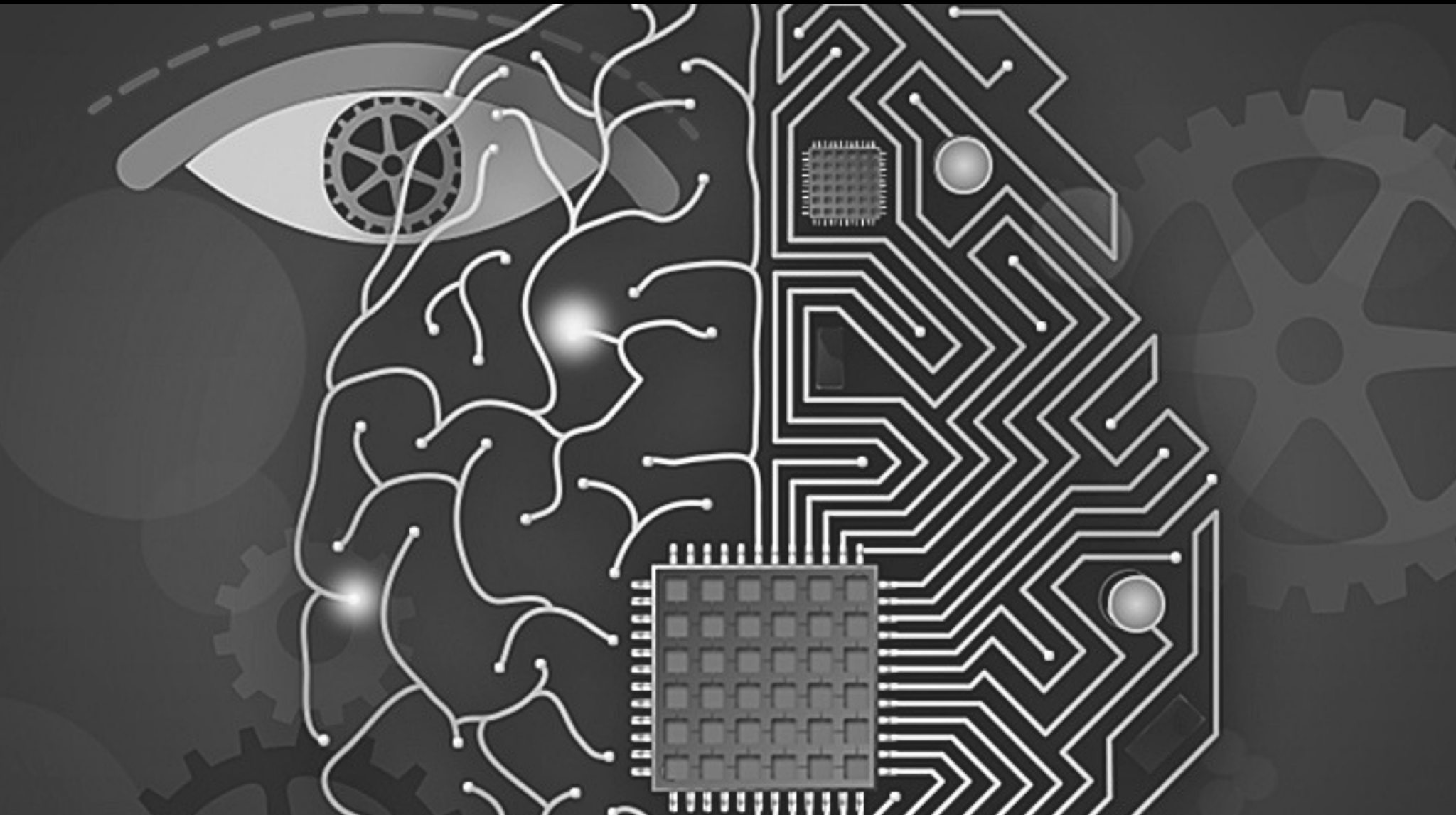
Cognitive Engine Design



There is a lot of algorithms out there which could be used as in a cognition process (AI>Machine learning).

