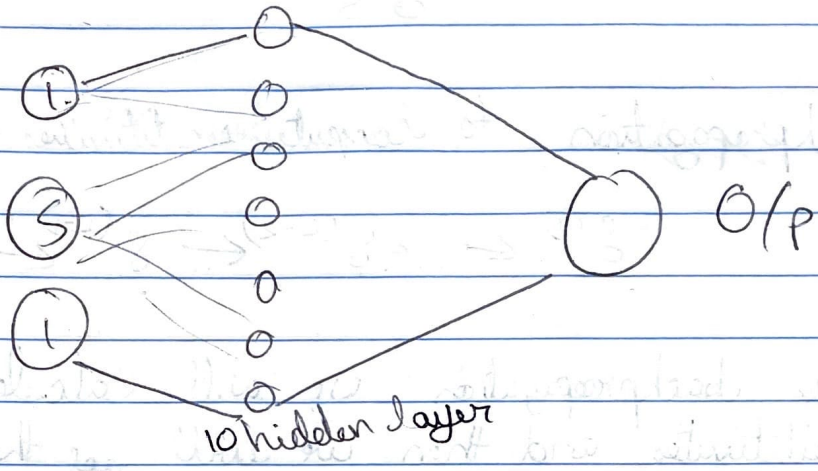
etdCKb

Lecture -21 etdCKb

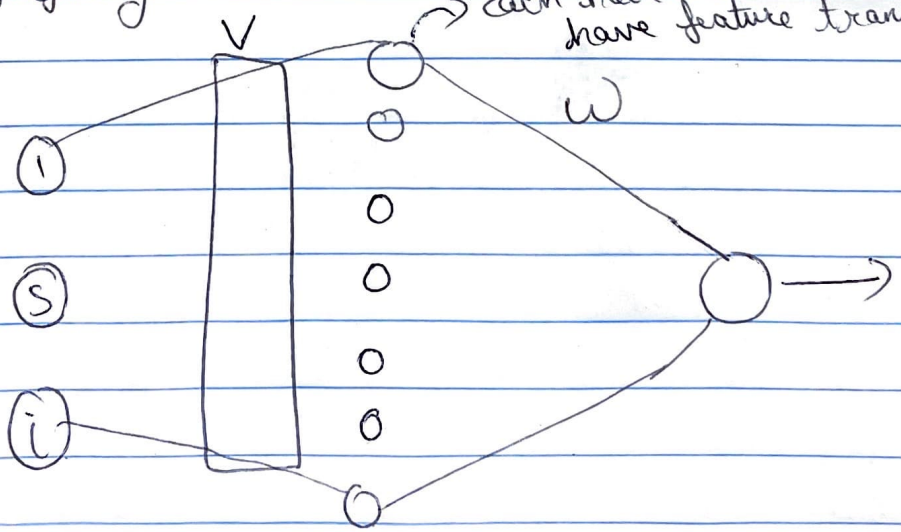
* Gradient descent Algorithm:-

$w(0)$ $t = 0$



* Neural Networks Overfit:-

Neural network can create a classifier which will classify the points distinctly but may lead to overfitting.



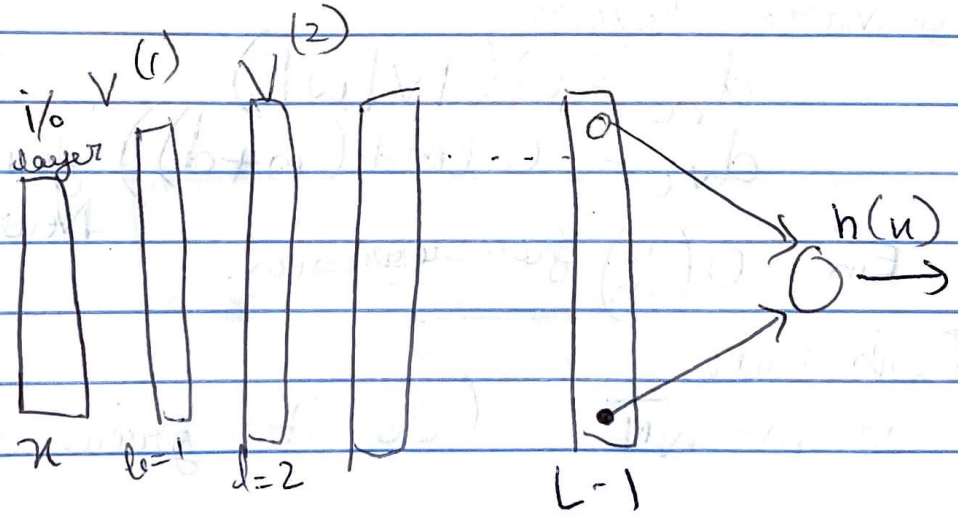
Neural Networks have a Tunable Transform.

- Neural Network.

$$h(x) = \theta\left(w_0 + \sum_{j=1}^{m} w_j \theta\left(v_j^T x\right)\right)$$

- Non-linear Transform.

$$h(x) = \theta\left(w_0 + \sum_{j=1}^{m} w_j \phi_j(x)\right)$$



- Approximation Capability of Neural Networks.

$$E_{in}(h) \leq \frac{(2RC_f)^2}{m} = O\left(\frac{1}{m}\right).$$

* Generalisation.

