DATE

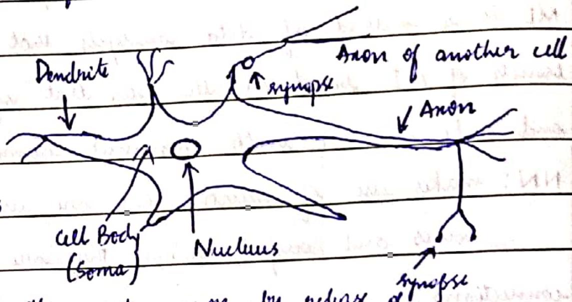
Introduction to Neural Networks

Introduction (From Josh)

ANN are m/c learning techniques which simulate the mech of learning biological organisms.

The human nervous system contains cells, referred as neurons.

The biological neuron



Dendrite → input ; Axon - output

Neurons Pass electrochemical impulses across synapses from one cell to the next.

Impulse must be strong enough to activate the next neuron by release of chemicals across a synaptic bifurcation.

The neuron is made up of a nerve cell consisting of a soma that has many dendrites but only one axon.

Axons are speci nerve fibres with a special cellular extension that comes from the cell body.

The single axon can branch hundreds of times, however.

Dendrites are thin structures that arise from the main cell body.

Synapses are connecting junction b/n axons and dendrites. Send signals from the axon of a neuron to the dendrite of another neuron.

Dendrites have fibers branching out from the soma in a bushy network around the cell.

Dendrites allow the cell to receive signals from connected neighbouring neurons. It performs multiplication by that dendrite's weight value. Multiplication means an increase or decrease in the ratio of synaptic neurotransmitters to signal chemicals introduced into the dendrites.

Axons are single, long fibers extending from the main soma. they stretch out longer distance than dendrites and measure generally 1cm in length. They will branch 100 of times and connects to other dendrites. Neurons are able to send electrochemical signal that travels along the cell's axon and activates synaptic connections with other neurons.

Information flow across the biological neuron

Synapses that increase the potential and those that decrease the potential.

Neurons also have been shown to form new connections over time and even migrate. These combined mechanics of connection

Total connection in the human brain

Biological neural networks are composed of 86 billion neurons connected to many other neurons

500 trillion connections between neurons in the human brain.

Computation happens $z = \frac{1}{1+e^{-y}}$

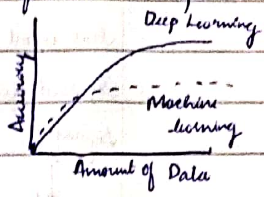
- Activation function: Tells what should be the output
- Various fns are there Sigmoid, Relu (Rectified linear unit)

| Eg: Baby understanding what cat is !
if $z = 1 \Rightarrow$ respond

DATE o/p function = Activation function

__Humans versus computers: Stretching the limits of AI__ (Charu)

Success of neural network: increased data availability and computational power of modern computers has outgrown the limits of traditional machine learning.



Smaller data sets = Machine learning is better.

Reasons: more choice, greater ease of model interpretation, tendency to handcraft interpretable features that incorporate domain specific insights.

__Neural Networks: (Josh)__

NN are computational model that shares some properties with the animal brain in which many simple units are working in parallel with no centralized control unit. Behaviour is shaped by its network Architecture (NA). NA can be defined by the following

(i) Number of neurons (ii) No. of layers (iii) Types of connections between layers.

__From biological to artificial__

Animal brain is fundamental components of the mind. Do not completely understand how this collection of decentralized functional units provides the foundation for thought and the seal of consciousness.

Modelling the neuron with beginning of perceptron.

__Perceptron__

Linear model used for binary classification (0 or 1). Neural networks considers perceptron as artificial neuron.

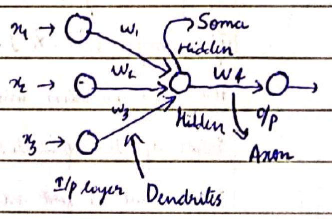
Preprocessing: Multiplication of input with weight.

$$Y = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

$$Z = Act(Y)$$

$$= Z * W_4$$

(i) Summation of product of feature with weights
(ii) Activation



__Definition of the perceptron__

The perceptron is a linear model binary classifier with a simple input-output relationship as, It shows we're summing n no. of inputs times their associated weights and then sending this "net input" to a step function with a defined threshold. Typically with perceptrons, this is a step function with a threshold value of 0.5. This function will output a single binary value (0 or 1), depending on input.

