R BENJAMIN FRANKLIN

aRMY iNSTITUTE OF TECHNOLOGY  BE IT

aRTIFICIAL nEURAL nETWORK

ARTIFICIAL NEURAL NETWORKS

Table of Contents

[Introduction: 3](#_Toc40994276)

[Basic Concepts: 4](#_Toc40994277)

[Neural Network: 4](#_Toc40994278)

[Biology - AI – The – Connection: 4](#_Toc40994279)

[Difference between biological neural network and artificial neural network: 5](#_Toc40994280)

[Building Blocks: 6](#_Toc40994281)

[Neurons: 6](#_Toc40994282)

[Network Topology: 6](#_Toc40994283)

[Feedforward Neural Network: 6](#_Toc40994284)

[Feedback Network: 7](#_Toc40994285)

[Commonly Used Activation Function: 7](#_Toc40994286)

[Supervised and Unsupervised learning: 8](#_Toc40994287)

[Supervised learning: 8](#_Toc40994288)

[Unsupervised learning: 9](#_Toc40994289)

[Learning and Adaption: 10](#_Toc40994290)

[Sources 12](#_Toc40994291)

[Learning Vector Quantization 13](#_Toc40994292)

[Codebook vector 13](#_Toc40994293)

[Euclidean Distance: 13](#_Toc40994294)

[Best Matching Unit: 13](#_Toc40994295)

[Sonar, Mines vs. Rocks: 13](#_Toc40994296)

[Code: 14](#_Toc40994298)

[Adaptive Resonance Theory 18](#_Toc40994299)

[ART Architecture: 19](#_Toc40994300)

[Kohonen Self-Organizing Feature Maps 21](#_Toc40994301)

[Structure and Properties of Self-Organising maps: 21](#_Toc40994302)

[Algorithm: 22](#_Toc40994303)

[Associate Memory Network 23](#_Toc40994304)

[Auto Associative Memory: 23](#_Toc40994305)

[Hetero Associative memory: 24](#_Toc40994306)

[Hopfield Networks: 25](#_Toc40994307)

[Hopfield network 1: 25](#_Toc40994308)

[Hopfield Network 2: 25](#_Toc40994309)

[Updating Rule: 26](#_Toc40994310)

[Attraction and repulsion of neurons in state space 26](#_Toc40994311)

[Energy: 27](#_Toc40994312)

[Training: 27](#_Toc40994313)

[Boltzmann Machine: 28](#_Toc40994314)

[Structure and working of Boltzmann Machine: 28](#_Toc40994315)

[Restricted Boltzmann Machine: 29](#_Toc40994316)

[Energy equation 29](#_Toc40994317)

[Brain-State-in-a-Box Network: 30](#_Toc40994318)

[Brain-State-in-a-Box Network Model: 30](#_Toc40994319)

[Energy function: 31](#_Toc40994320)

[Applications of BSB: 31](#_Toc40994321)

[Optimization Using Hopfield Network: 32](#_Toc40994322)

[Travelling Salesman Problem: 32](#_Toc40994323)

[Optimising the Travelling Salesman Problem using Hopfield Networks: 32](#_Toc40994324)

[Other Optimization Techniques: 33](#_Toc40994325)

[Gradient Descent: 33](#_Toc40994326)

[Simulated Annealing 34](#_Toc40994327)

[Algorithm: 34](#_Toc40994328)

[Genetic Algorithm: 35](#_Toc40994329)

[Algorithm: 35](#_Toc40994330)

[Advantages of Genetic Algorithm: 36](#_Toc40994331)

[Drawbacks of Genetic Algorithm: 36](#_Toc40994332)

[Applications of Neural Network: 37](#_Toc40994333)

[Image Processing: 37](#_Toc40994334)

[Text Classification and Recognition: 37](#_Toc40994335)

[Forecasting: 38](#_Toc40994336)

[Signature Detection and Verification: 38](#_Toc40994337)

[Artificial Neural Network Resources: 39](#_Toc40994338)

[Artificial Neural Network Interview Questions answer: 40](#_Toc40994339)

[References 44](#_Toc40994348)

Artificial Neural Network Tutorial

# Introduction:

T

he world of Artificial Intelligence amazes each and every one of us with its new discoveries and solutions to our everyday problems. Be it detecting diseases to boosting business through future predictions, AI has grown to be a household name in the present age.

A rather layman term to understand Artificial Neural Network would be to see it in the lens of Y.-S. Park.

“Artificial neural networks (ANNs) are biologically inspired computational networks.”

Y.-S. Park

Today the data scientist is one of the highest paid person in the corporate world. Companies are on the lurch of recruiting people with a good knowledge in data science or AI making it a popular field to study.

As we embark through this tutorial series I will make sure that you enjoy learning each and every bit of this wonderful learning journey.

Artificial Neural Network as the name suggests refers to a series of interconnected network across various nodes. This neural network is often compared with the neurons in our brain. Yes! You got me right. Understanding ANN using the window of a neuron can be quite intuitive and fun.

So let’s get started.

# Basic Concepts:

A

rtificial Neural Networks are one of the fastest growing fields in the technological world. Today automation of machines and various other technological tools have taken up pace like never before. Therefore, understanding of Artificial neural networks is of paramount importance if one intends to venture in this domain.

## Neural Network:

According to Wikipedia, the definition of neural network is as follows

"A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems."

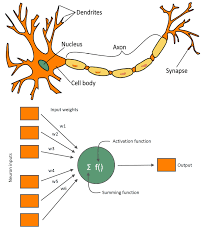
Source: Neural network - [https://en.wikipedia.org](https://en.wikipedia.org/wiki/special:search/Neural%20network)

We can infer from the definition that the neural network closely resembles our own biological neural network in the basic term. Neural networks help us in classifying and clustering data among the many things it can do. In a nutshell it is a computational model inspired by the human biological system.

*Source: File:Neural network example.svg -* [*https://en.wikipedia.org*](https://commons.wikimedia.org/wiki/File:Neural_network_example.svg)

## Biology - AI – The – Connection:

We all must have in some point of time studied about the neuron and its function in the brain. Well happens that a neuron the fundamental unit of a brain. Millions of neurons can be said to reside inside the brain. These neurons form synapses which is a structure that enables a neuron to pass a signal to another neuron in the brain. These synapses can be strong or weak depending on our memory to remember certain things.



(Image Source:https://www.ee.co.za/wp-content/uploads/2019/07/Application-of-machine-learning-algorithms-in-boiler-plant-root-cause-analysis-Fig-1.jpg)

In a computational model, a neural network is similar to the neuron. After receiving input from other sources it gives the output after processing or computation. The inputs received from other nodes are weighted which basically signify the significance of the input in determining the output. All these weights with the inputs are summed together in a function called the activation function. After this is computed and if the result is above a certain threshold, then the output is generated.

## Difference between biological neural network and artificial neural network:

|  |  |
| --- | --- |
| BNN | ANN |
| The biological neuron is undoubtedly complex and is a network of millions of neuron. | **The number of neurons is ANN is not that much of a number.** |
| There are hundreds of connections between neurons which can fire asynchronously. | **The layers in ANN are not connected to each other, rather they are computed step by step.** |
| There is no particular structure or way in which these neurons remember. | **The way ANNs works is very precise and structured.** |

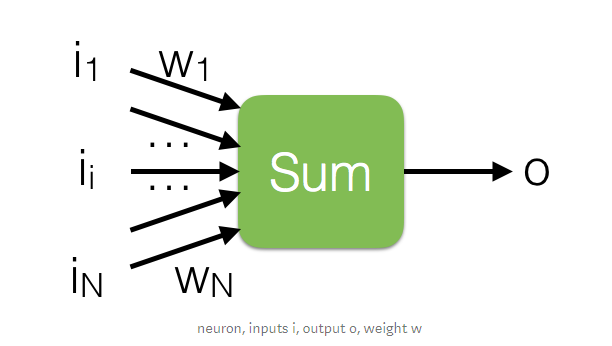
(What is the difference between biological and artificial neural networks?, n.d.) (Nagyfi, 2018; Fumo, 2017; Neural Network Definition, n.d.)

# Building Blocks:

Artificial Neural Networks are made up of some basic building blocks. Let’s look at these basic blocks which build up the neural network.

## Neurons:

Neuron is the basic building block of a neural network. These neurons take inputs from several other neurons, sum up the respective weights from these neurons and finally fires the output depending on the threshold value. Weights are assigned to each of these neurons.



(Image source: https://miro.medium.com/max/984/1\*9mndvDtRF0le-niGnpUKXg.png)

## Network Topology:

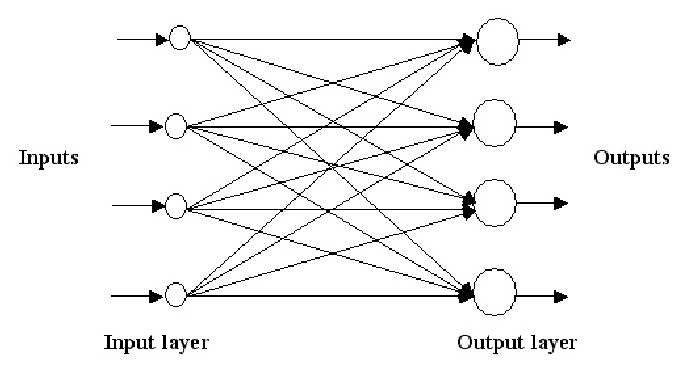
### Feedforward Neural Network:

*Source: File:Feed forward neural net.gif -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/File:Feed_forward_neural_net.gif)

This is the simplest form of neural network and was one of the very first networks to be designed. This network doesn’t form any cycles. This is an iterative network where the output of a particular step depends on the present outcome. Here the movement of information is only linear and moves in one direction.

