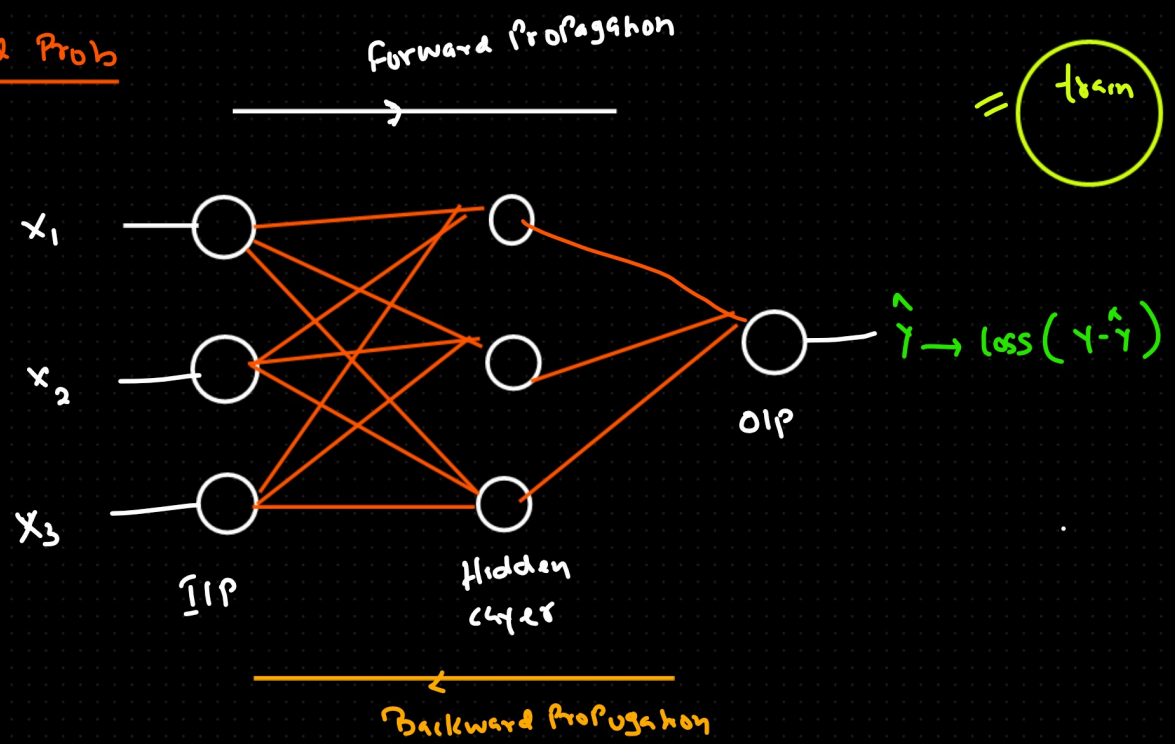
1. Vanishing grad | Exploding grad. ✓
2. Data Scaling → Batch Norm. ✓
3. Drop out ✓
4. Regularization ✓
5. Weight initialization ✓

