DEEP LEARNING

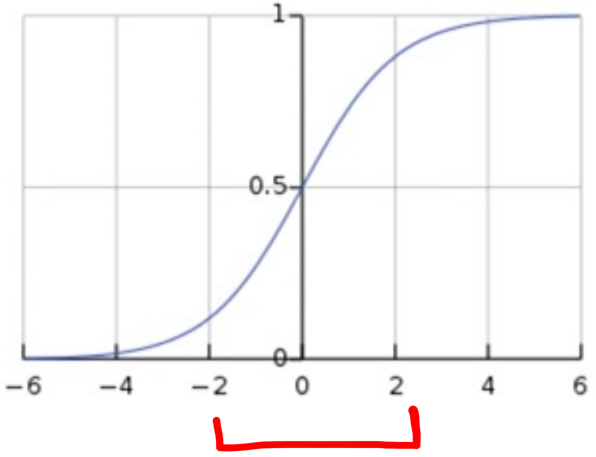
Trainer : Dr. Darshan Ingle

Revisiting Activation Functions

Sigmoid [0-1]
- makes NNs decision boundary non-linear.

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

$$= \frac{1}{1+e^{-(mx+c)}}$$



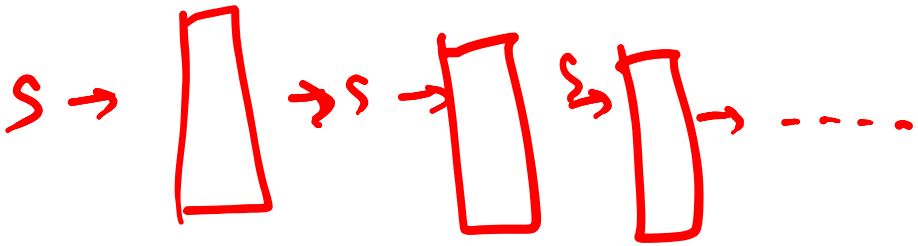
Standardization

1 ———————— 5 million

0 ———————— 0.0001

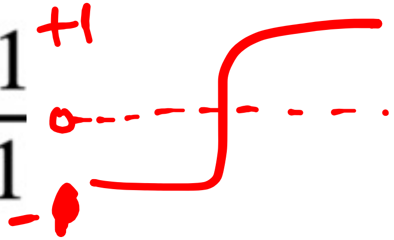
we dont want i/p in different range . ∴ we prefer to center them around 0

$\sigma = [0 - 1]$ → middle = 0.5] → its basically centered around 0.5



Hyperbolic tangent (tanh)

$$\tanh(a) = \frac{\exp(2a)-1}{\exp(2a)+1}$$



Unlike sigmoid, where the data is centered around 0.5,

In tanh [-1,+1], the data is centered around 0.

Still more problems

tanh is little better than sigmoid.

Still, both r problematic.

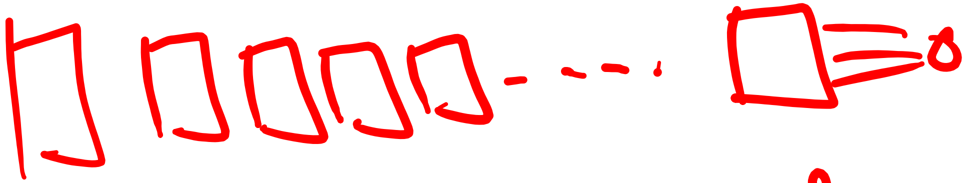
Problem: Vanishing Gradient.

गायब Slope

Vanishing gradient problem

1980-2000, researchers were not able to create a Deep NL.

HL7!



Why? They used Sigmoid in each & every neuron that we are using.