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

Net Activation function = dot product of the input and connection weights.

If the bias value is negative, it forces the learned weights sum to be much greater value to get a 1 classification output. Bias term is learned through perceptron learning algorithm.

__The perceptron learning algorithm__

The perceptron learning algorithm takes each input record, as we can see in Fig, and computes the output classification to check against the actual classification label.

classmate

PAGE

Hidden layer - Relu
Output layer - Anti Sigmoid

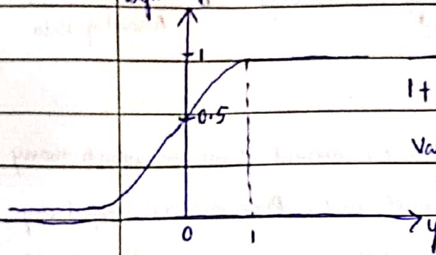
y = Summation of weight and inputs
$\hat{y} \Rightarrow$ Predicted

DATE

- To produce the classification, the columns are matched up to weights where n is the number of dimensions in both our input and ~~output~~ weights. Bias value - 1 because we don't affect the bias input.

Sigmoid function:

Sigmoid(y)



$$\frac{1}{1+e^{-y}}$$

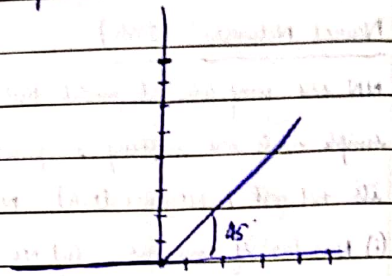
Varies from 0 to 1

Relu : max(y, 0)

Rectified Linear unit

$y = -ve$ max$(-ve, 0) = 0$

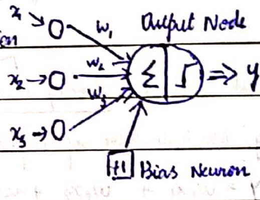
$y = +ve$ max$(+ve, 0) = +ve$



The Basic Architecture of Neural Networks (Charu C. Aggarwal)

In any type of application, one would have historical cases in which the class variable is observed and other cases in which the class variable has not yet been observed but needs to be predicted.

The input layer ~~does not~~ does not perform any computation on its own.



The predicted $\hat{y}$ is computed as follows.

$$\hat{y} = \text{sign}\{\bar{W} \cdot \bar{X}\} = \text{sign}\left\{\sum_{j=1}^{d} w_j \cdot x_j\right\}$$

The sign function maps a real value either +1 or -1, which is appropriate for binary classification.

Error of the prediction $E(\bar{x}) = y - \hat{y}$ (keep it minimum as possible).

Error value is ~~zero~~ nonzero, the weights in the neural network need to be updated in the direction off the gradient.

Perceptron contains single computational layer; it is considered a single layer network.

invariant part of the prediction (some entity that is not altered) which is referred to as bias

$$\hat{y} = \text{sign}\left\{\sum_{j=1}^{d} w_j \cdot x_j + b\right\}$$

The bias can be incorporated as the weight of an edge by using a bias neuron

Invariant: quantity that is unaltered.

Bias (Important): a neuron improves properties of the neuron. It allows moving the threshold of activation function.

bias value allows shifted ~~bot~~ to left or right to better fit data, it is always 1.

Weight represent the connection between units. Weights of NN can and will become positive or negative depending on training data.

Training: overall algo used to train NN to recognize a certain I/P and map it to O/P. Involves finding a set of weights in the network that proves to be good or good enough, at solving specific problem.

Learning: deals with individual weights and thresholds and decides how those would be manipulated.