** Start from how to improve the performance of the NN

Vanishing grad Prob

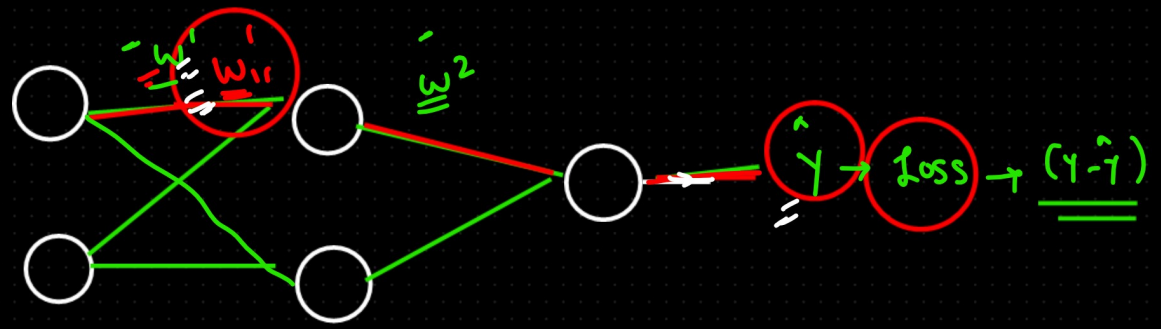


Gradient will be very very small → there won't be any sort of a weight update

Vanishing

Vanishing gradient

Neural Network ⇒ Layers Nodes ⇒ Multiple ⇒ Deep Neural Network



$$w_n = w_o - \eta \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial I} \times \frac{\partial I}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}^1}$$

= Calculation (I/P x weights) = O/P ⇒ Back Propagation

[0-1]

(Derivation / Partial Derivation)

Sigmoid → [0-0.25]

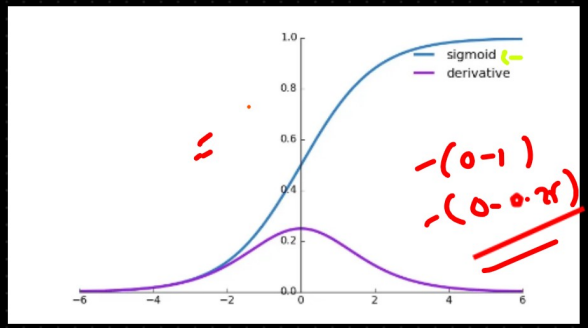
1

Neural Net → Deep Neural Net ⇒ Vanishing grad. ⇐ wrong activation

2

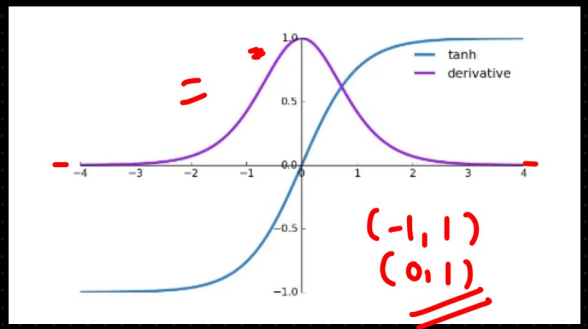
1. sigmoid
2. tanh

there is no update in the weight



-(0-1)
-(0-0.25)

Sigmoid



(-1, 1)
(0, 1)

tanh.

$0.1 \times 0.1 \times 0.1 \times 0.1 = 0.0001$

= Deep Neural Network ⇒ Weight ⇒ Forward Propagation ⇒ (loss)

Sigmoid / tanh

Update the weight

$W_n = W_o - \eta \frac{\partial L}{\partial W_o}$

Chain rule ← (0 – 0.25)

0.01, 0.1, 0.001, 0.2, 0.15

Small × Small × Small ... = very small value

1 − 0.01 × 0.0001

Where = $W_{old} - \eta \frac{\partial L}{\partial W}$

= 1 − 0.00001

= 0.9999

negligible

1 − 0.9999 = 0.000001

= gradient value

which is going to be vanish

Vanishing grad. ⟶ Deep Neural Ner

{ 1. Loss will not change
  2. Weights will not change

Vanishing grad (How you gonna resolve it)

1. Reduce the Complexity of your NN:
   - ↳ Reduce the Nb. of layer
   - ↳ Reduce the No. of Nodes

   (Shallow Neural Ner)

2. Use a diff act. fn

Relu → max(0, z)

(−∞, +∞)

(0, +∞)

{ It will remove negtive value from your data }

Diff ⇒ -ve ⇒ 0

+ve ⇒ 1

Dying relu ← 0 or 1

(chain rule $\frac{\partial L}{\partial w} = (\ldots\ldots\ldots)$ ← 0

Leaky relu

3 Proper weight initialization

1. Xavier
2. Glorot

4 Batch Normalization ←

How to improve a Performance of Neural Network ?