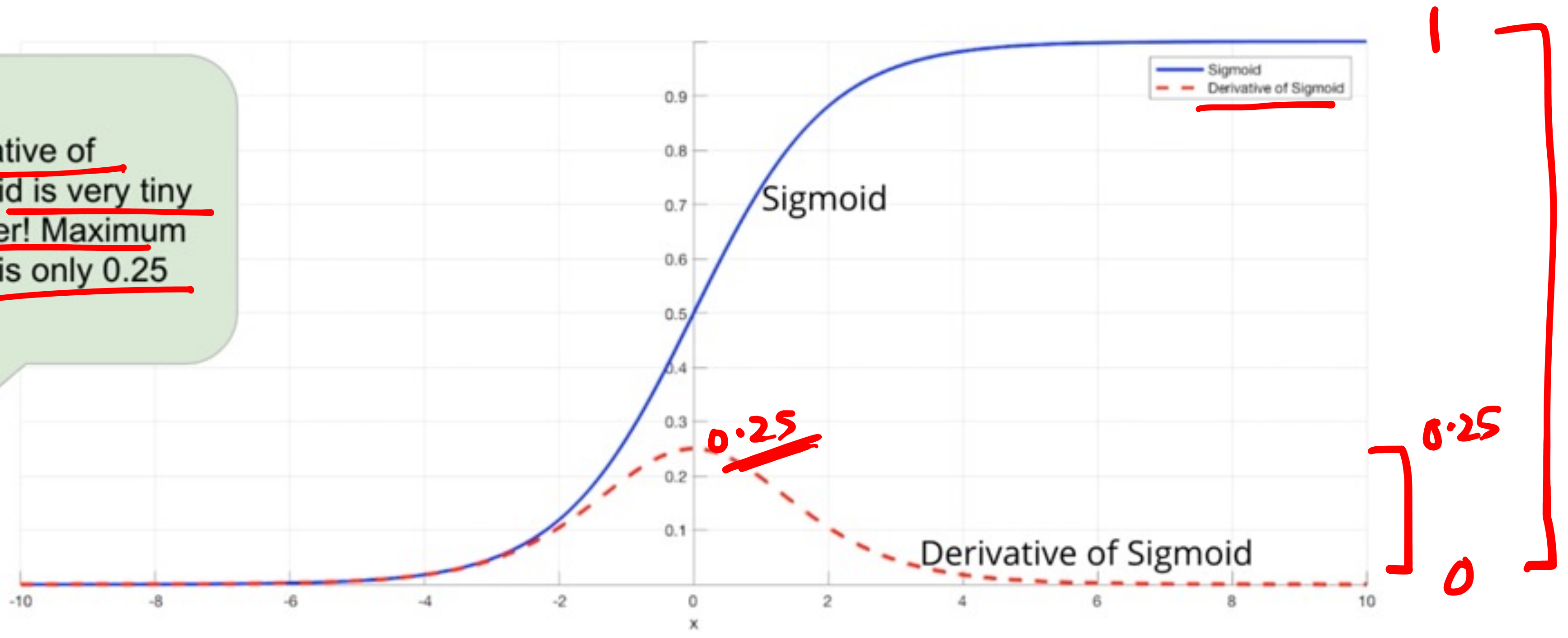
Bcz of this all of them were facing V.G. Prob.