$\eta \rightarrow$ learning ~~weight~~ rate
values from 0.01 - 0.0001, normally we take 0.001

$O_{11} \rightarrow$ output

W_{11}^{1} → hidden layer 1, → hidden layer node; ↓ Input node

$f_{11} \rightarrow$ Neuron; ↓ Hidden

$\boxed{Cost = \sum_{i=0}^{n} (y-\hat{y})^2}$

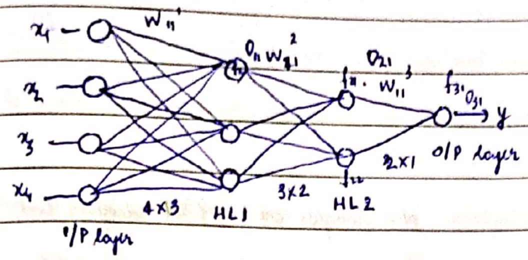
DATE

$\boxed{\text{Loss function}: (y-\hat{y})^2}$ to reduce this we use optimizers. Optimizers will undergo backpropagation.

$\boxed{W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}}$

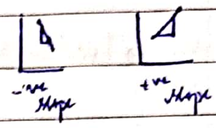
1 iteration = 1 forward propagation + 1 backward propagation
↓
referred to as epoch.

Multilayer Neural Networks



step 1: $\sum_{i=1}^{n} W_i x_i + bias$

step 2: Act (z)

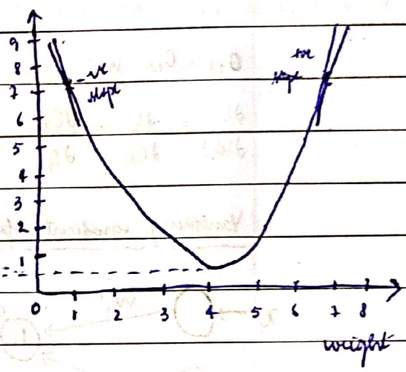


Matrix is formed w.r.t weight and next layer.

Gradient descent (optimizer) : Reduces the $(y-\hat{y})^2$; → Gradient descent; Stochastic Gradient : selects one feature randomly and performs F/W and B/W propagation

To update the weight such a way that the $\hat{y}$ is to be similar to y. To find out the minimum efficient weights to be updated to get best $\hat{y}$.

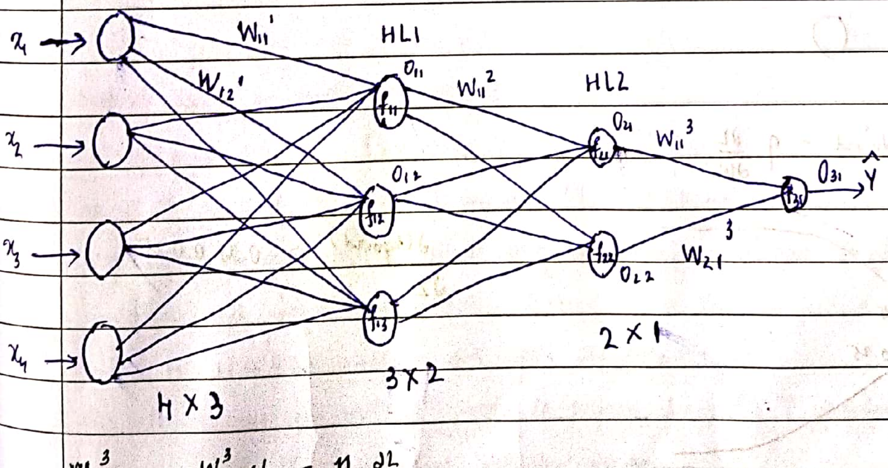
$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W} \Big\} \text{done by output node}$$



Neural network will not learn properly until it reaches global minima.

-ve slope reduces loss ; +ve slope ~~increases~~ converges to the global minimum point.

O_{31} is impacted by Act. function intern impacted with weights.



$$W_{11\,new}^{3} = W_{11\,old}^{3} - \eta \frac{\partial L}{\partial W_{11}^{3}}$$

PAGE

classmate

$O_{31} = W_{11}^{3} * O_{21} + W_{21}^{3} * O_{22}$

Dataset
Testing data } Goal problem.

