3S = MultR * C3, min(C2S, C3S, CX), qualifier(Use, CX, Qmult),
Ct = CX * C1 *Qmult, assertz(infer_summary(
imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1), Ct)),
retract(dbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)),
retract(tdbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)).
```

/* How Facility Sub Program */

```
Exsys_driver :- getallans, showresults,!.
Getallans :- not(prepare_answer).
Prepare_answer :- answer(X, Y), fail.
answer(X, Y) :-  hypothesis_node(X), allinfer(X, Y), assert(danswer(X, Y)).
Showresults :- not(displayall).
displayall :- diplay_aoe_answer, fail.
diplay_aoe_answer :- danswer(X, Y), clearwindow, write("For this hypothesis:"), nl,
write(" ", X),nl, write("The certainty is:", Y), nl, nl, not(how_describer(X)).

how_describer(Node) :- repeat, nl, write("Type h(how) nodename, or c(to continue),"),
nl, readln(Reply), nl, how_explain(Reply),!.
```

Prolog Program for E.S. Explanation

```
how_explain(Reply) :- Reply = "c".
how_explain(Reply) :- fronttoken(Reply, _, X1), fronttoken(X1, X, _),
infer_summary(imp(_, _, X, _, _, _, _, _), _),  clearwindow,!,
write("The rule(s) that bear upon this conclusion are:"), nl, nl,
 infer_summary(imp(A, A1, X, R, S, C, D, E),F), write("Concluded: ", X), nl, gettype(A, Z),
write("from an ", Z), nl, write(" premise 1 was: ",S), nl, write(" premise 2 was: ",D), nl,
write("The certainty from use of this rule alone was: ",F), nl, nl, fail.
how_explain(Reply) :- fronttoken(Reply, _, X1), fronttoken(X1, X, _), terminal_node(X),
evidence(X, C), write("You told me that: "), nl, write(" ", X), nl, write("has a certainty of:
",C), nl, fail.
```

/* Why Facility Sub Program */

```
infer(Node, Ct) :- terminal_node(Node), evidence(Node, Ct), !.
infer(Node, Ct) :- terminal_node(Node), repeat, nl, write("Type w(why) or give the certainty
for node ", Node), nl, readln(Reply), reply_to_input(Node, Reply, Ct), !.
reply_to_input(Node, Reply, Ct) :- not(isname(Reply)), adjuststack, str_real(Reply, CT),
asserta(evidence(Node,Ct)),!.
reply_to_input(_, Reply, _) :- isname(Reply), Reply = "w", nl, dbimp(U, V, R, S, S1, X, Y, Y1),
why_describer(U, V, R, S, S1, X, Y, Y1), retract(dbimp(U, V, R, S, S1, X, Y, Y1)),
putadjustflag, pauser, !, fail.
```

Prolog Program for E.S. Explanation

```
why_describer(U, U1, V, R, S, X, Y, Z) :- clearwindow, nl, U <>"s", gettype(U,UU),
write("I am trying to use an inference rule of the type "), nl, write(UU), write(", to support
the conclusion: "), nl, write(" ", V), nl, write("Premise 1 is: ",S), nl, getmode(R, RR),
write(" This premise will be used ", RR), nl, write("Premise 2 is: ",Y),nl, getmode(X, XX), nl,
write(" This premise will be used ", XX), nl, write("The certainty of the implication is: ", Z),
nl, !.
why_describer("s", V1, V, R, S, X, Y, Z) :- clearwindow, nl,
write("I am trying to use an inference rule of the type "), nl,
write("simple implication, to support the conclusion: "), nl,
write(" ", V), nl, write("premise 1 is: ", S), nl,
getmode(R, RR), write(" This premise will be used ", RR), nl
write("The certainty of the implication is: ", Z), nl, !.

gettype("a", "and implication").
gettype("o", "or implication").
gettype("s", "simple implication").

getmode("pos", "as you see it.").
getmode("neg", "prefaced by not.").
```

Concept of E.S. Shell

Designers and research centers and major companies specialized in software provide a software tool or integrated software simplifies the work of expert systems in many ways. This software called the (Shell) expert systems, and provides the kind of software the following services:

1. Easy to insert knowledge bases and included in the expert system, it provides the interfaces and the text editor and tools to facilitate the process of dealing with the knowledge base and make it easier to deal with in terms of expert system knowledge base is included, classifying and organizing within the expert system. Update process also facilitates and deletions as well as a knowledge base for the purpose of continuing the updating.
2. This tool facilitate the process of dealing with inquiries he wants beneficiary through modification, as well as a follow-up process and correction (Trace & Debug) for the steps of the implementation of an expert system, and this is very useful in cases of testing and confirm the rules contained in the expert system because they may contain some mistakes as a result of several things (e.g, an error in the input, the fault of the human source, or any type of error).

3. Ease of construction of a variety of expert systems through expert system shell because it provides single inference engine is working on several different applications in the knowledge base. It may take some action of a specific expert system and add it to another expert system, this is also provided by the expert system shell.
4. In many cases the human expert does not know that knowledge base subject to a reduction through the application of smart computation theories, some types of software expert system shell provide this feature in its ability to reduce the knowledge base leading to a reduction of the time of conclusion and reduce storage area of these the rules.
5. Some shell systems contain the possibility of providing tools to deal with the human expert for the purpose of acquiring knowledge of it without the need to conduct interviews with the designers and developers of software expert systems or even without the need for methods of acquiring knowledge, such as the questionnaire or other. Be beneficial to both sides of the human expert designers and expert system this property.

Sample of E.S. Shells

There are some software that offers expert in various disciplines and programming languages system shell, which have been selected from among many of these software.

1- **PESS**: an acronym (Prolog Expert System Shell) is a software package offers many possibilities to build an integrated expert system Prolog language is one of the cortex broader systems used.

2- **JESS**: an acronym (Java Expert System Shell) is a software package works in the Java language environment and provides a framework and multiple potential to build an integrated expert system is one of the widely-used systems shell.

3- **Xpert Rule**: a system that provides many capabilities in addition to the potential and facilities expert systems such as smart analysis and work in a variety of environments such as the various operating systems and systems of mobile phones and the creation of expert systems applications on the Internet pages of the (Web) and servers (Servers).

4- **FuzzyShell**: This shell is working style logic Almillb for the formulation of expert systems and are important in applications that require this type of processors.

5- **Clips**: This provides a kind of expert system shell possibility of drafting a number of technologies such as those based on the rules, based on object-oriented and procedural programming. It provides an environment similar to the language (LISP).

6- **Drools5**: This is the kind of expert system shell of the species that provides multiple capabilities for the formulation of a potential expert more than those of any regular system because it provides the potential for additional system expert and one of the systems deployed use.

Artificial Neural Networks (ANN)

Artificial neural networks are one of the most important techniques of artificial intelligence in the field of machine learning, and perhaps occupies center stage in this field among the different technologies in the field of machine learning, given the size of the areas and applications where this technique was used, the longer the machine learning using this technique the kind of clear (Explicit Learning).

Historical Background

In 1943 two scientists (Warren McCulloch) and (Walter Pitts) invent the first computational model for artificial neural networks relying on math. and algorithms, called (Threshold Logic Model). This model has paved the way to the artificial neural network research, and included two parts, the first focusing on ecological processes in the brain, and the second search in the artificial neural network applications in the field of artificial intelligence.

At the end of the forties of the twentieth century put the world (Donald Hebb) hypotheses based learning mechanical neural plasticity known as (Hebbian Learning). It is a perfect model for learning the rules without supervisor (Unsupervised Learning). These ideas and applied to Computational models of machine (Turing) of the type (B) in 1948.

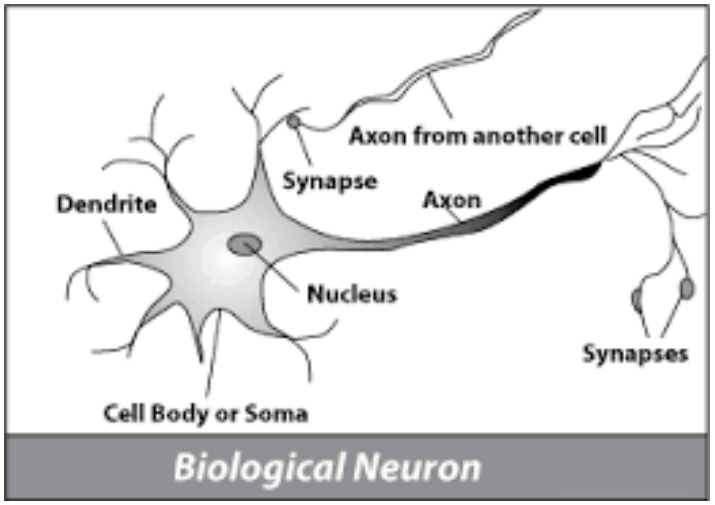
In 1954 for the first time use (Farley) and (Clark) computation machine (ie digital calculator) and through the simulation model (Hebb) at the Massachusetts Institute of Technology (MIT), and there are several scientists worked computation machines neural networks in 1956, such as (Rochester , Holland, Habit, Duda).

Historical Background

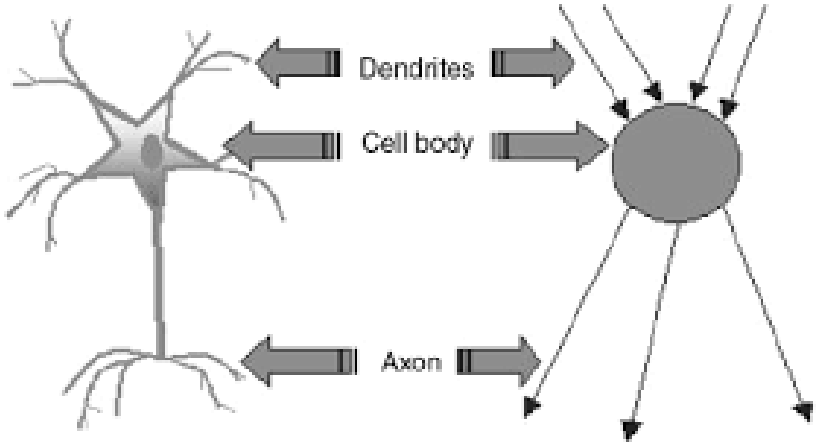
In 1958 the world (Rosenblatt) invented the (Perceptron), a pattern recognition algorithm through a network calculator to learn two-level (Two-Layer) using addition and subtraction. Subsequently developed to include more complicated than the first operations. Then developed the world thereafter (Paul Werbos) to become an algorithm called back propagation algorithm in 1975.