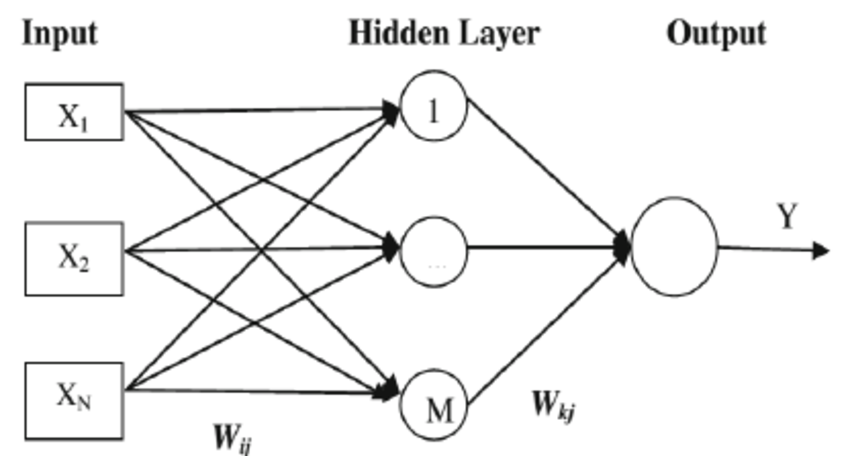
#### Single feedforward network:

Single feedforward network has only one weighted layer and thus the input layer is fully connected to the output layer.



#### Multilayer feedforward network:

Multilayer feedforward network has more than one weighted layer. This layer has hidden units which are the layers between the input and output layer.



## Feedback Network:

This is an iterative process where each iteration’s operations depends on the present output. This helps in making predictions in areas like control hypothesis. Here the network has feedback paths and the signal can flow in both directions using loops. It is a non-linear dynamic system wherein the change is continuous till it reaches the equilibrium state.

## Commonly Used Activation Function:

The activation function takes up some input and passes through a function and return the output after performing some functions. Some of the activation functions

* Sigmoid function
* ReLU
* Tanh
* Leaky RelU

(Fumo, A Gentle Introduction To Neural Networks Series — Part 1, 2017) (Understanding The Essential Blocks of Artificial Intelligence - Neural Networks, n.d.) (Culurciello, 2017) (Ruiz, 2018) (Building Blocks, n.d.) (Artificial Neural Network - Building Blocks, n.d.)

# Supervised and Unsupervised learning:

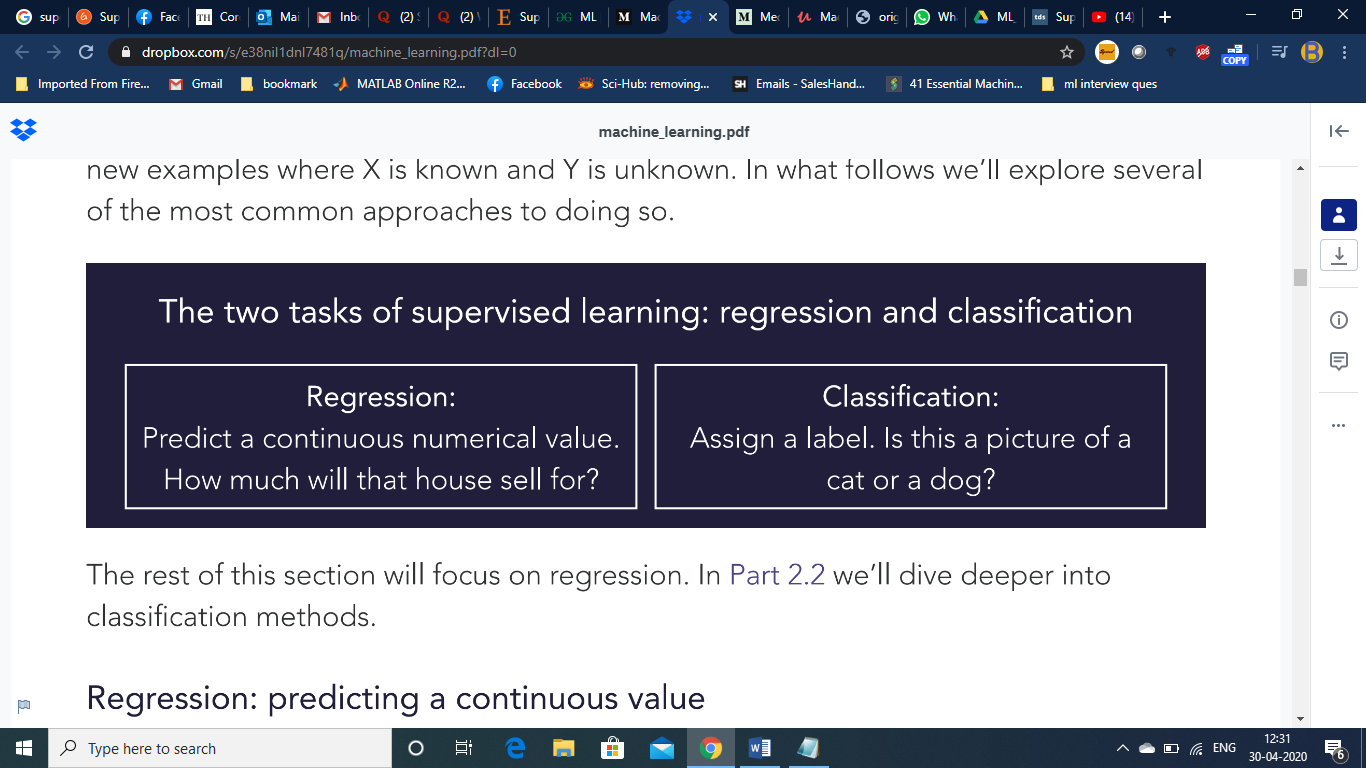
## Supervised learning:

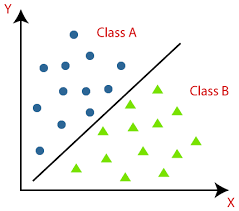
S

upervised learning works on labeled dataset. It takes in some input. In supervised learning a training set which can be in the form of well labeled spread-sheet is inserted as input to the model. This labeled dataset consists of input and output parameters.

Let’s say I need to predict how much time it will take to reach my college from my home. I have all the data related to the weather, traffic and based on whether my already faulty elevator works on that particular day. I call this as my training set. I feed it to my machine learning system. Now my system will try to find some really interesting relations between these labeled features. For example, it may predict that if I start at 7am for college when the traffic is less then I reach the college faster compared to starting for the college at 8am.

Regression and classification are two tasks of supervised learning.





Classification

(Image source: https://static.javatpoint.com/tutorial/machine-learning/images/classification-algorithm-in-machine-learning.png)

## Unsupervised learning:

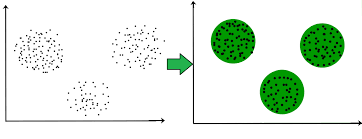
U

nsupervised learning tries to structure an unstructured dataset. Here the dataset is not labeled with the features. Here each piece of data passed through our model is unlabeled.

Well, the question arises as to how the model will be able to predict the output or find relations.

Let’s dive into an interesting example of unsupervised learning:

Imagine you want to classify people living in India and China. You are given the data in the form of tuples (Eg [162,89] which denotes the height and age of a person). This is fed into our system. Now imagine the graph looks somewhat like this.



(Image source: <https://media.geeksforgeeks.org/wp-content/uploads/merge3cluster.jpg>)

The system clusters the people into unlabeled groups based on the data provided by us. We can assume that one is predominantly an Indian and the other a Chinese.

Another interesting application of unsupervised learning is auto encoder. An auto encoder is a neural network that is trained to reconstruct its own input. That’s quite confusing right. Let’s make it simple.

Imagine you have a list of digits from 0-9 in handwritten form and you pass it through an auto encoder. The auto encoder encodes it followed by decoding. The aim is to make the output image as close as possible to the original image.

# Learning and Adaption:

We have learnt that ANN is quite similar to how our biological nervous system works. Isn’t it so interesting to know that we are learning something which is quite similar in working to our how we think, adapt, classify and respond. So let’s dive even deeper into more interesting stuff.

**Learning** in artificial neural network is an iterative process where the system learns depending upon the inputs and the weights associated with each of the inputs.

**Adaption** as the name suggests, is adapting and adjusting to different situations. ANN is a complex adapting system which canadapt to changes based on the information fed to it.

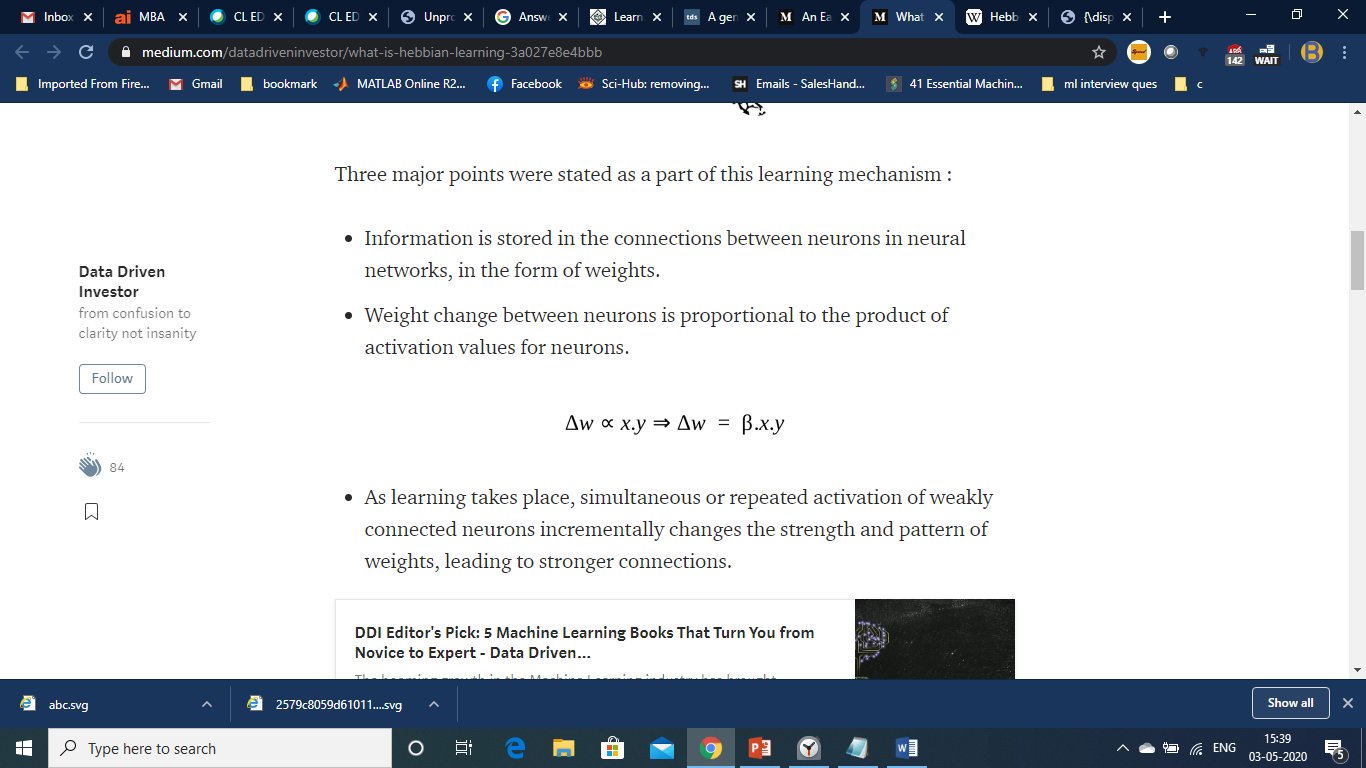
Let’s look at some of the learning rules. In order to change the behaviour of the input or the output, we need to change the weights and the same for which we need a method in place. These methods are called learning rules.

