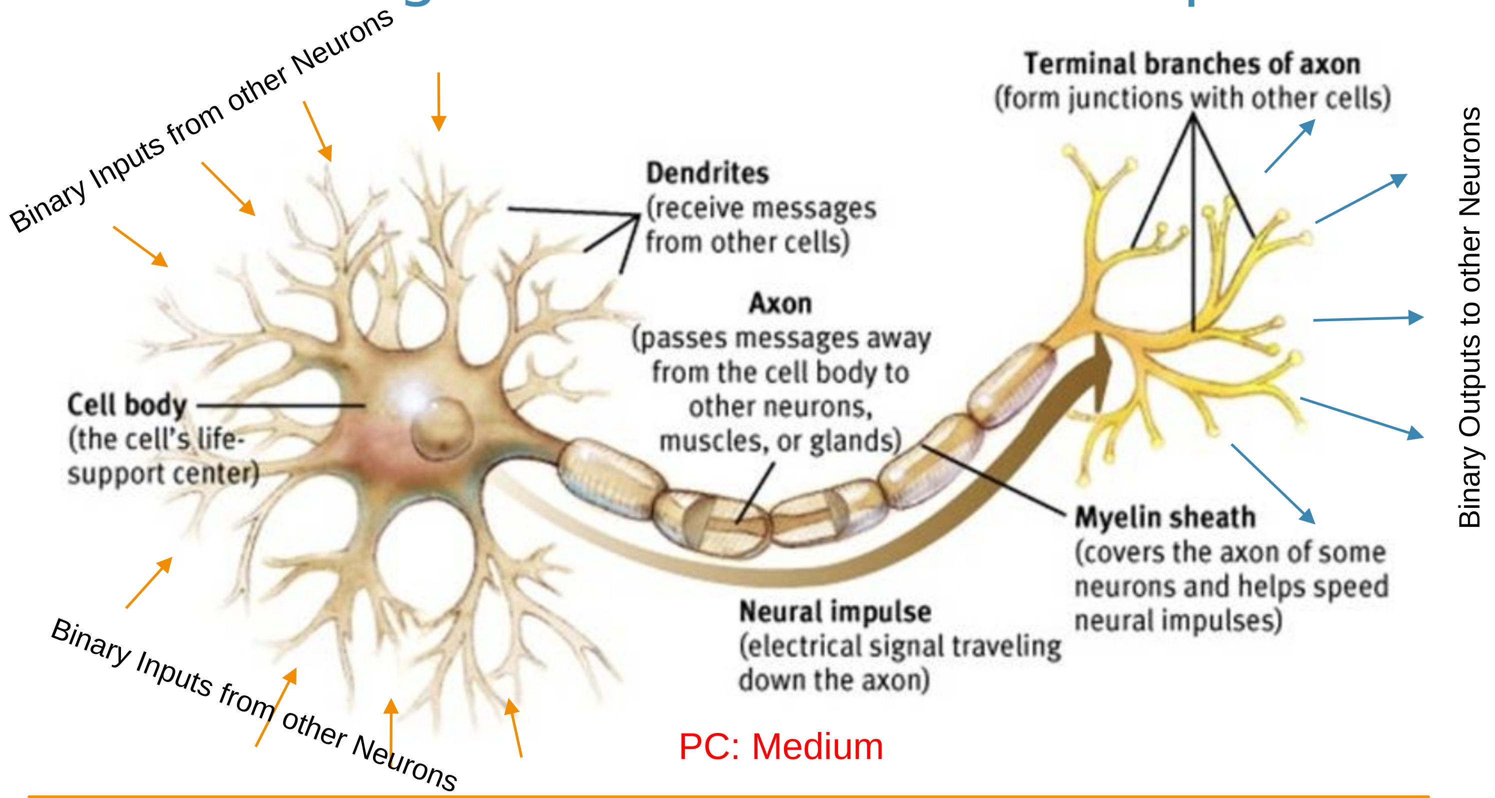
ANN & Bioinformatics Applications

What is Perceptron?

- **The term perceptron, in ML, refers to an algorithm for supervised learning of binary classifiers**
- **Perceptron is linear classifier**
  - **It makes its predictions based on a linear predictor function combining a set of weights with the feature vector**
- **Perceptron is binary classifier**
  - **There should be only 2 categories for classification**
- **A neural network is an interconnected system of perceptrons**
  - **Perceptrons are the foundation of any neural network**

The biological Motivation of Perceptron


Binary Inputs from other Neurons
Terminal branches of axon
(form junctions with other cells)
Dendrites
(receive messages from other cells)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Cell body
(the cell's life-support center)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)
Neural impulse
(electrical signal traveling down the axon)
Binary Outputs to other Neurons
Binary Inputs from other Neurons

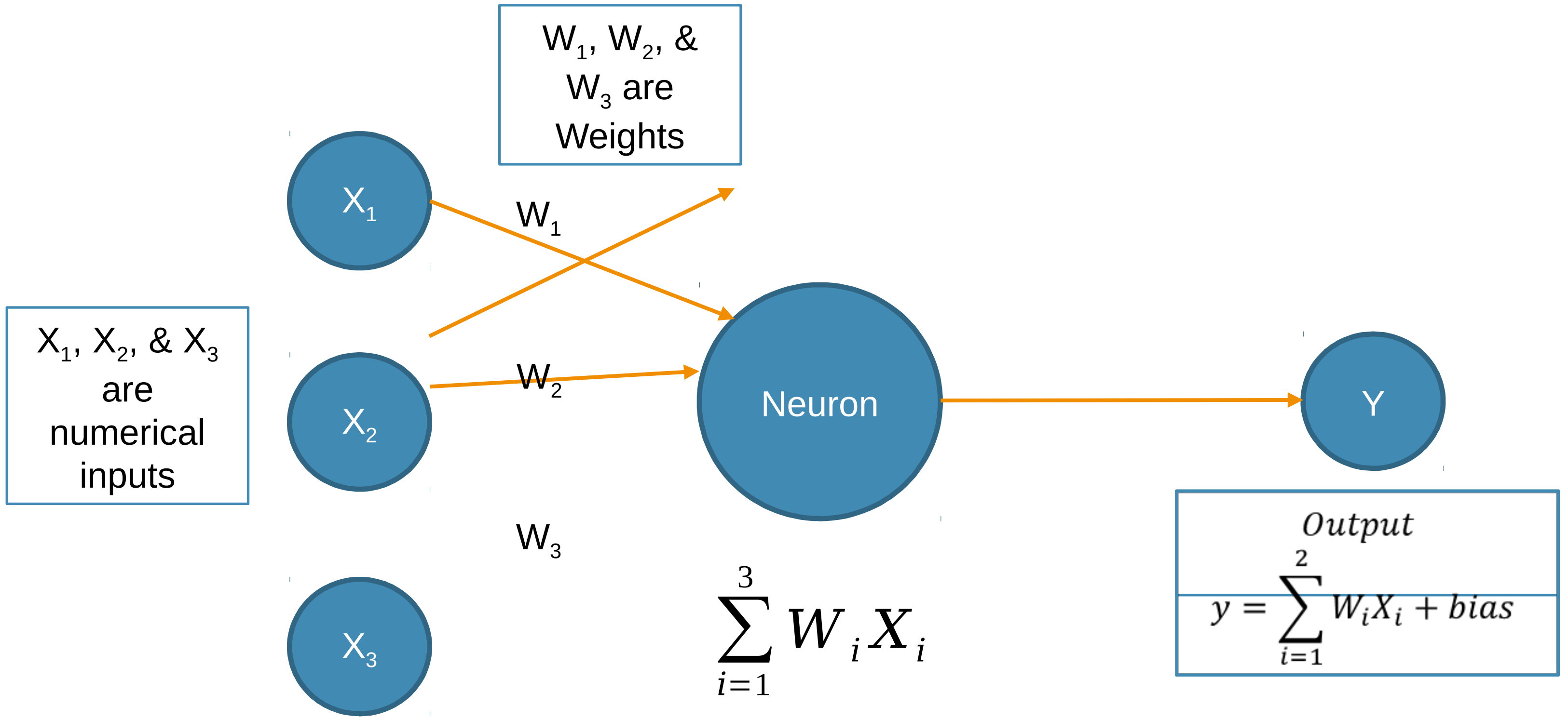
PC: Medium

In the biological neuron, the point of this cell is to take in some input (in the form of electrical signals in our brains), do some processing, and produce some output (also an electrical signal).

Human Neurons – A Quick Overview

- **An individual neuron accepts inputs, usually from other neurons, through its dendrites**
- **The dendrites connect with other neurons through the synapse — that assigns a weight to a particular input**
- **Then, all of these inputs are considered together when they are processed in the cell body**
- **All-or-none behavior of neurons**
  - **If the combination of inputs exceeds a certain threshold, then an output signal is produced**
    - **In this case, output travels to other neurons along the axon to the axon terminals**
  - **If the combination falls short of the threshold, then the neuron doesn’t produce any output**

How does Perceptron mimic human neurons?



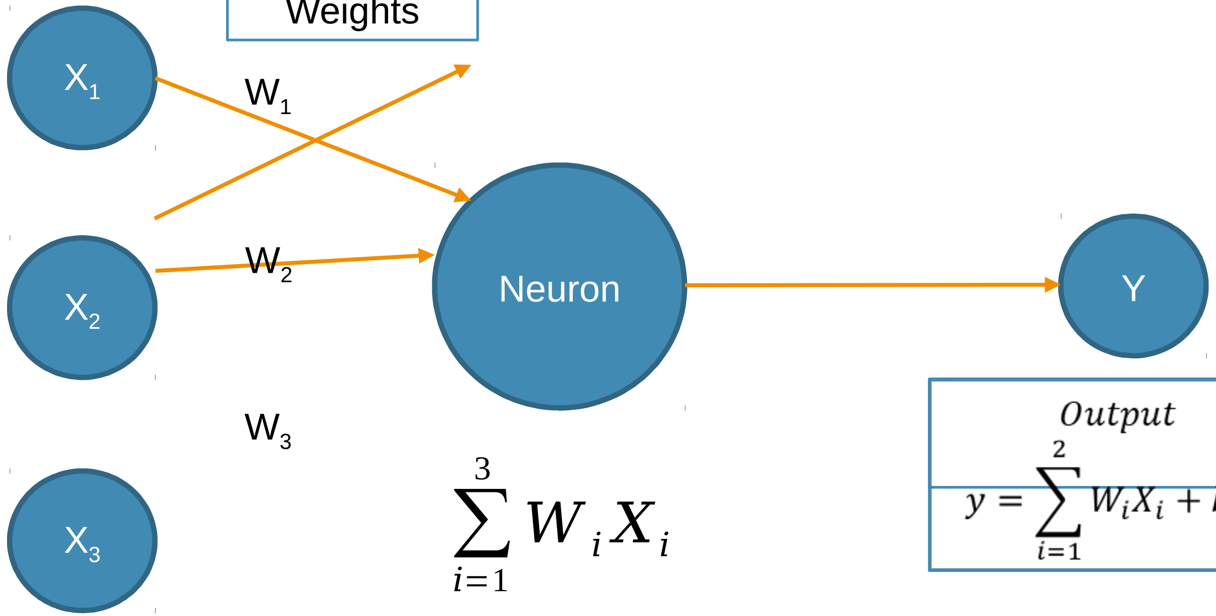
W_1, W_2, & W_3 are Weights
X_1, X_2, & X_3 are numerical inputs
X_1
X_2
X_3
W_1
W_2
W_3
Neuron
Y

$$\sum_{i=1}^{3} W_i X_i$$

Output

$$y = \sum_{i=1}^{2} W_i X_i + bias$$

How does Perceptron mimic human neurons?