1. Handle or TC vanishing gradient
   - Activation fn - Refer the Bappy's notes
   - Correct weight initialization -
2. Overfitting
   - Early stopping
   - Dropout -
   - Regularization
   - Reduce complexity (Layer, Node)
   - Increase data
3. Normalization (fast training)
   - Normalization / Std. - Can refer the Bappy's notes
   - Batch Normalization -
4. Optimizer -
   - Momentum
   - Adagrad
   - Adadelta / RMSProp
   - Adam
5. HyperParameter
   - No. of hidden layer -
   - Node / Neuron -
   - Batch size -
   - Learning rate -
6. Activation
   - Sigmoid
   - tanh
   - type Relu
   - Softmax
   - Linear / Step

Hyperparameter tuning using Keras Tuner

Can refer Bappy's notes

Data Scaling

Age → [1-95]

Salary → [1 - 1000000]

Keep the data in a same scale

= Input x weight + bias

Age × w_1 + Sal × w_2 + bias

50 → 1

100000

Contour

To ensure distribution of data is same ie; normally distributed with mean=0 and std dev=1

1. Std

$$\frac{X_i - \mu}{\sigma}$$

[-3, +3] → z-table → SND

Salary

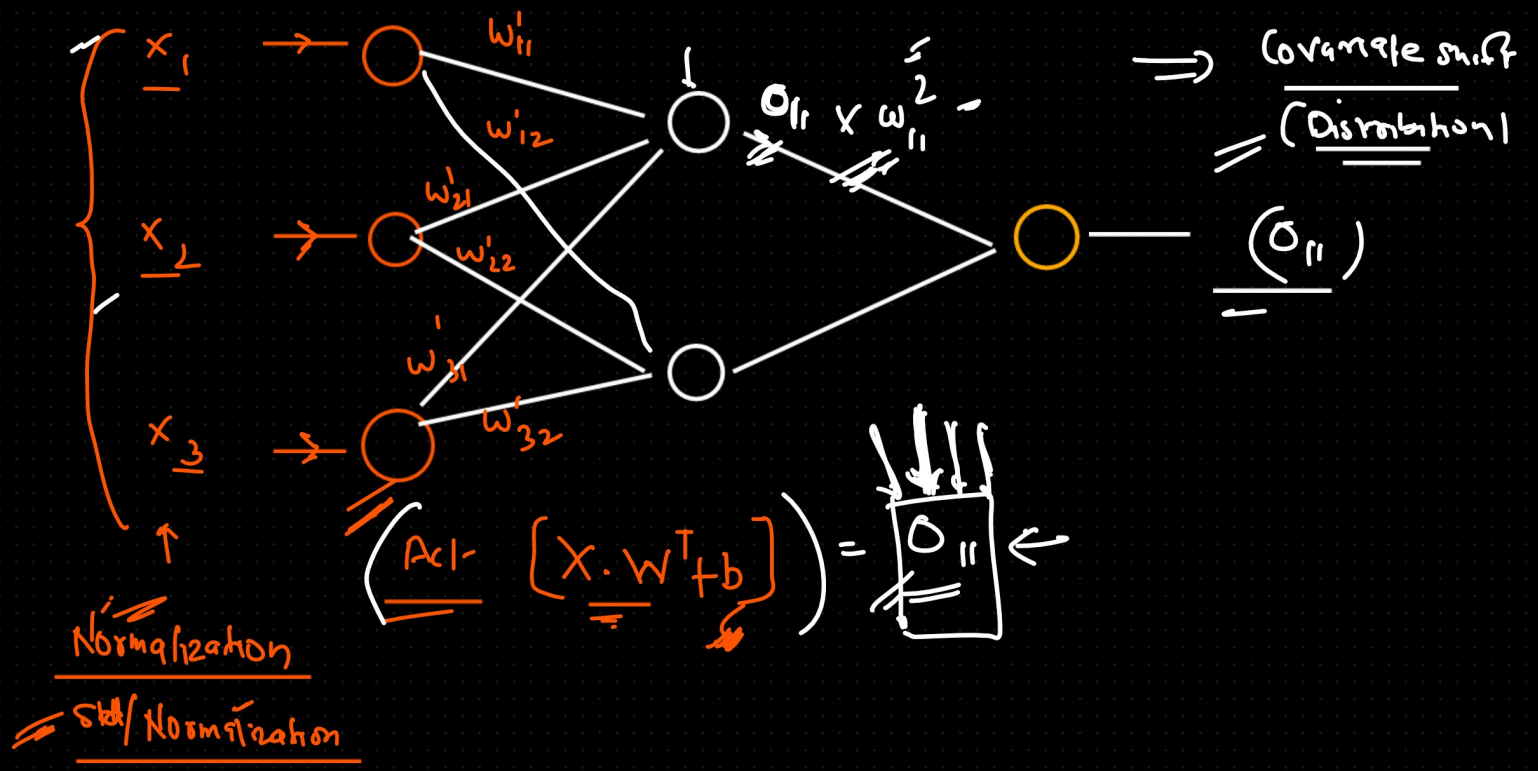
To compress the values in the range [0,1] so that to eliminate any bias that may arise due to different scale or magnitude of data(Recall Avnish's example)

