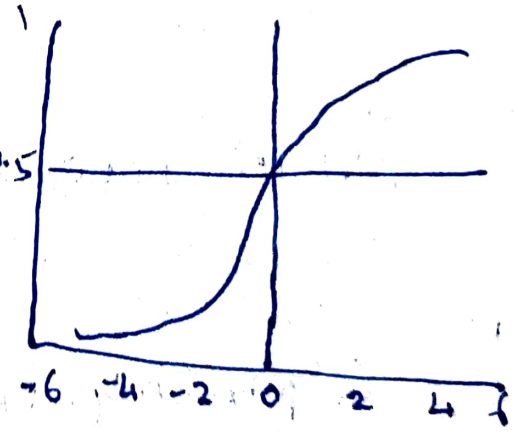
Activation Functions

Rajeev Koneru

-> Activation function ->

basically these are mathematical equations that determine the output of neural network. the function is attached to each neuron in network & it determines whether it should be activated or not based on whether each neuron's Input is relevant for model prediction.

1) Sigmoid



adv

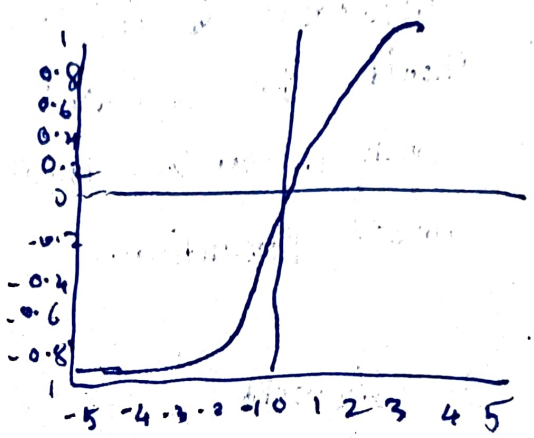
- Smooth gradient preventing Jumps in output values
- output values lies between 0 & 1

disadv

- vanishing gradient
- outputs not zero centered
- computationally expensive

2. TanH / hyperbolic Tangent

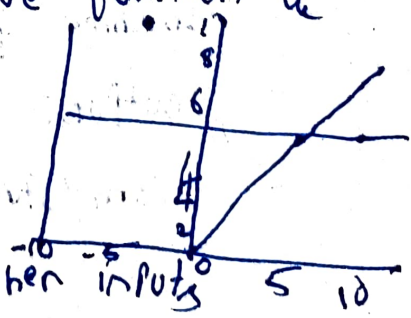
Adv - Zero centered - making it easier to model inputs that have strongly -ve, neutral, & strongly +ve values.
- otherwise like sigmoid func



disAdv
- like the sigmoid

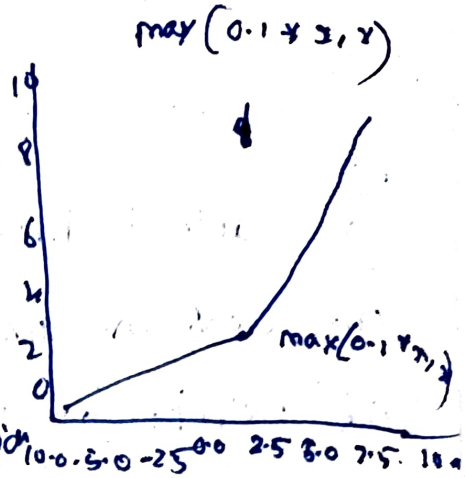
3) Rectified linear unit (ReLU)

Adv
- Computationally efficient - allows network to converge very quickly
- Non-linear - it has derivative function & allows for backpropagation.



Disadv
The Dying ReLU problem - when inputs approach zero, or -ve, the gradient of func becomes 0, the network cannot perform backpropagation & cannot learn.

4) Leaky ReLU



Adv - Prevents dying ReLU problem
This variation of ReLU has small +ve slope in -ve area
so it does enable back propagation even for -ve input values
- otherwise like ReLU

Dis Adv - Results not consistent - it does not provide consistent predictions for -ve input values

5) Parametric ReLU

Adv

$$f(x) = \max(\alpha x, x)$$

- Allows the -ve slope to be learned
unlike Leaky ReLU, this function provides the slope of -ve part of function as argument.
i.e it's possible to perform back propagation & learn most appropriate value of α
- otherwise like ReLU

Dis Adv

- may perform differently for different

6) Softmax

$$S(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{k} e^{x_i}}, \quad j = 1, 2 \ldots k$$

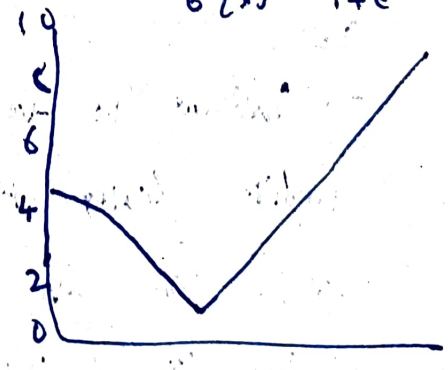
Adv

- able to handle multiple classes only one class in other activation functions. normalizes the outputs for each class b/w 0 & 1 & divides by their sum giving probability of input value being in a specific class.