DATE

Anaconda is a free and open-source programming python distribution and collection of hundreds of packages related to data science, scientific programming, development and more.

$$\frac{\partial L}{\partial W_{11}^3} = \frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial W_{11}^3}$$ From chain rule of derivative

$$W_{21\,new}^{3} = W_{21\,old}^{3} - \eta \frac{\partial L}{\partial W_{21}^3}$$

$$\frac{\partial L}{\partial W_{21}^3} = \frac{\partial L}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial W_{21}^3}$$

$$W_{11}^2 \rightarrow W_{11\,new}^{2} = W_{11\,old}^{2} - \eta \frac{\partial L}{\partial W_{11}^2}$$ New weights are computed during back propagation by using these computation

$$O_{21} = W_{11}^2 . O_{11}$$

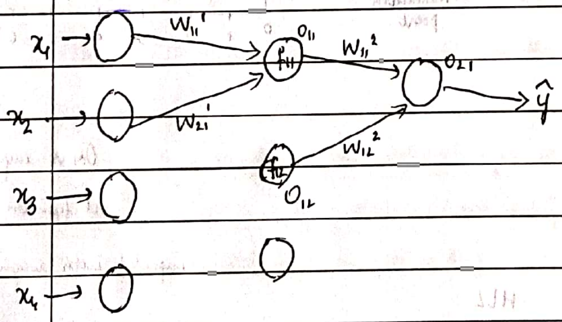
$$\frac{\partial L}{\partial W_{11}^2} = \frac{\partial L}{\partial O_{31}} . \frac{\partial O_{31}}{\partial O_{21}} . \frac{\partial O_{21}}{\partial W_{11}^2}$$

$$W_{12}^2 \Rightarrow W_{12\,new}^{2} = W_{12\,old}^{2} - \eta \frac{\partial L}{\partial W_{12}^2}$$

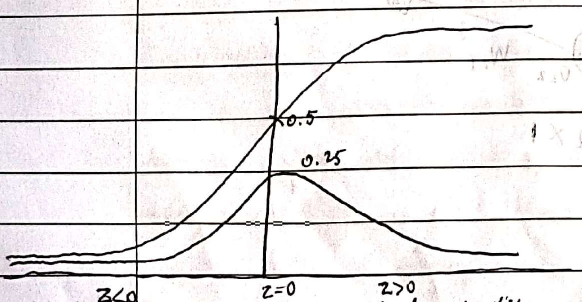
$$O_{22} = O_{12} . W_{12}^2$$

$$\frac{\partial L}{\partial W_{12}^2} = \frac{\partial L}{\partial O_{31}} . \frac{\partial O_{31}}{\partial O_{22}} . \frac{\partial O_{22}}{\partial W_{12}^2}$$

Vanishing gradient problem



$$W_{11\,new}^{1} = W_{11\,old}^{1} - \eta \frac{\partial L}{\partial W_{11}^1}$$



$\frac{\partial(\text{sigmoid})}{\partial Z} \rightarrow$ 0 to 0.025

Loss function value is less implies y and $\hat{y}$ are almost same and prediction is correct.

Vanishing gradient is because of sigmoid. Derivative of Relu is 1, hence the gradient will not vanish. Hence we will not get any problem.

If more number of layers are there both exploding and vanishing will occur together.

DATE

Only for sigmoid function loss is found.

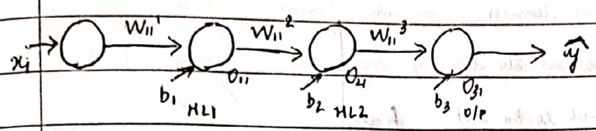
$$W_{11\,new}' = W_{11\,old}' - \eta \frac{\partial L}{\partial W_{11}'} \qquad loss = (y-\hat{y})^2$$

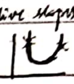
$$\frac{\partial L}{\partial W_{11}'} = \frac{\partial L}{\partial O_{21}} \times \overset{0 \text{ to } 0.25}{\frac{\partial O_{21}}{\partial O_{11}}} \times \overset{0 \text{ to } 0.25}{\frac{\partial O_{11}}{\partial W_{11}'}}$$

(Activation function) Derivative has caused the vanishing of gradient descent.