ANN
Artificial Neural Network

MLP
Multilayer perceptron

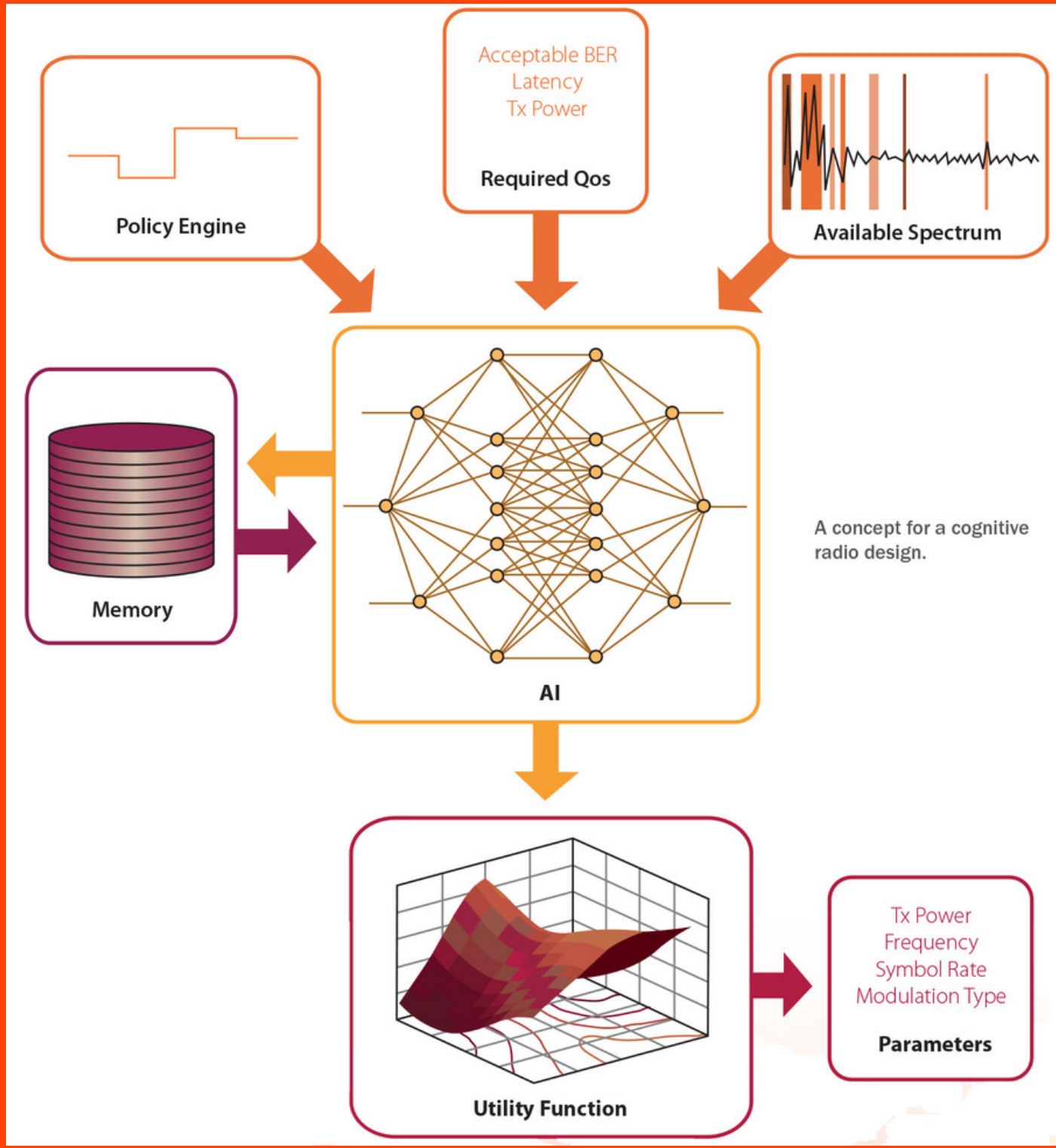
CBR
Case Based Reasoning

GA
Genetic Algorithm

So, we have to choose which one fits our desired target

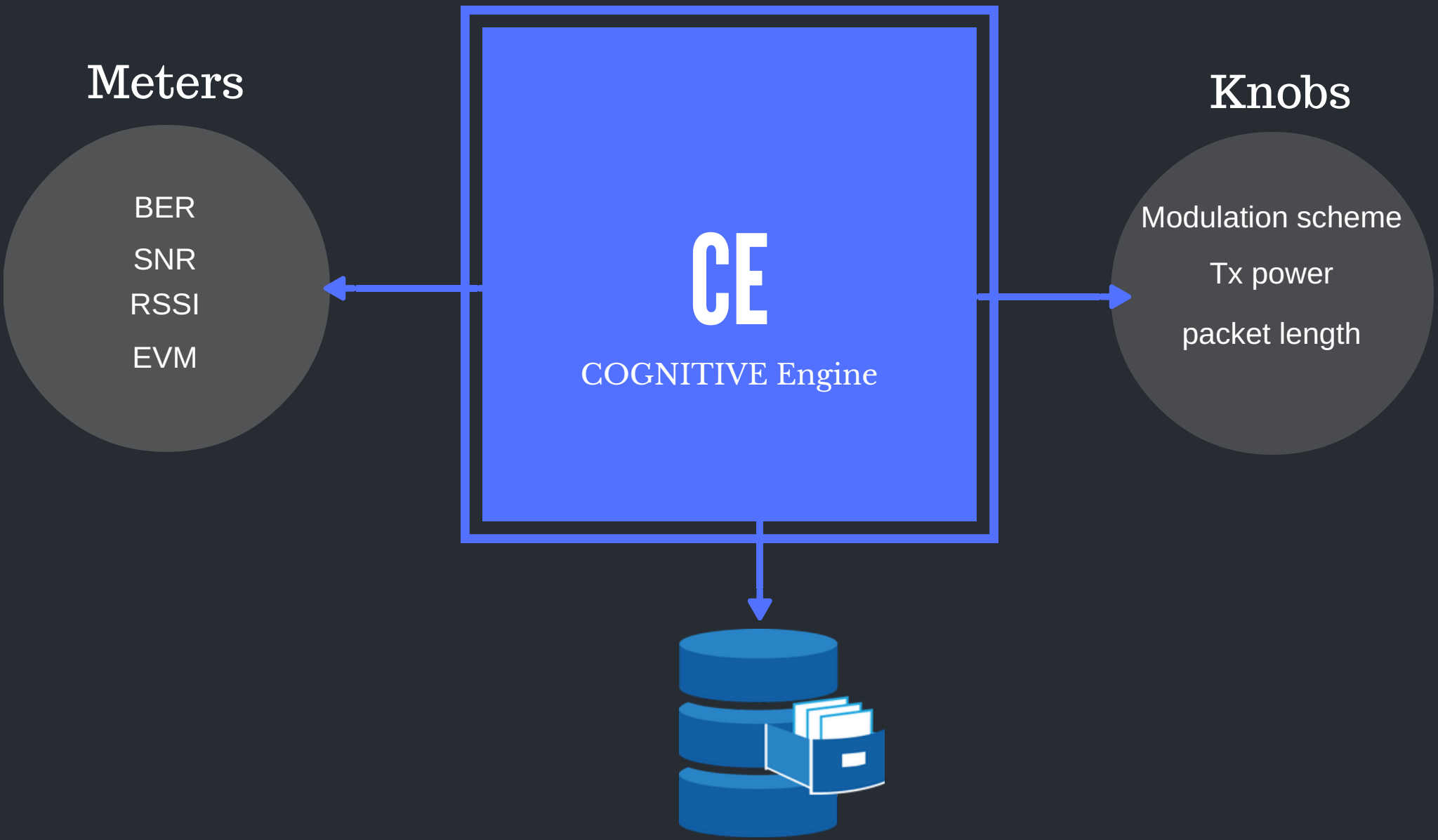
The interaction between the cognitive engine and the radio through its knobs and meters.
In cognitive radio terms, the waveform is the wireless signal transmitted that represents the current settings of all of the radio's knobs.

Meters represent the metrics used in the radio optimization. Knobs include the type of modulation and modulation parameters, frequency channel, symbol rate, and channel and source coding. Meters include bit error rate (BER), frame error rate (FER), signal power, battery life, and computational resources.