$\sigma = [0-1]$

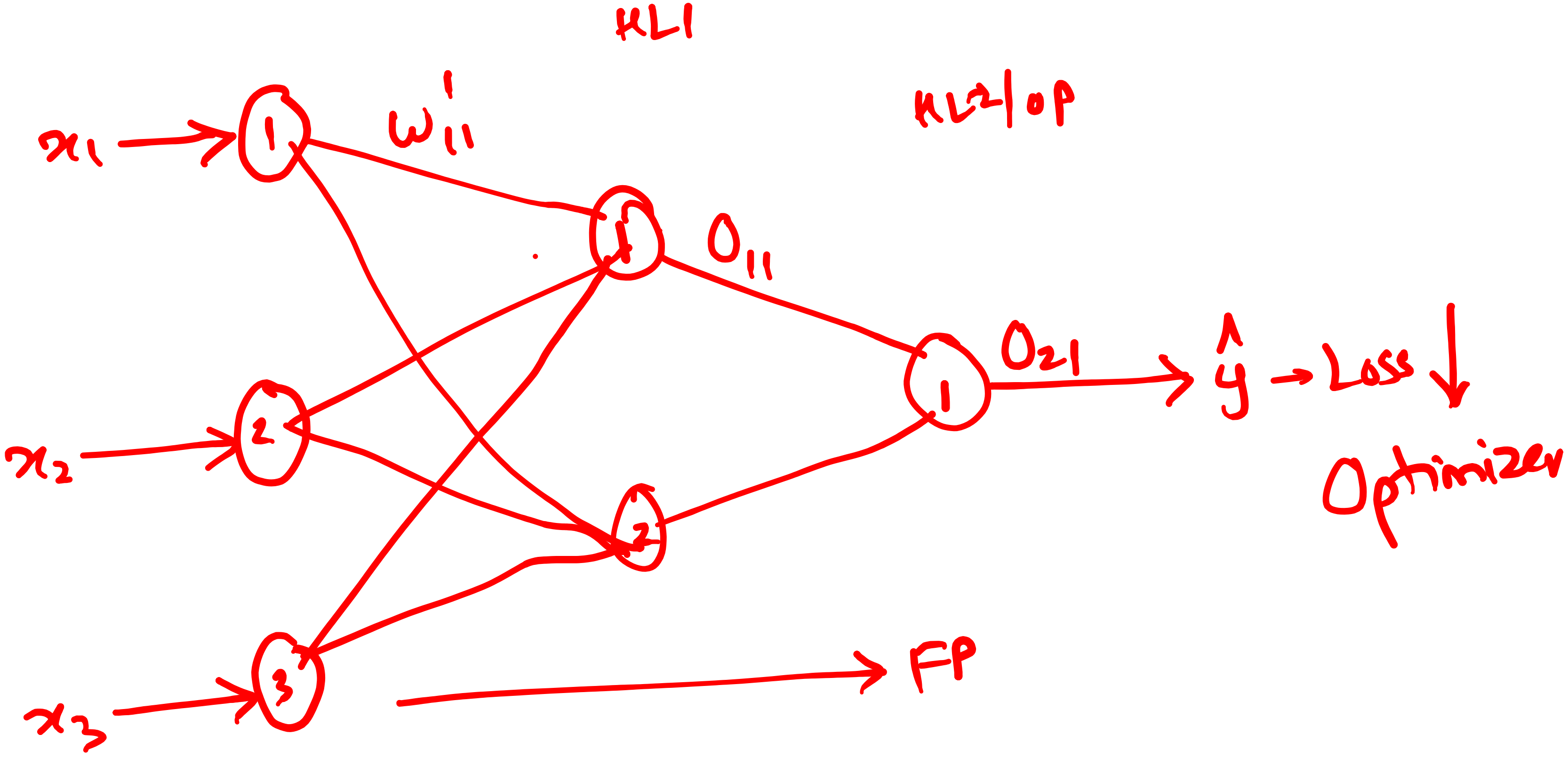
Derivative of $\sigma = [0-0.25]$

Vanishing gradient problem

Derivative of sigmoid is very tiny number! Maximum value is only 0.25



Vanishing gradient problem



$\sigma = [0-1]$

$Der.\sigma = [0-0.25]$

0.01 - 0.04

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

$\frac{\partial L}{\partial w^1_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial w^1_{11}}$

$[0-.25]\ [0-.25]$

$= 0.20 \times 0.02$

$= 0.04$

$w^1_{11\,new} = w^1_{11\,old} - 0.004$

$= 75 - 0.004$

$\boxed{w^1_{11\,new} \approx 75}$

Vanishing gradient problem

tanh = [-1, +1]

Der. of tanh = [0 - 1]

Soln: ReLU func. for hidden layers

& Sigmoid ~~for~~ for o/p. Layer.