To avoid this vanishing gradient we came up with relu activation function.

Exploding gradient problem (Because of initialization of weight and sigmoid function) Here we will never go to global minima but we will be moving between positive and negative slopes





$$W_{11\,new}' = W_{11\,old}' - \eta \frac{\partial L}{\partial W_{11\,old}}$$

$$\frac{\partial L}{\partial W_{11}} = \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W_{11}'}$$

b is bias, and it is there for all the neurons.

$$O_{21} = Act(z)$$

$$Z = W_{11}^2 O_{11} + b_2$$

$$\frac{\partial O_{21}}{\partial O_{11}} = \frac{\partial [Act(z)]}{\partial O_{11}}$$

$$\frac{\partial}{\partial O_{11}} \quad \frac{\partial [Act(z)]}{\partial z} \times \frac{\partial z}{\partial O_{11}} \Rightarrow 0 \text{ to } 0.25 \times \frac{\partial [W_{11}^2 O_{11} + b_2]}{\partial O_{11}}$$

$$0 \text{ to } 0.25 * [W_{11}^2 + 0]$$

$$\Rightarrow 0 \text{ to } 0.25 * W_{11}^2$$

Dropout Layers in Multi Neural Network

Underfitting in NN:

One of the problems that occur during neural network training

Underfitting is the case where the model has "not learned enough" from the training data, resulting in low generalization and unreliable predictions.

A model with too little capacity cannot learn the problem.

Too little learning will make the model to perform poorly on the training dataset and on new data.

An underfit model has high bias and low variance.

We deactivate some features in overfitting problem. we decide using dropout ratio, it is ...
Selection and deletion of neurons are random. In BP, updates only active neurons's weight.
Weight is reduced to not being overfit.

DATE

Overfitting in NN:

One of the problems that occur during the neural network's training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large.

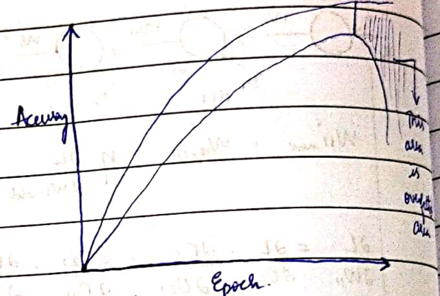
The network has memorized the training examples, but it has not learned to generalize to new situations.

A model with too much capacity can learn it too well and overfit the training dataset.

With too much learning, the model will perform well on the training dataset and poorly on new data.

An overfit model has low bias and high variance. The model learns the training data too well and performance varies widely with new unseen.

Good fit model: A model that suitably learns the training dataset and generalizes well to the old out dataset.



How to handle **underfitting problem**?

Underfitting can easily be addressed by increasing the capacity of the network.

How to handle **overfitting problem**?

(i) Reduce the network's capacity by removing layers or reducing the number of neuron elements in the hidden layers

(ii) Use **dropout layers** which will randomly remove certain features by setting them to zero.

Dropout layers are an easy and effective way to prevent overfitting in NN.

A dropout layer randomly drops some of the connections between layers. This helps to prevent overfitting.

We reduce the weights of the model to small during learning.

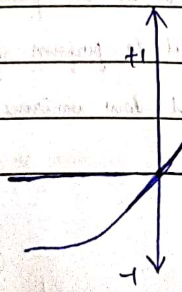
These techniques not only reduce overfitting, but they can also lead to faster optimization of the model and better overall performance.

In deep NN, there are many weights and bias parameters. When we have huge weights and bias parameters, the NN ~~error~~ tends to overfit the data set problem on a particular use scale.

In deep NN, underfit will never happen. Usually underfitting happens with one layer NN.