Let’s look at some of the popular learning rules:

1. Hebbian Learning rule

[Donald O. Hebb](https://en.wikipedia.org/wiki/Donald_O._Hebb) came up with an interesting way to update the weights between neurons in a neural network. And this became a cause for the neuron to learn and hence was called Hebbian learning rule.

This learning rule assumed that information between neurons to be weights associated with it. It also stated that the change in weight was proportional to the product of activation values of neurons



Formula for Hebb learning rule.

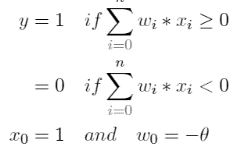
It stated that when both the nodes are either positive or negative at the same time, then a positive weight exists between them while if one of them is positive and the other negative or the vice versa, then there would exist a negative weight between them. If both doesn’t hold true then the weight would remain unchanged.

So we have learnt probably one of the first learning rules found by humans. Let’s see the next one which is quite popular when it comes to learning rules.

1. Perceptron Learning Rule

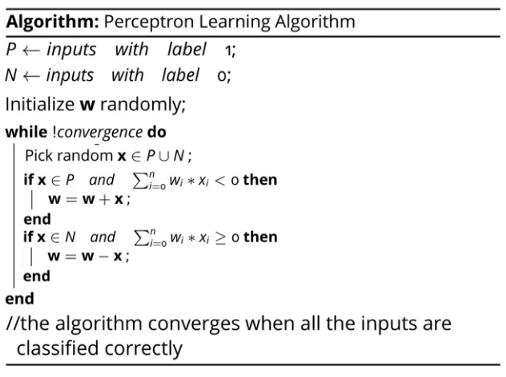
The perceptron model is a computational model which takes an input. Once the input is accepted it aggregates it, which simply means it sums up the weights and then either return 1 or 0 based on the threshold value. Here ‘w’ indicates the weights and the perceptron accepts the weights along with the bias.

Linearly separable functions are only possible to be implemented by a single perceptron.



Perceptron learning algorithm:

Perceptron as we well know is a linear classifier which basically which basically is used in classification. It basically works on weights and feature vector.



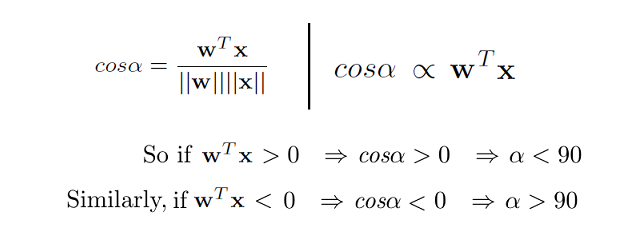
(Image Source:https://hackernoon.com/photos/JTw2M3rQabaxNg3EFoNIxjmC1ZB3-eq2332p6)

Now, our purpose is to classify the positive and negative inputs in our data using the weights(w). So let’s understand this with the help of this algorithm.

* First up, we initialize w with some random value.
* Our algorithm then loops through both the positive(P) and negative(N) inputs.
* Do you notice something odd in the algorithm? If you have, it great. Otherwise let’s dive into finding what it is.
* Ideally for x € P, the dot product of the vectors w.x >0 and for x € N, w.x<0.
* But here it’s the other way round and is not following the perceptron rule. Therefore, we add w to x if the product is less than 0 and the vice versa otherwise.

Now let’s see why it still works,

Now since the cosine of the angle is proportional to the dot product, the angle should be less than 90 when x € P.



(Image Source: [https://miro.medium.com/max/1066/1\*AjFrPxeZEot-apPCAC3mSQ.png](https://miro.medium.com/max/1066/1*AjFrPxeZEot-apPCAC3mSQ.png))

Wow! That really clears some things. Let’s move on.

Now we know that when x € P, w.x<0, we are adding x to w, Here, actually the value of cos is increased which means that α is getting reduced. The same mechanism works when w.x<0.

As w moves around the negative and positive axis, it finally does converge according to the proof of convergence.

And voila! The weights are learning by the machine.

Some of the other popular learning algorithms include,

1. Delta learning rule
2. Competitive learning rule
3. Outstar learning rule

## Sources

1. <https://www.upgrad.com/blog/perceptron-learning-algorithm-how-it-works/>

2.<https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_learning_adaptation.htm>

3. <https://ai.stackexchange.com/questions/2545/understanding-the-perceptron-algorithm-in-the-book-a-course-in-machine-learning?rq=1>

4. <https://nptel.ac.in/courses/106106184/>

# Learning Vector Quantization

S

ince we are learning ANN, I would assume you would know K-nearest neighbour. Because learning vector quantization works quite similar to it except that it is more efficient the reason being, it allows you to choose the number of training instances rather than working on the whole training set and thus being very efficient in terms of speed and processing(classification). Also, an important point to note is that LVQ is applicable for both binary and multiclass classification problems.

Before looking into the algorithm we must know some of the following terms.

Codebook vector:

A codebook vector is basically like a neuron. The attributes pertaining to codebook vector are called weights and the collection or group of codebook vectors is called network.

A codebook vector is an array of numbers that has the same input and output attributes as the training data. There are several patterns in the dataset. These patterns are learned by the machine. These pool of pattern is called codebook vector. A single pattern is called a codebook.

### Euclidean Distance:

In very basic terms, Euclidean distance is the distance between any two points. These two points can be linear or distributed in a two or three dimensional space.



Source: <https://www.cut-the-knot.org/pythagoras/DistanceFormula.shtml>

### Best Matching Unit:

A Best Matching Unit(BMU) is basically a codebook vector.

To understand Best Matching Unit(BMU), let’s imagine there are many different new data items coming up. Our system calculates the distance between a codebook and a new piece of data. Thus it is done for all new data items. Finally, we sort these distance in ascending order and select the first codebook vector which has the least distance or has the most matching codebook vector.

### Sonar, Mines vs. Rocks:

### I have used the Sonar dataset to show you the working of the LVQ algorithm. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. To know more about the dataset and to download it you can use the link given below.

I did experiment with the dataset and got an accuracy of 77.56% which is better than a K-means classification algorithm applied to it.

<https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)>

### Code:

I have basically referred a code by Dr Jason Brownlee, one of my favourite data science blog writer. You can try this code on a different binary or multiclass classification dataset and see the results for yourselves.

from random import seed

from random import randrange

from csv import reader

from math import sqrt

# Loading a CSV file

def load\_csv(filename):

  dataset = list()

  with open(filename, 'r') as file:

    csv\_reader = reader(file)

    for row in csv\_reader:

      if not row:

        continue

      dataset.append(row)

  return dataset

# string to float

def str\_column\_to\_float(dataset, col):

  for row in dataset:

    row[col] = float(row[col].strip())

# string column to integer

def str\_column\_to\_int(dataset, col):

  class\_values = [row[col] for row in dataset]

  unique = set(class\_values)

  lookup = dict()

  for i, value in enumerate(unique):

    lookup[value] = i

  for row in dataset:

    row[col] = lookup[row[col]]

  return lookup

# Splitting the dataset

def cross\_validation\_split(dataset, nfolds):

  dataset\_split = list()

  dataset\_copy = list(dataset)

  fold\_size = int(len(dataset) / nfolds)

  for i in range(nfolds):

    fold = list()

    while len(fold) < fold\_size:

      index = randrange(len(dataset\_copy))

      fold.append(dataset\_copy.pop(index))

    dataset\_split.append(fold)

  return dataset\_split

# Accuracy percentage

def accuracy\_metric(actual, predicted):

  correct = 0

  for i in range(len(actual)):

    if actual[i] == predicted[i]:

      correct += 1

  return correct / float(len(actual)) \* 100.0

# Cross validation split

def evaluate\_algorithm(dataset, algorithm, nfolds, \*args):

  folds = cross\_validation\_split(dataset, nfolds)

  scores = list()

  for fold in folds:

    train\_set = list(folds)

    train\_set.remove(fold)

    train\_set = sum(train\_set, [])

    test\_set = list()

    for row in fold:

      row\_copy = list(row)

      test\_set.append(row\_copy)

      row\_copy[-1] = None

    predicted = algorithm(train\_set, test\_set, \*args)

    actual = [row[-1] for row in fold]

    accuracy = accuracy\_metric(actual, predicted)

    scores.append(accuracy)

  return scores

# Euclidean distance between two vectors

def euclidean\_distance(r1, r2):

  distance = 0.0

  for i in range(len(r1)-1):

    distance += (r1[i] - r2[i])\*\*2

  return sqrt(distance)