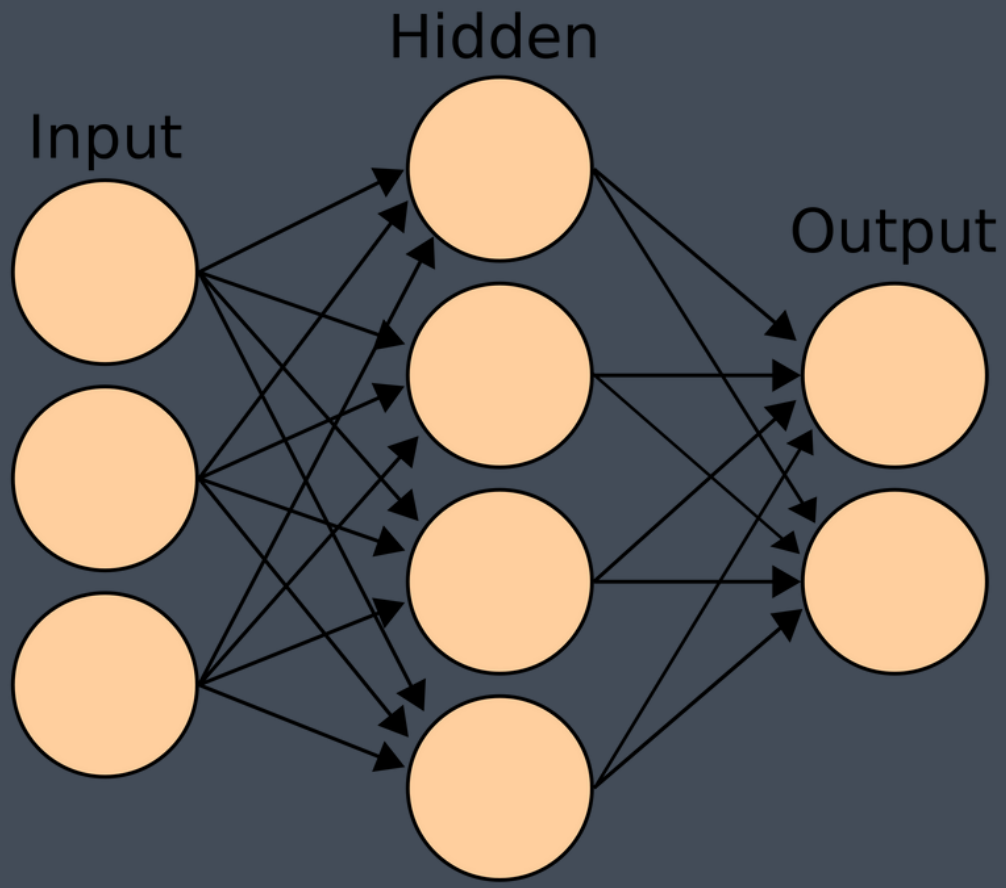
A concept for a cognitive radio design.