2. Normalization

$$\frac{X_i - X_{min}}{X_{max} - X_{min}}$$

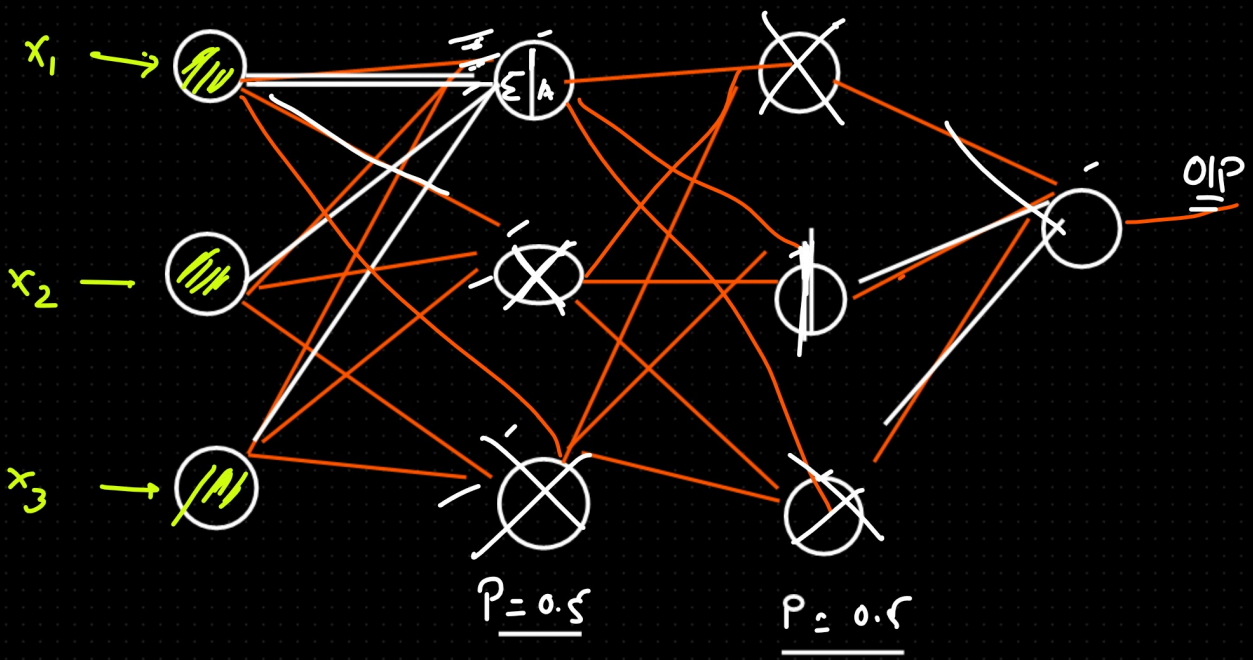
[0-1]