# Locating the best matching unit(BMU)

def get\_best\_matching\_unit(codebooks, test\_row):

  distances = list()

  for codebook in codebooks:

    dist = euclidean\_distance(codebook, test\_row)

    distances.append((codebook, dist))

  distances.sort(key=lambda tup: tup[1])

  return distances[0][0]

# Make a prediction with codebook vectors

def predict(codebooks, test\_row):

  bmu = get\_best\_matching\_unit(codebooks, test\_row)

  return bmu[-1]

# Create a random codebook vector

def random\_codebook(train):

  n\_records = len(train)

  n\_features = len(train[0])

  codebook = [train[randrange(n\_records)][i] for i in range(n\_features)]

  return codebook

# Train a set of codebook vectors

def train\_codebooks(train, n\_codebooks, lrate, epochs):

  codebooks = [random\_codebook(train) for i in range(n\_codebooks)]

  for epoch in range(epochs):

    rate = lrate \* (1.0-(epoch/float(epochs)))

    for row in train:

      bmu = get\_best\_matching\_unit(codebooks, row)

      for i in range(len(row)-1):

        error = row[i] - bmu[i]

        if bmu[-1] == row[-1]:

          bmu[i] += rate \* error

        else:

          bmu[i] -= rate \* error

  return codebooks

# LVQ Algorithm

def learning\_vector\_quantization(train, test, n\_codebooks, lrate, epochs):

  codebooks = train\_codebooks(train, n\_codebooks, lrate, epochs)

  predictions = list()

  for row in test:

    output = predict(codebooks, row)

    predictions.append(output)

  return(predictions)

# Test LVQ on Ionosphere dataset

seed(1)

# load and prepare data

filename = '/content/sonar.csv'

dataset = load\_csv(filename)

for i in range(len(dataset[0])-1):

  str\_column\_to\_float(dataset, i)

# convert class column to integers

str\_column\_to\_int(dataset, len(dataset[0])-1)

# evaluate algorithm

n\_folds = 5

learn\_rate = 0.3

n\_epochs = 50

n\_codebooks = 20

scores = evaluate\_algorithm(dataset, learning\_vector\_quantization, n\_folds, n\_codebooks, learn\_rate, n\_epochs)

print('Scores: %s' % scores)

print('Mean Accuracy of LVQ is: %.2f%%' % (sum(scores)/float(len(scores))))

Output:

Scores: [80.48780487804879, 90.2439024390244, 70.73170731707317, 75.60975609756098, 70.73170731707317]

Mean Accuracy of LVQ is: 77.56%

So that’s all folks about LVQ. Hope you got an intuitive idea on this algorithm. Let’s move on to a much more bigger and interesting topics.

# Adaptive Resonance Theory

Let’s try to understand the theory a little intuitively. Imagine a neural network with an input layer, some hidden layers and an output layer. Just imagine if the nodes in the output layer can’t remember what is there in the input layer. Won’t it be a disaster. Well, ART resolves to address this particular issue. The title would say it all. **Adaptive and Resonance** which basically means that it can adapt to any learning and it can do so by not forgetting the old information.

Do you know that this theory helps to solve the plasticity-stability dilemma of Conventional Neural Networks. ART can carry out ‘on-line’ training as well as remembering the previously trained patterns. And what’s so interesting is that it can change the code according to the changes in the environment (l am not referring to the weather Lol) and thus is quite self-organising. Thus it plays a major role in solving this dilemma of CNN.



(Source:<https://www.researchgate.net/profile/Facundo_Bre/publication/321259051/figure/fig1/AS:614329250496529@1523478915726/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o.png>)

Interesting so far, is it? An algorithm which can remember the past trained data and is adaptive to the environment. Well happens that we have different kinds of ARTs.

#### ART – 1:

This type of ART is designed in a way to cluster the binary vectors.

#### ART – 2:

This is just an extension to ART 1 and clusters continuous-valued input data.

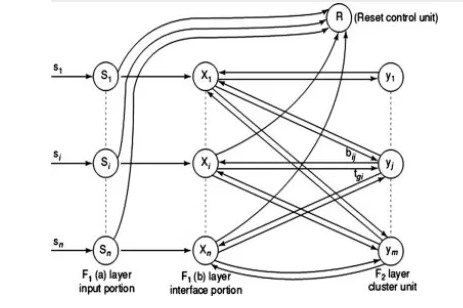
Other classifications include Fuzzy ART, ARTMAP AND FARTMAP.

### ART Architecture:

The ART architecture consists of two units

1. Computational unit
2. Supplement unit.

**Training Algorithm:**

****

We can consider 12 steps to train ART 1.

1. Initialize the weights.
2. Run a while loop till the condition is false
3. For each of the training data perform steps 4 to 11.
4. Set the activation of F2 units to zero and F1(a) units to input vectors.
5. Calculate 
6. Send input signal from F1(a) layer to F1(b) layer:

xi=si

1. For each F2 node that is not inhibited, the following rule should hold: If yj not=-1, then

yj=  ∑bij.xi

1. Perform step 8 to 11 when reset is true.
2. Find J for yJ>=yj for all nodes. If yJ =-1, then all the nodes are inhibited and this pattern cannot be clustered.
3. Activation X of F1(b) has to be recalculated
4. Norm of vector x is calculated(||x||=∑xi)
5. Test for reset condition.

If ||x||/||s||<ρ, then inhibit node J, yJ= -1. Go back to step 8 again.

#### Disadvantages and criticisms of ART:

Fuzzy ART and ART which are the learnt categories depend on the order in which training data is executed.

Consistency in ART models has always been a problem.

(art-adaptive-resonance-theor, n.d.)

# Kohonen Self-Organizing Feature Maps

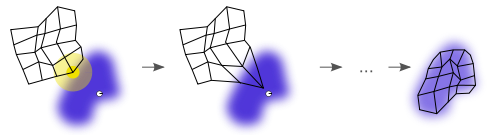
K

ohenen self-organizing feature maps are used to convert a high dimensional arbitrary space into a lower dimensional space(typically one dimensional). And this is done without disturbing the topology or the properties of the neighbouring input space. Hence feature mapping would be a quite useful tool in this process.

### Structure and Properties of Self-Organising maps:

As we know the Self-organising maps converts a high dimensional space into a low dimensional space, the definition provided by Wikipedia gives us more clarity on the same. It states.

“A **self-organizing map** (**SOM**) or **self-organizing feature map** (**SOFM**) is a type of [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) (ANN) that is trained using [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a **map**, and is therefore a method to do [dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction). “- Wikipedia



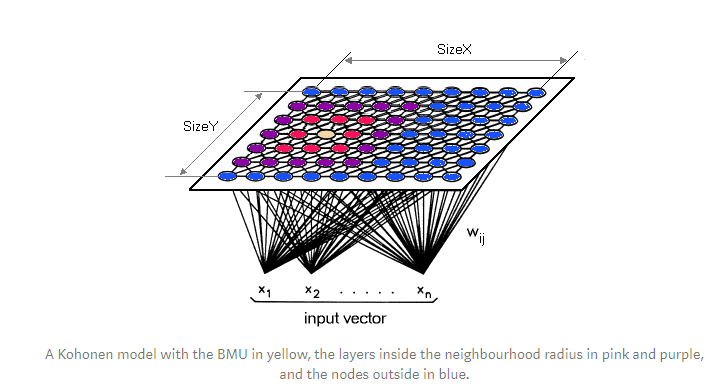
The image is illustrating a conversion from higher arbitrary dimensionality to a lower dimensionality

SOM involves two steps : Training and mapping

**Training** builds the map using competitive process.

**Mapping** classifies a new input vector automatically.

The structure of the SOM is a single layer linear 2D grid of neurons. The nodes though being connected to the input vector, do not know the values of the neighbours. The weights of these nodes are updated as a function of the input of the input data.



(Image source: https://miro.medium.com/max/840/1\*u9hfers-li-DOVerspOm4A.png)

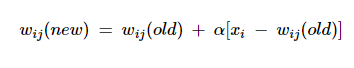
The self organising maps uses competitive learning to adjust and update the weights. At each iteration a single node is activated which is called the **Best Matching Unit**(BMU). Euclidean distance for the other nodes are calculated from the input node. The node with the least Euclidean distance is chosen along with its neighbouring nodes.

Thus nodes that are similar and those that are not are grouped separately.

### Algorithm:

Steps:

1. Initialise the weight w and the learning rate to any random value.
2. Next we should select a random input vector (say x).
3. Repeat the next two steps for all node.
4. Next we should calculate the Euclidean distance between the input vector x and the weight vector w of the first node.
5. Identify the node with the smallest distance t.
6. Find the smallest distance of the node(BMU) from the input vector node.
7. The neighbourhood function (β) and its associated radius *σ* is calculated.
8. The same is repeated for the nodes in the region of the BMU neighbourhood. The weight w is updated by the following formula



1. Repeat the loop until t=n.

(Eklavya, n.d.) (Eklavya, Kohonen Self-Organizing Maps, 2019) (Kohonen Self-Organizing Feature Maps, n.d.)

# Associate Memory Network

A

ssociate Memory Network, also called as Content-Addressable memory pertain to pattern association i.e. they store patterns and give out the best matching pattern with the input as its output.

They are classified mainly into two categories,

1.Auto associative memory

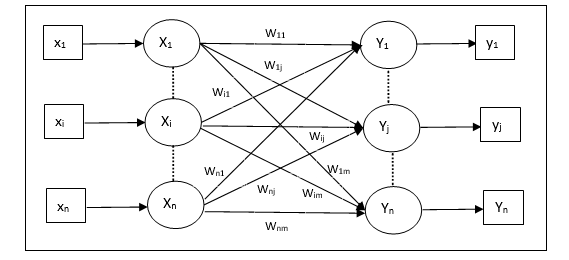
2.Hetero associative memory

### Auto Associative Memory:

Also known as **autoassociation network,** it is used to retrieve a piece of data from a small portion of data. It is a single layer neural network that stores a set of patterns.

