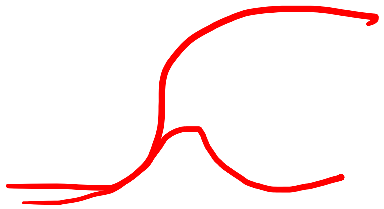
Vanishing gradient problem

Yesterday → Caused bcz of Sigmoid → 0 to 1



$$\frac{\partial \sigma(z)}{\partial z} = 0 \text{ to } 0.25$$

$$\boxed{\eta = 1}$$

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

0.0004

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

0.2 · 0.1 · 0.02 = 0.0004

v.v. small.

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

Exploding Gradient Descent

(RELU is not invented yet) we only as researchers know SIGMOID.

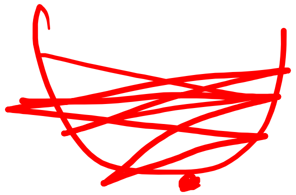
Culprit: $\boxed{\text{WEIGHTS.}}$

* In EX. Gr. Pb, the larger assigned weight when mult. by derivative of Sigmoid returns a v. big No.

assume: 450 / 600

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w_{old}}$$

→ chain rule



How?

small whole / small whole / big whole

$$\left.\begin{array}{l} 2 \times 4 = 8 \\ 3 \times 12 = 36 \end{array}\right\} \text{Exploding Gradient}$$

Vanishing Gradient

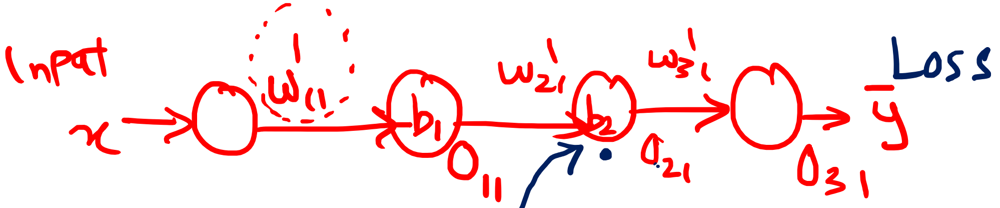
$$\left\{\begin{array}{l} 0.2 \times 0.4 = \underline{0.08} \\ 0.3 \times 0.12 = 0.036 \end{array}\right.$$

small point / small point / more smaller no.

Exploding Gradient Descent

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

$\eta = 1$



$z = w \cdot x + b$

$z = w'_{21} \cdot O_{11} + b_2$

$$w'_{11_{new}} = w'_{11_{old}} - \eta \frac{\partial L}{\partial w'_{11_{old}}}$$

large

$w'_{11\,new} \approx$ v.diff. $w'_{11\,old}$

$$\frac{\partial L}{\partial w'_{11\,old}} = \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial w'_{11}}$$

$150 * 100 * 75$

} v.v. huge no

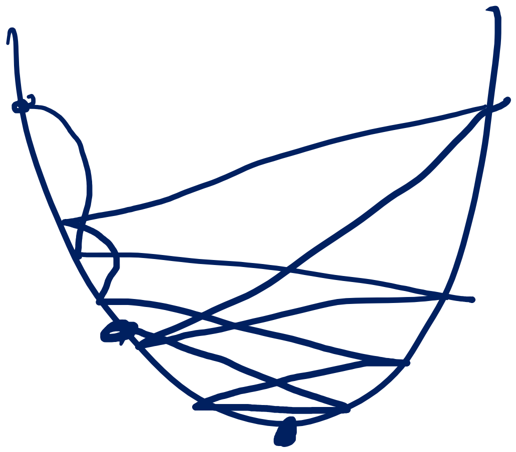
$$\frac{\partial O_{21}}{\partial O_{11}} = \frac{\partial \phi(z)}{\partial z} \cdot \frac{\partial z}{\partial O_{11}}$$

$$0 \text{ to } 0.25 * \frac{\partial}{\partial O_{11}}(w'_{21} \cdot O_{11} + b_2)$$