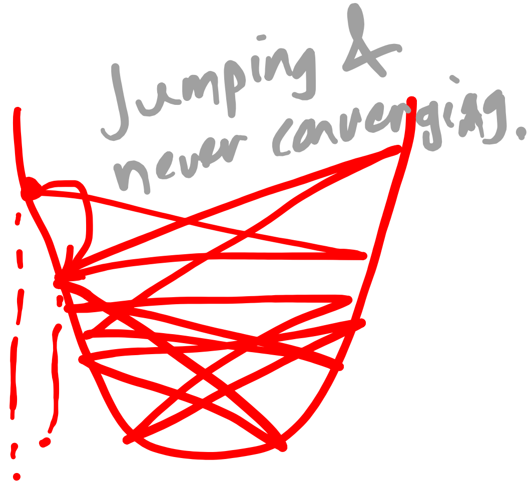
Vanishing gradient problem

Vanishing gradient problem

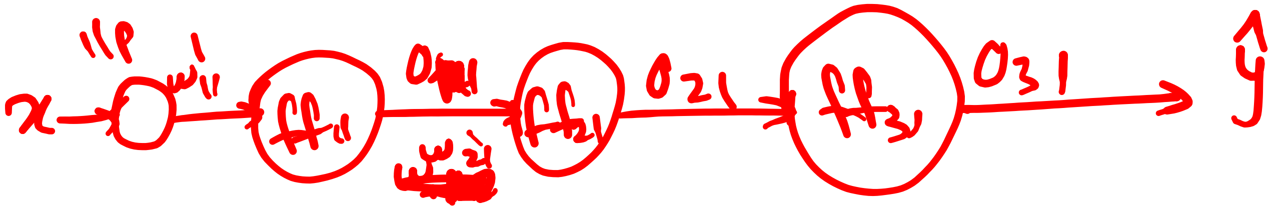
Vanishing gradient problem

Exploding Gradient Descent

This is mainly caused bcz of weights.



Exploding Gradient Descent



$$0 \le \phi(z) \le 0.25$$

$$\frac{\partial O_{21}}{\partial O_{11}} = \underbrace{\frac{\partial \phi(z)}{\partial z}}_{[0-0.25]} \cdot \frac{\partial z}{\partial O_{11}}$$

$$* \frac{\partial}{\partial O_{11}}(w_{21} \cdot O_{11} + b_2)$$

$$= [0-0.25] * w_{21}$$

$$= 0.25 * 500 = \underline{\underline{125}}$$

0.01×2500000

$$w'_{11\,new} = w'_{11\,old} - \eta \cdot \frac{dL}{dw'_{11}}$$

$200 \times 125 \times 100$

$$\frac{dL}{dw'_{11}} = \frac{\partial O_{31}}{\partial O_{21}} \cdot \boxed{\frac{\partial O_{21}}{\partial O_{11}}} \cdot \frac{\partial O_{11}}{\partial w'_{11}}$$

$$O_{21} = \phi(z) \qquad \frac{1}{1+e^{-z}}$$

$$\boxed{z = w_{21} \cdot O_{11} + b_2}$$

Exploding Gradient Descent

Dropout in MNN

Multilayer → we always face overfitting i.e. model starts to memorize the values.

In short, we face a high Variance problem.

To solve this: (1) Regularization: Ridge & Lasso.

(2) DropOut: PhD Thesis — 2014 by Nitish Shrivastav & Jeoffrey Hinton.

Dropout in MNN

Predicti we remove all dropouts.

∴ all neurons are activated with all links

enabled &

we do just one additional step i.e. Weight × Prob.

w × p

How to Select p i.e. dropout?

Hyperparameter Optimization.

If NN is overfitting, p-val. to be a little higher i.e. atleast greater than 0.5 .

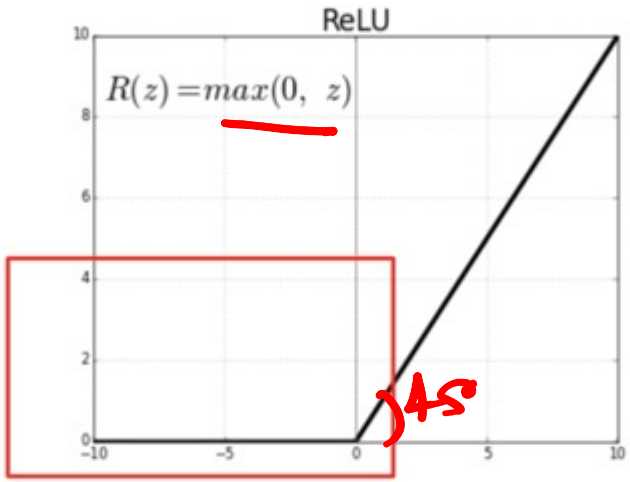
Dropout in MNN

ReLU

max(0,z)

It has +ve side slope.

Slope is always 45°.



Derivative of ReLU is always 1 for any +ve value

Der. of ReLU is a prob. for neg. side as angle is 0.