This is worth noting that research on artificial neural networks have experienced a recession between 1960 and 1975 for several reasons, including the models failed to address the issues are not complex algorithms proposed Another reason is the lack of fast computers at that time to deal with such advanced technology needed to execution speed to get the desired learning and these reasons outlined in the report to the worlds (Marvin Minsky) and (Seymor Papert). The period saw the eighties and nineties of the twentieth century, the emergence of many artificial neural network algorithms and helped to steady growth this rapid evolution of computing and increase the memories used, and especially the development that took place in the field of parallel computing processors (Parallel Computers).

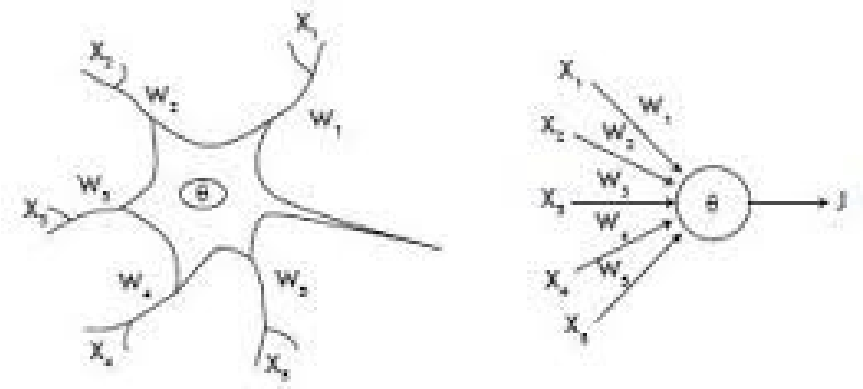
Biological Neuron



Biological to Artificial Neuron



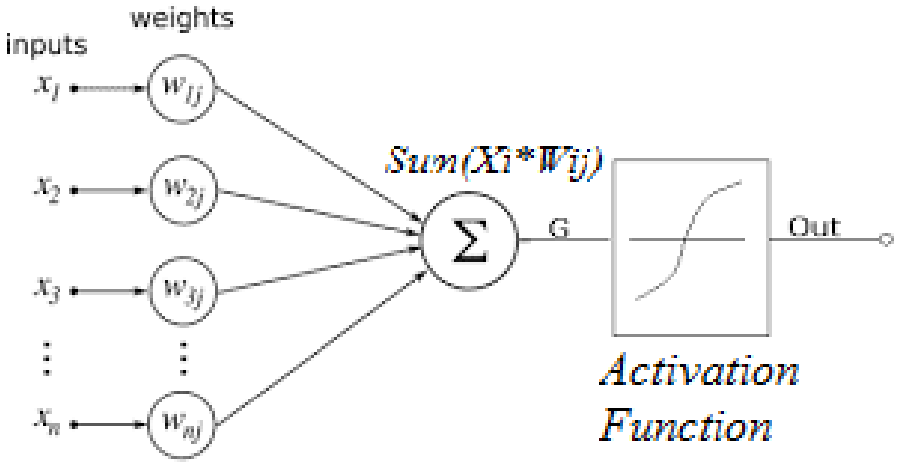
Biological neuron



Artificial neuron

From the biological neuron to the artificial neuron

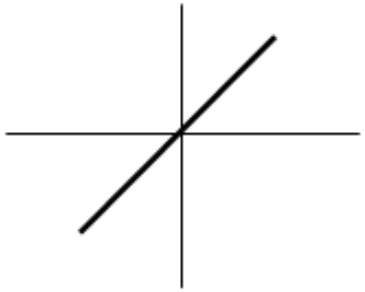
Artificial Neuron



ANN Components:

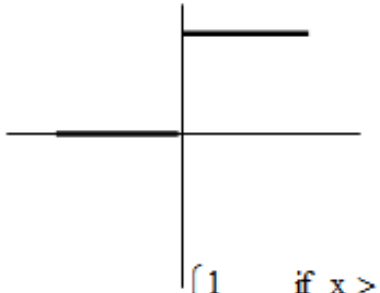
- Inputs (Int., real, binary).
- Output (Int., real, binary).
- Activation function (linear, non-linear).
- Weights (fixed, variable).
- ANN topology.

Activation Function Samples



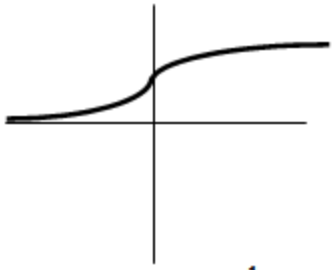
$f(x) = x$

Simitric fun.



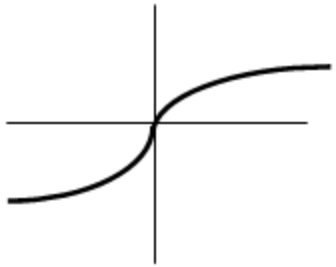
$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$$

Binary fun.



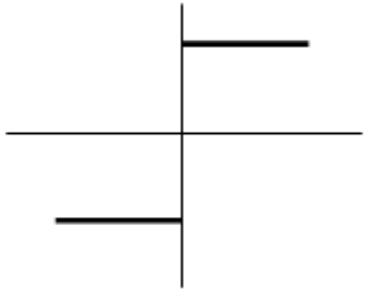
$$f(x) = \frac{1}{1+e^{-x}}$$

Sigmoid fun.



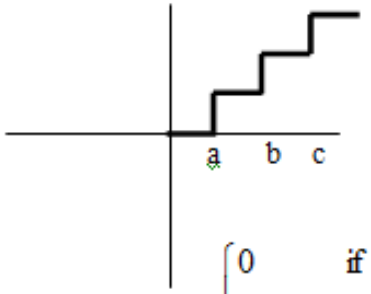
$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Tanh fun.



$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \le 0 \end{cases}$$

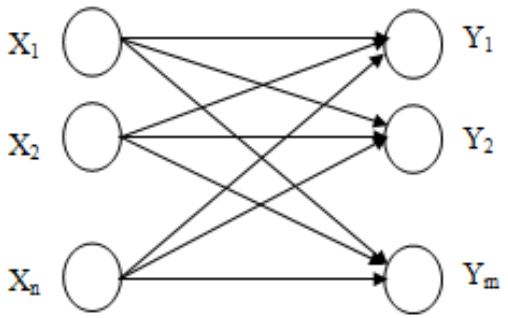
Bipolar Hard Limit fun.



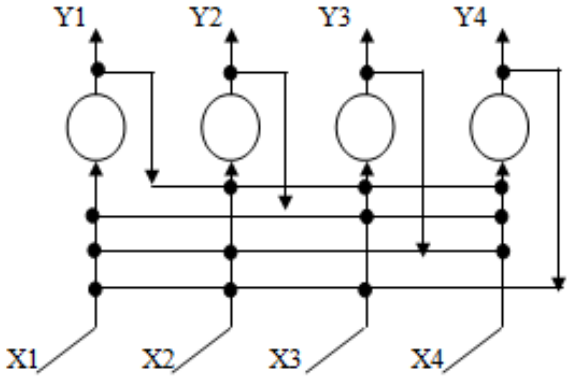
$$f(x) = \begin{cases} 0 & \text{if } 0 \le x < a \\ 1 & \text{if } a \le x < b \\ 2 & \text{if } b \le x < c \\ 3 & \text{if } x \ge c \end{cases}$$

Steps fun.

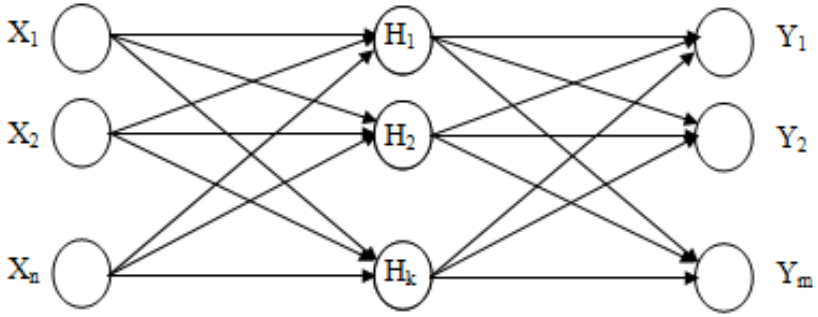
ANN Topology Samples



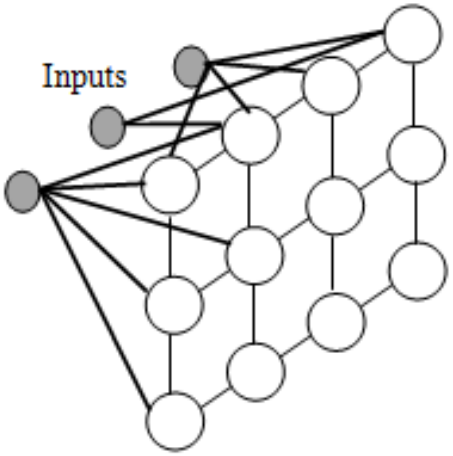
Single Layer NN



Feedback Single Layer NN



Multi-Layer NN



Self-Organization NN

Prof. Dr. Ahmed Tariq Saadeq

ANN Properties

1. Parallelism or synchronization at work: a fundamental feature of the advantages of artificial neural networks, ie if promised each cell represents a processing unit (Processor Element) itself linked to turn a large number of units similar treatment, they will we increase the processing calculator for orders speed tremendously and this is what you did mainframes, which produced tremendous speed using artificial neural network technology to computers able to record time to solve very complex problems companies.