CGPA (0-10)


x_1
x_2
x_3
w'_{11}
w'_{12}
w'_{21}
w'_{22}
w'_{31}
w'_{32}
$O_{11} \times w^2_{11}$
Covariate shift
(Distribution)
(O_{11})
Act- $[X \cdot W^T + b]$ = O_{11}
Normalization
Std/Normalization

Drop out

Training (1 epoch)

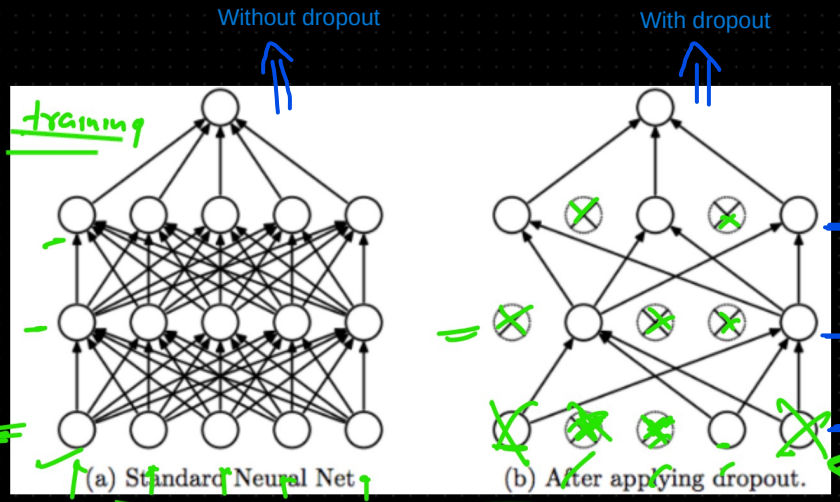


* So, basically in Drop out we are shutting down few neurons in each hidden layer randomly. Due to this the data is passed only through the neurons which are not shut down and overfitting is addressed.

* Shutting down of the neuron is done randomly in each epoch. Therefore in each epoch we are going to get the different set of neurons through which data is basically getting passed. This is similar to what we see in the random forest. This is how we are enabling randomization in this.

* No. of neurons to be picked up randomly in each epoch for shutdown is determined using Dropout ratio. We assign Dropout ratio for each hidden layer

