Machine Learning Lect-2

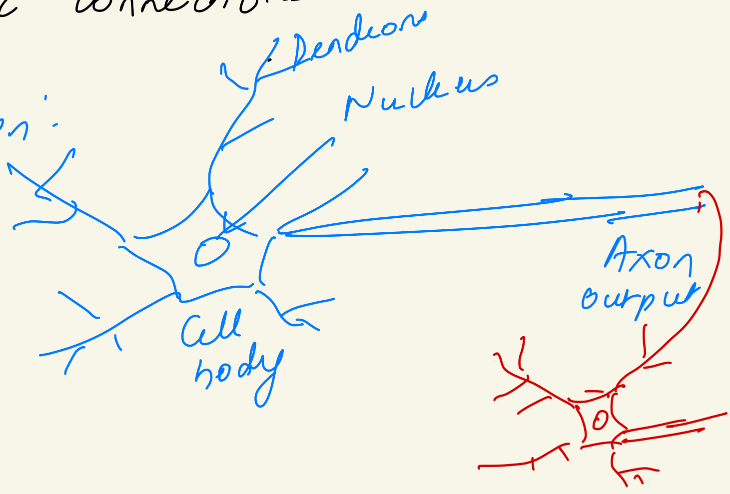
Introduction to Neural Networks

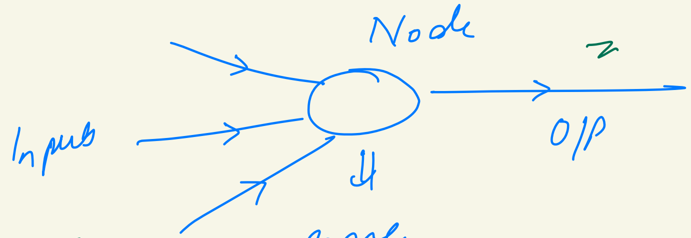
DR. PRERANA MUKHERJEE

Introduction

- Neural Networks
  ↳ Inspired by how human brain works & maps the inputs to logical outputs.
- Brain is the most complex part of human body & consists of 10 billion + neurons
- Neurons are highly interconnected & act as individual units.
- Characteristics of neurons:
  - Massive parallelism
  - Highly interconnected (solve the complex problems of your real world)
  - Model distributive associative memory through weights in the synaptic connections

Diagram of neuron:





Structure of neuron simulated by NN

↓

some simple computations weighted sum of inputs & then applies some squashing/activation function (could be sigmoid or other)

$z = \phi(\sum w_i x_i)$

ANNs through synaptic connections O/p meets another neuron input using electric impulses through the synaptic layers.

Incorporate the 2 fundamental component of the biological neural nets.

- Nodes – Neurons
- Weights – synapses

Basic unit in NN : Perception

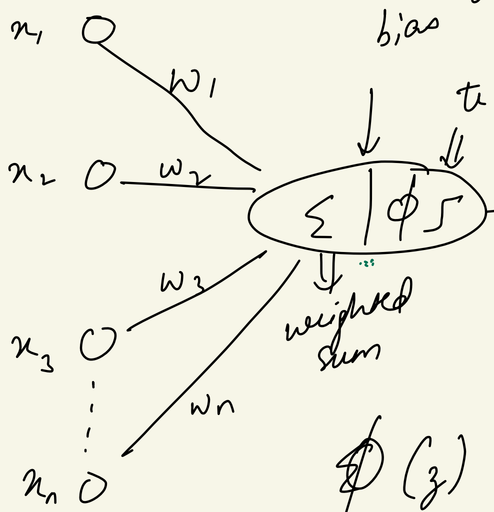
Perceptron

Inputs : n

Input $x_0 = 1$

bias or w_0

transfer function



y (0 or 1)

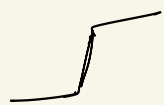
Training Data

$\bar{X}$	y
$\bar{x}_1$	y_1
$\bar{x}_2$	y_2
$\vdots$	$\vdots$
$\bar{x}_m$	y_m

Training the n/w means learning the weights to have good fit for training data.