2. The ability to adapt (Adaptation): It is a very important characteristic of artificial neural networks properties, as they are solving the problem through a particular algorithm rather than through the programming problem is, they adapted themselves to solve the problem through the private data of the problem. In other words, there is no programming to solve the problem but the fact that no application for a specific learning algorithm and its function to cope with the requirements of the problem and adapt.
3. Distributed Memory : When the artificial neural network learns the data problem, it certainly will be storing data or keep it in its own way as any learning base, that is, they become a real problem memory for data broken down by type of artificial neural network. There are types of artificial neural networks are used for storage only serves as a working memory.

4. The ability to generalize (Generalization): If artificial neural network trained on a particular set of models in a particular application, and then were tested this network over other models is that may I trained them and were successful test results of any that artificial neural network introduced new models this means it has circulated the base that came out through her training on the given models, the new models and this is very important characteristic of the properties of artificial neural networks. In other words, it scrapped the concept of rule: "If ... Then ..." through a generalization feature.
5. fault tolerance (Fault Tolerance): As known, the architectural Von Neumann rely on serial execution of operations (Sequential Processing) meaning that errors that occur in the suggestions (steps) program affect what follows from the meta, while in architectural artificial neural networks, that bears the errors that occur in the inputs and correct them or something like that so it will not affect the work of the artificial neural network as well as for software solutions to problems using artificial neural network technology.
6. Definition of the problem: As we know, the basis of the work of artificial neural networks is learning a set of input models or problem situations through certain algorithms, and thus was a paragraph defining the problem exceeds what has to do so in the traditional solve the problem of programming cases, this is very important in many applications because it is difficult to understand what the problem is a mechanism.

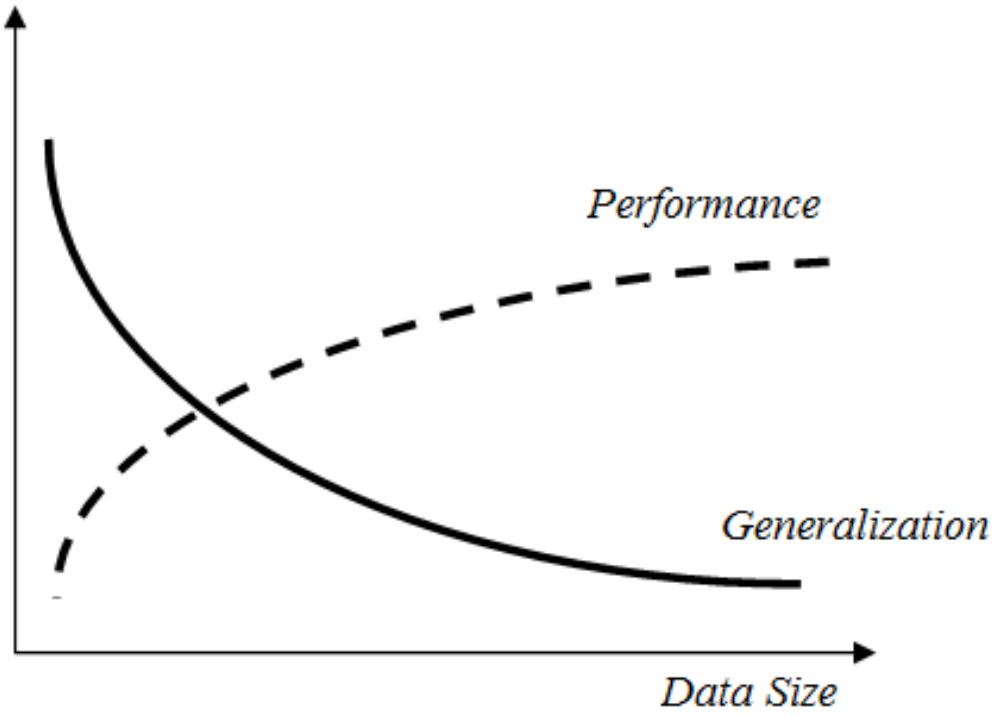
7. Solving confusing data problems (Noisy Data): Using artificial neural network technology (can deal with confusing data, for example, data that are some of the values are not true (the result of a particular fault or negligence is) and incomplete data might be missing value) and this property is the result of carrying the aforementioned error property.

8. Ease of construction and learning: as we shall see later, the building, programming and learning artificial neural networks are an uncomplicated often because of the record (Standard) algorithms and rules to learn and that's what made this technique is desirable in solving a lot of minor ones and complex problems.

Generalization in ANN

Generalization means (learning within minimum learning data size).

Performance of ANN increase when learning data size increase.



Learning Types in ANN

1- Supervised Learning: Based on the data of the problem and included inputs and output target, learning process stops when the neural network learns on the inputs and outputs of the problem.

2- Unsupervised Learning: Based on the Input data (no output), learning process stops when the neural network weights become stable.

3- Self-Organization : It is similar to unsupervised learning with additional processing such as clustering or same as it.

Some Learning Rules

Hebb Rule, $\Delta W(i,j) = \eta\, X(i)\, O(i)$

Widrow-Hoff Rule, $\Delta W(i,j) = \eta\, [Y(i)-O(i)]\, X(i)$

Competitive Rule, $\Delta W(i,j) = \eta\, [X(i)-W(i,j)]\, O(i)$

Generalized Delta Rule (GDR),

$$\Delta W(i,j) = \eta\, \delta(j)\, X(i) - W(i,j)$$

Generalized Delta Rule (GDR) with momentum,

$$\Delta W(i,j) = \eta\, \delta(j)\, X(i) - \alpha\, W(i,j)$$

Some Parameters

η : learning rate ($0< \eta <1$).

Bias or Threshold (θ): it is an added value to the weights or a constant node to increase the convergence or decrease the time learning.

α : momentum term ($0< \alpha <1$), it is also use to increase the convergence or decrease the time learning during the weights updates.

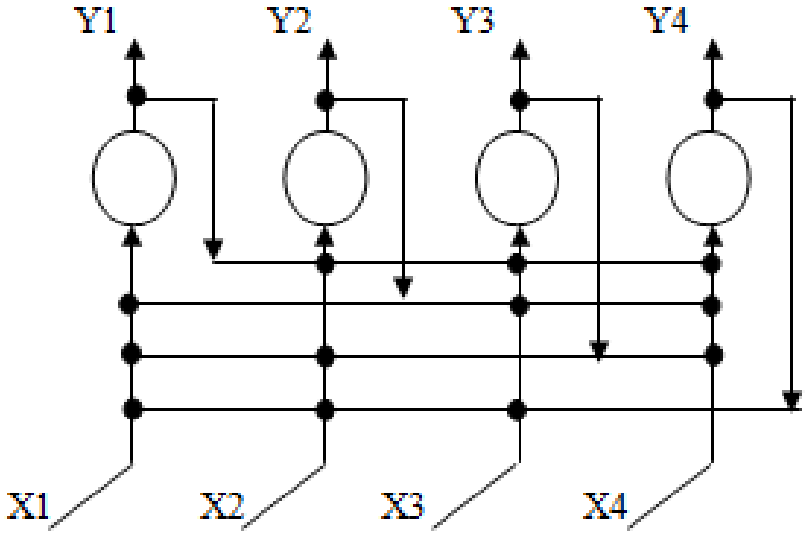
Hopfield NN

Invented by the physician (John Hopfield) in 1982.

Features:

- **Unsupervised learning.**
- **Associative memory.**
- **Full connection.**
- **Single layer.**
- **Feedback.**
- **Fixed weight.**
- **Bipolar.**
- **Linear activation function.**
- **Input Nodes equal to Output Nodes.**

Hopfield NN Topology



Hopfield NN Learning Algorithm

-Initialize, (N) no. of input node, (P) no. of samples, fun. $f(x) = \begin{cases} 1 & if\ x > 0 \\ -1 & if\ x <= 0 \end{cases}$

- Convert 0 to -1,

- Compute weight matrix, $W_{i,j} = \begin{cases} 0 & if\ i = j \\ \sum_{1}^{P} X_i X_j & otherwise \end{cases}$ $where\ i,j = 1,2,.....N$

-

Example

Assume N=4, P=3 as below

1	1	1	0
0	0	0	0
1	1	0	0

1	1	1	-1
-1	-1	-1	-1
1	1	-1	-1

Compute the weight matrix

$W_{1,2}$=1*1 + -1*-1 + 1*1 =3 = $W_{2,1}$

$W_{1,3}$=1*1 + -1*-1 + 1*-1 =1 = $W_{3,1}$

$W_{1,4}$=1*-1 + -1*-1 + 1*-1 =-1 = $W_{4,1}$

$W_{2,3}$=1*1 + -1*-1 + 1*-1 =1 = $W_{3,2}$

$W_{2,4}$=1*-1 + -1*-1 + 1*-1 =-1 = $W_{4,2}$

$W_{3,4}$=1*-1 + -1*-1 + -1*-1 =1 = $W_{4,3}$

$$W = \begin{pmatrix} 0 & 3 & 1 & -1 \\ 3 & 0 & 1 & -1 \\ 1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$$

Testing:

Assume we have the sample [1 1 1 1], apply the fun.

$$O_i = f(Z_i + \sum_1^j Z * W_{i,j})$$

$$where\ i,j = 1,2,.....N$$

O1= f(1+(1*0+1*3+1*1+1*-1)) = f(4) =1

O2= f(1+(1*3+1*0+1*1+1*-1)) = f(4)=1

O3= f(1+(1*1+1*1+1*0+1*1)) = f(4) =1

O4= f(1+(1*-1+1*-1+1*1+1*0)) = f(0) = -1

output = [1 1 1 0]

Again,

O1= f(1+(1*0+1*3+1*1+ -1*-1)) = f(6) =1

O2= f(1+(1*3+1*0+1*1+ -1*-1)) = f(6)=1

O3= f(1+(1*1+1*1+1*0+ -1*1)) = f(6) =1

O4= f(0+(1*-1+1*-1+1*1+ -1*0)) = f(-1) = -1

Output = [1 1 1 0], it is same as previous, therefore stop, this final output.