W_1, W_2, & W_3 are Weights

X_1, X_2, & X_3 are numerical inputs

$$\sum_{i=1}^{3} W_i X_i$$

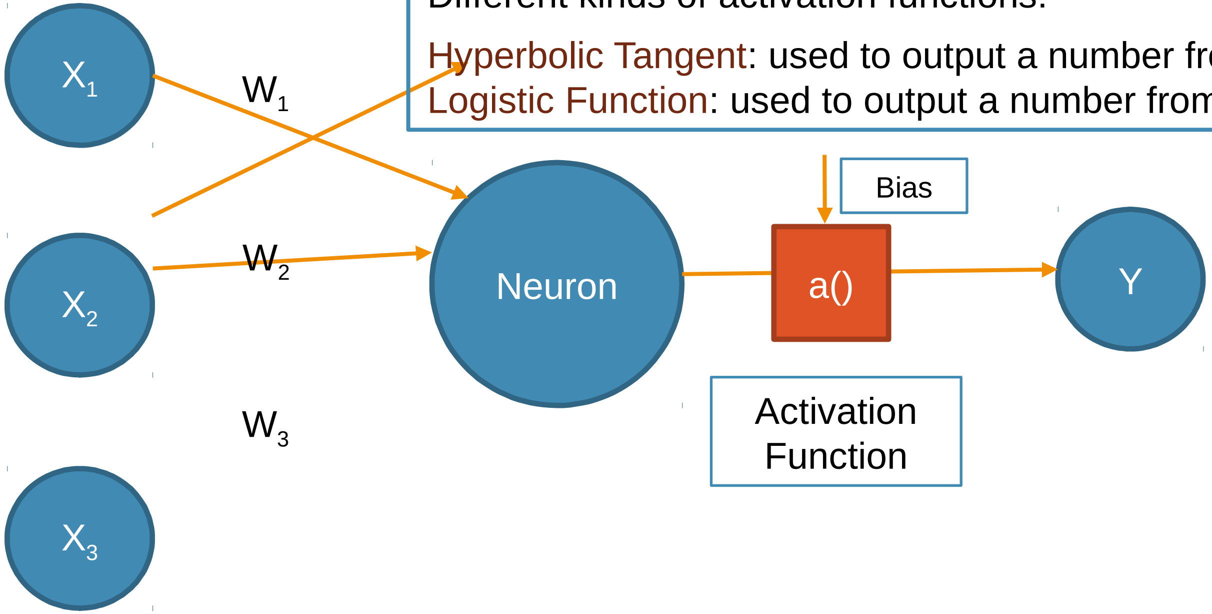
$Output$

$$y = \sum_{i=1}^{2} W_i X_i + bias$$

What if we want the Perceptron’s output to be within a specific range i.e. 0 & 1?

Different kinds of activation functions:

Hyperbolic Tangent: used to output a number from -1 to 1.
Logistic Function: used to output a number from 0 to 1.



An activation function is a function that converts the input given (the input, in this case, would be the weighted sum and the bias) into a certain output based on a set of rules.

Representation Power of Perceptron

- AND Gate

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

- OR Gate

A perceptron without activation function can only perform linear transformation on the inputs and hence it is less powerful…!

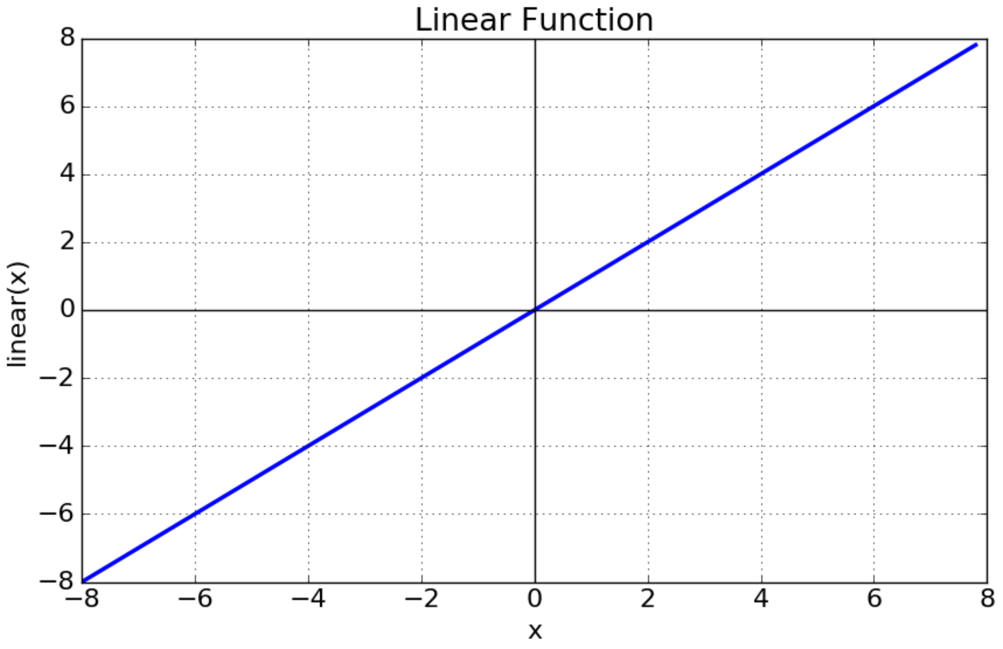
Activation Functions

- **The activation/transfer function defines the output of a perceptron given an input or set of inputs**
  - **It maps the resulting values in between 0 to 1 or -1 to 1 etc.**
- **The Activation Functions can be basically divided into 2 types-**
  1. **Linear Activation Function**
  2. **Non-linear Activation Functions**

Through activation function, the nonlinear separation of data points is also possible.

Linear Activation Function

- **As you can see the function is a line or linear**
- **Therefore, the output of the functions will not be confined between any range**



PC: Towardsdatascience

$Function: f(x) = x$

$Function: f(x) = x$

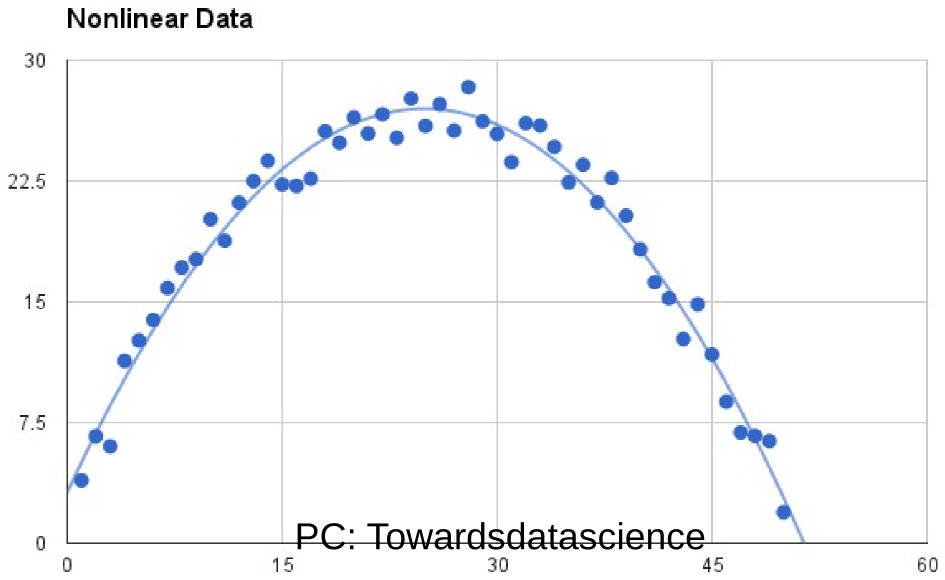
$Output\ Range: -\infty\ to + \infty$

$Output\ Range: -\infty\ to + \infty$

It doesn’t help in the presence of complex data that is usually fed to neural networks.

Non-linear Activation Function

- **The Nonlinear Activation Functions are the most used activation functions**
- **Nonlinearity helps to make the graph look something like below**



PC: Towardsdatascience

Terminologies:

1. Derivative or Differential:

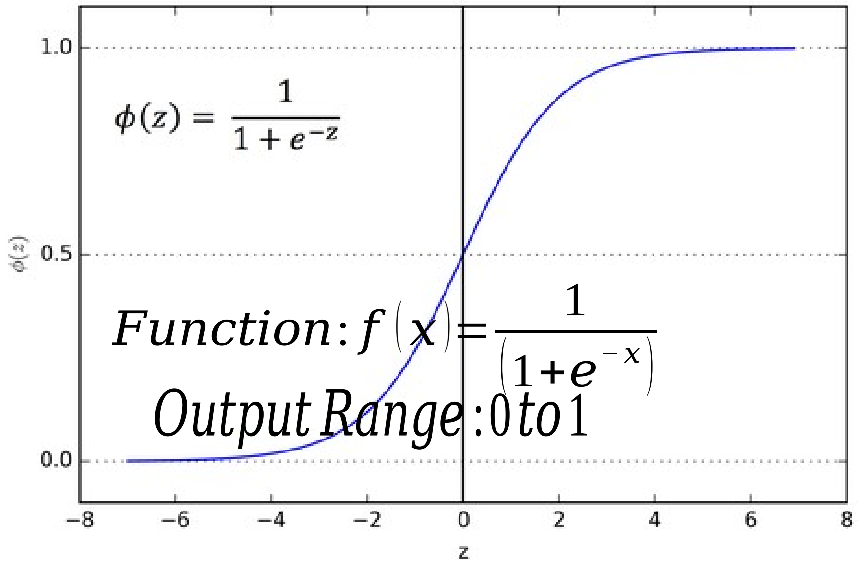
- Change in y-axis w.r.t. change in x-axis
- It is also known as slope