$= (0 \text{ to } 0.25) * w'_{21}$

$= (0 \text{ to } 0.25) *$ large no

$= 0.25 * 400 = 100$



Weights

$\Downarrow$

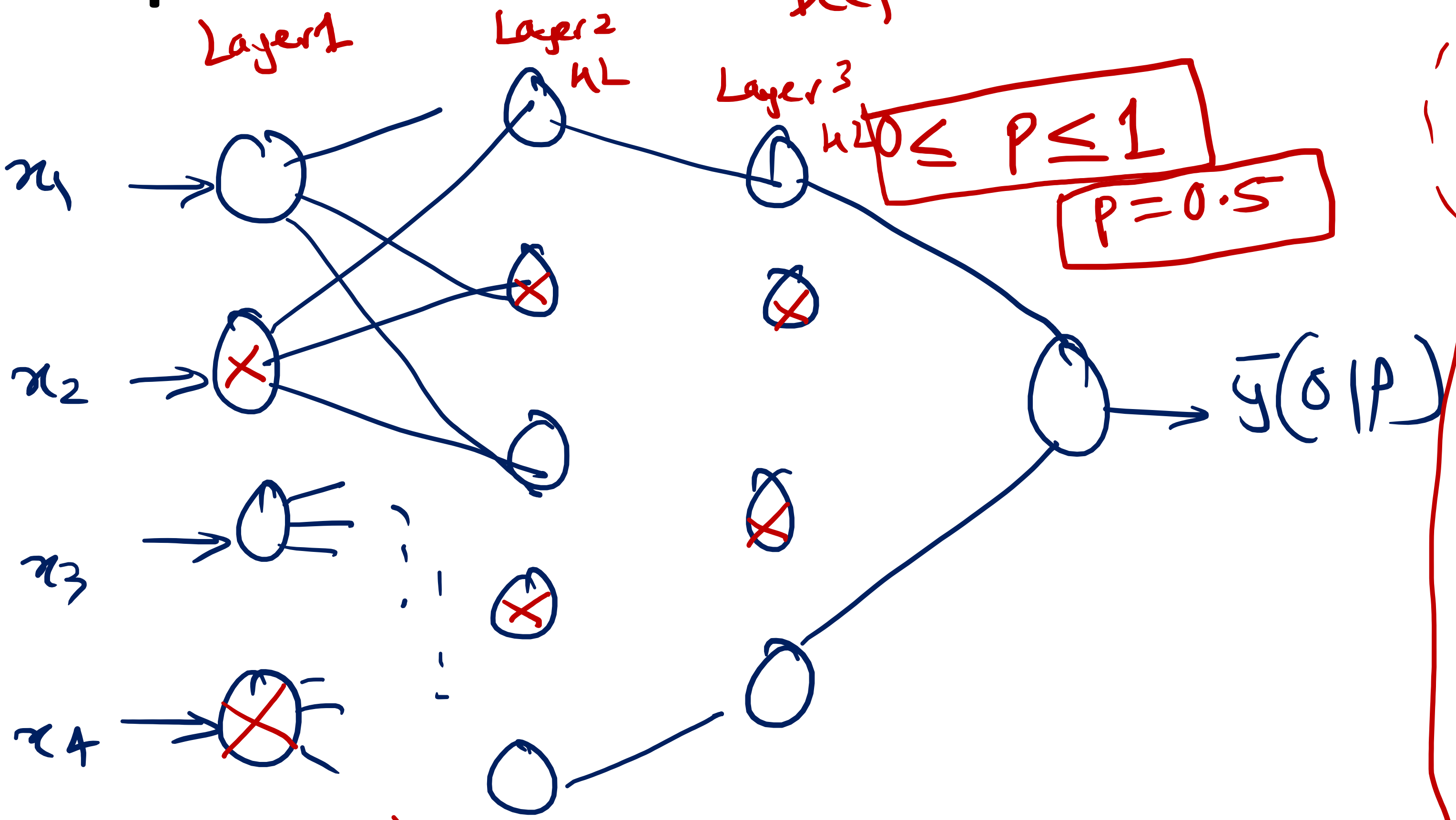
Exploding Gradient Problem

Exploding Gradient Descent

Dropout in MNN

more Deep ⇒ Overfitting

M.L. ⇒ Random Forest is used for feature selection.