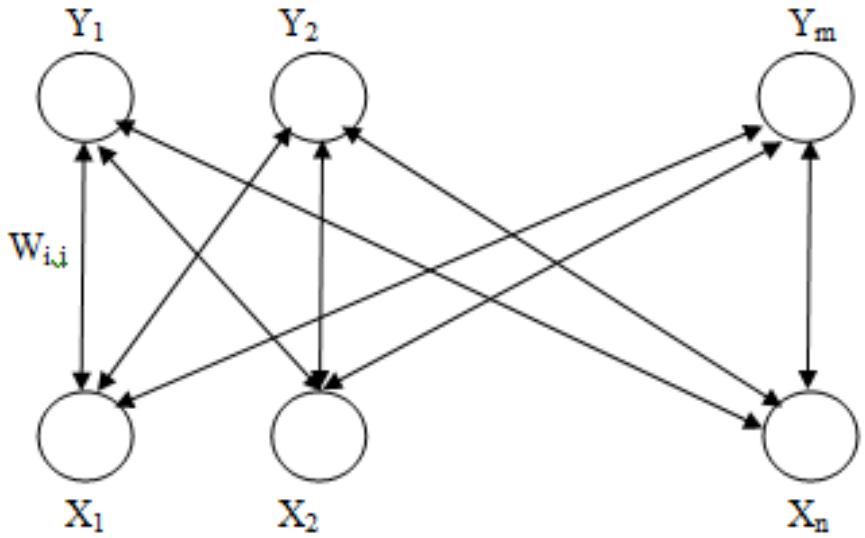
[1 1 1 0]

Relatively it is good.

Binary Associated Memory (BAM)

(Bart Kosko) propose this type of artificial neural networks in 1988, which is very similar to a Hopfield NN, but in BAM there are input layer & output layer.

BAM-NN Topology



BAM features:

- **Unsupervised learning.**
- **Associative memory.**
- **Full connection.**
- **I/O layer.**
- **Feedback.**
- **Fixed weight.**
- **Bipolar.**
- **Linear activation function.**
- **Input Nodes not necessary equal to Output Nodes.**

BAM Algorithm is same as Hopfield, except the weight computation as below equation:

$$W_i = A_i^T * B_i$$

Example

Input	Output
1 0 0	0 0 1
0 1 0	0 1 0
0 0 1	1 1 0

Convert 0 to -1, then

$W_1=[A_1^T][B_1]$

$$W_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$W_2=[A_2^T][B_2]$

$$W_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

$W_3=[A_3^T][B_3]$

$$W_3=\begin{bmatrix}-1\\-1\\1\end{bmatrix}\begin{bmatrix}1 & -1 & -1\end{bmatrix}=\begin{bmatrix}-1 & 1 & 1\\-1 & 1 & 1\\1 & -1 & -1\end{bmatrix}$$

$W = W_1 + W_2 + W_3$

$$W=\begin{bmatrix}-1 & -1 & 1\\1 & 1 & -1\\1 & 1 & -1\end{bmatrix}+\begin{bmatrix}1 & -1 & 1\\-1 & 1 & -1\\1 & -1 & 1\end{bmatrix}+\begin{bmatrix}-1 & 1 & 1\\-1 & 1 & 1\\1 & -1 & -1\end{bmatrix}=\begin{bmatrix}-1 & -1 & 3\\-1 & 3 & -1\\3 & -1 & -1\end{bmatrix}$$

Testing : 1 0 0

*B = f(A*W)*

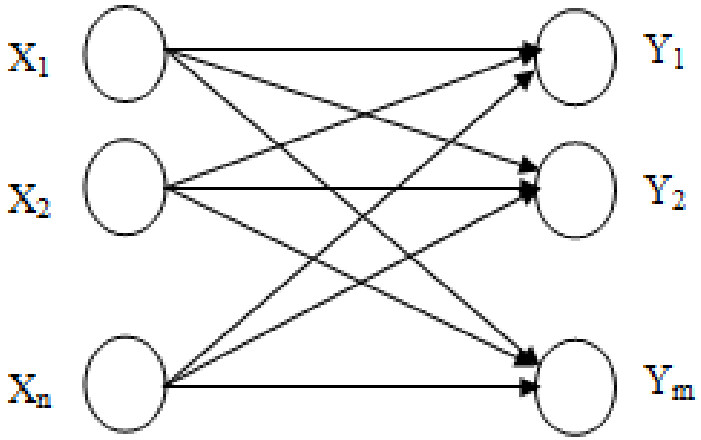
$$\begin{bmatrix}1 & -1 & -1\end{bmatrix}\begin{bmatrix}-1 & -1 & 3\\-1 & 3 & -1\\3 & -1 & -1\end{bmatrix}=\begin{bmatrix}-3 & -3 & 5\end{bmatrix}=\begin{pmatrix}0 & 0 & 1\end{pmatrix}$$

Single Layer NN

Features:

- Supervised learning.
- Single layer (I/O).
- Linear Activation Function.
- Full connection.
- Based on Delta learning rule.
- Variable weights.

Single layer NN topology



Single Layer NN Algorithm

- Initialize X input nodes (N nodes), Y output nodes (M nodes), No. of samples (P), η learning rate and activation fun. *f(x)*.
- Generate weight matrix with size (NxM).
- While error ratio not acceptance Do

  for each sample compute the below equation:

$$O_{k,j} = f(\sum_{i=1}^{N} X_{k,i} * W_{i,j}) \quad \text{where } j = 1,2,....,M \ \& \ k = 1,2,....,P$$

  compute the error ratio as below:

$$\delta = (Y - O)$$

  if error not equal zero then adjust the weight matrix as:

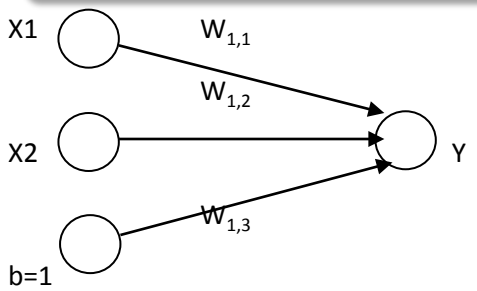
$$W_{i,j} = W_{i,j} \pm \eta * \delta * X_i \quad \text{where } i = 1,2,....,N \ \& \ j = 1,2,....,M$$

  end while

End.

Example: Logical AND Gate

X1	X2	B	Y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



Where X1, X2 are input nodes, B is bias node & Y is output node.

Initialize W = [1 1 -1], activation function is

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \le 0 \end{cases}$$

First sample [1 1 1]

$O_1 = f(1 * 1 + 1 * -1 + 1 * 1) = f(1) = 1$

$\delta = Y_1 - O_1 = 1 - 1 = 0$ O.K.

Second sample [1 -1 1]

$O_2 = f(1 * 1 + -1 * -1 + 1 * 1) = f(3) = 1$

$\delta = Y_2 - O_2 = -1 - 1 = -2$

Adjust W

$W_{1,1} = 1 + 0.7 * (-2) * 1 = -0.4$

$W_{1,2} = -1 + 0.7 * (-2) * (-1) = 0.4$

$W_{1,3} = 1 + 0.7 * (-2) * 1 = -2.4$

W = [-0.4 0.4 -2.4]

Then $O_2 = f(1 * -0.4 + -1 * 0.4 + 1 * -2.4) = f(-3.2) = -1$

Third sample [-1 1 1]

$O_3 = f(-1 * -0.4 + 1 * 0.4 + 1 * -2.4) = f(1.6) = 1$

$\delta = -1 - 1 = -2$

Adjust W

$W_{1,1} = -0.4 + 0.7 * (-2) * (-1) = 1$

$W_{1,2} = 0.4 + 0.7 * (-2) * 1 = 1$

$W_{1,3} = -2.4 + 0.7 * (-2) * 1 = -3.8$

$W = [\ 1 \quad 1 \quad -3.8]$

Then $O_3 = f(-1 * 1 + 1 * 1 + 1 * -3.8) = f(-3.8) = -1$

Fourth sample [-1 -1 1]

$O_4 = f(-1 * 1 + -1 * 1 + 1 * -3.8) = f(-5.8) = -1$

Now again from the begin

$O_1 = f(1 * 1 + 1 * 1 + 1 * -3.8) = f(-1.8) = -1$

Adjust W

$W_{1,1} = 1 + 0.7 * (2) * 1 = 2.4$

$W_{1,2} = 1 + 0.7 * (2) * 1 = 2.4$

$W_{1,3} = -3.8 + 0.7 * (2) * 1 = -2.4$

$W = [\ 2.4 \quad 2.4 \quad -2.4]$

And,

$O_2 = f(1 * 1 + -1 * 1 + 1 * -3.8) = f(-3.8) = -1$

$O_3 = f(-1 * 1 + 1 * 1 + 1 * -3.8) = f(-3.8) = -1$

$O_4 = f(-1 * 1 + -1 * 1 + 1 * -3.8) = f(-5.8) = -1$

O.K.

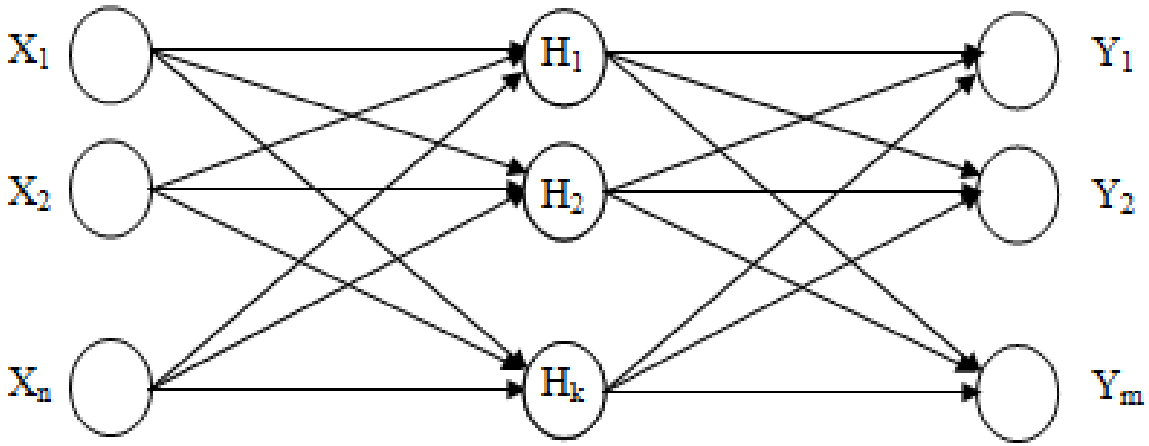
Back-Propagation NN

(Paul Werbos) was developed back propagation algorithm in 1975. It is one of the most artificial neural network algorithms in applications

Features:

- Supervised learning.