& we cannot find derivative of 0 ∵ 0 is not differentiable.

Simply put: 1 if z > 0

0 if z < 0

ReLU

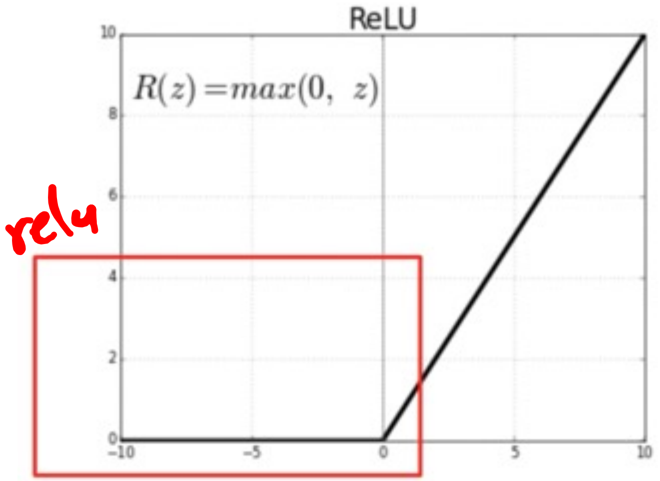
- Now if we apply this in the chain rule, works perfectly.
- But if any derivative gives value as 0, chain rule will straightaway make the calculation value as 0 thereby making it a dead neuron or a dead activation function.

Relu: Weight initialized using : He Normal or He Uniform.

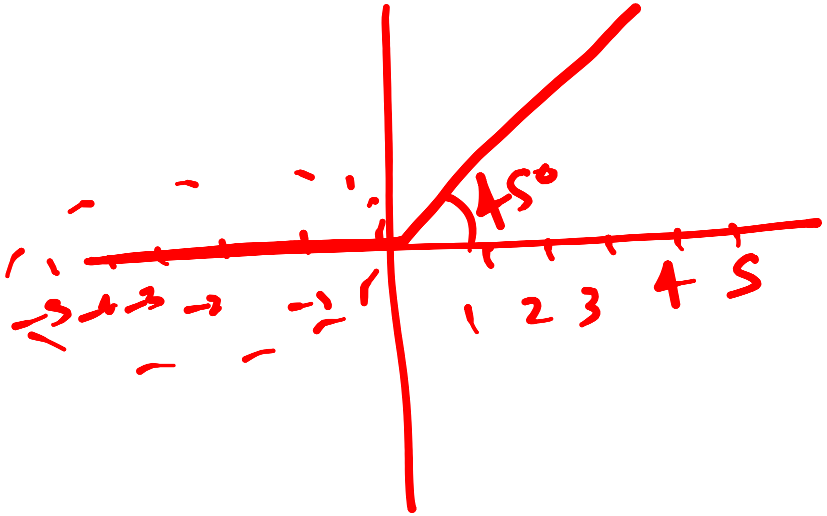
Sigmoid: Weight initializer : Glorot Uniform

- Therefore we go for Leaky ReLU
- With ReLU, always use weight initializer as he normal or he uniform.
- With Sigmoid, use weight initializer as glorot_uniform