2. Monotonic function:

- A function which is either entirely non-increasing or non-decreasing

Sigmoid or Logistic Activation Function

- **The Sigmoid Function results in a S-shape curve as**



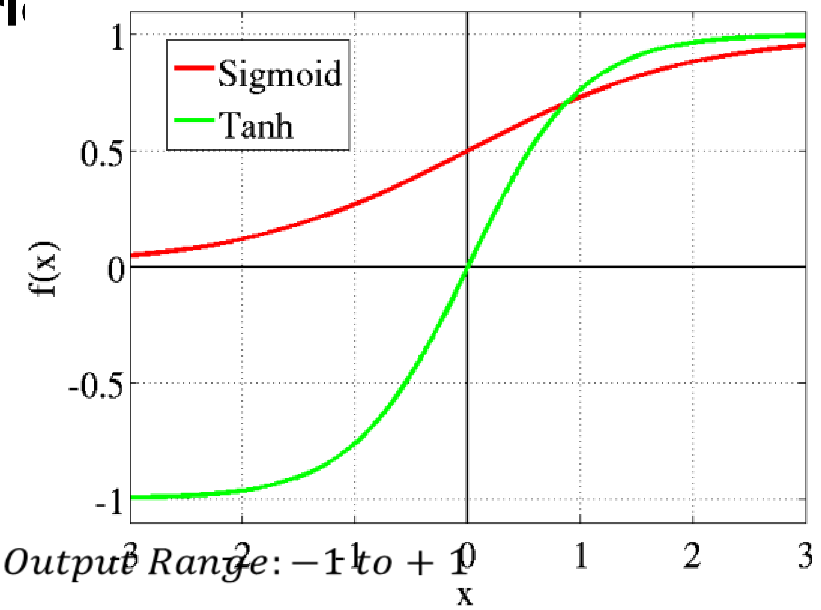
PC: Towardsdatascience

$$\boxed{\begin{array}{c} Function:\ f(x) = \dfrac{1}{(1+e^{-x})} \\ \\ Output\ Range: 0\ to\ 1 \end{array}}$$

- It is especially used for models where we have to predict the probability as an output (0 <= P <= 1)
- The function is differentiable meaning we can find the slope of the sigmoid curve at any two points
- The function is monotonic but not derivative
- The logistic sigmoid function can cause a neural network to get stuck at the training time
- Softmax function : a more generalized logistic activation function which is used for multiclass classification

Tanh or Hyperbolic tangent Activation Function

- **The tanh is also like logistic sigmoid except it is symmetric around the ori**
  - The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph
  - The function is differentiable
  - The function is monotonic while its derivative is not monotonic
  - The tanh function is mainly used classification between two classes



Output Range: -1 *to* $+1$

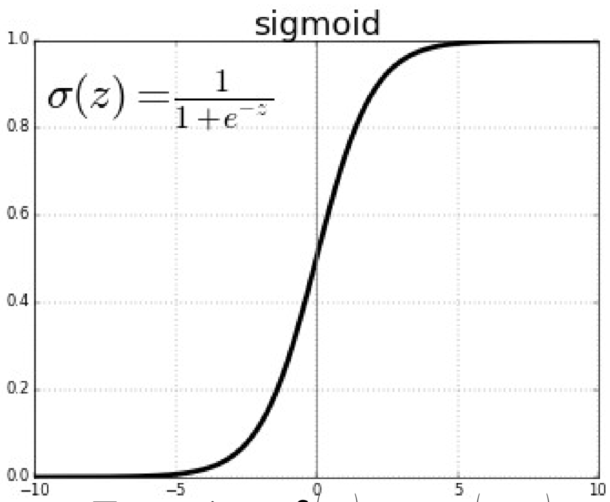
PC: Towardsdatascience

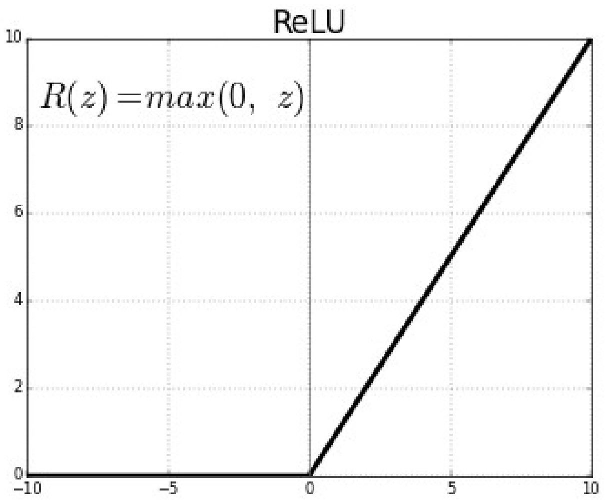
$$Function: tanh(x) = 2 * Sigmoid(2x) - 1$$

$$Output\ Range: -1\ to +1$$

ReLU (Rectified Linear Unit) Activation Function

- **The ReLU function is another non-linear activation function that has gained popularity in the deep le**





PC: Towardsdatascience

$Function: f(x) = max(0, x)$

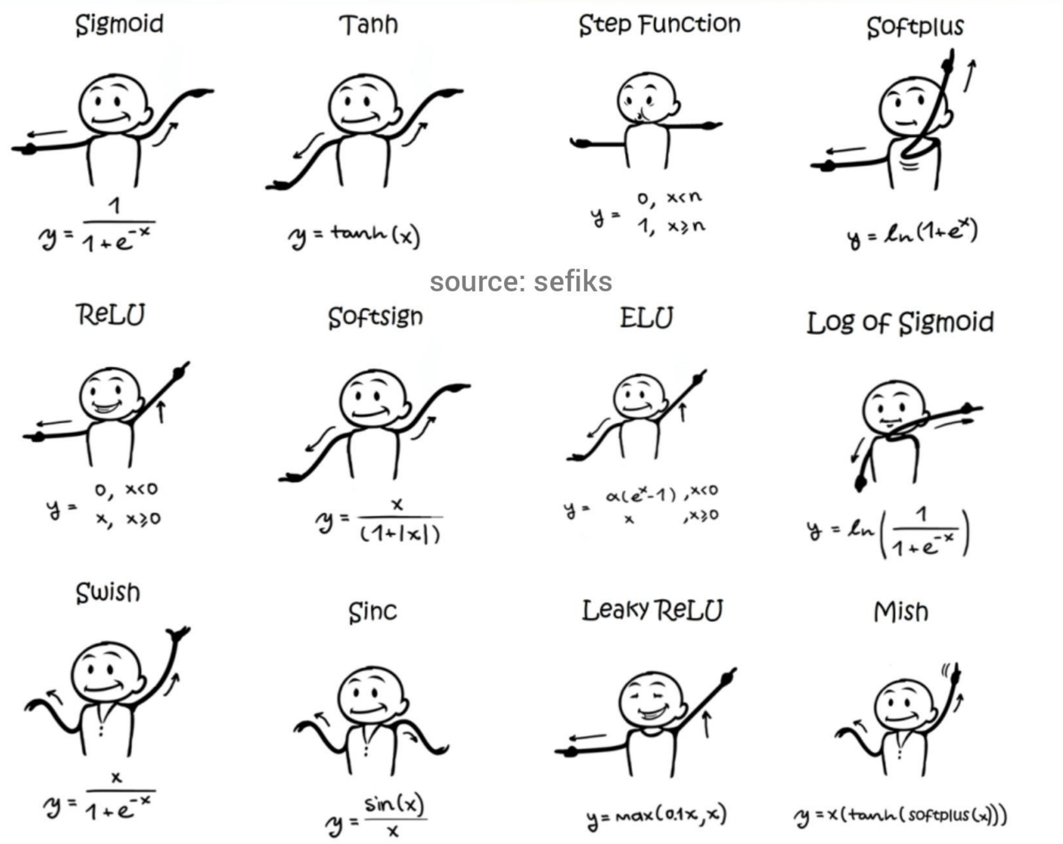
$Function: f(x) = \max(0, x)$

$Output\ Range: 0\ to\ \infty$

$Output\ Range: 0\ to\ \infty$

- As you can see, the ReLU is half rectified (from bottom) i.e., f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero
- The function and its derivative both are monotonic

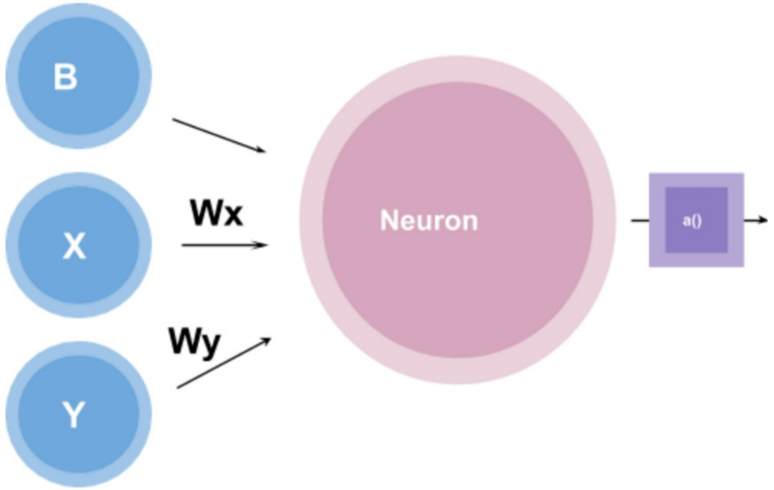
Dance Moves of Deep Learning Activation Functions


Sigmoid
$y = \frac{1}{1+e^{-x}}$
Tanh
$y = \tanh(x)$
Step Function
$y = \begin{cases} 0, & x<n \\ 1, & x\geq n \end{cases}$
Softplus
$y = \ln(1+e^{x})$
source: sefiks
ReLU
$y = \begin{cases} 0, & x<0 \\ x, & x\geq 0 \end{cases}$
Softsign
$y = \frac{x}{(1+|x|)}$
ELU
$y = \begin{cases} \alpha(e^{x}-1), & x<0 \\ x, & x\geq 0 \end{cases}$
Log of Sigmoid
$y = \ln\left(\frac{1}{1+e^{-x}}\right)$
Swish
$y = \frac{x}{1+e^{-x}}$
Sinc
$y = \frac{\sin(x)}{x}$
Leaky ReLU
$y = \max(0.1x, x)$
Mish
$y = x(\tanh(\text{softplus}(x)))$

PC: Reddit

Binary Classification using Perceptron

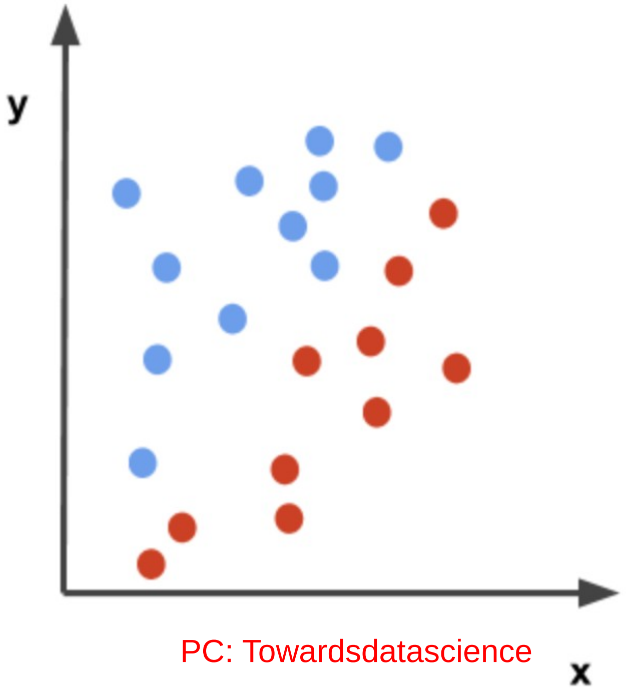
- **Let’s take a simple perceptron, given inputs x and y, which are multiplied with the weight w as w*x and w*y**
- **A bias is also given**



PC: Towardsdatascience

Binary Classification using Perceptron contd.