- Multi Layers.
- Non-Linear Activation Function.
- Full connection.
- Based on GBDR.
- Variable weights.
- Process Forward-Backward.

BP-NN topology



BP-NN Algorithm

- Initialize the parameters (A: no. of input nodes), (B: no. of hidden nodes), (C: no. of output nodes), (P: no. of samples), (η: learning rate) and activation function.
- Generate the weight matrices (W1[AxB], W2[BxC]) randomly.
- While error is acceptance Do
  - For each sample Do
  - Compute the hidden nodes as below

$$H_j = \frac{1}{1+e^{-V}} \quad \text{where } j = 1,2,.....B \text{ , } V = \sum_{i=1}^{A} W1_{i,j} X_i$$

  - Compute the output nodes as below

$$O_j = \frac{1}{1+e^{-V}} \quad \text{where } j = 1,2,.....C \text{ , } V = \sum_{i=1}^{B} W2_{i,j} H_i$$

  - End for

- Compute error as $$error = \sqrt{\sum_{k=1}^{C}(Y_k - O_k)^2}$$

- If error not acceptance Then
  - Adjust the weight matrices as below

$$\delta 2_k = O_k * (1 - O_k) * (Y_k - O_k) \qquad \text{where } k = 1,2,.....,C$$

$$\delta 1_k = Z_k * (1 - Z_k) * \sum_{j=1}^{C} \delta 2_j W 2_{k,j} \qquad \text{where } k = 1,2,.....,B$$

$$W 2_{i,j} = W 2_{i,j} + \eta * \delta 2_j * H_i \qquad \text{where } i = 1,2,....,B \ \& \ j = 1,2,....,C$$

$$W 1_{i,j} = W 1_{i,j} + \eta * \delta 1_j * X_i \qquad \text{where } i = 1,2,....,A \ \& \ j = 1,2,....,B$$

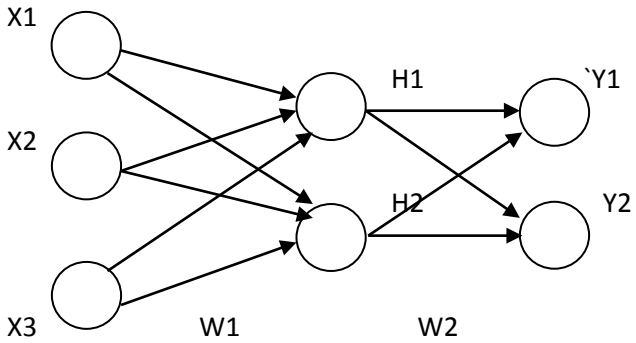
- End If;
- End While;
- Save the last weight values;
- End Algorithm.

Example: Assume we have 3 input nodes, 2 output nodes and 3 samples as below:

X1	X2	X3	Y1	Y2
1	0	0	1	0
0	0	1	0	1
1	1	1	1	1

Suppose we have 2 hidden nodes with the following weight matrices & BP-NN topology:

$$W1 = \begin{bmatrix} 0.2 & 0.1 \\ 0.3 & 0.7 \\ 0.8 & 0.5 \end{bmatrix} \qquad W2 = \begin{bmatrix} 0.1 & 0.6 \\ 0.7 & 0.8 \end{bmatrix}$$



Where η=0.6 and error acceptance ratio = 0.1

Solution:

First sample [1 0 0 1 0]

Sum = $\sum$ W1* X

Sum1 = 1*0.2 + 0*0.3 + 0*0.8 = 0.2

$$h_1 = \frac{1}{1+e^{-0.2}} = 0.5498$$

Sum2 = 1*0.1 + 0*0.7 + 0*0.5 = 0.1

$$h_2 = \frac{1}{1+e^{-0.1}} = 0.525$$

Sum = $\sum$ W2* H

Sum1 = 0.5498*0.1 + 0.525*0.6 = 0.37 $o_1 = \frac{1}{1+e^{-0.37}} = 0.5915$

Sum2 = 0.5498*0.7 + 0.525*0.8 = 0.805 $o_2 = \frac{1}{1+e^{-0.805}} = 0.691$

The error is not acceptance (Desired [1 0], Actual [0.5915 0.691])

Error in output layer

$\delta 2_1$=0.5915*(1-0.5915)*(1-0.5915)=0.0987

$\delta 2_2$=0.691*(1-0.691)*(0-0.691)=-0.1475

Error in hidden layer

$\delta 1_1$=0.5498*(1-0.5498)*(0.1*0.0987+0.6*-0.1475)=-0.0195

$\delta 1_2$=0.525*(1-0.525)*(0.7*0.0987+0.8*-0.1475)=-0.0196

Adjust the weights

W2

$W2_{1,1}$=0.1+0.6*0.0987*0.5498=0.1326

$W2_{1,2}$=0.6+0.6*-0.1475*0.525=0.551

$W2_{2,1}$=0.7+0.6*0.0987*0.5498=0.7311

$W2_{2,2}$=0.8+0.6*-0.1475*0.525=0.7535

W1

$W1_{1,1}$=0.2+0.6*-0.0195*1=0.1883, $W1_{1,2}$=0.1+0.6*-0.196*1=0.0882

$W1_{2,1}$=0.3+0.6*-0.0195*0=0.3, $W1_{2,2}$=0.7+0.6*-0.196*0=0.7

$W1_{3,1}$=0.2+0.6*-0.0195*0=0.8, $W1_{3,2}$=0.1+0.6*-0.196*0=0.5

Some drawbacks in BP-NN

- **Parameter values** : like η, α values, there are no canonical method to determine the reasonable values for these parameters, therefore the experiments is the way to choose the reasonable values.
- **No. of Hidden Nodes** : hidden layer play a big role in the convergence of weights and results, also there is no way to determine the reasonable no. of nodes in this layer, the solution is the experiments. Also sometimes there are more than hidden layer in the NN.
- **Choose the activation function** : there are several non-linear activation function, we must select one or more be carefully.
- **Weights update** : there are few weight update formals and parameters, we must select one from these be carefully.
- **No. of Bias nodes** : How many Bias nodes in the input & hidden layers? By experiments !!
- **Error ratio computation** : sample-by-sample or accumulatively.

Kohonen - NN

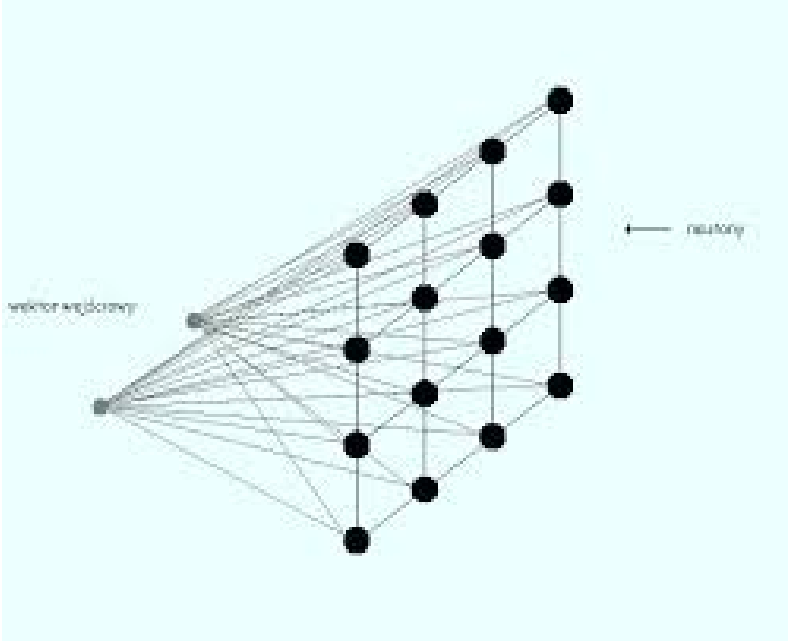
(Teuvo Kohonen) developed this NN in 1982, and adopted this algorithm as a kind of algorithms that does not need to supervisor and the competitive type are also used for the purposes of clustering. The structure of this network includes two layers, the first concerning the inputs (features) and the second is the output (clusters or groups), the weights determining which outputs a matrix fits the input. Given the values of the vectors weights randomly and then examine the inputs start and the extent of bias to the output through the equation of Euclidean dimension, takes less output value in terms of the Euclidean dimension siding designated entrance into this category, and so the second and third inputs to the end, that is, network self-reorganizing through mathematical equations that each input to the exit is biased or a particular class. In other words, that there is competition between them (input) on the bias to the output. Certainly the inputs would side similar to the same items in other words, it will be based on the principle of neighboring (Neighborhood).