DEMONSTRATION OF CE



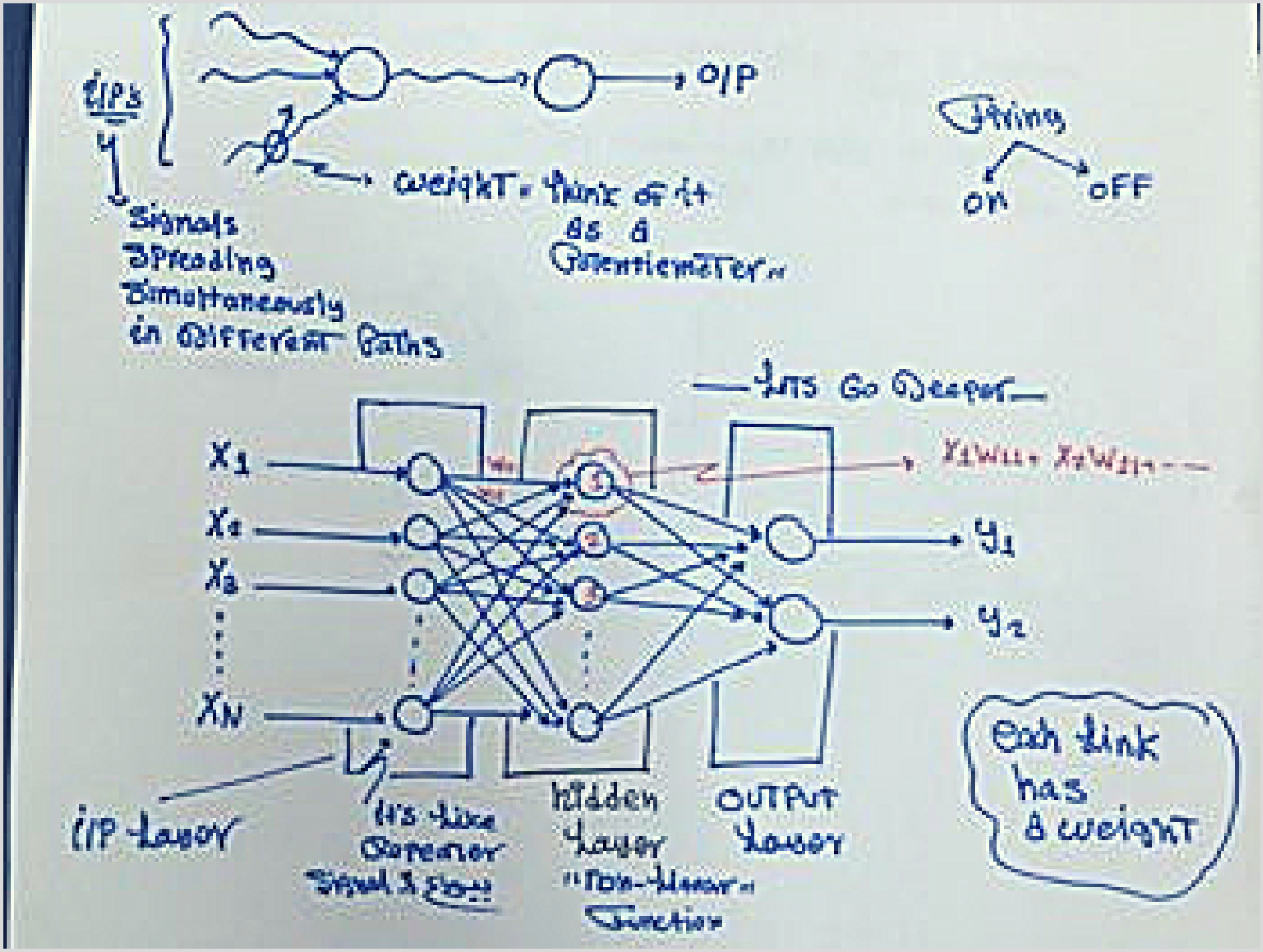
After searching I decide to work with the ANN algorithm And Algorithm selection still negotiable.

Why I selected ANN?
A huge amount of resources.
+
I Like it.



GO TO THAT LINK TO SEE THE FUNCTION OF ANN

http://gifmaker.cc/PlayGIFAnimation.php?folder=20170111122rcKRspBeQ0Cuq1ipgyVzfH&file=output_DOZHV5.gif

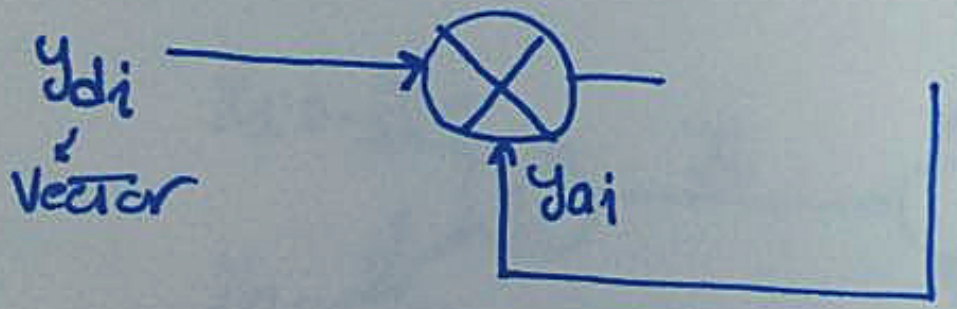

I/Ps
Signals
Spreading
Simultaneously
in Different Paths
weight - think of it
as a
Potentiometer
O/P
String
ON
OFF
Ins Go Deeper
X1
X2
X3
XN
Y1
Y2
I/P Layer
Hidden
Layer
"Non-Linear"
Function
OUTPUT
Layer
Each link
has
a weight

Modeling of Weights

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} \cdots \\ W_{21} & W_{22} & W_{23} \cdots \\ W_{31} & W_{32} & W_{33} \cdots \\ \vdots & \vdots & \vdots \\ W_{N1} & W_{N2} & W_{N3} \cdots \end{bmatrix}$$

TRAINING Process ⟹ Simply: بستخدم الـ I/P DATA واشوف الـ O/P

واقارنه بالـ O/P اللي انا عارفه وبناءً عليه، بعمل Adjusting للـ Weights



$$e(error) = Yd_{(n)} - Ya_{(n)}$$

Yd(n) ← Desired
Ya(n) ← actual

: Another important Parameters

α → learning Rate.