#### Architecture:

The architecture of auto associative memory is quite simple which has same number of input and output training vectors.



(Image source: <https://www.tutorialspoint.com/artificial_neural_network/images/auto_associative_memory.jpg>)

The algorithm for this network is as follows

Steps

1. Initialise all the weights(w) to zero
2. Execute the next two steps for each input vector
3. Activate the input units one at a time as follows



1. Activate the input units one at a time as follows



1. Finally the weights should be adjusted

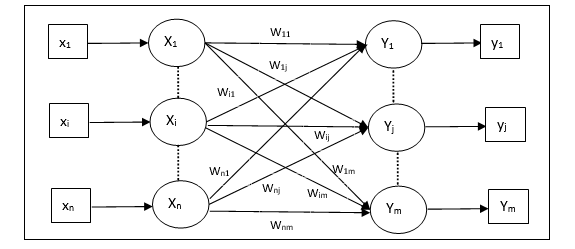


### Hetero Associative memory:

This is just like Associate Memory Network a single layer neural network with the difference that it does not have the same number of input and output vectors. There is no non linear and delay operations as the network is static in nature and the network stores the patterns with the help of the weights.

#### Architecture:

The diagram clearly indicates that the number of input and output vectors is not the same.



The algorithm for this network is as follows

Steps

1. Initialise all the weights(w) to zero
2. Execute the next two steps for each input vector
3. Activate the input units one at a time as follows



1. Activate the input units one at a time as follows



1. Finally the weights should be adjusted



(Tutorialspoint, n.d.)

# Hopfield Networks:

H

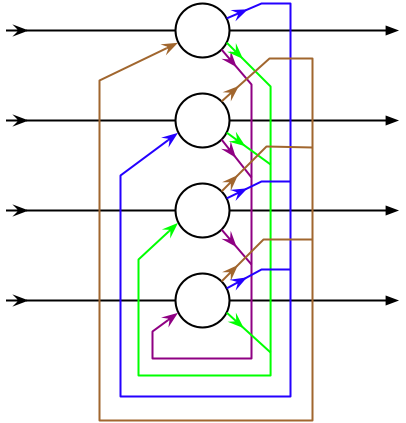
opfield networks are nothing but recurrent artificial neural networks. It’s one of the special neural networks which uses converging iterative process. The interesting part of this network is that it is guaranteed to converge at some point of time. Well, don’t you think that is great. But wait, are we missing out on something? Imagine what if the input provided is not sufficient or the model is faulty. It would still converge, but this convergence can lead us to the wrong local minimum instead of the expected local minimum or the stored pattern.

### Hopfield network 1:

John Hopfield introduced an ANN to store and retrieve memory like human brain.

Hopfield networks work on binary threshold units and recurrent connections. Here you can imagine in the sense of a neuron which can be turned on and off(0 or 1). An important point to note is that it doesn’t result in self loops when it comes to graph plotted.

So basically a Hopfield network is trained to store a number of patterns and then is able to recognize any of the learned patterns.

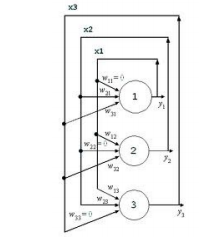


A Hopfield net with four units.

(Image source: By Zawersh at the English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4107292)

### Hopfield Network 2:

It is a single layered recurrent network. Here we have a concept called self connectivity weight w which is symmentric.



(Image source: <http://www.doc.ic.ac.uk/~ae/papers/Hopfield-networks-15.pdf>)

In the diagram, the given three neurons having values **i = 1, 2, 3** with values **Xi=±1** have connectivity weight **Wij**.

### Updating Rule:

An update can be provided by two ways

**Asynchronous update**

Here only unit is updated at a particular instance in time and can be picked randomly or pre-defined.

**Synchronous update**

Here, all the unit are updated in one go. The concept of central clock comes into picture here (although some see it as unrealistic) which maintains synchronisation.

The update rule states,

i

Where hi = ∑wijxj+bi which is called the field at i with bi€R a bias

Wij is the connecting strength weight between the units I and j.

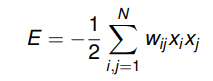
### Attraction and repulsion of neurons in state space

Remember the connecting weights (Wij)? Yes, the connecting weights between the units determine if the neuron will converge or diverge.

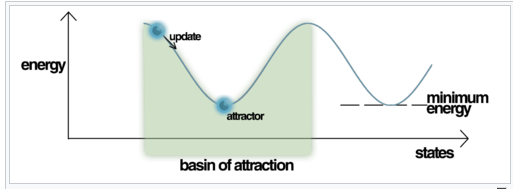
* If hj=1, hi is pulled by j towards si = 1. Here the neurons converge since the weights between them is positive.
* If hj=-1, hi is pushed by j towards si = 1. Here the neurons diverge since the weights between them is negative.

### Energy:

Hopfield has an energy function and what’s so interesting about it is that it either decreases or remains the same upon the network being updated.



The energy will eventually converge when it is updated again and again to a local minimum.



(Image Source: By Mrazvan22 - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=23796681Learning Rule)

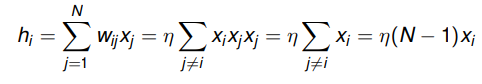
There are two properties of the learning rule that a Hopfield network model should typically possess.

1. The learning rule should be **local** i.e. the each and every weight is updated based on the information provided by the neighbouring neurons.
2. The learning should be **incremental** i.e. when a new pattern is learned, the new weight values depend on the old values and the new pattern.

### Training:

Consider, we want to learn a pattern p. We must store it first of all.

* So let’s say is some pattern we are interested in storing in a Hopfield network.
* To store this we should consider the weight wij. If we choose this weight wij= ηxixj between 1 and N then we find that the value xi doesn’t change. Let’s look how.
* Consider the formula given below which clearly clarifies and proves the above mentioned point.



* This clearly shows us that sign doesn’t really matter when it comes to Hopfield networks. You can try changing the value of xi to 1 or -1. You will notice even then will remain a fixed point.

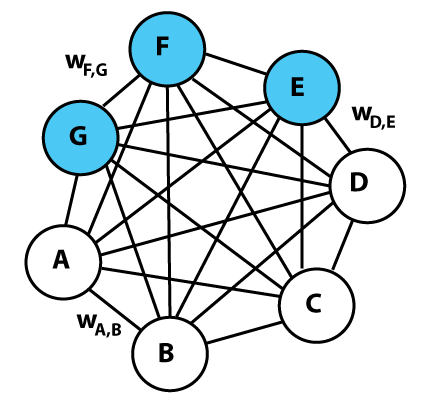
(Hopfield Networks, n.d.) (Hopfield network, n.d.) (Hopfield Network, n.d.)

# Boltzmann Machine:

B

oltzmann machine is a recurrent neural network. This machine is able to learn internal representations and able to solve combinatoric problems. The Boltzmann machine is a discovery by [Geoffrey Hinton](https://wikipedia.firstpartyapps.oaspapps.com/wikipedia/wikipedia_dev.html?et=%2BADwAZAA%2BAGwAMwBSAG0AYwBQAGcAWQBnAE4AQwBvAEYAdQBDAGwAZABtAHIASQBhAGwAaQBwAHoAVgB1AEEAOQA4ADIAagB6AEcARAA0AFEAQQA3AGIATQB0AHMAPQA8AC8AZAA%2BADwALwByAD4A&_host_Info=Word|Win32|16.00|en-US) who was a professor at Johns-Hopkins University. We can say that “Restricted Boltzmann machine” which was developed subsequently is a better version of Boltzmann machine as the learning rate is faster in the latter. They find their application in the field and thermodynamics and other energy fields and hence come under Energy-based models.

### Structure and working of Boltzmann Machine:



(Image Source: By Vera D - Created by Vera D, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=55007260>)

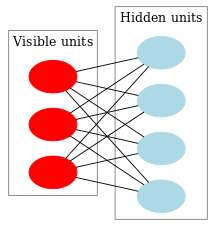
A Boltzmann machine is quite similar to Hopfield networks but is stochastic in nature and they decide whether to be on or off based on the training data and the cost function always try to reduce to a local minimum. The nodes are said to be either hidden or *visible.* The visible units receive information from the 'environment’.I know some of you must be wondering where the output for the model is in the diagram. Well, it happens that Boltzmann machine doesn’t produce any output. Well that’s quite weird isn’t it? But this machine works a bit different and we will see how. What makes this machine so special is that they need not remember patterns and output 1 or 0 after optimization. They fall into the category of unsupervised learning.

Another interesting thing to note is that the nodes are connected one to another unlike other traditional machines. And this is irrespective of whether they are hidden or visible. This allows for much efficient information sharing where they are able to store more patterns.

The Boltzmann Machine use neural networks where the neurons are connected to other neurons in other layers as well as within the same layer. This makes the Boltzmann machine a less favourable and highly inefficient machine.

### Restricted Boltzmann Machine:

Restricted Boltzmann Machine(RBM) is again a discovery by [Geoffrey Hinton](https://wikipedia.firstpartyapps.oaspapps.com/wikipedia/wikipedia_dev.html?et=%2BADwAZAA%2BAGwAMwBSAG0AYwBQAGcAWQBnAE4AQwBvAEYAdQBDAGwAZABtAHIASQBhAGwAaQBwAHoAVgB1AEEAOQA4ADIAagB6AEcARAA0AFEAQQA3AGIATQB0AHMAPQA8AC8AZAA%2BADwALwByAD4A&_host_Info=Word|Win32|16.00|en-US). They are a two layered ANN with generative capabilities. Two of these layers are connected by a fully bipartite graph. Well what I basically mean to say it every node of a particular layer (say visible layer) is connected to every node of the other layer (say hidden layer). But no node is connected within the same layer(intra-layer). This makes the learning more efficient and faster.