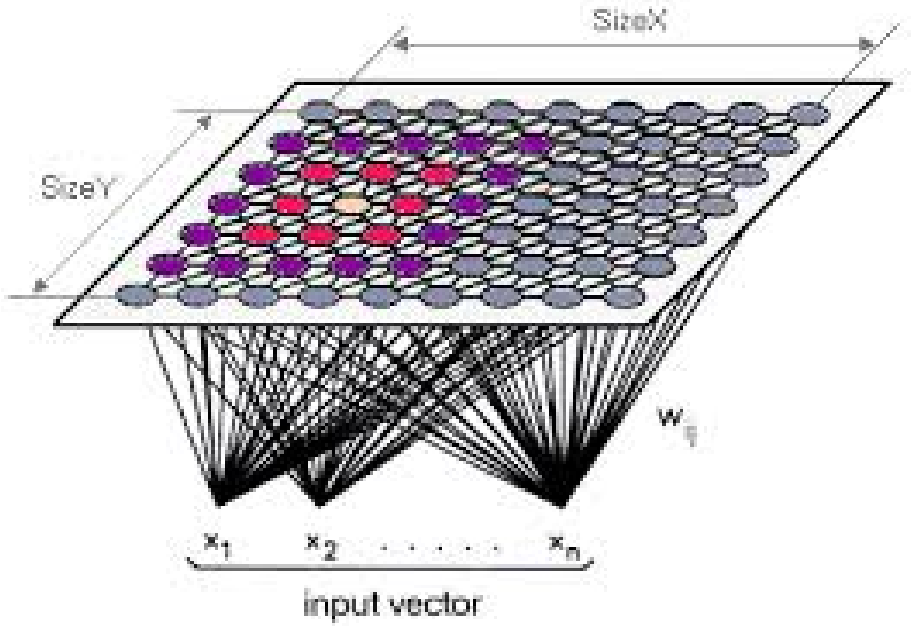
Kohonen – NN Features & Topologies

Features:

- Self-Organization (Unsupervised).
- Depends on competitive learning.
- Used for complex problems.

There are several topologies for Kohonen-NN as:





Kohonen-NN Algorithm

- Initialize the parameters (N: no. of input nodes X), (M: no. of clusters C), (P: no. of samples), (α: learning rate), (min- α).
- Generate the weight matrix W[NxM] randomly.
- While α < min- α Do
  - For each sample Do
  - Compute D for each cluster (j) $D(j) = \sum \left(x_i - w_{ij} \right)^2$
  - Select Minimum D(j)
  - Update column j in the weight matrix $w_{i,j} = w_{i,j} + \alpha(X_i - w_{i,j})$
  - End For
  - Update α value
  - End While

End Algorithm.

Update of α

- Linear : $\alpha = \alpha - \beta$, $(\beta>0)$

- Non-Linear : $\alpha = \alpha\ \beta$, $(0<\beta<1)$

$\alpha = \alpha / \beta$, $(\beta>1)$

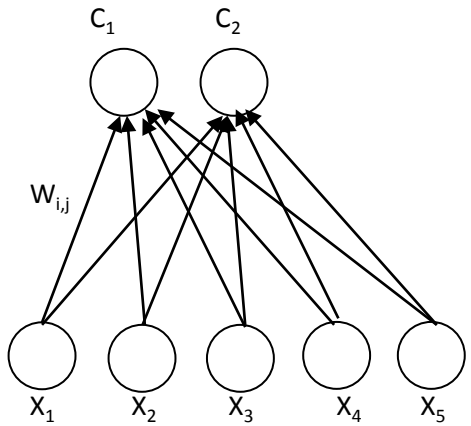
$\alpha = \log(\alpha)$

Example: Assume we have the following samples

1 1 0 0 0
0 0 0 1 1
1 0 1 0 0
0 0 0 0 1

Suppose there are 2 clusters, with initial α =0.6, min- α = 0.1, β=0.1 and initial W matrix as below:

$$w = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.2 \\ 0.3 & 0.6 \\ 0.1 & 0.5 \\ 0.5 & 0.4 \end{bmatrix}$$



For the first sample 1 1 0 0 0 , compute D(j) as below

D(1)= $(1-0.3)^2+(1-0.4)^2+(0-0.3)^2+(0-0.1)^2+(0-0.5)^2=1.2$

D(2)= $(1-0.7)^2+(1-0.2)^2+(0-0.6)^2+(0-0.5)^2+(0-0.4)^2=1.5$

The min j=1 so,

$W_{1,1}$=0.3+0.6*(1-0.3)= 0.72

$W_{1,2}$=0.4+0.6*(1-0.4)= 0.76

$W_{1,3}$=0.3+0.6*(0-0.3)= 0.12

$W_{1,4}$=0.1+0.6*(0-0.1)= 0.04

$W_{1,5}$=0.5+0.6*(0-0.5)= 0.2

$$W = \begin{bmatrix} 0.72 & 0.7 \\ 0.76 & 0.2 \\ 0.12 & 0.6 \\ 0.04 & 0.5 \\ 0.2 & 0.4 \end{bmatrix}$$

For the first sample 0 0 0 1 1, compute D(j) as below

$D(1)= (0-0.72)^2+(0-0.76)^2+(0-0.12)^2+(1-0.04)^2+(1-0.2)^2=2.672$

$D(2)= (0-0.7)^2+(0-0.2)^2+(0-0.6)^2+(1-0.5)^2+(1-0.4)^2=1.5$

The Min j= 2

$W_{2,1}=0.7+0.6*(0-0.7)= 0.28$

$W_{2,2}=0.2+0.6*(0-0.2)= 0.08$

$W_{2,3}=0.6+0.6*(0-0.6)= 0.24$

$W_{2,4}=0.5+0.6*(1-0.5)= 0.8$

$W_{2,5}=0.4+0.6*(1-0.4)= 0.76$

$$W = \begin{matrix} 0.72 & 0.28 \\ 0.76 & 0.08 \\ 0.12 & 0.24 \\ 0.04 & 0.8 \\ 0.2 & 0.76 \end{matrix}$$

The Third sample 1 0 1 0 0 , d1 = 1.472 , d2 = 2.32

$$W = \begin{matrix} 0.888 & 0.28 \\ 0.304 & 0.08 \\ 0.648 & 0.24 \\ 0.016 & 0.8 \\ 0.08 & 0.76 \end{matrix}$$

The fourth sample 0 0 0 0 1, d1 = 2.1475, d2 = 0.84

$$W = \begin{matrix} 0.888 & 0.112 \\ 0.304 & 0.032 \\ 0.648 & 0.096 \\ 0.016 & 0.32 \\ 0.08 & 0.904 \end{matrix}$$

Update the learning rate (α), as : = α - β = 0.6 - 0.1= 0.5

Then again from the first sample, d1 = 0.9235, d2 = 2.6544

0.944 0.112
0.652 0.032
W= 0.324 0.096
0.008 0.32
0.04 0.904

The second sample, d1 = 3.3269, d2 = 0.4944

0.944 0.056
0.652 0.016
W = 0.324 0.048
0.008 0.66
0.04 0.952

The Third sample, d1 = 0.8869, d2 = 3.1396

0.972 0.056
0.326 0.016
W = 0.662 0.048
0.004 0.66
0.02 0.952

The fourth sample, d1 = 2.4497, d2 = 0.4436

0.972 0.028
0.326 0.008
W = 0.662 0.024
0.004 0.33
0.02 0.976

Genetic Algorithms

"Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime."- Salvatore Mangano, *Computer Design*, May 1995.

- Originally developed by John Holland (1975)
- The genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution
- Uses concepts of "Natural Selection" and "Genetic Inheritance" (Darwin 1859)

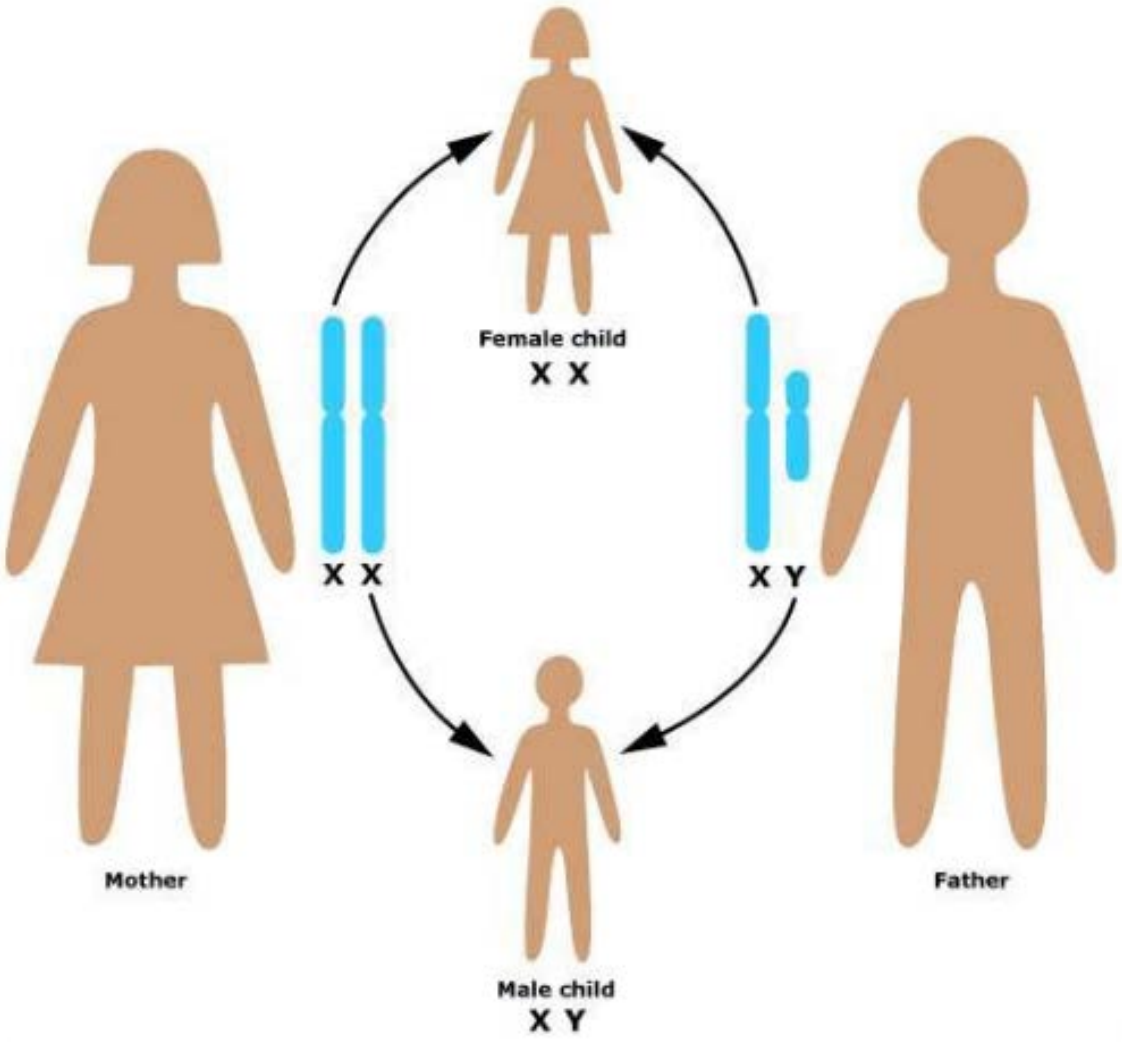
A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators

Biologically - GA

Our body is made up of trillions of **cells**. Each cell has a core structure (**nucleus**) that contains your **chromosomes**.