$\boxed{\text{OVERFITTING}}$

→ Decision Tree

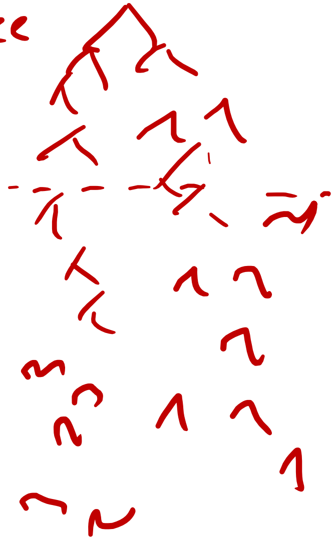
↳ Overfitting

H. Variance

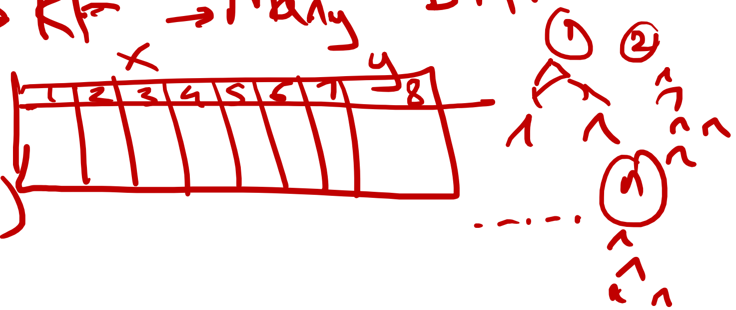
Overcome:

max_depth = 3

Hyperparameter



RF → Many D.T.



Solve: 2 ways: ① Regularization: L_1, L_2

② Dropout: Outcome of a PhD student (5yrs)

Nitish Shrivastav & Jeoffery Hinton (2014)

Dropout in MNN

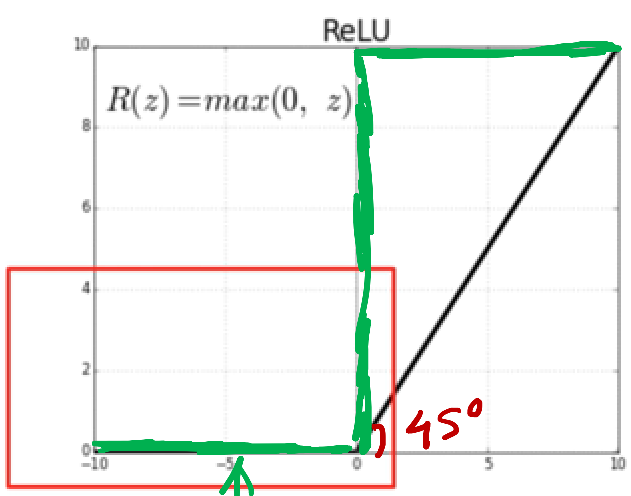
How to select p-value? → Hyperparameter Optimization

(K. Fold C.V.)