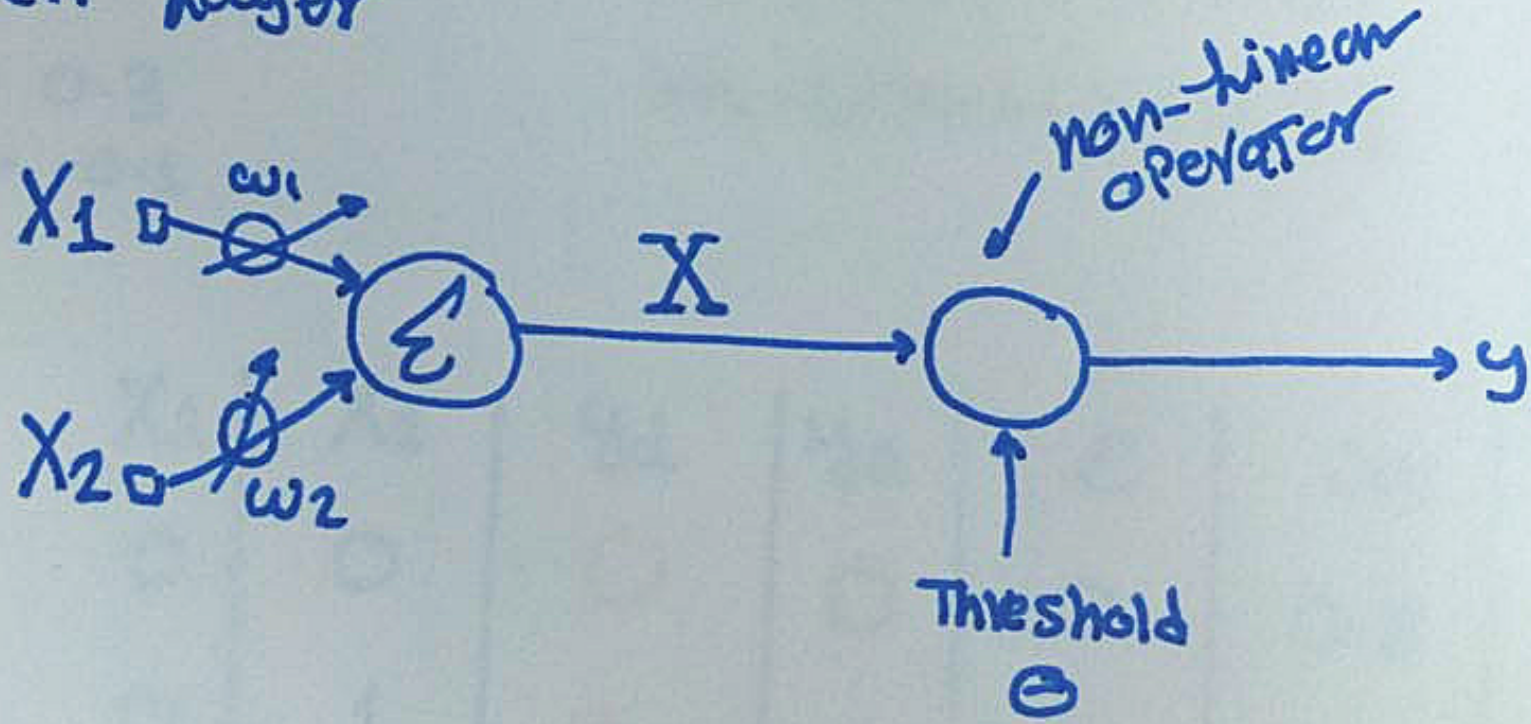
Simple neural network

2 input → one output →

Assumption:

Input is digital 1 or 0

no hidden layer

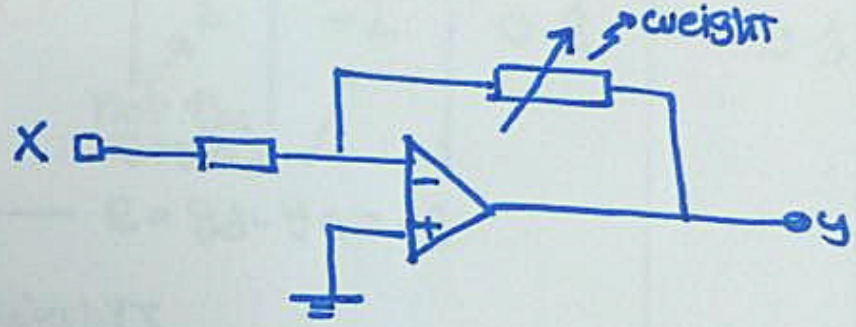


X_2 w_2

Threshold θ

Weight ??