- **Our goal is to separate the data plotted below so that there is a distinction between the blue dots and the red dots**



PC: Towardsdatascience

- A perceptron can create a decision boundary for a binary classification as follows
- Lets say, wx = -0.5, wy = +0.5, and b = 0
- Then, the function for the perceptron will be

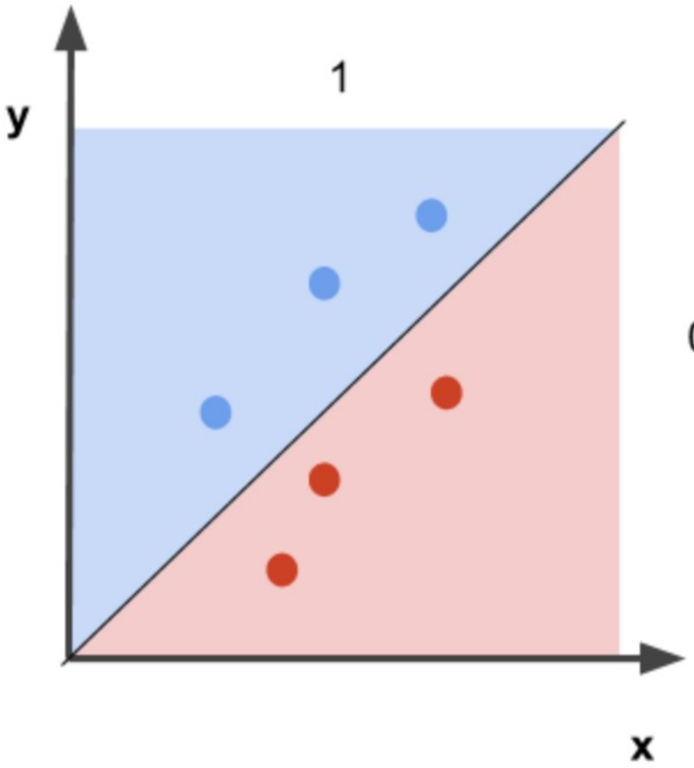
$$0.5x + 0.5y = 0$$

- Let's suppose that the activation function, in this case, is a simple step function that outputs either 0 or 1

Binary Classification using Perceptron contd.

- **The perceptron function will then label the blue dots as 1 and the red dots as**
- **In other words,**
  - **if $0.5x + 0.5y \Rightarrow 0$, the**
  - **if $0.5x + 0.5y < 0$, then**

Therefore, the function $0.5x + 0.5y = 0$ creates a decision boundary that separates the red and blue points



PC: Towardsdatascience

Perceptron Learning Algorithm

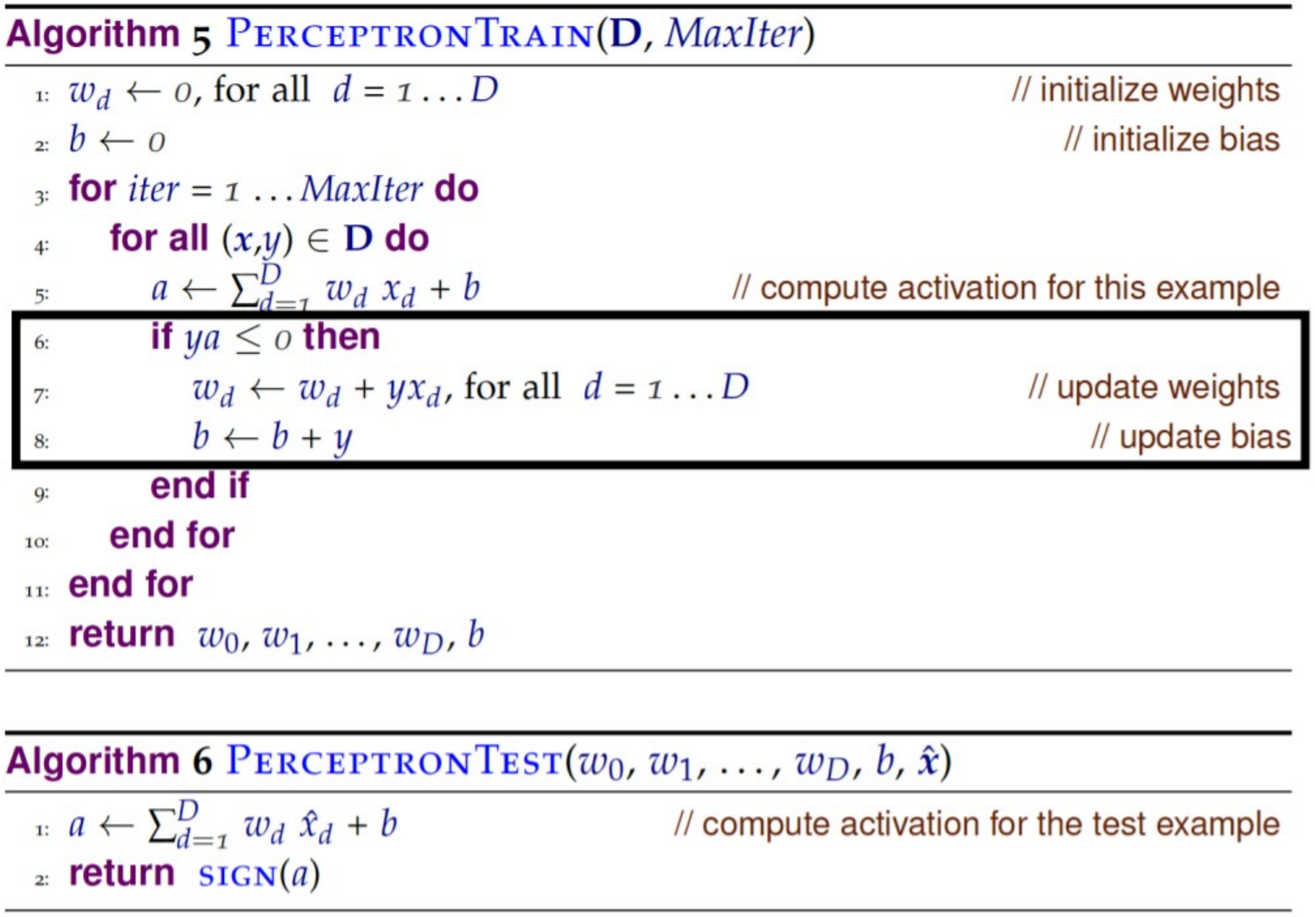
Intuition: Change the weight vector in such a way that it will do better on a misclassified example the next time around

- **Initialize w_0 … w_d randomly,**
- **For each example, make a prediction**
  - **If it is correct, do nothing**
  - **If it is incorrect, update the weight vector as**
    - **If the output unit incorrectly outputs 0, add input vector to the weight vector**
    - **If the output unit incorrectly outputs 1, subtract input vector from the weight vector**
- **Repeat**

$$w_i \leftarrow w_i + \eta\left(y - a\right) x_i$$

$$\boxed{w_i \leftarrow w_i + \eta(y - a)x_i}$$

Detailed Perceptron Learning Algorithm



Algorithm 5 PerceptronTrain($\mathbf{D}$, $MaxIter$)

```
$w_d \leftarrow 0$, for all $d = 1 \ldots D$                     // initialize weights
$b \leftarrow 0$                                                  // initialize bias
for $iter = 1 \ldots MaxIter$ do
    for all $(x,y) \in \mathbf{D}$ do
        $a \leftarrow \sum_{d=1}^{D} w_d\, x_d + b$               // compute activation for this example
        if $ya \leq 0$ then
            $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
            $b \leftarrow b + y$                                  // update bias
        end if
    end for
end for
return $w_0, w_1, \ldots, w_D, b$
```

Algorithm 6 PerceptronTest($w_0, w_1, \ldots, w_D, b, \hat{x}$)

```
$a \leftarrow \sum_{d=1}^{D} w_d\, \hat{x}_d + b$                 // compute activation for the test example
return sign($a$)
```

Decision Boundary

- **A perceptron is more specifically a linear classification algorithm, because it uses a line to determine an input's class**
  - **If we draw that line on a plot, we call that line a decision boundary**

$$B = \left\{ x : \sum_d w_d x_d = 0 \right\}$$

  - **The decision boundary is precisely where the sign of the activation, a changes from -1 to +1**
  - **We can rewrite this equation in terms of dot product between the two vectors w = <w$_1$, w$_2$,… w$_D$> and x = <x$_1$, x$_2$, … x$_D$>**

$$(x_i, y_i)\ \textit{is correctly classified if and only if}: y_i(w^T x_i) > 0$$

  - **The two vectors only has a zero dot product if and only if they are perpendicular**

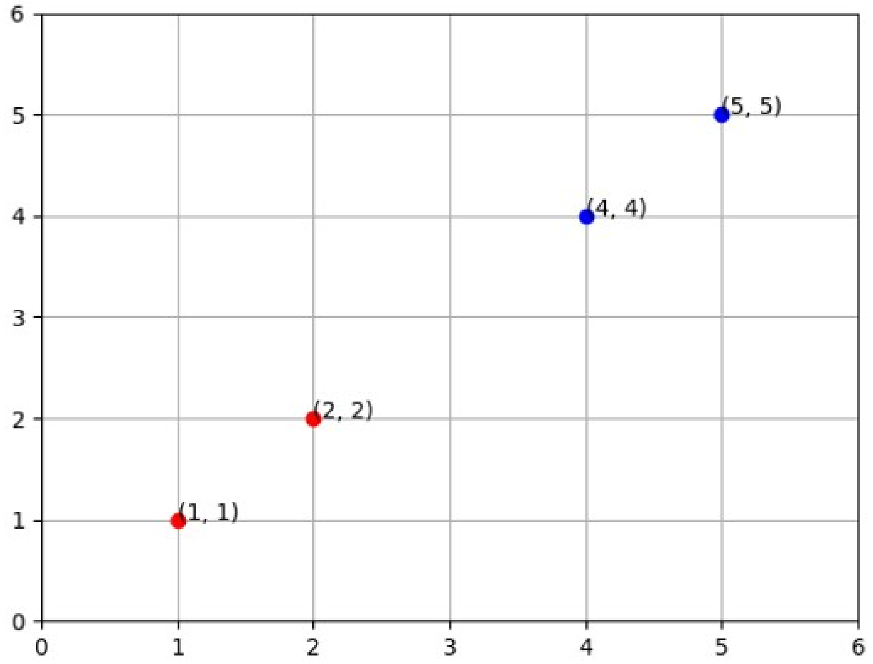
$$w^T x = 0 \qquad w^T x = 0$$

  - **Thus, the decision boundary is simply the plane perpendicular to w.**

Decision Boundary Example

x	y	Label
1	1	0
2	2	0
4	4	1
5	5	1

weights: [0.2, -0.1]
bias: -0.29



PC: Towardsdatascience

Decision Boundary

- **The summation of that our perceptron uses to determine its output is the dot product of the inputs and weights vectors, plus the bias** $w.x + b$

- **When our inputs and weights vectors of are of 2-dimensions, the long form of our dot product summation looks like this:**

$$w_1 . x + w_2 . y + b$$

- **That now looks an awful lot like the standard equation of a line**

$$w_1.x = -w_2.y - b$$

  - **We can now solve for two points on our graph, the x-intercept:**

$$w_1.x = -(w_2.y + b)$$

$$x = \frac{-(w_2.y + b)}{w_1}$$

$$x = \frac{-b}{w_1}, when\ y = 0$$

Decision Boundary contd.