Dropout in MNN

ReLU

$= max(0, z)$



Derivative of ReLU

Calculus: Der. of 0 is not differentiable

Simply Saying: 1 if $z > 0$

0 if $z < 0$

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w} \qquad \frac{\partial}{\partial} \cdot \frac{\partial}{\partial} \cdot \frac{\partial}{\partial} = 1$$

$$= 1 \qquad 1 \cdot 1 \cdot 1$$

- Vanishing Gradient Prob. is thus solved by ReLU

ReLU

$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w_{old}}$ ← DEAD NEURON/deactived

$\Rightarrow \frac{\partial}{\partial} \cdot \frac{\partial}{\partial} \cdot \frac{\partial}{\partial}$

$0 \cdot 1 \cdot 1$

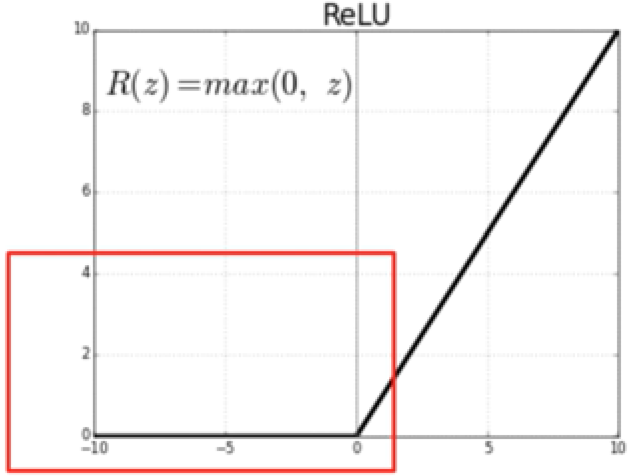
$\boxed{w_{new} = w_{old}}$

- 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.
- 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

Der. of ReLU

1 if $z > 0$

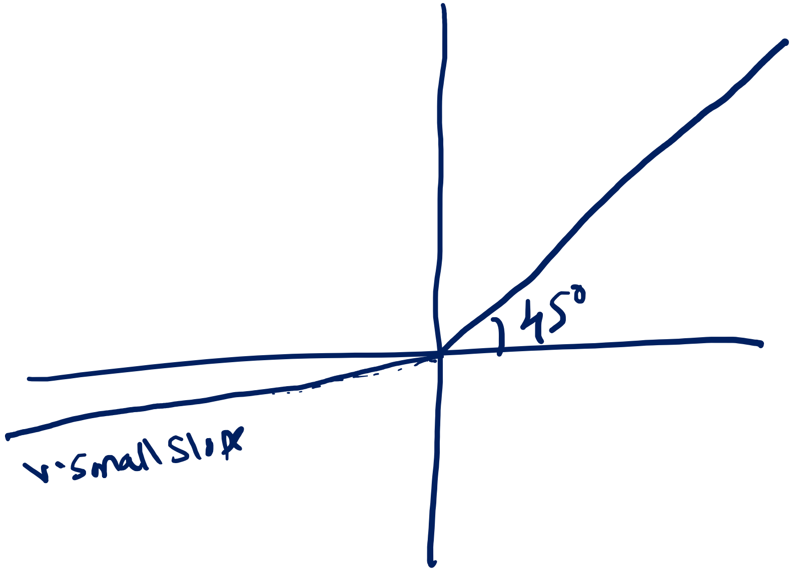
0 if $z < 0$



ReLU

ReLU

Leaky ReLU



$$\frac{\partial}{\partial} \cdot \frac{\partial}{\partial} \cdot \frac{\partial}{\partial}$$