$\in \mathbb{R}^d$

$\phi(z) = z$ (identity)

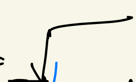
$\phi(z) =$  (unit step)

$\phi(z) :$  (sigmoid)

tanh

$$y = \sum_{i=1}^{n} w_i x_i + b \quad \text{or} \quad \sum_{i=0}^{n} w_i x_i$$

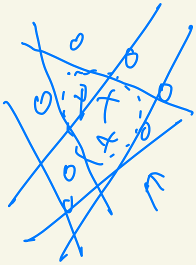
ϕ is identity function ∃ same passed as output

Or $\phi_2(z) =$  1

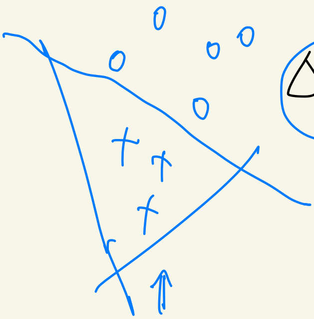
threshold function

Perceptron training rules

- we feed 1 eg at a time & update the weights based on output.



if output (predicted) == output (target) ⇒ no update in weight required.



$$w_{i_{next}} = w_{i_{prev}} + \Delta w_i$$

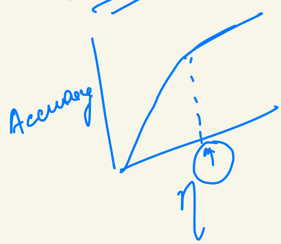
$$\Delta w_i = \eta \, (y - \hat{y}) \, x_i$$

η: Learning rate — based on error you change weights

y: Target

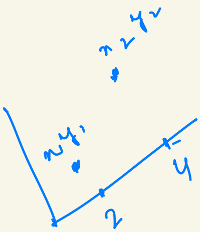
$\hat{y}$: Predicted

Grid search



Stopping criteria ⇒ Perceptron learning convergence if D (training data) is linearly separable. & linear decision surface to separate +/-

What if data is not linearly separable ??



4-2

Solution : Gradient descent

can be used when wrt to particular parameter set / weight values you can define the error of the n/w. Perform GD on error func. to find the optimal parameter set for which the error fn. is minimized

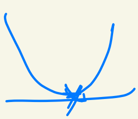
$$E = \frac{1}{2}\sum_{d \in D}\left(y_d - \hat{y}_d\right)^2$$

$\frac{1}{2}$ so that differentiation is easy

$\Rightarrow$ is like a surface & we want to find minima of this surface

Find partial derivative of error fn. with each & every weight and find the direction of the gradient & we go towards the -ve direction of gradient in order to go towards minima.

In certain cases, this error surface can be convex or quadratic & there will be single minima $\Rightarrow$ thus gradient descent guarantees the minima.



Multilayer networks error surface will be ill-behaved (non-convex)

there will be local minima

⇓

get stuck

$\phi_1(z) =$ step function → is not differentiable ⇒ so gradient descent can't be done

$\phi_1(z) = z$ is differentiable ⇒ gradient descent can be done

Stochastic gradient descent

1 eg. from training set at a time

$$E = \frac{1}{2}\left(y_d - \hat{y}_d\right)^2$$

Training rule of G.D

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = x_{ij}\left(y_j - \hat{y}_j\right)$$

Single layer n/w capture linear decisions / linear separable data.

Capture non-linear or complex functions ⇒ multi-layer n/ws

→ different n/ws connected with each other

→ If linear units are connected combination is again linear fn.

$$\phi_3(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$

Assignment
Prove it

$$\boxed{\phi_3'(z) = \phi_3(z) \cdot (1 - \phi_3(z))} \quad (*)$$

$$E = \frac{1}{2} \sum_{d \in D} \Big(y_d - \underbrace{\sigma(w \cdot x_d)}_{\hat{y_d}} \Big)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_d \frac{\partial E}{\partial \hat{y_d}} \cdot \frac{\partial \hat{y_d}}{\partial w_i}$$

$$= \cancel{\frac{1}{2}} \sum_d (y_d - \hat{y_d}) \cdot$$

$$\frac{\partial E}{\partial w_i} = \sum_d (y_d - \hat{y_d}) \cdot \sigma'(w \cdot x_d) \cdot x_{id}$$

$$\frac{\partial (y_d - \sigma(w x_d))}{\partial w_i}$$

$$\sigma'(w \cdot x_d) \cdot x_{id}$$

to represent complex functions we want non-linear units (which are differentiable)
⇓
eg sigmoid/ logistic transfer function

$$= \sum_d (y_d - \hat{y_d}) x_{id} \cdot \sigma'(w \cdot x_d) \quad (1)$$

Using π in 1

$$\frac{\partial E}{\partial w_i} = \sum_d (y_d - \hat{y}_d)\, x_{id}\, \hat{y}_d (1 - \hat{y}_d)$$

$$\Delta w_i = -\eta \sum_d (y_d - \hat{y}_d)\, \hat{y}_d (1 - \hat{y}_d)\, x_{id}.$$

Training rule of sigmoid unit

Single layered NN → for linearly seperable function.