(Image Source: By Qwertyus - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22717044)

Training at each level the model gets improved. After a RBM gets trained, the activities of its hidden units can be treated as data for training higher level RBM. Thus many units can be trained in this way making it effective than the conventional Boltzmann machine.

### Energy equation

The Boltzmann machine if often related with thermodynamic systems. The energy function is quite similar to that of the Hopfield Networks. The energy equation is given by



Where,

Wij is the connection between i and j

Si is the state

Θi is the bias

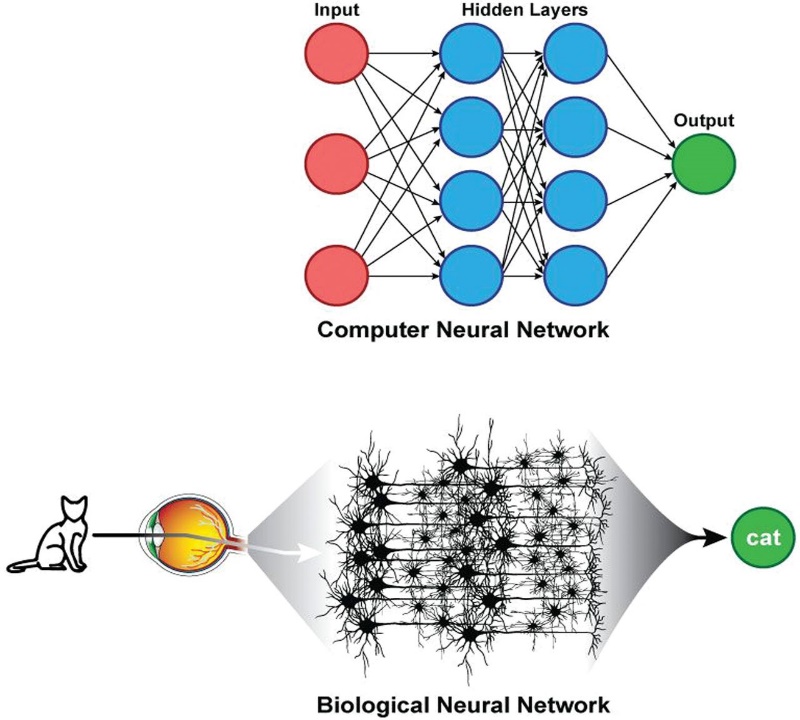
(Boltzmann machine, n.d.) (Nayak, 2018)

# Brain-State-in-a-Box Network:

**B**

**rain State in a box** is a pattern categorization device branched from the neurophysiological subject which was proposed by Anderson, Silverstein, Ritz and Jones in 1977. This model is quite intuitive as it opens to explain the effects and mechanisms often seen in cognitive science.

* Here, neurons can accept values between -1 and 1 and they are updated at the same time.
* Brain state in a box network is a fully connected network which depend on the dimension n of the input space.



BSBs are quite compared with the way cognitive science works

(Image Source: https://www.ajnr.org/content/ajnr/early/2018/02/01/ajnr.A5543/F2.large.jpg)

### Brain-State-in-a-Box Network Model:

Assume W to be a symmetrical weight matrix and it is positive semi-definite (which means xTWx >= 0 and assume that x(0) is the initial state vector.

The algorithm for this model is given by two equations given below

y(n) = x(n)+η W x(n)

x(n+1) = f(y(n))

Or we can say,

X <- f(x+η Wx)

Where η is called the feedback factor.

f(x) = +1     if x > 1;

f(x) = x     if  -1 < x < 1;

f(x) = -1     if x < -1.

Note that this system given binary output i.e. output in the form of 1 or -1 depending on the input and the weight W which has to be chosen with the required property. This output in the for -1 or 1 indicates the position of the state, whether it is in the corner of the box or is moving towards the wall.

### Energy function:

The energy equation for BSB is given as,

**E = -(ɳ/2) Kijwijxixj= -(ɳ/2) xT W x**

This energy function is called the *Lyapunov function.* The energy is proved to be decreasing but not increasing in any way but more general condition exists to show if this function really exists.

### Applications of BSB:

BSBs have the following applications:

* Large scale gBSB nets.
* Classification of radar signals emerging from the source of emitters.
* To store and retrieve images.

(www.physics.nus.edu.sg, n.d.) (The “Brain-State-in-a-Box” neural model is a gradient descent algorithm, 1986) (Brain State in a box, n.d.) (Brain-State-in-a-Box Network, n.d.) (htt)

# Optimization Using Hopfield Network:

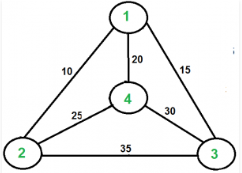
O

ptimization simply means to make the network as efficient as it can be through employing and experimenting with different algorithms and networks. The aim should be to reach the local minimum with the greatest efficiency. The solution of the problem is not exact but satisfactory to a large extent. Convergence depends on the coefficients weighting the cost function terms.

### Travelling Salesman Problem:

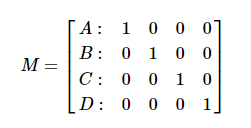
Most of us would have studied about Travelling Salesman Problem. Do you know that the Hopfield network can be used to optimize this problem? Let’s look at how this is done.

The basic concept of TSP can be imagined like this. Consider a given set of cities and a user is required to arrive at the specified city by following the shortest route possible. He must also ensure that he visits each city exactly once and returns to the starting point



(Image Source: <https://www.geeksforgeeks.org/wp-content/uploads/Euler12-300x225.png>)

Any four cities A,B,C,D can be represented in the form of a matrix in the following manner.



### Optimising the Travelling Salesman Problem using Hopfield Networks:

As we know, that the Hopfield Network is a single layer feedback network which is fully interconnected, the dynamics of each neuron is given by,



Where , is called the time constant,

T is the connection matrix

Ii is the bias

By the above equation HNN can be said to have the energy equation as follows,



Also,



We can clearly see from the above equations that the energy function decreases as the network state evolves and finally reaches the local minimum.

(Feng & Douligeris, 2000) (htt1) (htt2) (Hopfield neural networks for optimization: study of the different dynamics) (Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming), n.d.)

# Other Optimization Techniques:

Let’s look some of the other optimizations techniques normally found in the field of Artificial Neural Network.

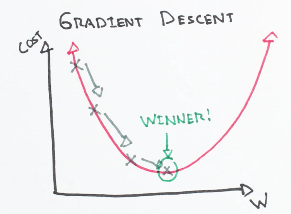
## Gradient Descent:

Well, we know that optimisation means to arrive somewhere in the local minimum of our function. We can also say that it means to minimize the cost function J. Optimisation is usually used to find the global minimum, but we are interested in finding the local minimum in this case.

Imagine a 2 dimensional space with peaks and a valley. We have a weight w with us which we want to reduce iteratively so that it reaches the local minimum.

Following steps are to be followed to bring this weight to a local minimum position

* Starting at the top of the peak, we take a downhill step specified by the negative gradient.
* We continue this process iteratively till reaches the local minimum point.
* Hence, the algorithm will eventually converge.



If we are to find the minimum of a function f(x), then we need to follow the following steps.

* We need to find the initial value x0.
* We need to find the gradient J which will eventually give us the slope.
* Next, we need to update the value of x

Xn+1 = xn – ΘJ(xn).

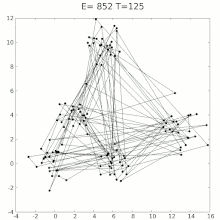
A point to note is that we must not choose a substantially larger or smaller step as it may diverge instead of converging or make take a lot of time to reach the local minimum.

(Other Optimization Techniques, n.d.) (coursera, n.d.) (Gradient Descent, n.d.) (Gradient Descent Algorithm and Its Variants, n.d.)

## Simulated Annealing

"Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem."

Source: Simulated annealing - [https://en.wikipedia.org](https://en.wikipedia.org/wiki/special:search/Simulated%20annealing)



(Image source: *Source: File:Travelling salesman problem solved with simulated annealing.gif -* [*https://en.wikipedia.org*](https://commons.wikimedia.org/wiki/File:Travelling_salesman_problem_solved_with_simulated_annealing.gif))

### Algorithm:

Steps:

1. We initialize the parameter at the beginning
2. We then find the associated cost function.
3. We then generate a random neighbouring solution.
4. The new solution is calculated from the old one.
5. Compare the cost functions.
6. We need to stop at a particular instance. Therefore keep in mind the stopping condition.

(ComprehensiveSimulatedAnnealing, n.d.) (simulated-annealing-algorithm, n.d.) (geeksforgeeks, n.d.) (SimulatedAnnealing, n.d.)

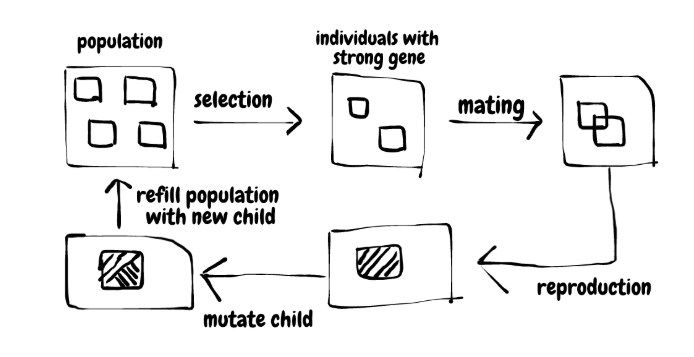
# Genetic Algorithm:

In [computer science](https://en.wikipedia.org/wiki/Computer_science) and [operations research](https://en.wikipedia.org/wiki/Operations_research), a **genetic algorithm** (**GA**) is a [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic) inspired by the process of [natural selection](https://en.wikipedia.org/wiki/Natural_selection) that belongs to the larger class of [evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm) (EA). –

Wikipedia

 Genetic algorithms, [which](https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems) in 1960 by John Holland, is an extension to Alan Turing’s concept of a “[learning machine](https://www.csee.umbc.edu/courses/471/papers/turing.pdf)” . It is an example of technology imitating nature to solve problems.