- useful for output neurons - it is used only for output layer that need to classify inputs into multiple categories.

7) Swish

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

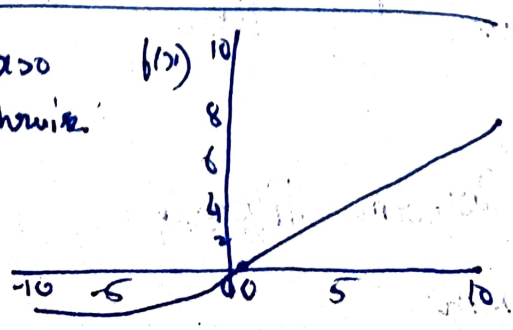


- it is self gated activation function.

- it performs better than Relu with similar level of computation efficiency.

8) ELU (Exponential linear units)

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$



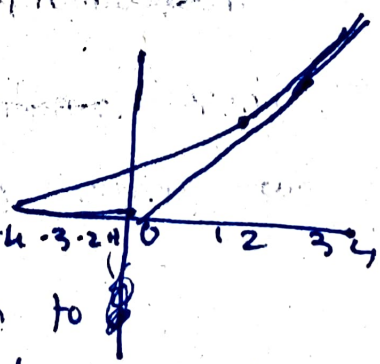
- it has all the advantage of ReLU
  - no dead ReLU issues
  - The mean of output is close to 0 zero-centered.
- one tiny problem is it slightly more computationally intensive.

9) Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

- it generalizes the ReLU & leaky version.
- both ReLU & leaky ReLU are special case of this form.
  - it is learnable Activation function.
- it added a layer of activation. which contain k. it adds k neurons & outputs largest value.

10) Softplus



- the softplus is similar to ReLU, but it is relatively smooth. it is unilateral suppression like ReLU. it has wide acceptance range (0, + inf)

$$f(x) = \ln(1 + \exp x)$$

Loss Functions

1) L1 & L2 Loss

- these are used to minimize the error.

L1 known as least Absolute Deviations.

L2 known as least square errors.

- L1 used to minimize error which is sum of all absolute differences b/w true & predicted value.

$$L_1 = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

- L2 minimizes error which is sum of all squared differences b/w true & predicted value

$$L_2 = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

- disadvantage is with outliers.
- these ~~components~~ points will account for main component of loss.

Huber

2)

- it is often used in regression problems. compared with L2 loss, huber loss is less sensitive to outliers.
- because its residual is too large.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

δ is set parameter, y - real value

$f(x)$ - predicted value.

3) hinge loss

- used often for binary classification problems, such as ground true: $t = 1$ or -1, predicted value $y = wx + b$
- score of correct category should be greater than sum of scores of all incorrect categories by some safety margin. most notably for SVM.

$$\text{SumLoss} = \sum_{j \neq y_i} \max(0, s_j - s_{y_j} + 1)$$

4) ~~Sigmoid~~ Cross-entropy loss

- it is most common setting for classification problems cross-entropy loss ~~loss~~ as predicted probability diverge is from actual ~~val~~ label

cross entropy loss

$$= -(y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i))$$

when actual label is 1 ($y(i)=1$) second half of function disappears whereas in case actual label is 0 ($y(i)=0$) 1st half is dropped off.

5) Sigmoid cross entropy loss

scores = x*w+b, entering this value into the sigmoid function (or) compressing value range (0,1)

- sigmoid func smooths predicted value. which makes the predicted value of sigmoid-ce from the label loss growth is not so steep.

6) Softmax / cross-entropy loss

---

it softmax func converts a set of fraition vectors into corresponding probability vectors.

- ~~Soft~~ it uses softmax function to convert the score vector into probability vector.

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

$p(x)$ - probality that classifiction x is correct
& P can only be 0 or 1

$q(x)$ ~~Probab~~ prediction probality that x category is correct classification.

value range (0,1)

$$L_i = -\log\left(\frac{e^{b_{yi}}}{\sum_j e^{b_j}}\right)$$

or

$$L_i = -b_{yi} + \log \sum_j e^{b_j}$$

b_j is score of all possible categories.
& b_y is score of ground truth class

7) mean Bias error

- This is same as MSE with only difference that we don't take absolute values clearly there's a need for caution as +ve & -ve errors could cancel each other out. Although less accurate in practice, it could determine if model has +ve bias or -ve

$$MBE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)}{n}$$

8) KL - Divergence

- it is a measure of how Probality distribution differs from another distribution. A KL-d of zero indicates that distributions are identical.

$$D_{KL}(P||Q) = \begin{cases} -\sum_x P(x) \cdot \log \frac{Q(x)}{P(x)} = \sum_x P(x) \cdot \log \frac{P(x)}{Q(x)} & \text{for discrete distribution} \\ -\int P(x) \cdot \log \frac{Q(x)}{P(x)} \cdot dx = \int P(x) \cdot \log \frac{P(x)}{Q(x)} \cdot dx & \text{for continuous distribution} \end{cases}$$

= expectation of logarithmic difference b/w P & Q wrt P