$$0.01 \cdot 1 \cdot 1$$

Der. of Leaky

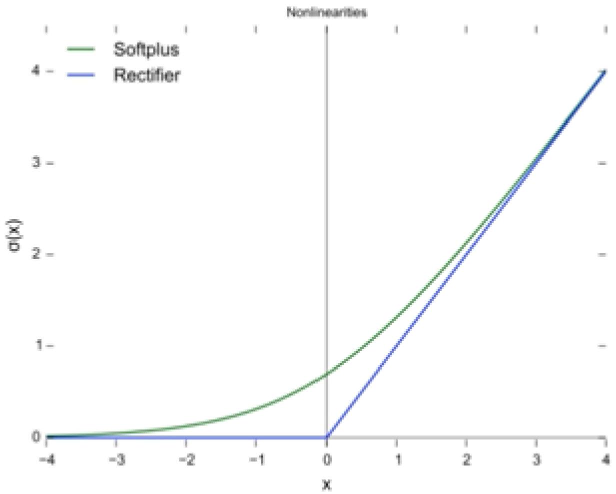
$$\text{ReLU}: \begin{cases} 1 & \text{if } z > 0 \\ 0.01(z) & \text{if } z < 0 \end{cases}$$

add some small value for -ve derivate

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.

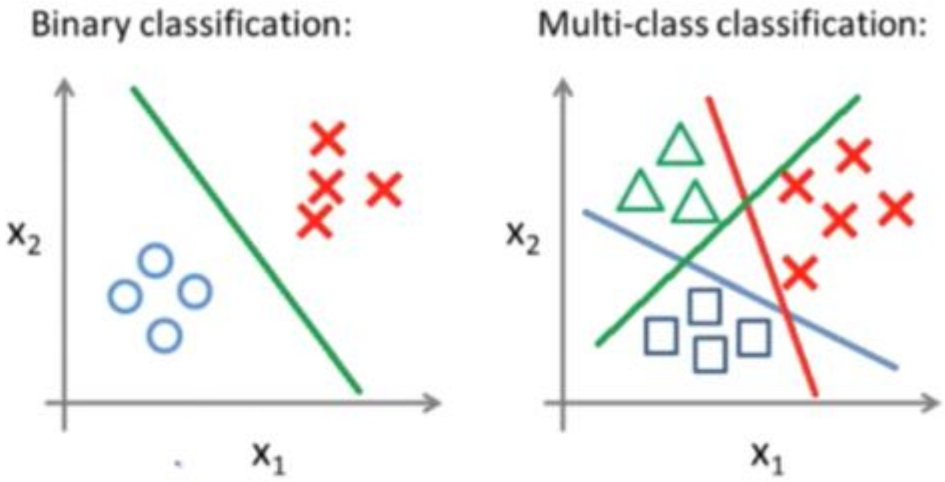
Softplus Activation Function

- 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~~ ReLU ~~you~~ not a good choice in the end?
- Now despite all this work to find alternatives to the value activation these days most people still use the ~~value~~ ReLU as a reasonable default choice.
- It works well and sometimes 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

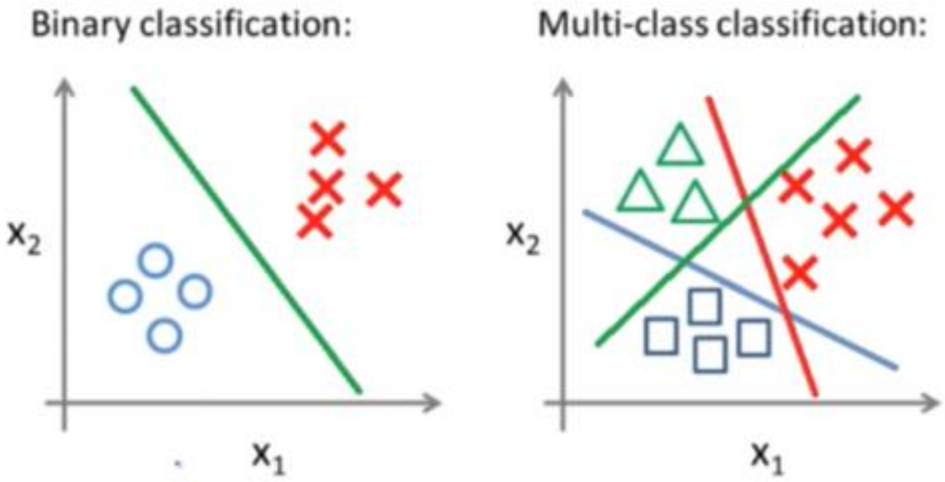
- My motto which a lot of my students are tired of hearing by this point is that machine learning is experimentation and not philosophy.
- Never use your mind to try and predict the outcome of a computer program.
- If you have a computer that is always the suboptimal course of action why not simply run 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