What makes Genetic Algorithm so famous is its ability in learning hyper parameters. Let’s look how we can solve the problem of hyper parameters using Genetic Algorithm.



A step by step picture of Genetic Algorithm

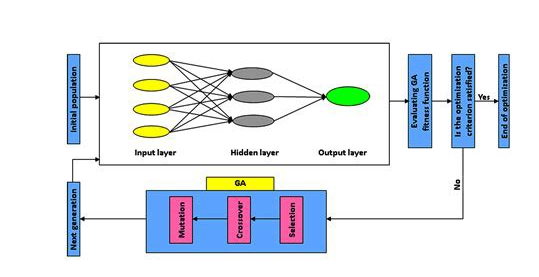
(Image Source: [https://miro.medium.com/max/1400/1\*5dvZcjfKVLWJK47SfhNmGQ.png](https://miro.medium.com/max/1400/1*5dvZcjfKVLWJK47SfhNmGQ.png))

### Algorithm:

The following steps are employed to solve the hyper-parameter problem.

1. A population of several neural networks needs to be created on the onset
2. Random hyper-parameters should be assigned to all these neural networks.
3. Train the neural networks one by one.
4. Calculate the training cost
5. Keep track and calculate the fitness of the neural network. Fitness is the measure of how it fared in that iteration.
6. The maximum fitness should hence be determined
7. Pick up two neural networks based on probability system
8. Crossover the genes of the 2 neural networks. This will create a child neural network which has some properties of first neural network.
9. Mutate the genes of the child
10. Repeat the last three steps for the number of neural networks.

At the end the model would be able to find the neural network with the optimum hyper parameters.



(Image Source: https://miro.medium.com/max/948/0\*eydAywp8fITbboo0.)

### Advantages of Genetic Algorithm:

Genetic algorithm finds a great deal of applications in various fields. Let us see some of it advantages.

* Here the number of parameters is very large.
* Unlike some of the other algorithms there can be multiple local optima here
* The optimisation is good for noisy environments .
* Answer to a question is always available, which gets better over time.
* It is inherently parallel.

### Drawbacks of Genetic Algorithm:

* The major setback of GA is that there is no guarantee of finding a global minimum
* Another drawback is that with this algorithm one needs to have a substantially large population and hence it may take a lot of time to fetch the output.
* Random convergence of solutions is a big problem with respect to the fitness function.
* It’s unguided mutation is also a major setback.
* This may computationally be expensive given the repeated calculation of the fitness function.

(Artificial Neural Network - Genetic Algorithm, n.d.) (S., 2018) (benefits-of-using-genetic-algorithm, n.d.) (a-gentle-introduction-to-genetic-algorithm, n.d.) (Introduction to Genetic Algorithms — Including Example Code, n.d.) (Genetic\_algorithm, n.d.)

# Applications of Neural Network:

A

rtificial Neural Network finds a wide range of application across various fields and domains because of its ability to do classification, clustering, regression, object detection, anomaly detection, pattern recognition, visualisation and a lot more. Artificial Neural Network is going to be a part and parcel of our day to day lives in some or the other way. As I am writing this article the pandemic of Corona-virus has already brought the world to an utter standstill. This is the time that AI has come to the fore and has helped solved many problems right from helping in tracking the people having the virus to automatically monitoring several cameras and sending the alerts on detecting people with having a temperature. Let’s look at some of the interesting applications of ANN.

## Image Processing:

A lot of image processing can be done with the help of ANNs. Just take for example my recent project on Sign Language Detection Using CNN. The model when fed with a particular hand gesture through the camera or an image in the database can detect the character corresponding to the gesture. Be it facial recognition to identifying type of cells, and identifying with various types of cells in our body, ANN has had it footfall in almost all domains. More advanced machines are in the making like self-driving cars which primarily have a complex image processing mechanism.



Self-driving cars are going to be the norm in the future

## Text Classification and Recognition:

Text Classification is employed in various applications like web searching, information filtering, language identification etc.

Take for example the model being developed by one of my professors in Army Institute of Technology which aims to detect and convert the sign boards in Indian languages into the corresponding text. While this may pose a huge challenge given the uneven and often blurred or tainted sign boards in India, but nonetheless as the model learns new texts, this project is tend to become a success. There are several numerous example. Sentiment analysis on twitter data set or the Amazon review data set also is a part of text classification and categorization.



## Forecasting:

Do you even the people with the most experience in the stock market (say for example my grandpa) are in a dilemma as to how some of the people new to this field are making more gains. Well, turns out to be simple they are more reliant on the forecasting of stock prices by one of the many predicting machines for the stock prices. And it will be no brainer that ANN will rule the market in the days to come. Well one of the significant advantages of ANN in predicting stock prices is that it can identify any hidden relationships and features which is not normally identifiable by even an expert in this field.



Stock market prediction and analysis will be the driving seat for the future businesses

## Signature Detection and Verification:

One of the greatest issues with technology today Is the various aspects of safety being compromised with it. This is where ANN comes to the rescue with its latest advancements. One such advancement is signature verification probably using LSTM and CNN. This can be used to authenticate the document and identify if the signature is forged or not. This will go a long way in solving the problem of forging in banks, hospitals and even in business firms.

There are many more such applications that has come up in the field of ANN. I have just covered a few of them. You can find out some of the cool applications by surfing on the net and probably start working on the area of your interest.

(Mahanta, 2017; Applications of artificial neural-networks for energy systems, 2000; Gill, n.d.; Łątkowski, 2020; Applications of Neural Networks, n.d.; 10 Applications of Artificial Neural Networks in Natural Language Processing, n.d.)

# Artificial Neural Network Resources:

One of the sites which I can say as my goto site if I need to understand any concept in ANN would be Medium(<https://medium.com/>). The concepts are beautifully explained by some of the most experienced ones in this field.

Some of the other resources would be:

1. <https://towardsdatascience.com/>
2. <https://www.sciencedirect.com/>
3. <https://www.wikipedia.org/>
4. [Learn Python the Hard Way (Online Book)](https://learnpythonthehardway.org/book/)
5. [Statistics and Probability (Khan Academy)](https://www.khanacademy.org/math/statistics-probability)
6. [Data Visualization in Python (Video Series)](https://www.youtube.com/watch?v=q7Bo_J8x_dw&list=PLQVvvaa0QuDfefDfXb9Yf0la1fPDKluPF)
7. [Modern Machine Learning Algorithms: Strengths and Weaknesses](http://elitedatascience.com/machine-learning-algorithms)
8. [Machine Intelligence and Data Products (Video)](https://www.youtube.com/watch?v=SxxqaC5hf04)
9. Machine learning by Andrew Ng

Well, that’s all folks, if you have any doubts please don’t hesitate to mail me at [benjamin10051999@gmai.com](mailto:benjamin10051999@gmai.com). Till then, good bye and good luck with your journey with Artificial Networks.

# Artificial Neural Network Interview Questions answer:

### **Differentiate supervised and unsupervised deep learning procedures.**

* Supervised learning works on well-defined data whereas unsupervised learning works on unlabelled data
* In supervised learning the number of classes are known which is not the case with unsupervised learning

### **What are the applications of deep learning?**

There are several applications of deep learning. Some of them are:

* Sentiment analysis
* Image Processing
* Forecasting
* Object Classification and Detection
* Text Classification
* Character and Signature Recognition
* Computer vision
* Natural language processing and pattern recognition

1. **How Artificial Neurons Learns?**

Two **things can be taken into consideration,**

* **Associative mapping: The output pattern of the network is obtained by working in a pattern on the input data.**
* **Regularity Detection:** **In this, units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular ‘meaning’. This type of learning mechanism is essential for feature discovery and knowledge representation.**

1. **What Are The Disadvantages Of Artificial Neural Networks?**

**They require sometimes a huge training data set which are not very sure to give us accurate results.**

### **What do you mean by "overfitting"?**

When the data in the model is trained a lot, this problem of overfitting occurs. This means that the noise or random fluctuations are also learned by the model. Hence when presented with a new data, it may not be able to predict the correct output although it may claim to have 99 or 100 percent accuracy.

*Source: File:Overfitting.svg -* [*https://en.wikipedia.org*](https://commons.wikimedia.org/wiki/File:Overfitting.svg)

### **What are the deep learning frameworks or tools?**

Some of the deep learning frameworks and tools are,

Tensorflow, Keras, Chainer, Pytorch, Theano & Ecosystem, Caffe2, CNTK, DyNetGensim, DSSTNE, Gluon, Paddle, Mxnet, BigDL

1. **Define precision and recall.**

**Recall is known as the number true positive rates. It is the number of positives the model claims to the actual number of positives in the data. Precision is a measure of the amount of accurate positives the model predicts.**

1. **Why is “Naive” Bayes naive?**

**The ability of the Naïve Bayes algorithm to make assumptions which is actually impossible to see in real life earned its name as “naïve”. The conditional probability is calculated as the pure product of the individual probabilities of components. This implies the absolute independence of features — a condition probably never met in real life.**

### **What are the supervised learning algorithms in Deep learning?**

The supervised learning algorithms in deep learning are:

1. Artificial Neural Network
2. Convolution Neural Network
3. Recurrent Neural Network

### **What are the unsupervised learning algorithms in Deep learning?**

The supervised learning algorithms in deep learning are:

1. Self Organizing Maps
2. Deep belief networks (Boltzmann Machine)
3. Auto Encoders
4. **How many layers are found in the neural network?**

**The layers in the neural network can be classified as,**

* **Input Layer**
* **Hidden Layer**
* **Output Layer**

### **What is the use of the Activation function?**

This function is used to introduce non-linearity into the neural network so that it would be able to learn more and improve its accuracy. This coverts the inputs into the outputs and decides on which neurons to fire to obtain the most accurate predictions and outputs.