Each **chromosome** is made up of tightly coiled strands of deoxyribonucleic acid (**DNA**). **Genes** are segments of DNA that determine **specific traits**, such as eye or hair color. You have more than 20,000 genes.

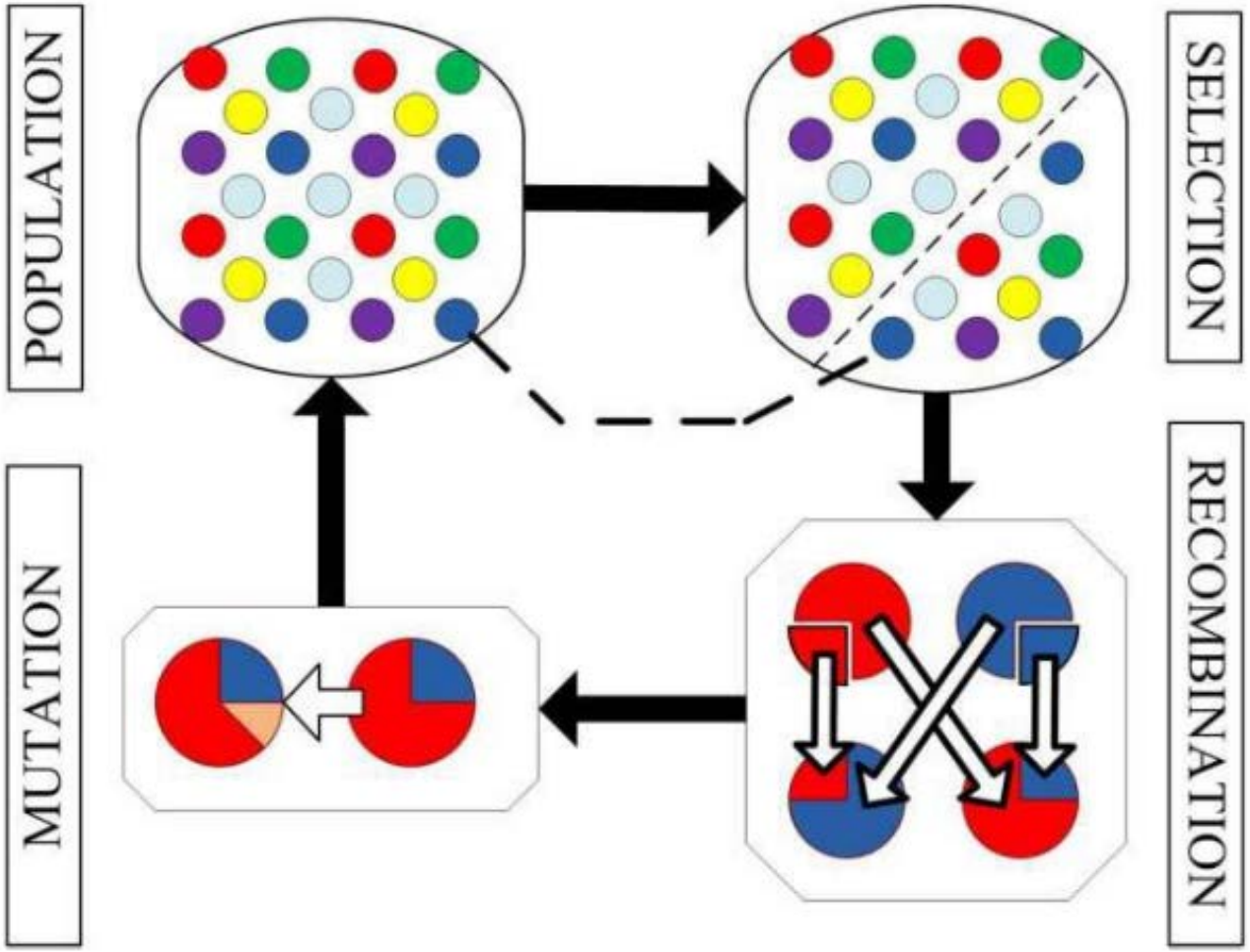
A gene **mutation** is an alteration in your DNA. It can be inherited or acquired during your lifetime, as cells age or are exposed to certain chemicals. Some changes in your genes result in genetic disorders.


Female child
X X
X X
X Y
Mother
Father
Male child
X Y

Bio-Genetic Vs. Genetic Algorithm

Biological Genetic	Genetic Algorithms
Chromosomes	Strings or Symbols
Genes	Strings or Symbols (May Be One)
Allele	Feature
Position	Position of String or Symbol
Genotype	Representation or Structure
Phenotype	Coding of Representation or Structure

Components of GA



Classical Form of GA

Begin
Encode the problem;
Generate initial population of individuals;
Evaluate the fitness of all individuals;
While not (Termination Conditions) **Do**
    Begin
        Select pair of individuals;
        Crossover between the individuals (recombine);
        Mutate individuals;
        Produce a new population;
        Evaluate the fitness of the modified individuals;
    End While
End Algorithm

Encoding

- Binary : 01001111
- Integer : 12 5 38 11 09
- Real : 1.2 0.3 11.4 0.11 1.0
- Char : A C V F
- String : abc cvc123

Initial Population

We start with a population of n random strings. Suppose that $l = 10$ and $n = 6$

We toss a fair coin 60 times and get the following initial population:

$s_1 = 1111010101$

$s_2 = 0111000101$

$s_3 = 1110110101$

$s_4 = 0100010011$

$s_5 = 1110111101$

$s_6 = 0100110000$

Fitness Function : *F(X)*

$s_1 = 1111010101 \quad f(s_1) = 7$

$s_2 = 0111000101 \quad f(s_2) = 5$

$s_3 = 1110110101 \quad f(s_3) = 7$

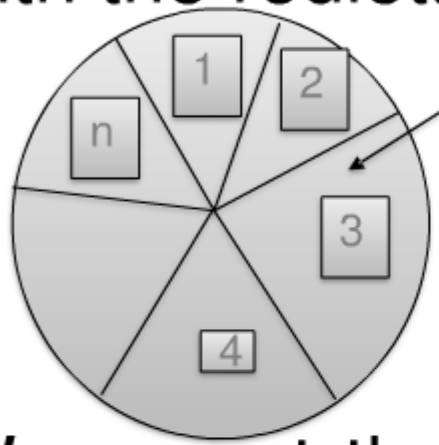
$s_4 = 0100010011 \quad f(s_4) = 4$

$s_5 = 1110111101 \quad f(s_5) = 8$

$s_6 = 0100110000 \quad f(s_6) = 3$

Selection Operator

- Next we apply fitness proportionate selection with the roulette wheel method:



Area is Proportional to fitness value

Individual *i* will have a probability to be chosen

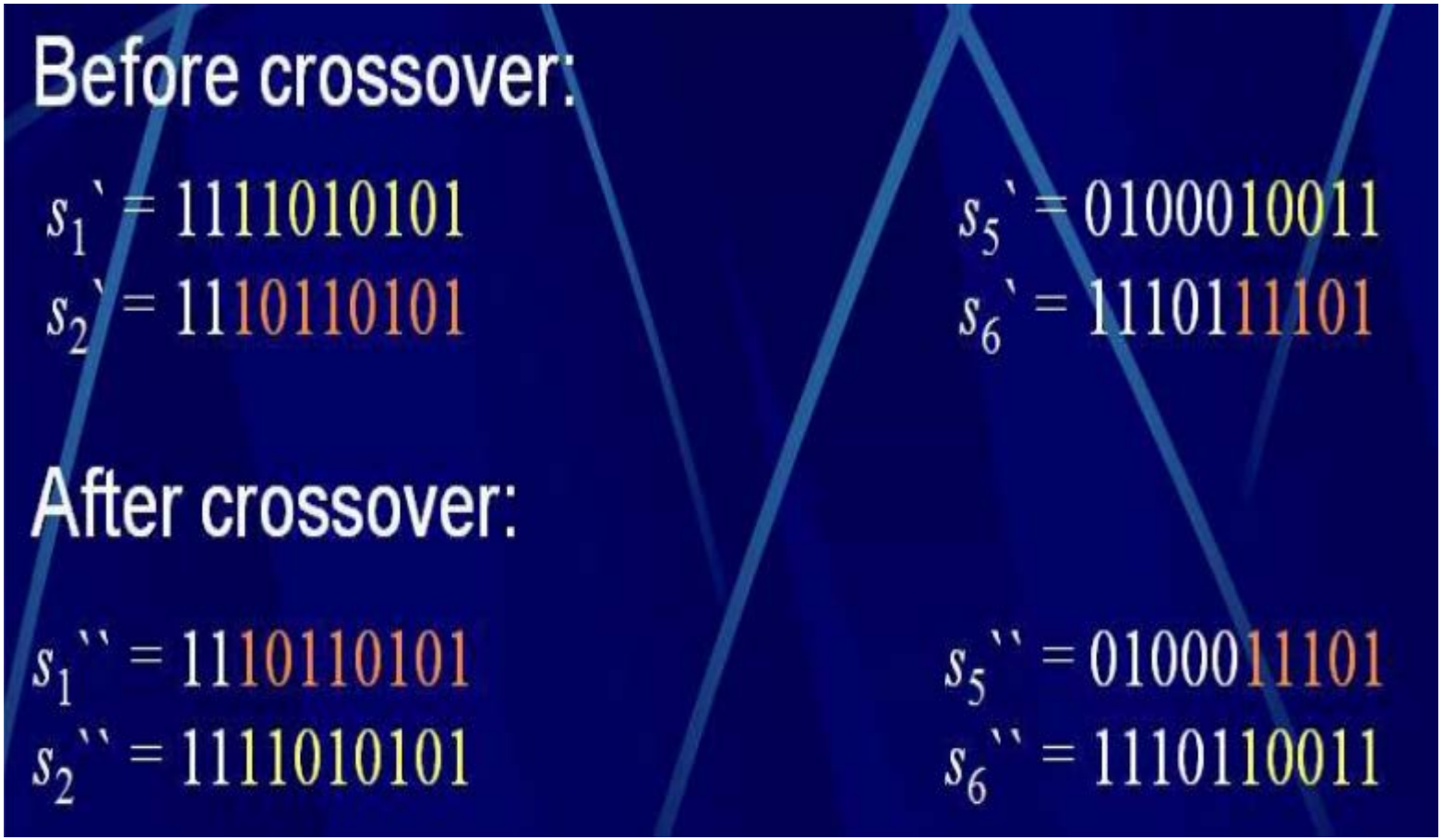
$$\frac{f(i)}{\sum_{i} f(i)}$$

- We repeat the extraction as many times as the number of individuals
- we need to have the same parent population size (6 in our case)

Other Selection Operator

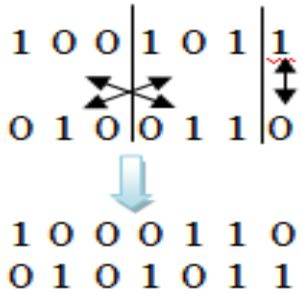
- Elitism.