To fix this problem technique used called **dropout**.

derivative will vary from 0 to 1



$$\tanh(z) = \frac{1-e^{-2z}}{1+e^{-2z}}$$

Hyperbolic tan

DATE

If we use sigmoid and tanh activation function, gradient descent (may or may not) converge and vanishing descent problem occurs. (Disadvantages)

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W}$$

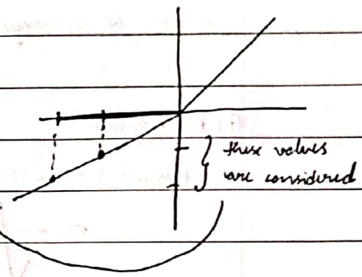
we find negligible difference in W_{new} and W_{old} when sigmoid and tanh activation function are used.

Gradient descent may or may not converge when sigmoid and tanh AFs are used. It may converge when no. of iterations are performed for many times.

This problem does not arises in Relu function, so it is used in deep neural network.

In Relu also, gradient descent will not converge when $z=0$ i.e. $W_{new} = W_{old}$, so you need to ensure that where $z<0$, you need to have some value ~~great~~ apart from zero.

This is leaky relu when $z=0, z<0$



How to get this component ?

$= (-)z \times 0.01$

$= -0.01$

To avoid the use of dead neuron in the network, we make use of leaky neuron. Leaky neuron need not be applied when less number of neurons are dead.

With leaky Relu vanishing gradient problem never arises.

Weight Initialization techniques

Key points while initializing the weights

(i) Weights should be small (but must not be very very small)

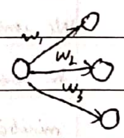
(ii) weights should not be same

(iii) Weight should have good variance

uniform distribution technique

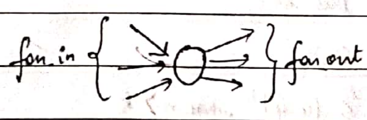
weights are taken and sampled from uniform distribution

It should be within a range



will be range.

$$\left[\overset{a}{-\frac{1}{\sqrt{fan\_in}}}, \overset{b}{+\frac{1}{\sqrt{fan\_in}}} \right]$$



Sigmoid activation function works better for this.

classmate PAGE

Scanned with CamScanner

In Minibatch SGD the curve is somewhat smoother than GD w.r.t some data (Fast)

GD > Minibatch > SGD

DATE

2. Xavier / Gorat distribution:

In this we have two

(select the smaller value)

(i) Xavier / Gorat uniform distribution

Wi will range from $\left[-\frac{\sqrt{6}}{\sqrt{fan\text{-}in + fan\text{-}out}}, +\frac{\sqrt{6}}{\sqrt{fan\text{-}in + fan\text{-}out}} \right]$

(ii) Xavier / Gorat Normal:

Wi will range from Normal (mean = 0 and σ = standard deviation)

$$\sigma = \sqrt{\frac{2}{fan\text{-}in + fan\text{-}out}}$$

sigmoid activation function works well.

3. He initialization

(i) He uniform:

Wi will be in range $\left[-\sqrt{\frac{6}{fan\text{-}in}}, +\sqrt{\frac{6}{fan\text{-}in}} \right]$

(ii) He normal:

Normal (mean = 0, std deviation = σ)

$$\sigma = \sqrt{\frac{2}{fan\text{-}in}}$$

ReLU activation function works well.

Stochastic Gradient:

Only one i/p feature is randomly picked out of 'n' features, it is forward propagated and then backward propagation is performed.

Error = $(y-\hat{y})$; Loss = $(y-\hat{y})^2$

Cost = $(y-\hat{y})^2$

Benefit: Accuracy is more in this model

Training takes more time.

We consider subset of dataset

We consider this as

minibatch SGD.

Loss = $\sum_{i=1}^{k} (y-\hat{y})^2$ where $n > k$

Gradient Descent

Error = $(y-\hat{y})$

Loss = $(y-\hat{y})^2$

Cost = $\sum_{i=1}^{n} (y-\hat{y})^2$

Accuracy is less than SGD.

Training is done more quickly

We consider full dataset

For (GD): To compute huge data, we need to have all GD in the RAM. Computational power required is also large. More amount of time is required to compute $\hat{y}$.

For (Minibatch): To handle this less amount of memory is enough and also less computation is enough. Only the required ... allocate. After sometime SGD overtakes GD ⇒ SGD minibatch is faster than GD

CLASSMATE

PAGE