Example :

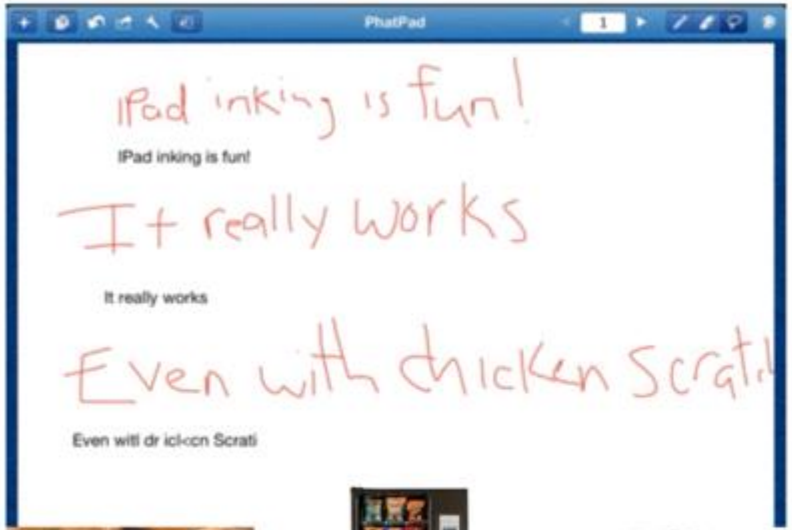
① Character Recognition / Handwriting Recognition →

② Speech Recognition: Each word counts as a separate category.

③ Image Classification: ImageNet (million of images of ~~th~~ thousands of categories like bagels, vending m/c, soccerball, etc.