Think of weight as an Amplifier with Variable Gain



Perceptron??

e.g. linear Function

$y = \text{step}(X - \theta)$! where $X = x_1 w_1 + x_2 w_2$

o/p

as a matrix notation $[X] = [x_1 \; x_2] \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

(ex) TRAIN neural network To Solve AND GATE

function

o/p ← $y = \text{step}(X - \theta)$; where $X = x_1 w_1 + x_2 w_2$

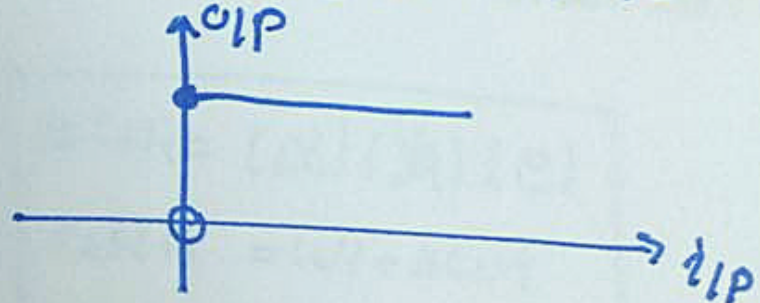
as a matrix notation $[X] = [x_1 \; x_2] \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

(ex) TRAIN neural network To Solve AND GATE

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

← TRAINING DATA

$$\text{step}(x) = \begin{cases} 1 & : x \geq 0 \\ 0 & : x < 0 \end{cases}$$



Assume: Learning Rate $\alpha = 0.1$

Θ (Threshold) $= 0.2$

Random Values For weights $[-0.5 : 0.5]$

$W_1 = [\text{Rand}() - 0.5]$

$W_2 = [\text{Rand}() - 0.5]$

Initial Values $W_1 = 0.3$

$W_2 = -0.1$

Random Values For weights [-0.5 : 0.5]

$W_1 = [Rand() - 0.5]$

$W_2 = [Rand() - 0.5]$

Initial Values $W_1 = 0.3$, $W_2 = -0.1$

Epoch	X_1	X_2	y_d	y_a	e	w_1	w_2
1	0	0	0	0	0	0.3	-0.1
	0	1	0	0	0	0.3	-0.1
	1	0	0	1	-1	0.2	-0.1
	1	1	1	0	1	0.3	0

$step[0.2 + 0] = 0$

$y_a = step[0.2 + X] = 0$

y_a not the Desired

$e = y_d - y_a = -1$

So I have to adjust the weights

$\Delta w_1 = [\alpha][X_1][e]$

$= 0.1 * 1 * -1 = -0.1$

So

$W_{1\text{-new}} = w_1 + \Delta w_1 = 0.3 + (-0.1) = 0.2$

$\Delta w_2 = \alpha * 0 * -1 = 0$

epoch	X_1	X_2	y_d	y_a	e	w_1	w_2
2	0	0	0	0	0	0.3	0
	0	1	0	y0	0	0.3	0

hey Wait How to calc y_a

$y_a = \text{step}[X - \Theta]$

$\text{step}[(0 * 0.3 + 1 * 0) - 0.2] = \text{step}[-0.2] = 0$

1	0	0	1	-1	0.2	0

$y_a = \text{step}[0.3 - 0.2] = 1$

So

$\Delta w = \alpha \times e$

$\Delta w_1 = (0.1) * (1) * (-1) = -0.1$; $w_1 = 0.3 - 0.1 = 0$

$\Delta w_2 = (0.1) * (0 + 1 - 1) = 0$; $w_2 = 0$

1	1	1	1	0	0.2	0

$y_a = \text{step}(0.2 - 0.2)$

$\text{step}(0) = 1$

dw no error

epoch n

ref:
https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/