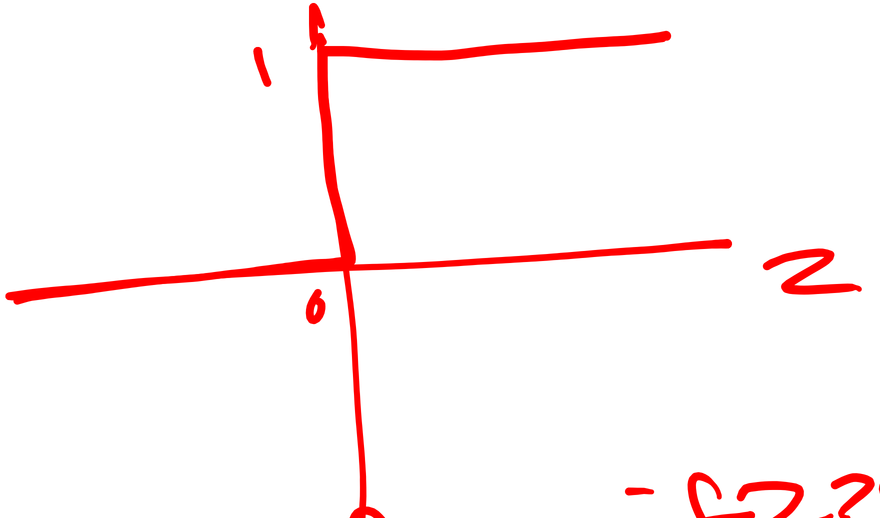
Leaky relu



ReLU



Der. of ReLU



$$\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z < 0 \end{cases}$$

$$W_{old} = W_{new} - \eta \cdot \frac{dL}{dw}$$

$\square \cdot \square \cdot \square$

$1 \times 1 \times 1$

$1 \times 0 \times 1$

No Vanishing Gradient Prob

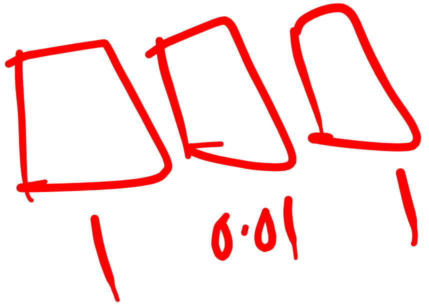
But -ve side is an issue.

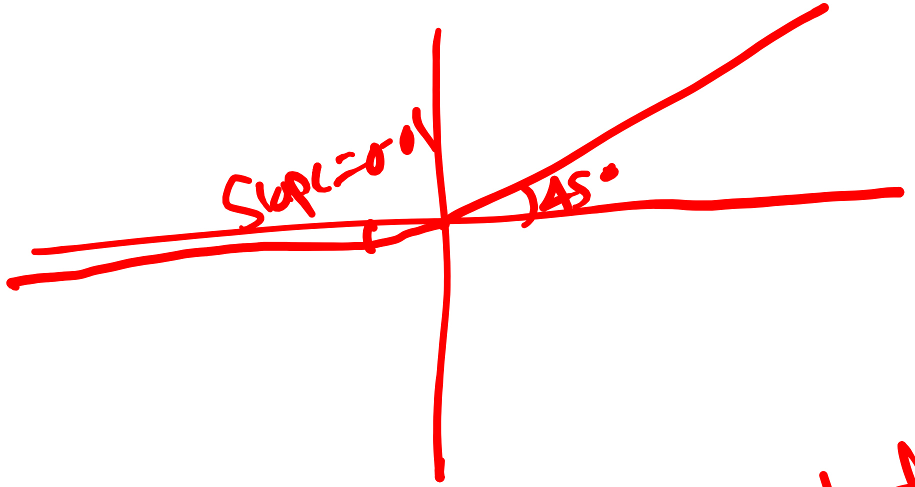
$\therefore$ Dead neuron / dead AF.

Soln: Leaky relu.

ReLU

Leaky ReLU



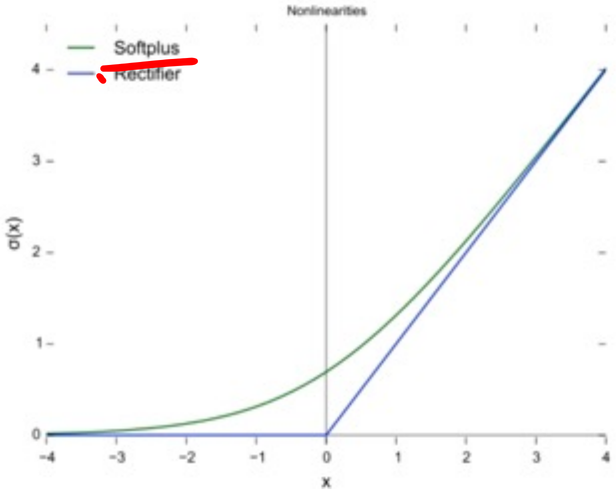


Addition of some val. for -ve, der. val. doesn't become 0.

Softplus Activation Function

- Another option which is very similar is the soft plus activation which because you're taking the log of the exponent looks very linear when the input is reasonably large.

$$f(x) = \log(1 + e^x)$$



- But for both of these previous activation functions, there is the vanishing gradient on the left side but we've established that it's not so much of a problem since we know that the ~~real you~~ already works and it has gradients that are equal to zero also.