- **And the y-intercept:**

$$w_1.x + w_2.y + b$$

$$w_1.x + w_2.y + b$$

$$w_2.y = -(w_1.x - b)$$

$$y = \frac{-(w_1.x - b)}{w_2}$$

$$y = \frac{-(w_1.x - b)}{w_2}$$

$$\boxed{y = \frac{-b}{w_2}, when\, x = 0}$$

$$y = \frac{-b}{w_2}, when\, x = 0$$

- **With those two points, we can find the slope, m:**

point$_1$ = $(x_1, y_1) = (0, -b / w_2)$

point$_2$ = $(x_2, y_2) = (-b / w_1, 0)$

Decision Boundary contd.

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

$$m = \frac{(0 - -(b / w2))}{(-(b / w1) - 0)}$$

$$m = -\frac{(\frac{b}{w_2})}{(\frac{b}{w_1})}$$

- **Now, we have the two values we need to construct our line in slope-intercept form:**
  - **slope = $-(b/w_2)/(b/w_1)$**
  - **y-intercept = $-b/w_2$**

$$y = -((b/w_2))/(b/w_1)\,x + (-b/w_2), like\; y = mx + c$$

Decision Boundary contd.

weights: [0.2, -0.1]
bias: -0.29

$$y = -\left(\left(-0.29/-0.1\right)\right)/\left(-0.29/0.2\right)x + \left(-\left(-0.29\right)/-0.1\right)$$

$$y = \left(-\frac{0.29}{-1.45}\right)x - 2.9$$

$$y = 2x - 2.9$$

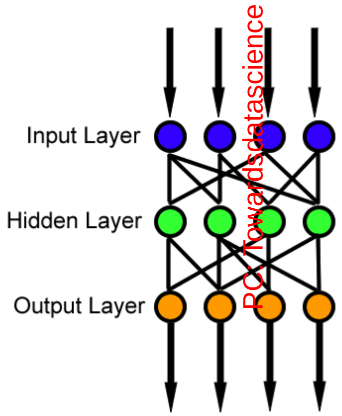
- **Plugging in our numbers from the dataset, we get the above.**

Introduction to Neural Networks

Neural networks are a series of algorithms that mimic the operations of a human brain to recognize relationships between vast amounts of data

Neural Networks

- **A neural network is made of artificial neurons that receive and process input data**
  - **A neural network process starts when input data is fed to it**
  - **Data is then processed via its layers to provide the desired output**
  - **The essential building block of a neural network is a perceptron or neuron**

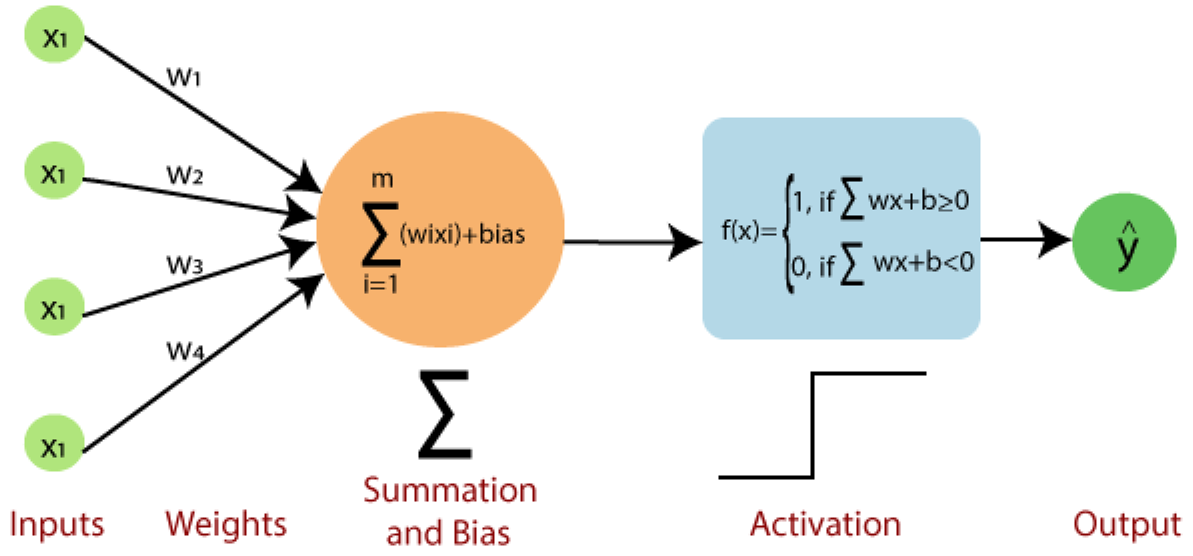


Layers in Neural Networks

- **Input Layer**
  - **The Input layer communicates with the external environment that presents a pattern to the neural network**
  - **Every input neuron should represent some independent variable that has an influence over the output of the neural network**
- **Hidden Layer**
  - **The hidden layer is the collection of neurons which has activation function applied on it**
  - **Its job is to process the inputs obtained by its previous layer**
- **Output Layer**
  - **The output layer of the neural network collects and transmits the information accordingly in way it has been designed to give**
  - **The pattern presented by the output layer can be directly traced back to the input layer**

How Neural Networks work

- **Neural Networks are complex systems with artificial neurons**



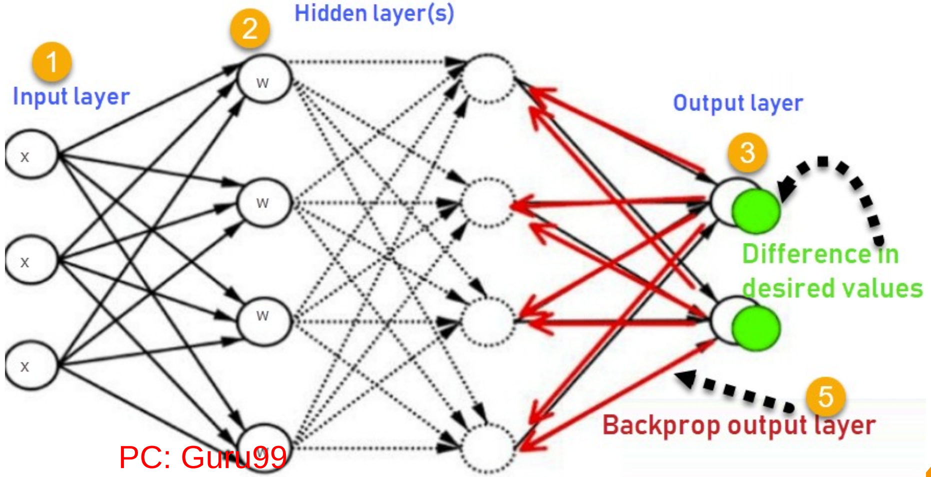
PC: Section.io

How Neural Networks work contd.

- **The input layer receives data represented by a numeric value**
  - **Once the input layer receives data, it is redirected to the hidden layer**
  - **Each input is assigned with weights**
- **The inputs and weights are multiplied, and their sum is sent to neurons in the hidden layer**
  - **Bias is applied to each neuron**
  - **Each neuron adds the inputs it receives to get the sum**
  - **This value then transits through the activation function**
- **The activation function outcome then decides if a neuron is activated or not**
  - **An activated neuron transfers information to the other layers**
- **The output layer produces the desired output**

Back Propagation

- **If the output of a neural network is wrong, backpropagation takes place**
    - **While designing a neural network, weights are initialized to each input**
    - **Backpropagation means re-adjusting each input's weights to minimize the errors, thus resulting in a more accurate output**



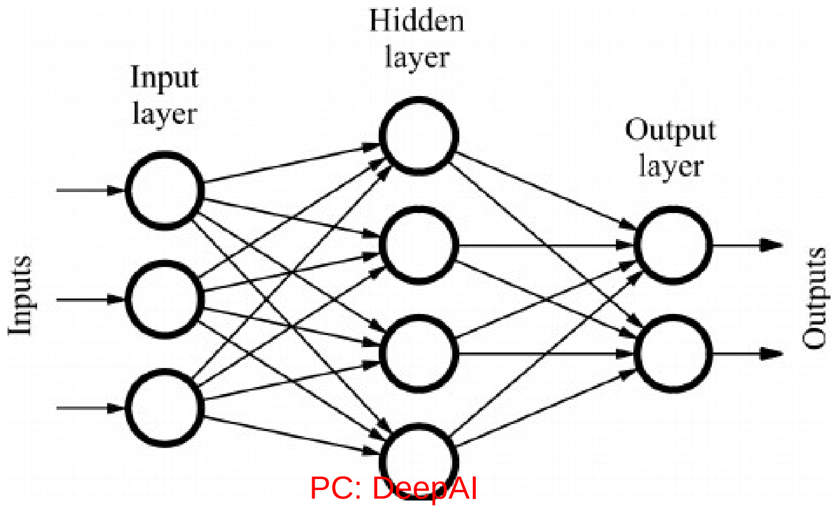
PC: Guru99

Types of Neural Networks

- Perceptron
- Feed-Forward Neural Network
- Recurrent Neural Network
- Convolutional Neural Network

Feed Forward Neural Network

- **In a feed-forward network, data moves in a single direction**
  - **It enters via the input nodes and leaves through output nodes**
  - **There is no backpropagation possible here**