5 epoch



Dropout → 0.5 → 50%

0.30 → 30% → Shutdown

0.25 → 25%

0.5

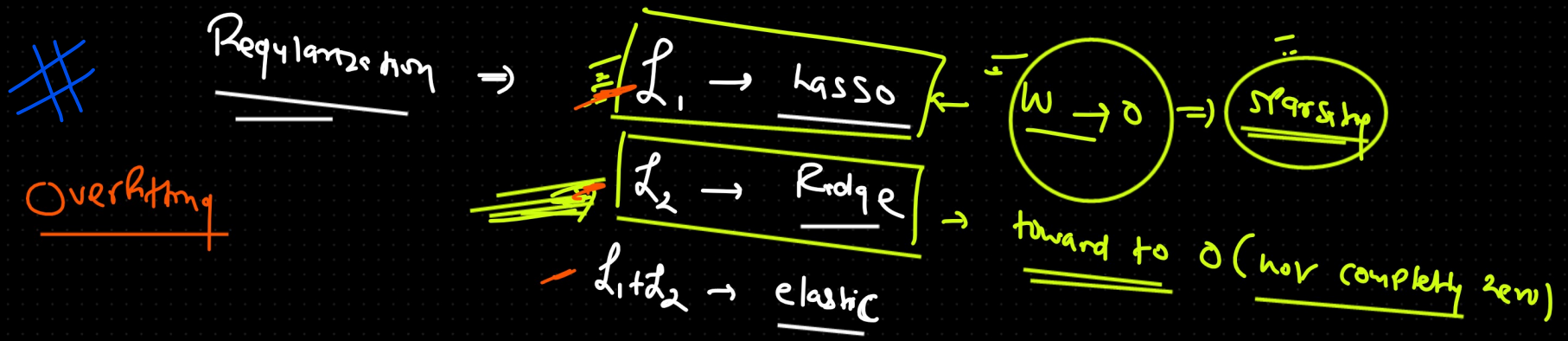
Drop out ratio for 1st hidden layer

Drop out ratio for 2nd hidden layer

Drop out ratio for 3rd hidden layer

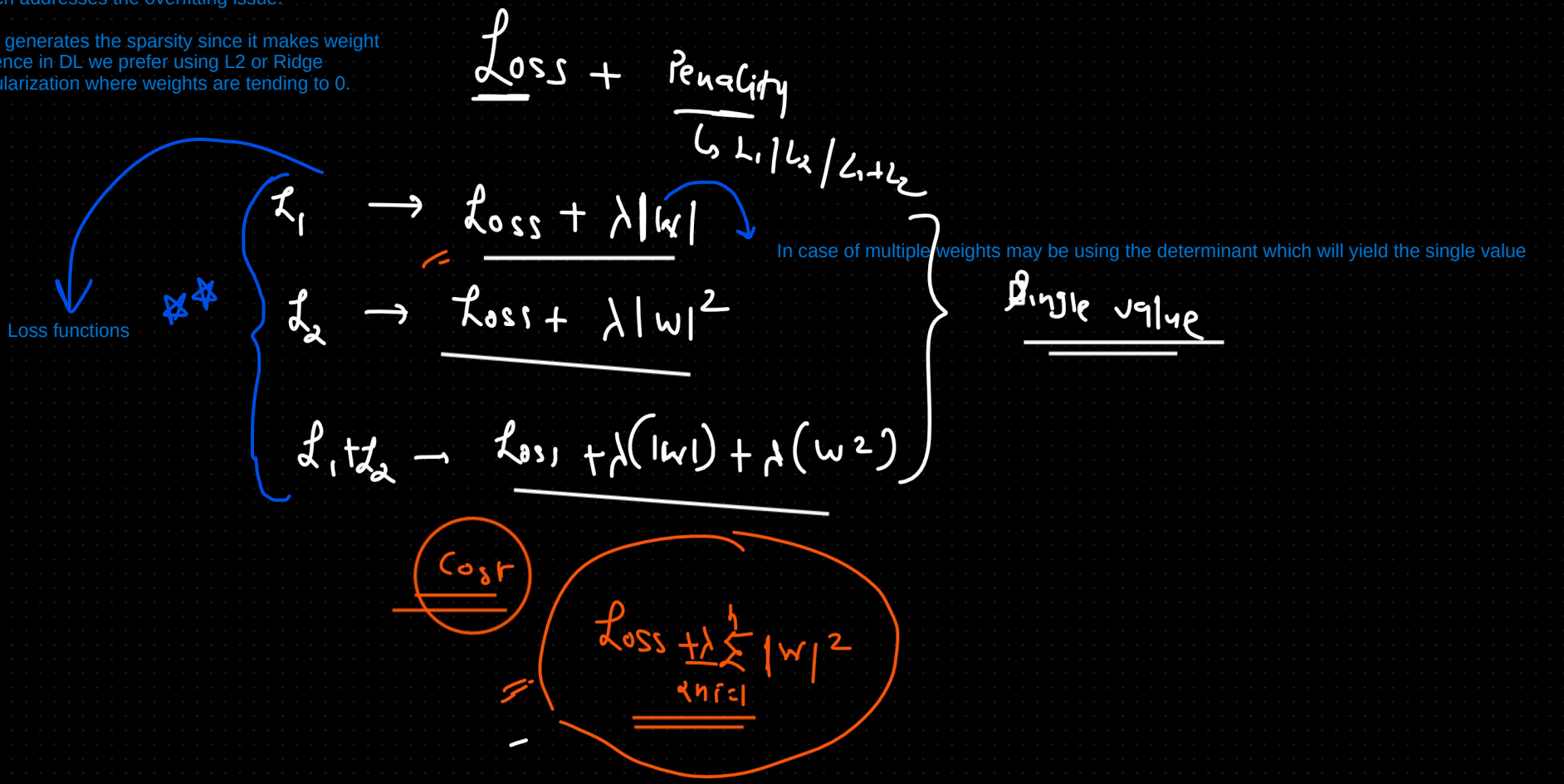
Randomization

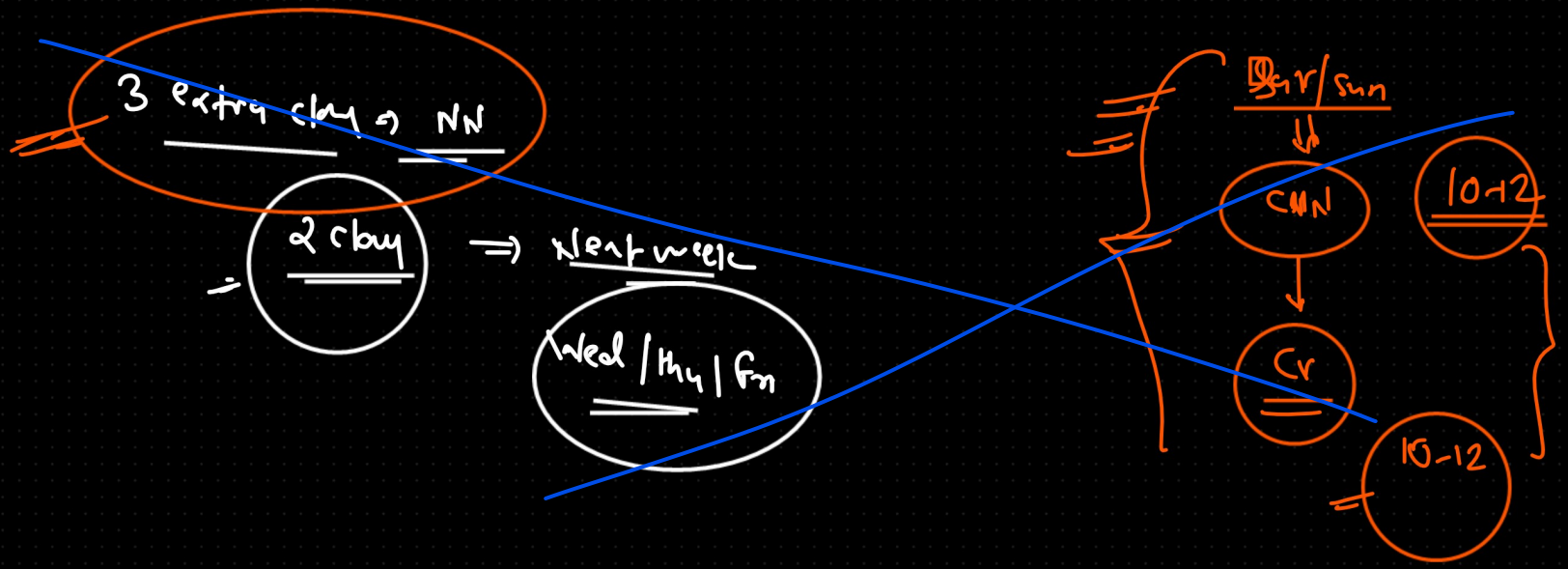
RF



* Regularization can be used to add the penalty (regularization parameter) in the loss/cost function which addresses the overfitting issue.

* L1 generates the sparsity since it makes weight 0 hence in DL we prefer using L2 or Ridge regularization where weights are tending to 0.




3 extra cbuy → NN
2 cbuy
→ Next week
Wed / Thu / Fri
Sat / Sun
CMN
10-12
Cr
10-12