MLP (sign(.) activation func^n):

$$d_{vc} = O(md \log md)$$

where

d is # features
m is # hidden units.

Neural Networks

For tanh → the d_{vc} more larger.

$d_{vc} = O(|V||W|)$

$d_{vc} = O(md(m+d))$ for 2 layer Neural Network.

$E_{in} = O\left(\frac{1}{m}\right)$ for regression.

* Thumb rule.

Set $m = \sqrt{N}$ (so it grows sublinearly)

$E_{in} = O\left(\frac{1}{\sqrt{m}}\right)$ for classification.

$$E_{in} \sim \frac{1}{N^{1/4}}, \quad O\left(\sqrt{\frac{d_{vc}}{N} \log N}\right) \sim \frac{1}{N^{1/4}}$$

As $N \to \infty$.

$E_{out} \to E_{in}$ &

$E_{in} \to E^*_{out}$

* Regularization - Weight Decay.

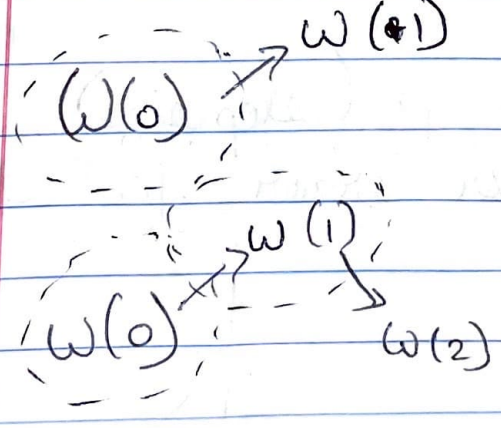
$\lambda = 0$
(No regularization)
overfitting

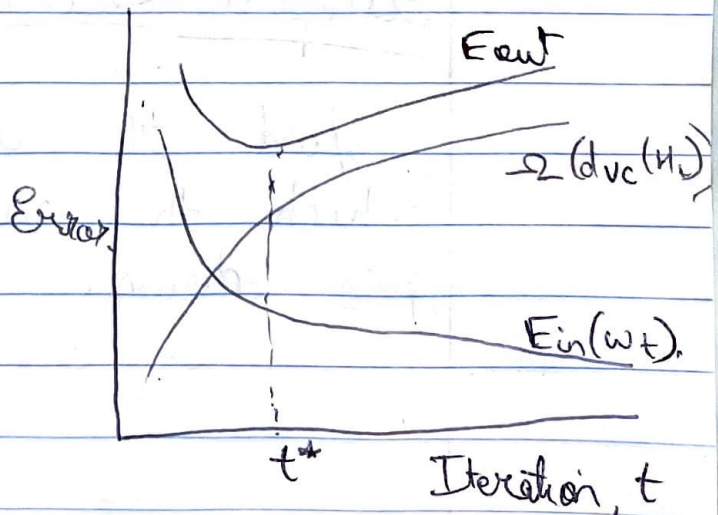
$\lambda = 0.01$
(some regularisation)
less overfitting.

$$E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} \text{ (Square of wt)}$$

* Another way to regularize is to stop early.

When to stop?





Use validation set to find t^*.

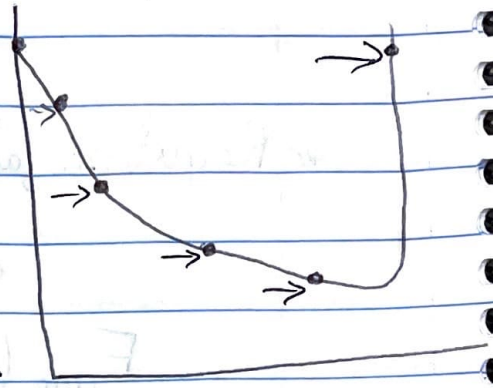
* Early stopping.

Variable learning Rate.

$\alpha > 1$, $\beta < 1$

$\eta = \alpha \eta$

learning rate will be α times earlier value.



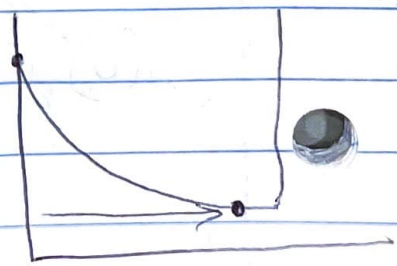
Initially we will have smaller η and slowly it will grow to a high value.

$\beta \rightarrow$ our angle.

We will go in the -ve direction of our gradient descent.

Steepest Descent:-

Find the largest η (step size) for which the value for error fun^n still goes down.



In batch all of your data points &
In stochastic we pick random data points.

In stochastic steepest - We pick random data pt^n & find the max steep we can go, for every random data point.

If all you ~~comp~~ are concerned about Computational time, use Stochastic gradient descent rather than variable learning rate.