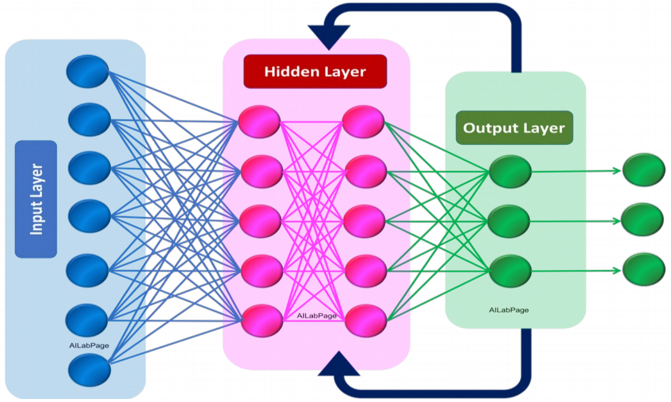
PC: DeepAI

Applications
1. Speech Recognition
2. Facial Recognition

Recurrent Neural Network

- **A Recurrent Neural Network (RNN) is a network good at modeling sequential data**
  - **Sequential data means data that follow a particular order in that a thing follows another**
  - **RNN is a feedback neural network**
  - **In RNNs, data runs through a loop, such that each node remembers data in the previous step**

RNN are used to solve problems in stock predictions, text data, and audio data

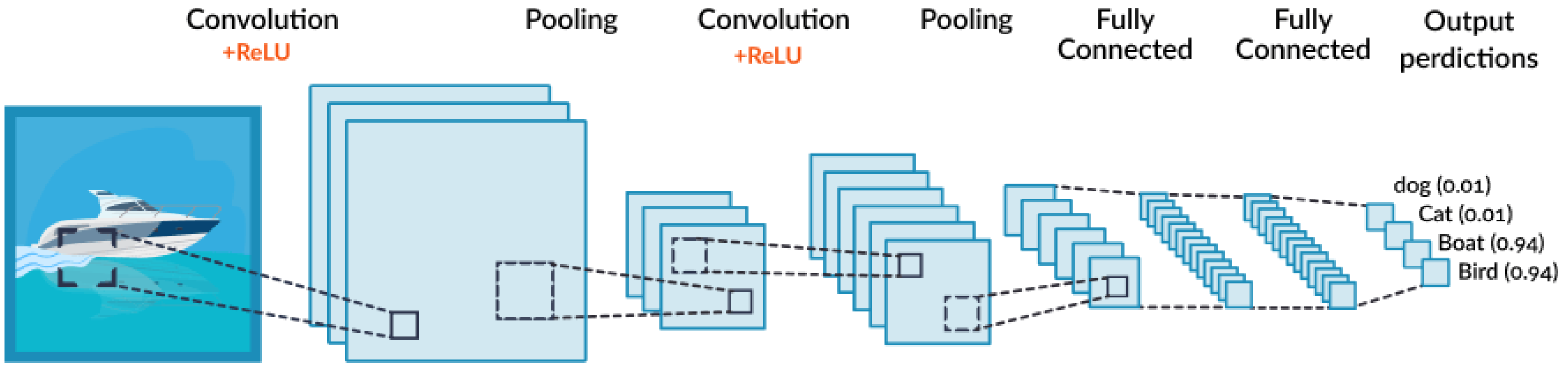


PC: Morioh

Convolution Neural Network

- **Convolutional Neural Networks (CNN) are commonly used for image recognition**
  - **CNNs contain three-dimensional neuron arrangement**
  - **The first stage is the convolutional layer: Neurons in a convolutional layer only process information from a small part of the visual field**
  - **The second stage is pooling: It reduces the dimensions of the features and, at the same time, sustaining valuable data**
  - **CNNs launch into the third phase (fully connected neural network) when the features get to the right granularity level**
  - **At the final stage, the final probabilities are analyzed and decide which class the image belongs to**
  - **CNN is mainly used in signal and image processing**

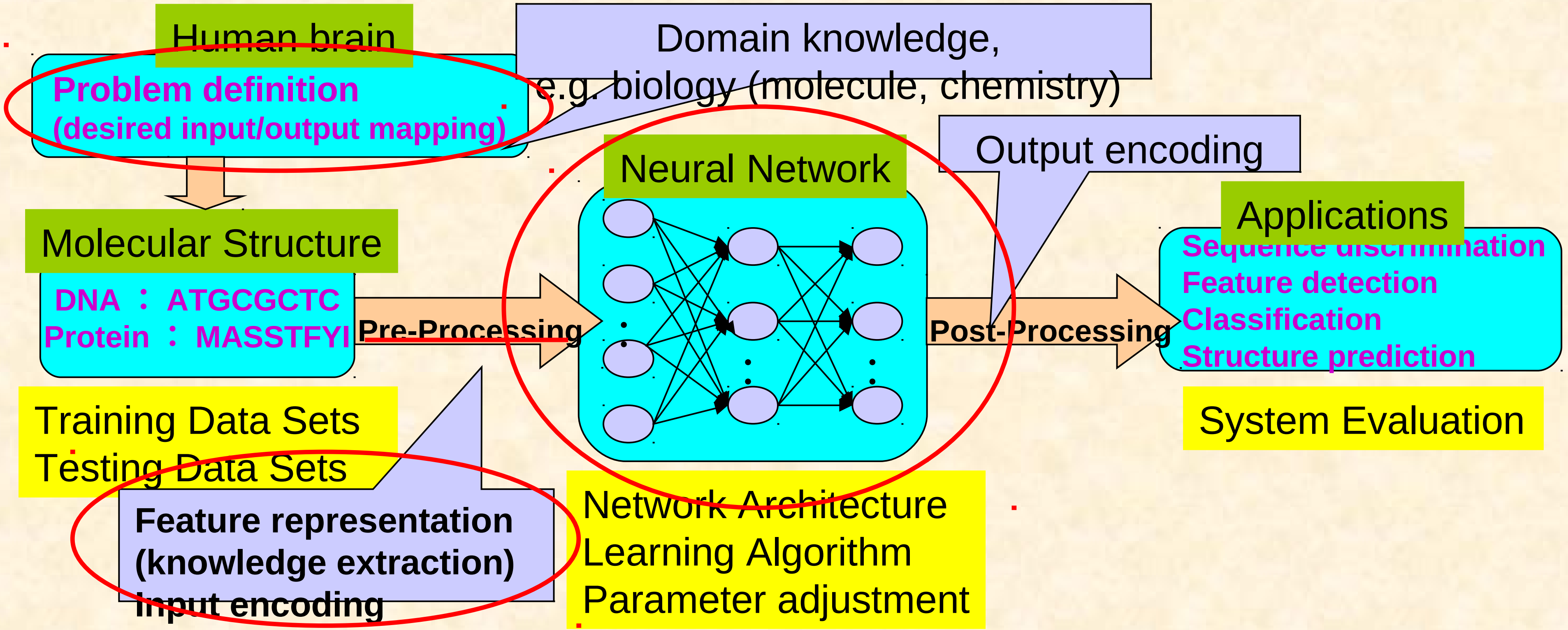
Convolution Neural Network contd.



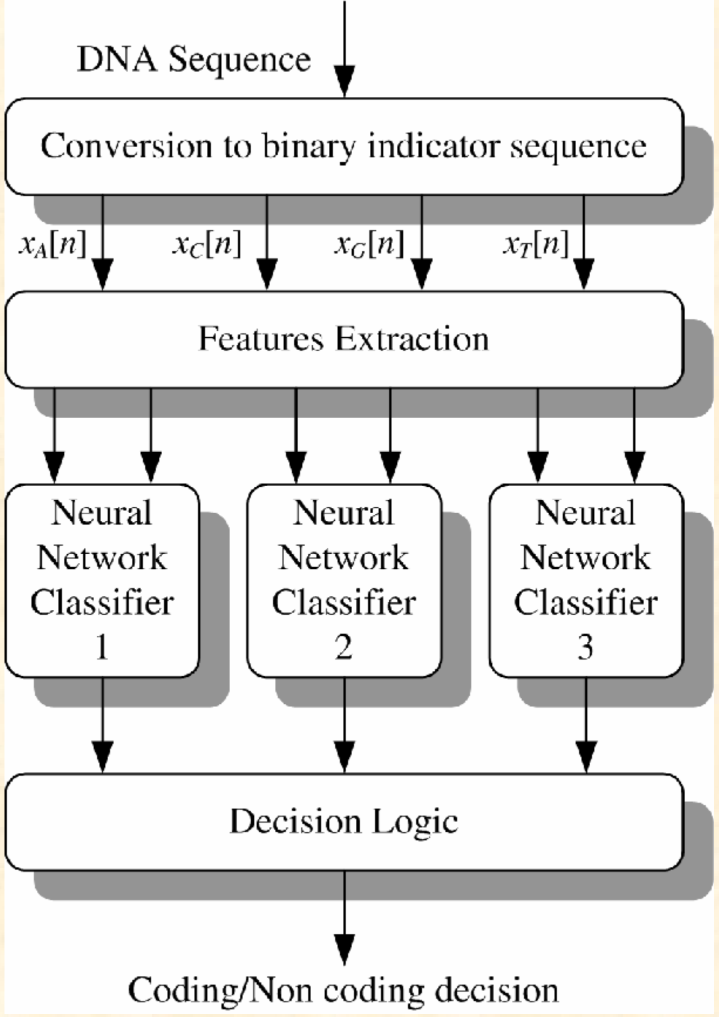
PC: Section.io

ANN in Bioinformatics

Design Issues

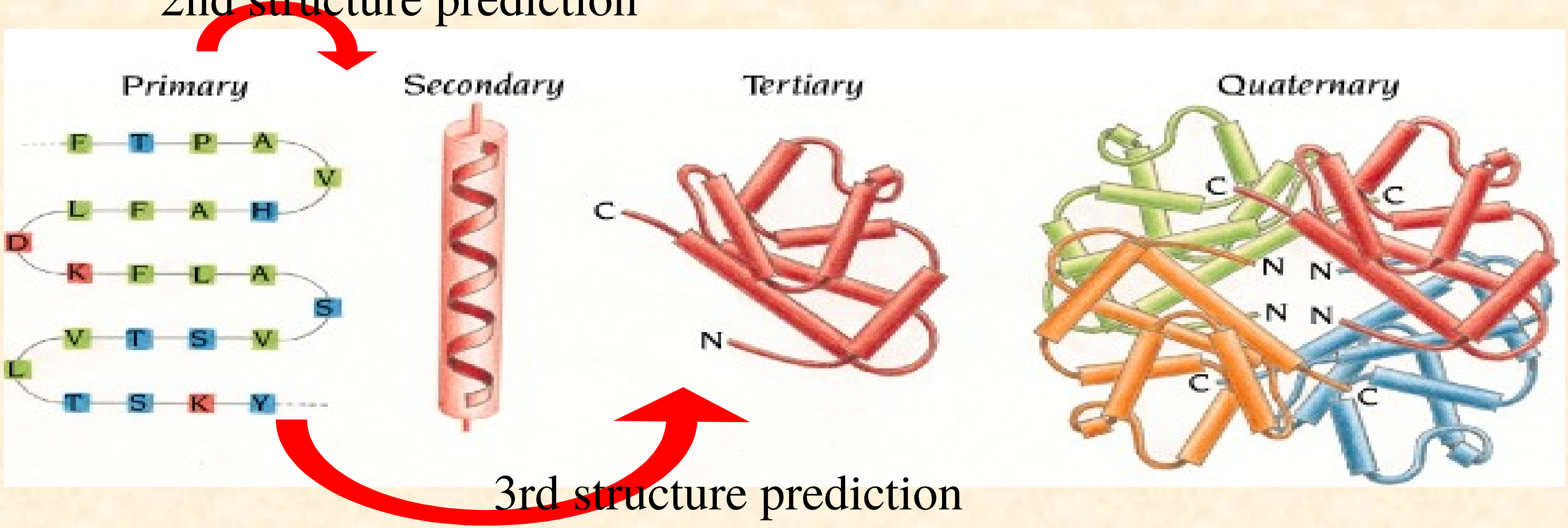


Gene Identification


DNA Sequence
Conversion to binary indicator sequence
$x_A[n]$
$x_C[n]$
$x_G[n]$
$x_T[n]$
Features Extraction
Neural Network Classifier 1
Neural Network Classifier 2
Neural Network Classifier 3
Decision Logic
Coding/Non coding decision

