Activation Function

Deep Learning

* To introduce non-linear properties between the input and output
* Neural-Networks are considered Universal Function Approximators.
* We need to apply a Activation function f(x) so as to make the network more powerfull and add ability to it to learn something complex and complicated form data and represent non-linear complex arbitrary functional mappings between inputs and outputs. Hence using a non linear Activation we are able to generate non-linear mappings from inputs to outputs.
* **“Input times weights , add Bias and Activate”**
* **Types**
  + **Sigmoid or Logistic**
    - f(x) = 1 / 1 + exp(-x) .
    - Its Range is between 0 and 1.
    - It is a S — shaped curve.
    - It is easy to understand and apply but it has major reasons which have made it fall out of popularity -
    - **Disadvantages**
      * Vanishing gradient problem
      * Secondly , its output isn’t zero centred.
      * It makes the gradient updates go too far in different directions.
      * 0 < output < 1, and it makes optimisation harder.
      * Sigmoids saturate and kill gradients.
      * Sigmoids have slow convergence.
  + **Hyperbolic Tangent function**- Tanh :
    - It’s mathamatical formula is f(x) = 1 — exp(-2x) / 1 + exp(-2x).
    - Now it’s output is zero centered because its range in between -1 to 1
    - Hence optimization is easier in this method hence in practice it is always preferred over Sigmoid function .
      * But still it suffers from Vanishing gradient problem.
* **ReLu- Rectified Linear units :** I
  + It was recently proved that it had 6 times improvement in convergence from Tanh function.
  + It’s just R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x. H
  + ence as seeing the mathamatical form of this function we can see that it is very simple and efficinent .
  + Hence it avoids and rectifies vanishing gradient problem .
  + But its limitation is that it should only be used within Hidden layers of a Neural Network Model.
* Hence for **output layers we should use a Softmax** function for a Classification problem to compute the probabilites for the classes , and for **a regression problem it should simply use a linear function.**
* Another problem with ReLu is that some gradients can be fragile during training and can die.
* It can cause a weight update which will makes it never activate on any data point again. Simply saying that ReLu could result in **Dead Neurons**.
* To fix this problem another modification was introduced called Leaky ReLu to fix the problem of dying neurons. It introduces a small slope to keep the updates alive.
* We then have another variant made form both **ReLu and Leaky ReLu called Maxout function .**
* **Sigmoid functions and their combinations generally work better in the case of classifiers**
* **Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem**
* **ReLU function is a general activation function and is used in most cases these days**
* **If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice**
* **Always keep in mind that ReLU function should only be used in the hidden layers.**
* **As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn’t provide with optimum results**