ReLU

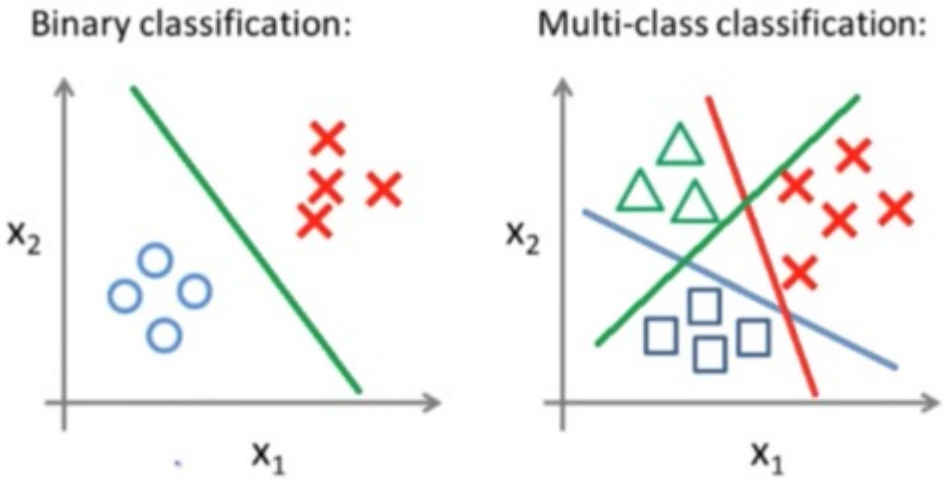
Softplus Activation Function

- Although we initially stated that we would like the inputs at each layer to be centered around zero we can see that the ReLU and Softplus do not accomplish this.
- For soft plus and the ReLU the minimum value is zero while the maximum value is infinity.
- This definitely means they won't be centered around zero. So is the ~~real~~ ~~yo~~u not a good choice in the end?
- relu Now despite all this work to find alternatives to the value activation these days most people still use the ~~value~~ as a reasonable default choice.
- It works well and sometimes relu you'll find that using other alternatives such as the leaky ReLU or the ELU offer no benefit. But Sometimes they do, which is why you always have to experiment for yourself.

Softplus Activation Function

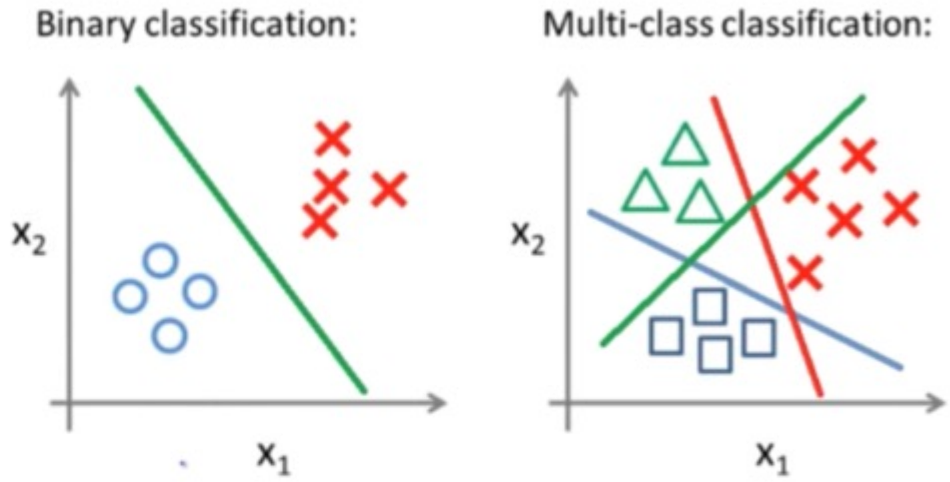
- My motto which a lot of my students are tired of hearing by this point is that machine learning is experimentation and not philosophy.
- Never use your mind to try and predict the outcome of a computer program.
- If you have a computer that is always the suboptimal course of action why not simply run the computer program with a computer.
- Your mind is not suitable for running computer programs but computers are therefore follow the rule.
- Don't use philosophy use experimentation.