Hierarchy of Protein Structure

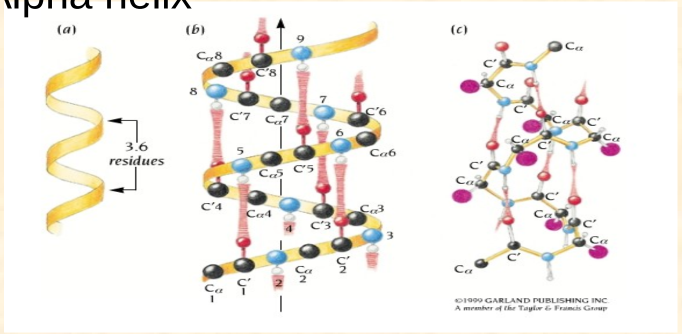
2nd structure prediction



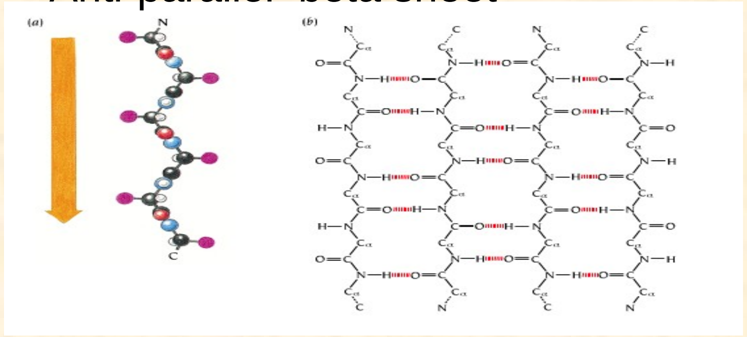
3rd structure prediction

Protein Secondary Structures

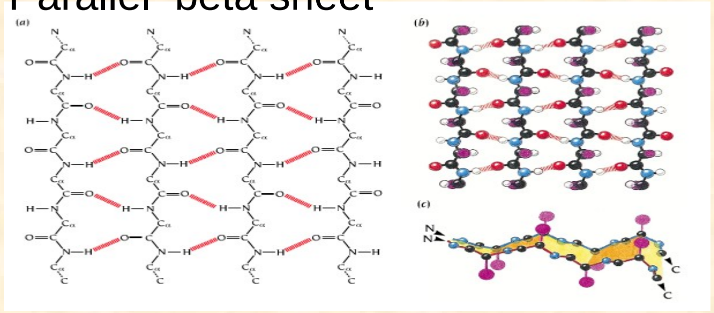
Alpha helix


(a)
(b)
(c)
3.6 residues
©1999 GARLAND PUBLISHING INC.
A member of the Taylor & Francis Group

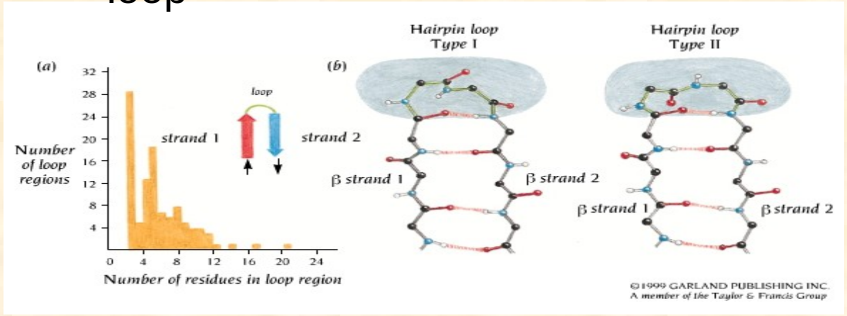
Anti-parallel beta sheet



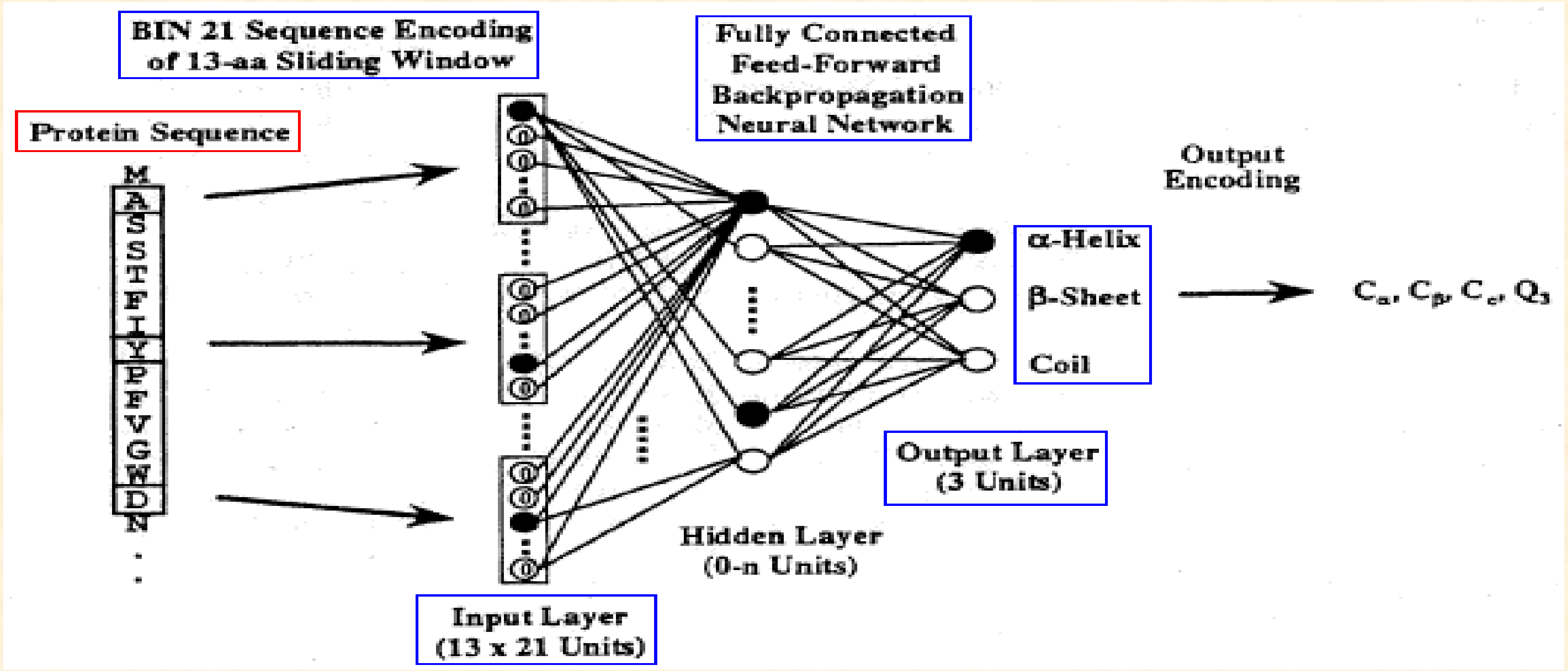
Parallel beta sheet



loop



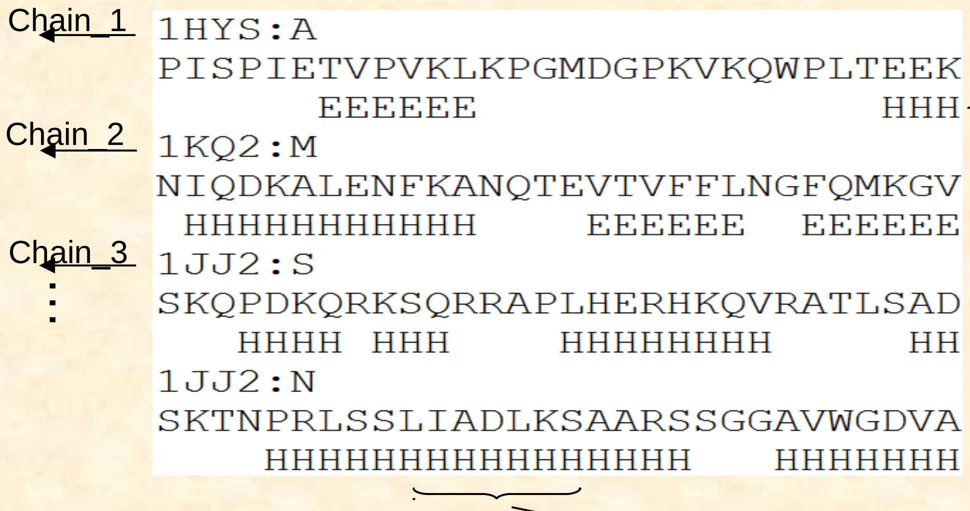
Prediction of Protein 2nd Structures



BIN 21 Sequence Encoding of 13-aa Sliding Window
Fully Connected Feed-Forward Backpropagation Neural Network
Protein Sequence
M A S S T F I Y P F V G W D N
Output Encoding
α-Helix
β-Sheet
Coil
C_α, C_β, C_c, Q_3
Output Layer (3 Units)
Hidden Layer (0-n Units)
Input Layer (13 x 21 Units)