*Source: File:Logistic-curve.svg -* [*https://en.wikipedia.org*](https://commons.wikimedia.org/wiki/File:Logistic-curve.svg)

1. **What is binary step function?**

This is a step function which is based on the threshold. It decides whether to activate a neuron or not based on whether the value calculated by the function is above or below a certain threshold value.

1. **What is softmax function?**

**Softmax is a really useful function, which takes up the inputs and converts the outputs into probabilities that sum up to one. It outputs a vector that represents a probability distributions of a list of potential outcomes.**

### **What are the main benefits of Mini-batch Gradient Descent?**

The main benefit of mini-batch gradient descent is that it does not select the input data one by one but rather in a batch (group of data). This saves a lot of time and improves accuracy. The gradient of the entire training set can be approximated, which helps to avoid the local minimum.

# References

(n.d.). Retrieved from Tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_associate\_memory.htm

(n.d.). Retrieved from www.physics.nus.edu.sg: https://www.physics.nus.edu.sg/~phywjs/CZ3205/Notes9\_2.htm

(n.d.). Retrieved from https://www.bonaccorso.eu/2017/09/22/hopfield-networks-addendum-brain-state-box-model/

(n.d.). Retrieved from https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_optimization\_using\_hopfield.htm

(n.d.). Retrieved from https://www.ece.uic.edu/~rgandhi/resume/Hope\_tsp.PDF

(n.d.). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/simulated-annealing/

*10 Applications of Artificial Neural Networks in Natural Language Processing*. (n.d.). Retrieved from medium: https://medium.com/@datamonsters/artificial-neural-networks-in-natural-language-processing-bcf62aa9151a

*a-gentle-introduction-to-genetic-algorithm*. (n.d.). Retrieved from medium: https://medium.com/the-z/a-gentle-introduction-to-genetic-algorithm-30695a6084c2

*ANN*. (n.d.). Retrieved from Java T Point: https://www.javatpoint.com/artificial-neural-network-adaptive-resonance-theory

*Applications of artificial neural-networks for energy systems*. (2000, September). Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/abs/pii/S0306261900000052

*Applications of Neural Networks*. (n.d.). Retrieved from tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_applications.htm

*art-adaptive-resonance-theor*. (n.d.). Retrieved from oureducation: https://blog.oureducation.in/art-adaptive-resonance-theory/

*Artificial Neural Network - Building Blocks*. (n.d.). Retrieved from tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_building\_blocks.htm

*Artificial Neural Network - Genetic Algorithm*. (n.d.). Retrieved from tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_genetic\_algorithm.htm

*benefits-of-using-genetic-algorithm*. (n.d.). Retrieved from stackexchange: https://stats.stackexchange.com/questions/23106/benefits-of-using-genetic-algorithm

*Boltzmann machine*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Boltzmann\_machine

*Boltzmann Machines*. (n.d.). Retrieved from javatpoint: https://www.javatpoint.com/boltzmann-machines

Bo-Suk Yang, T. H.-S. (2003). Integration of ART-Kohonen neural network and case-based. 9.

*Brain State in a box*. (n.d.). Retrieved from everybodywiki: https://en.everybodywiki.com/Brain\_State\_in\_a\_box

*Brain-State-in-a-Box Network*. (n.d.). Retrieved from tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_brainstateinabox.htm

*Building Blocks*. (n.d.). Retrieved from javatpoint: https://www.javatpoint.com/artificial-neural-network-building-blocks

Coinnigh, E. Ó. (2018). *quora*. Retrieved from quora: quora.com/What-is-the-stability-plasticity-dilemma-in-neural-networks-in-simple-words

*ComprehensiveSimulatedAnnealing*. (n.d.). Retrieved from http://mat.uab.cat/~alseda/MasterOpt/ComprehensiveSimulatedAnnealing.pdf

*coursera*. (n.d.). Retrieved from Gradient Descent in Practice II - Learning Rate: https://www.coursera.org/learn/machine-learning/supplement/TnHvV/gradient-descent-in-practice-ii-learning-rate

Culurciello, E. (2017). *Neural Networks Building Blocks*. Retrieved from medium: https://medium.com/breathe-publication/neural-networks-building-blocks-a5c47bcd7c8d

Eklavya. (2019, October 29). *Kohonen Self-Organizing Maps*. Retrieved from Towards Data Science: https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da

Eklavya. (n.d.). *EMNIST-Kohonen-SOM*. Retrieved from github: https://github.com/EklavyaFCB/EMNIST-Kohonen-SOM

Feng, G., & Douligeris, C. (2000). Using Hopfield networks to solve traveling salesman problems based on stable state analysis technique. *IEEE-INNS-ENNS International Joint Conference.* IEEE Xplore. Retrieved from Using Hopfield networks to solve traveling salesman problems based on stable state analysis technique

Fumo, D. (2017). *A Gentle Introduction To Neural Networks Series — Part 1*. Retrieved from towardsdatascience: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

Fumo, D. (2017). *A Gentle Introduction To Neural Networks Series — Part 1*. Retrieved from towardsdatascience: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

G.A. Carpenter, S. G. (2002). Fuzzy ART: an adaptive resonance algorithm for rapid, stable classification of analog patterns. *IJCNN-91-Seattle International Joint Conference on Neural Networks.* Seattle, WA, USA, USA: IEEE.

*Genetic\_algorithm*. (n.d.). Retrieved from wikipedia: https://en.wikipedia.org/wiki/Genetic\_algorithm

Gill, N. S. (n.d.). *Artificial Neural Networks Applications and Algorithms*. Retrieved from xenonstack: xenonstack.com/blog/artificial-neural-network-applications/

*Gradient Descent*. (n.d.). Retrieved from https://ml-cheatsheet.readthedocs.io/: https://ml-cheatsheet.readthedocs.io/en/latest/gradient\_descent.html

*Gradient Descent Algorithm and Its Variants*. (n.d.). Retrieved from towardsdatascience: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Grossberg, D. S. (2013). *Adaptive resonance theory*. Retrieved from Scholarpedia: http://www.scholarpedia.org/article/Adaptive\_resonance\_theory

*Hopfield network*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Hopfield\_network#Updating

*Hopfield Network*. (n.d.). Retrieved from javatpoint: https://www.javatpoint.com/artificial-neural-network-hopfield-network

*Hopfield Networks*. (n.d.). Retrieved from http://www.doc.ic.ac.uk/: http://www.doc.ic.ac.uk/~ae/papers/Hopfield-networks-15.pdf

*Hopfield neural networks for optimization: study of the different dynamics.* (n.d.). Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/abs/pii/S092523120100337X

*Introduction to Genetic Algorithms — Including Example Code*. (n.d.). Retrieved from towardsdatascience: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

IpsitaDalai. (n.d.). *adaptive-resonance-theory-art*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/adaptive-resonance-theory-art/

*Kohonen Self-Organizing Feature Maps*. (n.d.). Retrieved from Tutorials point: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_kohonen\_selforganizing\_feature\_maps.htm

Łątkowski, K. (2020). *Artificial Neural Networks – How it is Used in the Fight Against the COVID-19 Pandemic*. Retrieved from stayinfront: https://www.stayinfront.com/blog/artificial-neural-networks-how-it-is-used-in-the-fight-against-the-covid-19-epidemic/

*Learning Vector Quantization (LVQ) Neural Networks*. (n.d.). Retrieved from MathWorks: https://in.mathworks.com/help/deeplearning/ug/learning-vector-quantization-lvq-neural-networks-1.html;jsessionid=d82a04440c05dda6e7df85c563fb

Mahanta, J. (2017, June 10). *Introduction to Neural Networks, Advantages and Applications*. Retrieved from towardsdatascience: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207

Maini, V. (2019). Machine Learning for Humans. *Machine Learning for Humans, Part 2.1: Supervised Learning*.

Nagyfi, R. (2018, September 04). *The differences between Artificial and Biological Neural Networks*. Retrieved from towardsdatascience: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Nayak, M. (2018, 11 19). *An Intuitive Introduction Of Boltzmann Machine*. Retrieved from Medium: https://medium.com/datadriveninvestor/an-intuitive-introduction-of-boltzmann-machine-8ec54980d789

*Neural Network Definition*. (n.d.). Retrieved from https://pathmind.com/wiki/neural-network: https://pathmind.com/wiki/neural-network

*Other Optimization Techniques*. (n.d.). Retrieved from tutorialspoint: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_other\_optimization\_techniques.htm

Ruiz, P. (2018). *Neural Networks I: Notation and building blocks*. Retrieved from towardsdatascience: https://towardsdatascience.com/neural-networks-i-notation-and-building-blocks-817b1d2ea04b

S., S. (2018, March 26). *Genetic Algorithms + Neural Networks = Best of Both Worlds*. Retrieved from towardsdatascience: https://towardsdatascience.com/gas-and-nns-6a41f1e8146d

*SimulatedAnnealing*. (n.d.). Retrieved from mathworld: https://mathworld.wolfram.com/SimulatedAnnealing.html

*simulated-annealing-algorithm*. (n.d.). Retrieved from sciencedirect: https://www.sciencedirect.com/topics/engineering/simulated-annealing-algorithm

Smith, C. P. (2015). *FUZZY ADAPTIVE RESONANCE THEORY:.*

*Stochastic Gradient Descent — Clearly Explained*. (2019, 09 07). Retrieved from towardsdatascience: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

*The “Brain-State-in-a-Box” neural model is a gradient descent algorithm*. (1986). Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/abs/pii/002224968690043X

*Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)*. (n.d.). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

*Understanding The Essential Blocks of Artificial Intelligence - Neural Networks*. (n.d.). Retrieved from zeolearn: https://www.zeolearn.com/magazine/understanding-the-essential-blocks-of-artificial-intelligence-neural-networks

Wasserman, P. D. (1989). *Adaptive Resonance Theory*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Adaptive\_resonance\_theory

*What is the difference between biological and artificial neural networks?* (n.d.). Retrieved from stackexchange: https://psychology.stackexchange.com/questions/7880/what-is-the-difference-between-biological-and-artificial-neural-networks