- Activation Function-to normalise our output

Types of Activation Function

Sigmoid :
f(x)=1/(1+e^-x)

- Smooth function and is continuously differentiable
- Non Linear output , in range (0,1)
- Nearly flat at extremes 0,1 . Hence gradient is almost 0 (No learning)
- outputs values of same sign(i.e. +ve) which is often not desirablese
- Useful for Two classes

tanh:
tanh(x)=2/(1+e^(-2x)) -1 OR tanh(x)=2sigmoid(2x)-1

- More steeper than sigmoid which means gradient descent is easy
- output in range (-1,1) so solves same sign output problem of sigmoid
- rest properties same as Sigmoid

Relu:
f(x)=max(0,x)

- it does not activate all neurons , since when input is negative it would convert them to 0
- making the network sparse making it efficient and easy for computation.
- Since the negative side of relu has no gradient , it can create dead neurons since no gradient descent would update them

Leaky Relu
f(x)= ax, x<0
= x, x>=0

- same as relu in which negative part of Relu i(horizontal line) is replaced with a non-zero, non-horizontal line of small scope like 0.001
- it solves problem of zero gradient in relu and hence no dead Neurons
- However the non zero horizontal line can slow the learning if its slope is not properly chooosed.
- An Up-gradation is **parametrised Relu** in which parameter **a** in Leaky Relu is also trainable.

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

- Used for multi class classification
- outputs the probability of each class
- ideally used in output layer

One Hot Encoding

Converting Categorical data ([Coffee, Tea, Tea, Coffee]) into sequence of binary numbers (binary vector)

1. [[0,1],
2. [1,0]

Data in numbers is easy to work as compared to categorical data

Process

1. **Data :** [house, car, tooth, car]
2. **Integer Encoding :** [0,1,2,1]
3. **One Hot Encoding :**
   [[1,0,0,0],
   [0,1,0,1],
   [0,0,1,0],
   [0,1,0,1]]

Advantage:

- In integer encoding weight with higher input would train more example tooth in above case
- Hot encoding uniforms data and gives equal opportunity for all nodes to get trained equally

Maximum Likelihood

- Probability of a model is product of probabilities of all events in that model.
- Maximum Likelihood is the model with maximum probability (Maximum chances of output being labelled correctly)
- How to find?
  - Product all probabilities of points classified
  - Compare to other models
- But above method is not used for two reasons
  - computationally expensive to multiply large outputs
  - We need some relation between probability and error to simplify Calculation

Cross Entropy

Cross Entropy is sum of negative of all probabilities of events in a model

- Low entropy : Good Model
- High Entropy : Bad Model

Binary Cross Entropy

$$CE = -\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

Multi Class cross Entropy

$$\text{Multi Class Entropy} = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

j=number of classes , i = num of events

Error Function

Since , negative of probability of point also signifies how close it to accurate classification we use it for error calculation hence
the Error equation becomes similiar to Cross Entropy but divided with total number of events ,
Hence in this way , Minimising the error means maximizing the probability .
Error of Point is : **Error(y, ˆy) = − ylog(ˆy) − (1 − y)log(1 − ˆy)**
Error of whole model

$$Error = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Gradient Descent

tion give us the maximum change possible in a function.
t would give us the minimum change
ninimise the Error Function.

Lets get our Hands dirty

$$\begin{aligned}\sigma'(x) &= \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} \\ &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\ &= \sigma(x)(1-\sigma(x))\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial w_j}\hat{y} &= \frac{\partial}{\partial w_j}\sigma(Wx+b) \\ &= \sigma(Wx+b)(1-\sigma(Wx+b)) \cdot \frac{\partial}{\partial w_j}(Wx+b) \\ &= \hat{y}(1-\hat{y}) \cdot \frac{\partial}{\partial w_j}(Wx+b) \\ &= \hat{y}(1-\hat{y}) \cdot \frac{\partial}{\partial w_j}(w_1x_1 + \cdots + w_jx_j + \cdots + w_nx_n + b) \\ &= \hat{y}(1-\hat{y}) \cdot x_j\end{aligned}$$