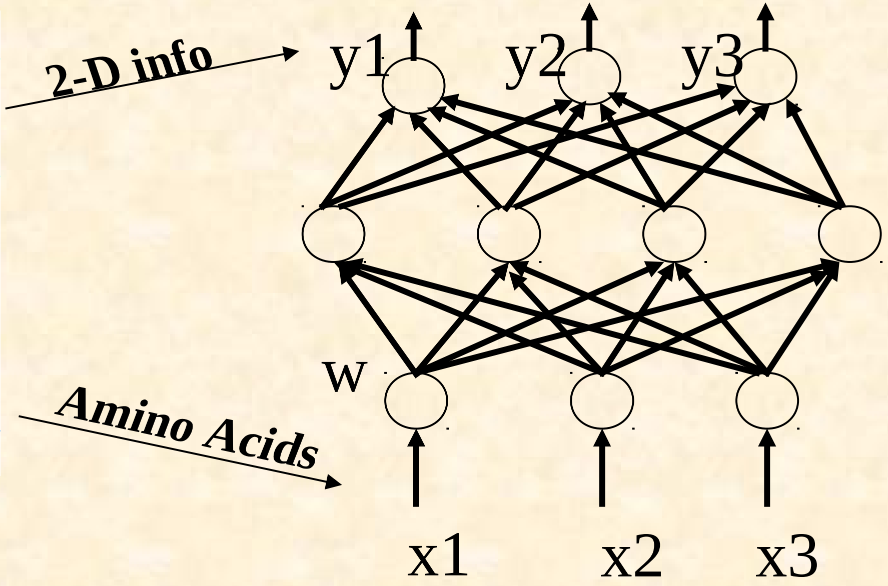
Adopted from Qian and Sejnowski, 1988

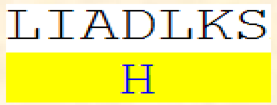
Sliding Window



Chain_1
Chain_2
Chain_3
⋮

```
1HYS:A
PISPIETVPVKLKPGMDGPKVKQWPLTEEK
      EEEEEE                HHH
1KQ2:M
NIQDKALENFKANQTEVTVFFLNGFQMKGV
 HHHHHHHHHHH    EEEEEE  EEEEEE
1JJ2:S
SKQPDKQRKSQRRAPLHERHKQVRATLSAD
   HHHH HHH    HHHHHHHH     HH
1JJ2:N
SKTNPRLSSLIADLKSAARSSGGAVWGDVA
    HHHHHHHHHHHHHHHH   HHHHHHH
```



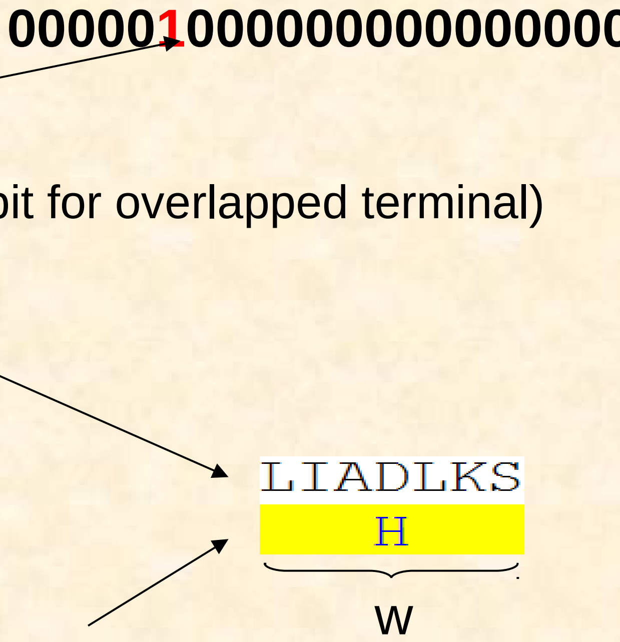


```
LIADLKS
   H
```

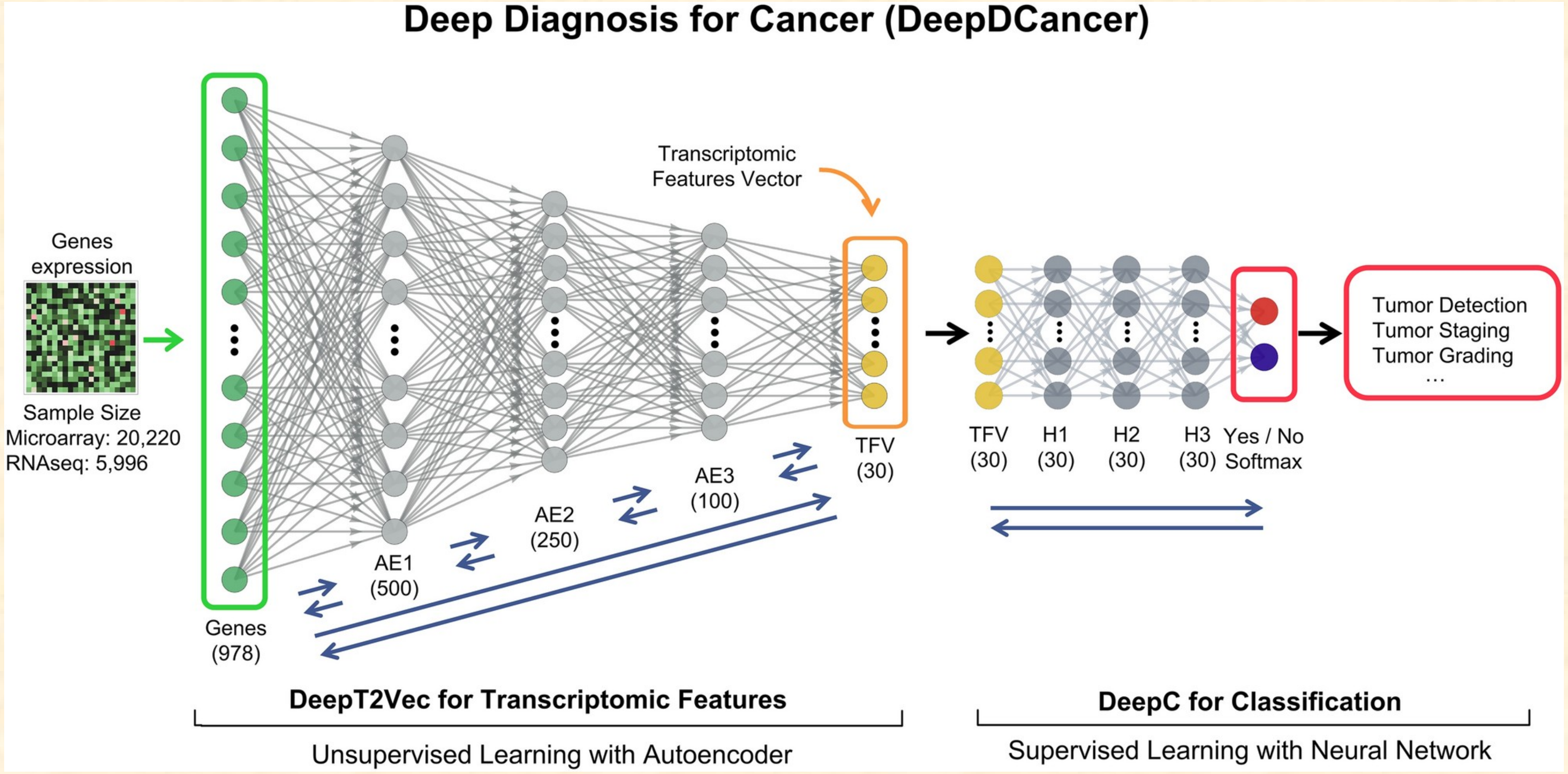
- Sliding window concept
  - Considering a piece of strings as inputs
  - Only looking at central position in a piece of strings to detect what kind of 2-D info. happens

Binary Bit Encoding Method

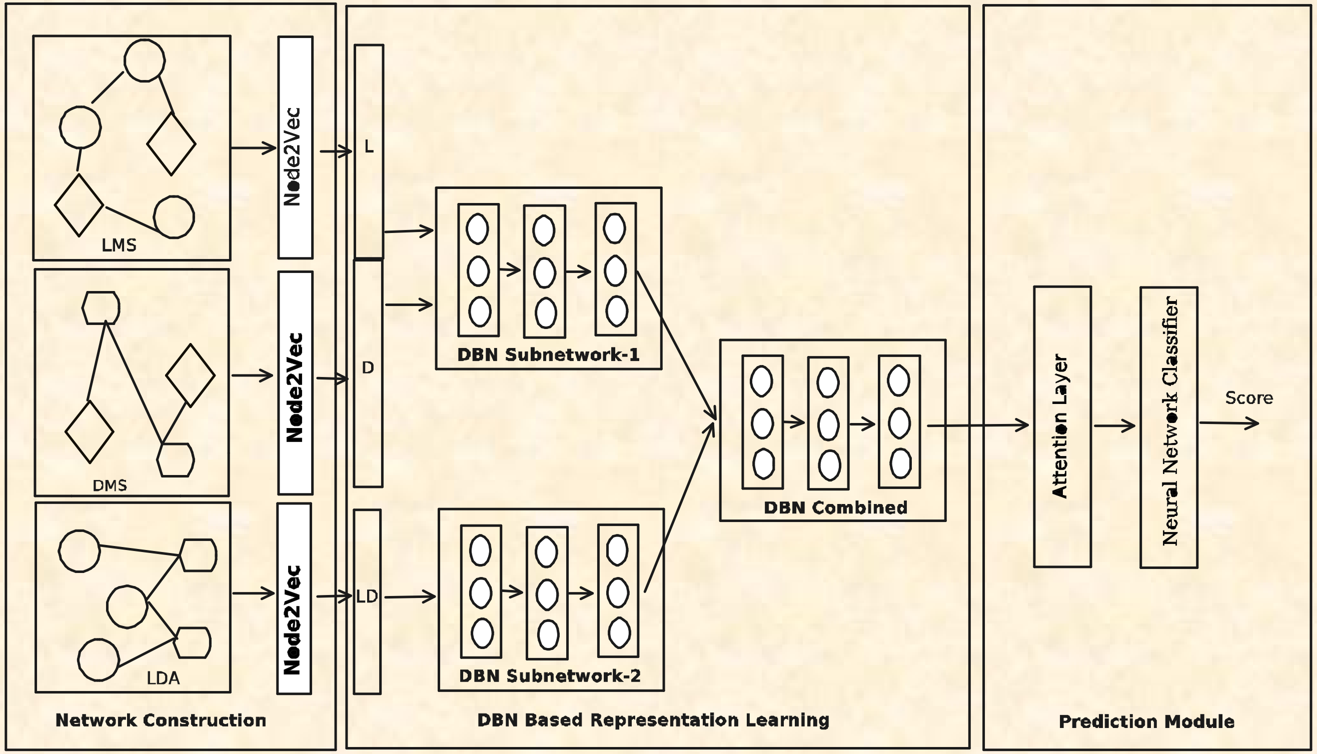
- Input encoding for each input pattern
  - Unary encoding scheme for protein sequence
    - 21 binary bits for 20 kinds of amino acid type (1 bit for overlapped terminal)
- Input layer with multiple Input patterns
  - A window size ‘w’ of consecutive residues been considered.
  - ‘21 * *w*’ units for sequence only
- Output layer with 3 units
  - To describe what kind of 2-D info. Happens
    (‘1, 0, 0’ for helix, ‘0, 1, 0’ for sheet, ‘0, 0, 1’ for coil)
- One hidden layer for non-linear 2-class pattern classificatio
  n



ANN for Microarray Analysis



DBNLDA: DL for Disease Association Prediction


LMS
DMS
LDA
Node2Vec
Node2Vec
Node2Vec
L
D
LD
DBN Subnetwork-1
DBN Subnetwork-2
DBN Combined
Attention Layer
Neural Network Classifier
Score
Network Construction
DBN Based Representation Learning
Prediction Module