Multiclass Classification



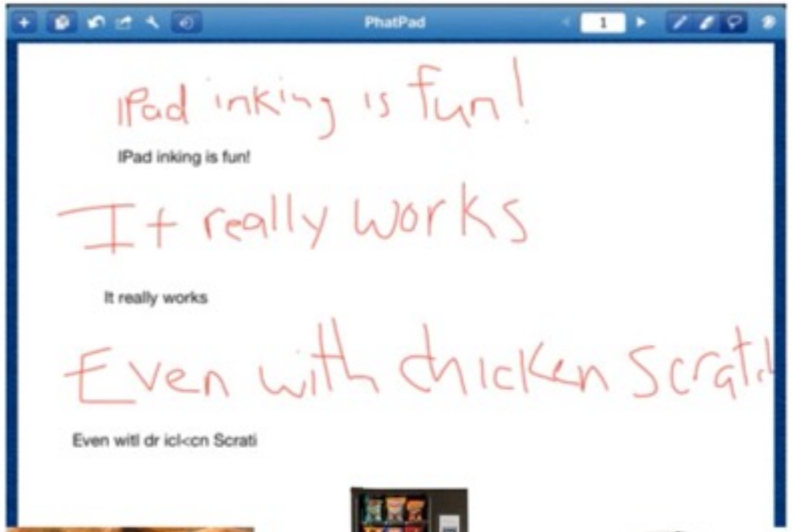
- We saw that for a binary classification, we use sigmoid at the output.
- We replaced sigmoid with ReLUs in the hidden layer
- However, for output, sigmoid is still the right choice for binary classification
- Applications of Binary Classification:
  - Disease vs No Disease
  - Fraud vs No Fraud
  - Click vs No Click
  - Accept Friend Request

Multiclass Classification



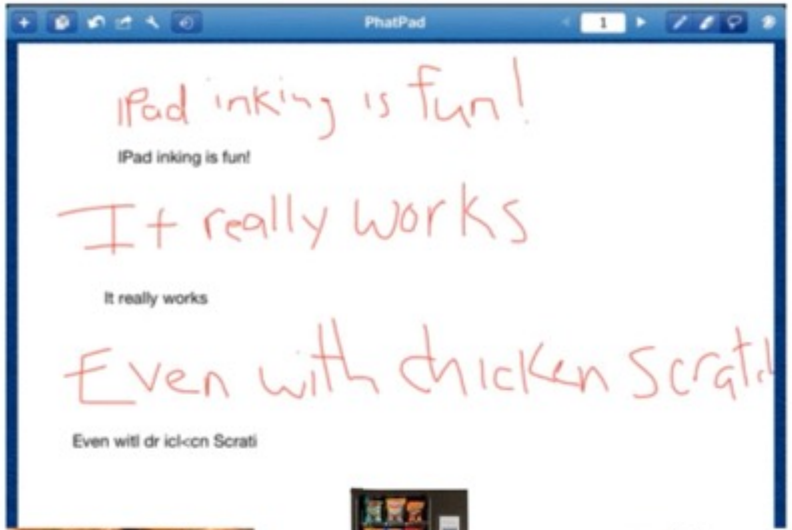
- But there are situations that binary classification cannot handle such as when we might have multiple categorical outcomes.

Multiclass Classification



Multiclass Classification

1. OCR : a-z 0-9
   26 + 10 = 36 possibilities.
2. Speech Recognition.
3. Image Classification



Using Softmax in Tensorflow

- Well just like most of the other functions we've applied so far it's very simple.
- We just pass in the string 'softmax'.
- In other words the only requirement is that you spell it correctly.
- Of course you can always implement the softmax yourself but in Tensorflow, there's no need.
- As a side note, the softmax is considered an activation function but unlike the ReLU, sigmoid and tanh, it is not meant for hidden layer activations.
- If you want to try using the softmax as a hidden layer activation, you are most welcome to but you'll generally find that it doesn't work that well.
- So the softmax is technically an activation function but we normally only use it when we're trying to get an output probability from a vector of activation values such as at the end of a neural network.

```
tf.keras.layers.Dense(K, activation='softmax')
```