Gradient with respect to weight w_j would give us way to minimise Error E with respect to w_j

$$\begin{aligned}\frac{\partial}{\partial w_j}E &= \frac{\partial}{\partial w_j}[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})] \\ &= -y\frac{\partial}{\partial w_j}\log(\hat{y}) - (1-y)\frac{\partial}{\partial w_j}\log(1-\hat{y}) \\ &= -y \cdot \frac{1}{\hat{y}} \cdot \frac{\partial}{\partial w_j}\hat{y} - (1-y) \cdot \frac{1}{1-\hat{y}} \cdot \frac{\partial}{\partial w_j}(1-\hat{y}) \\ &= -y \cdot \frac{1}{\hat{y}} \cdot \hat{y}(1-\hat{y})x_j - (1-y) \cdot \frac{1}{1-\hat{y}} \cdot (-1)\hat{y}(1-\hat{y})x_j \\ &= -y(1-\hat{y}) \cdot x_j + (1-y)\hat{y} \cdot x_j \\ &= -(y-\hat{y})x_j\end{aligned}$$

also,

$$\frac{\partial}{\partial b}E = -(y-\hat{y})$$

Updating Weights and bias

- **$w_i' \leftarrow w_i - \alpha[-(y-y\hat{})x_i]$**
- **$b' \leftarrow b + \alpha(y-y\hat{})$**

Here **,α** is learning rate

Logistic Regression Algorithm

1. Start with Random weight and bias
2. For any point that is misclassifed
    1. Calculate Gradient
    2. upate weights and bias
3. Repeat Until Error is small

transfering inputs / Data through different layers c

reaches to output layer.

A feed foward network has

- Input layer
- Middle Layers/ Hidden Layers
- Output Layers

Backpropagation

Backpropagation is process of sending back error from output nodes to input nodes through hiddle nodes
The process is as follows

- Doing a feedforward operation.
- Calculating the error.
- Running the feedforward operation backwards (backpropagation) to spread the error to each of the weights.
- Use this to update the weights, and get a better model.
- Continue this until we have a model that is good.

Overfitting and Underfitting

Overfitting is over generalisation of data such that It has adjusted to patterns in training data.
It can be due to

- Having more hidden layers
- Doing a large number of epochs
- Large weights

Underfitting is insufficient training of network , such that it fails to recognise necessary characterstics
It can be due to

- Less Epochs
- Having less hidden layers

Early Stopping
Early stopping is process of stopping the training when model has been sufficently trained.
Usually a validation set is used to perform Early setting , A graph of validation error and training error is plotted.
When , validation error starts to increase we stop the trainign

Dropout
Dropout is process of randomly turning some weights off , usualy done by specifying probability parameters.
It may happen that some heavy weights may train more than not so heavy leading to overfitting to overfitting to avoid thsi Droupout Technique is applied.

Random restart
Sometimes we can encounter a local minima in our error function which may lead to insufficent training to solve this we randomly restart next training from random starting points

Regularization
Regularisation is way of preventing model to learn too much (i.e., learn noise too) , It works by penalising the larger weights/coefficient and shrinking them towards zero

Vanishing Gradient

Stochastic Gradient Descent
Calculating gradient and doing back propagation for whole data is takes lot of computation power. So what we do is per batch of data sets we calculate gradient for only one data and use it to update weights

Learning Data Decay
As we reach near to minimum our gradient decrease and we should take smaller steps to learn , i.e., decrease learning rate.
This is usually done to prevent model from overshooting that is crossing the minima if steps are big enough and keep oscillating there.

Momentum
Momentum is a technique used to avoid getting stuck at local minima in Error function. What we do is take a parameter **β** with value in range 0-1 and multiply it with preceding Example : $Step(n)=\beta step(n-1)+\beta^2 step(n-2)+\beta^3 step(n-3) + \beta^4 step(n-4)$ ... Even though the gradient will overshoot it ultimately would settle at minima