Multiclass 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')
```

Task Summary

Task	A.F
Binary	Sigmoid
Multi-class	Softmax
Regression	None/Linear

The Model Type doesn't matter

Linear Reg.

Dense

Binary Logistic Reg.

Dense	Sigmoid

Multi-class Logistic Reg.

Dense	Softmax

ANN. Reg.

Dense	Dense

ANN. Binary Class.

Dense	Dense	Sigmoid

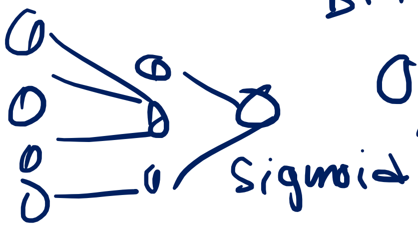
ANN Multi-class Classification

Dense	Dense	Softmax

Softmax is more general

Softmax — Multiclass

Can it be also used for Binary Classification?

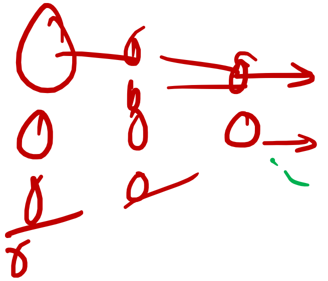


Binary

0/1 (Sigmoid at O/P)

Can I use Softmax here? Softmax = Multi-class (greater than 1)

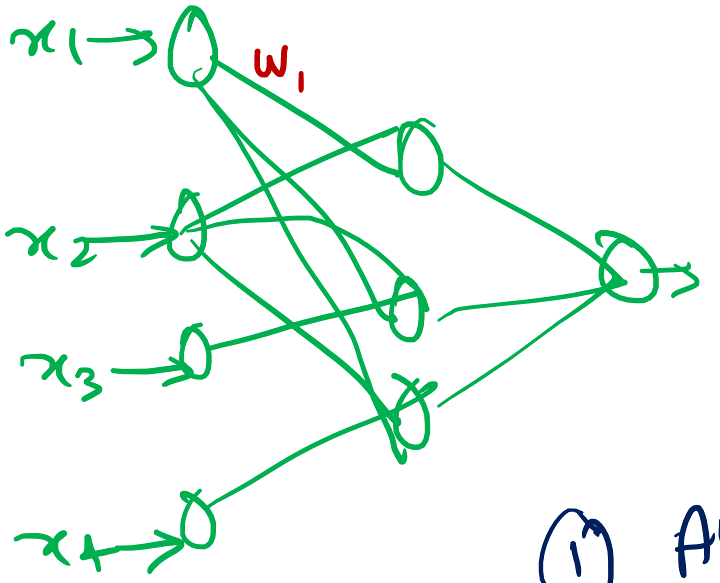
Yes (Class K = 2)



0 ✓ off ---- $P(y=0|x)$

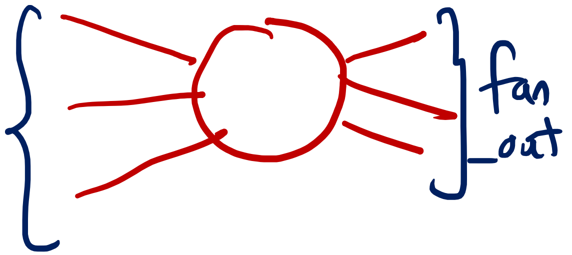
1 off ✓ ---- $P(y=1|x)$

Weight Initialization



Research Paper Convention/ Notation :

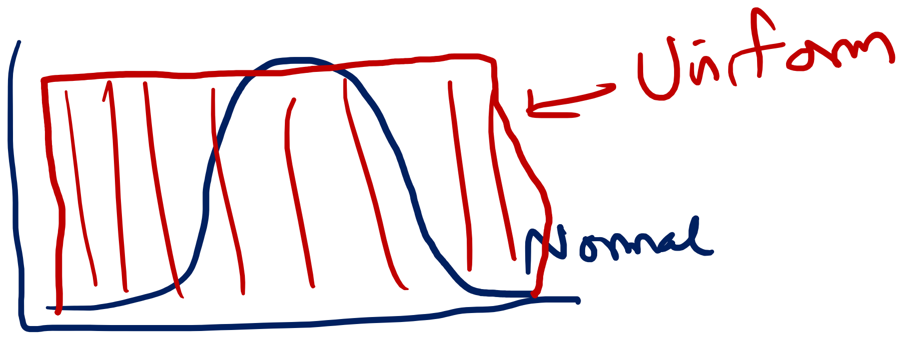
fan_in



Important Points:

1. Always initialize weights to a small value.
2. Don't keep all weights same.
3. All weights should have good amount of variance.

Weight Initialization



① Uniform Distribution:

$$W_{ij} \sim \text{Uniform}\left[\overset{a}{\frac{-1}{\sqrt{\text{fan\_in}}}}, \overset{b}{\frac{1}{\sqrt{\text{fan\_in}}}}\right]$$

Weight Initialization

② Xavier | Gorat : Use Sigmoid

Two Types:

ⓐ Xavier | Gorat Normal

$$w_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{(fan\_in + fan\_out)}}$$

ⓑ Xavier | Gorat Uniform

$$w_{ij} \sim U(0, \sigma)$$

$$\sigma = \left[\frac{-\sqrt{6}}{\sqrt{fan\_in + fan\_out}}, \frac{\sqrt{6}}{\sqrt{fan\_in + fan\_out}} \right]$$

Weight Initialization

3) He init: (ReLU):

Two Types:

a) He Uniform

$$w_{ij} \sim U\left[-\sqrt{\frac{6}{fan\_in}}, \sqrt{\frac{6}{fan_{out}}}\right]$$

b) He Normal

$$w_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{fan\_in}}$$