- Tournament.
- Rank.
- Boltezman.
- Sigma Scaling.
- Steady State.

Crossover (Recombination)

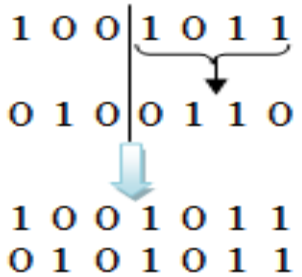


Other Crossover Operator

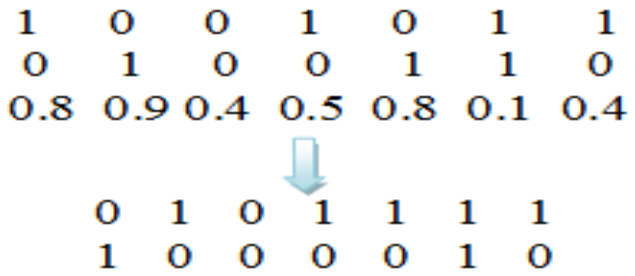
- Multi-Points Crossover



- Bacteria Conjugation.



- Uniform Crossover



Other Crossover Operator

- Crowding.
- Fitness Sharing Function.
- Inversion.
- Segregation.
- Migration.
- Translocation.
- Flat.
- Partially Matched.

Mutation

- Before applying mutation:

$s_1'' = 1110110101$

$s_2'' = 1111010101$

$s_3'' = 1110111101$

$s_4'' = 0111000101$

$s_5'' = 0100011101$

$s_6'' = 1110110011$

- After applying mutation:

$s_1''' = 1110100101$

$s_2''' = 1111110100$

$s_3''' = 1110101111$

$s_4''' = 0111000101$

$s_5''' = 0100011101$

$s_6''' = 1110110001$

Prof. Dr. Ahmed Tariq Saadeq

GA Parameters

- Population Size (Pop_Size): Its play a major role in providing appropriate diversity in the nature of the solutions generated by increasing the size of the community, but the greater the size of the community has increased processing time and this is contrary to what is required of a genetic algorithm to find a solution as soon as possible.
- Chromosome Length (Chrom_Len): This parameter depends mainly on the size of the problem addressed by the genetic algorithm it represents the length of the solution and not the number of solutions that we have dealt with in the size of the community coefficient. Here, the length of the solution is the chromosome length, for example, it is required to find the shortest path to map where 7 cities surely that the length of the chromosome must not exceed the number 7, and if the number of cities 100 the length of the chromosome does not exceed the number 100. It is possible to make this a coefficient variable in the event that the problem is dealing with the subject so or are variable by itself or can be doing this if it leads to the production of high quality solutions for high-appropriate time.

GA Parameters

Prof. Dr. Ahmed Tariq Saadeq

- Maximum Generation (Max_Gen): It represents a number of loops implementing the genetic algorithm operations, any number of generations generated starting from the first community generated randomly. Certainly it is the greater problem is the data to be solved size preferably increase the number of generations generated on the grounds that the problem cannot find a solution it quickly. Number of generations has generated up to several hundred volumes when they are required to solve large problems, and sometimes the number of generations could be up to four decimal places (thousands) in a very large problems can not be sufficient to prepare the hundreds of generations, where due to the large size of the problem.
- Crossover Probability (Pc): There is no doubt that the crossover of genetic algorithm process is one of the key processes in this algorithm because it is responsible for the changes in the next generation changes, so there is a possibility that the Pc process with a very big impact on future generations and therefore the speed of finding a solution (or solutions) is required. As is known, the likelihood of something between zero and 1, it has zero indicates not happen and the one referring to the constant occurrence, as we explained earlier, the chromosomes of the fetus born consists of a combination of the chromosomes of the mother and the chromosomes of the father, and sometimes they tend chromosomes of the fetus to the mother and sometimes those tend chromosomes to strongly Father, this means that the likelihood of the exchange between chromosomes means that the fetus takes on both sides, so the likelihood of boom should not be a small value (for example, less than 0.2) so that they reduce the mutation process and should not be so big that strongly occur (for example, greater than 0.8).

GA Parameters

Prof. Dr. Ahmed Tariq Saadeq

- Mutation Probability (Pm): Unlike the exchange process, the probability of occurrence of a genetic mutation (Pm) in the fetus are certainly a few, but they occur out of the question can not be neglected in the next generation as a result of several medical and genetic and social reasons are not in the process of addressing them. It is the probability of an adverse genetic mutation (Pm) is definitely going to be a few as it is in the best cases, does not exceed 0.2 according to most research and sources, and sometimes much less may be the value of the boom (Pm) is equal to 0.003, for example. In fact, to the benefit of genetic mutation in the genetic algorithm has many people think is useless and the opposite is true.
- Fitness Function: A mathematical function reflects how close the current solution of the desired solution (or current solutions required) solutions, which play a very important role in obtaining the required solutions and whenever selected successful was getting faster solution, but sometimes represent the difficulty of obtaining congruence resolve function a specific problem by using a genetic algorithm is a problem in itself. There are those who assumed certain mathematical function and try them in terms of execution speed and accuracy solutions and after experiments to be agreed on a specific function.

GA Parameters

Prof. Dr. Ahmed Tariq Saadeq

- Stop Conditions: When we stop work the genetic algorithm running in search of a solution (or solutions)? We can be summarized in terms of stopping three (of course the occurrence of one of those conditions leads to stop the execution of a genetic algorithm and not combined) conditions. The first condition is that finding the solution according to the criteria and parameters given. The second condition is the arrival of the number of generations generated to the limit (Max-Generation) given in execution. The third condition is the occurrence of the case of no return and are intended to here the case of (Local-Minima), in other words, that the generations that breed in turn where there is no significant diversity or change in her condition, and is therefore in a state of inertia leading to inaccessibility of the solution It is required in view of the lack of diversity property (diversity) which plays an important role in reaching a solution. The availability of any of the above conditions will result in an end to the implementation of a genetic algorithm and reboot again, or return to the first step of generating a primary community and call the main loop implementation of genetic algorithm may be that this treatment if there is no solution to the result.

Samples of Fitness Function

$$f(x) = x^2 + 2x + 1$$

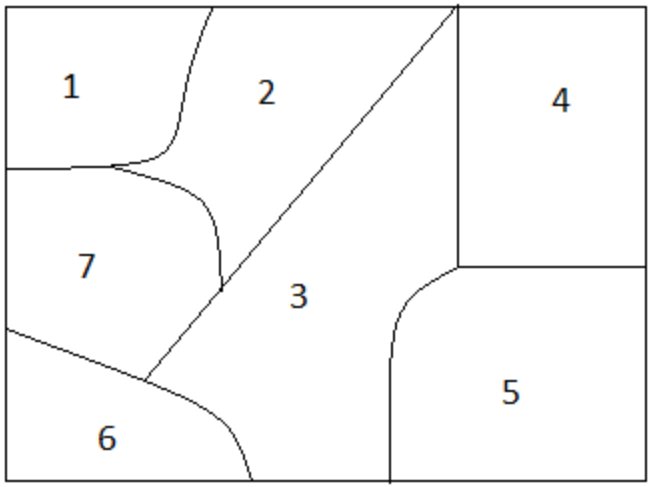
$$f(x) = x^2 + 1$$

$$f(x) = \frac{x}{\sum x^2}$$

Problem	Fitness Function
4-Color Mapping	A number of the same color neighboring cities
Shortest Path	Total cost of the path from the starting point to the end point
Traveling Salesman	Total cost of the first track of the city and return to it passing through all the cities at once
Graph Planarization	The number of intersections in the diagram
Best Path Robot	The remaining distance between the current situation and the state of the target

Application: 4-Color Mapping Problem

There are 7 cities as in the map, the adjacency matrix as below:



$$Adjcency = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Assume the GA generate 4 solutions (pop_size) randomly as :

G R R Y B G B

R B Y Y G R B

B G G Y R R Y

Y G R R B G B

Apply the fitness function on the population :

	f(x)
G R R Y B G B	1
R B Y Y G R B	2
B G G Y R R Y	1
Y G R R B G B	1

Assume the selection operation choice chromosome 1 & 2, to perform crossover operation as below:

G R | R Y B G B
R B | Y Y G R B

crossover in the 2nd position

R B R Y B G B
G R Y Y G R B

The mutate chromosome 1 (last gene) and chromosome 2 (2th gene) as below:

R B R Y B G Y
G G Y Y G R B

Compute the fitness function for the 2 new chromosome :

	f(x)
R B R Y B G Y	0
G G Y Y G R B	2

That means there is no 2 adjacency cities has same color as in the below map:

R B R Y B G Y