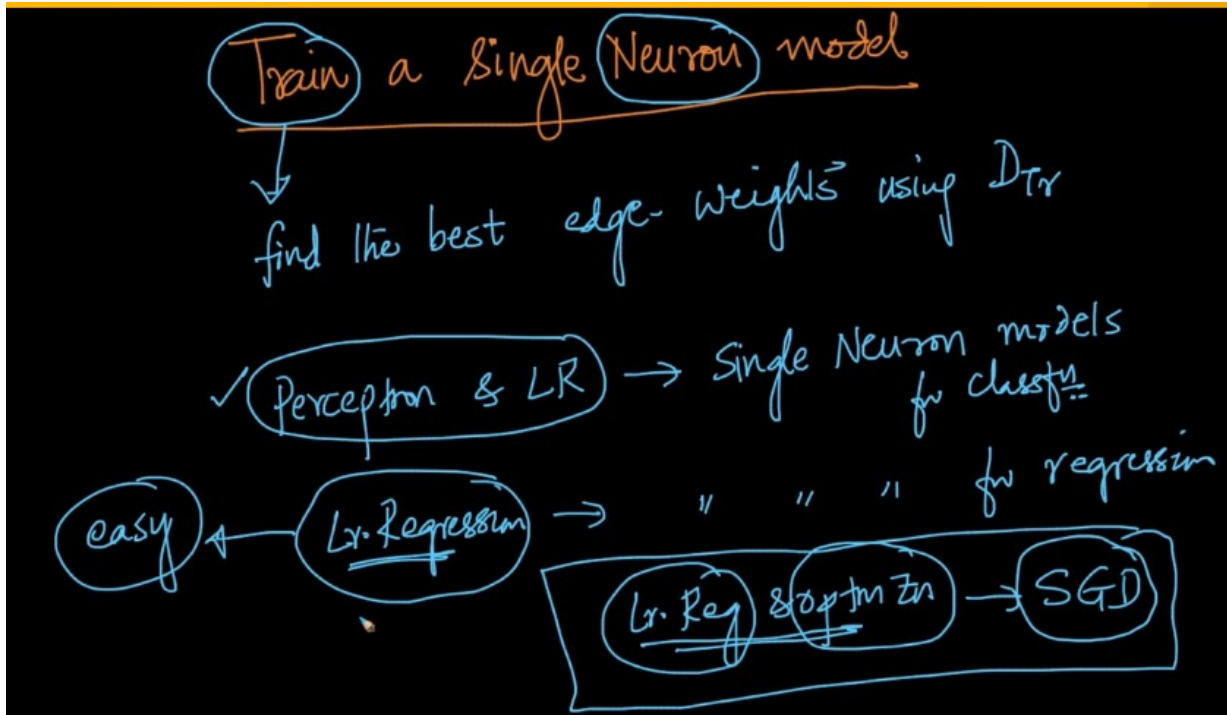


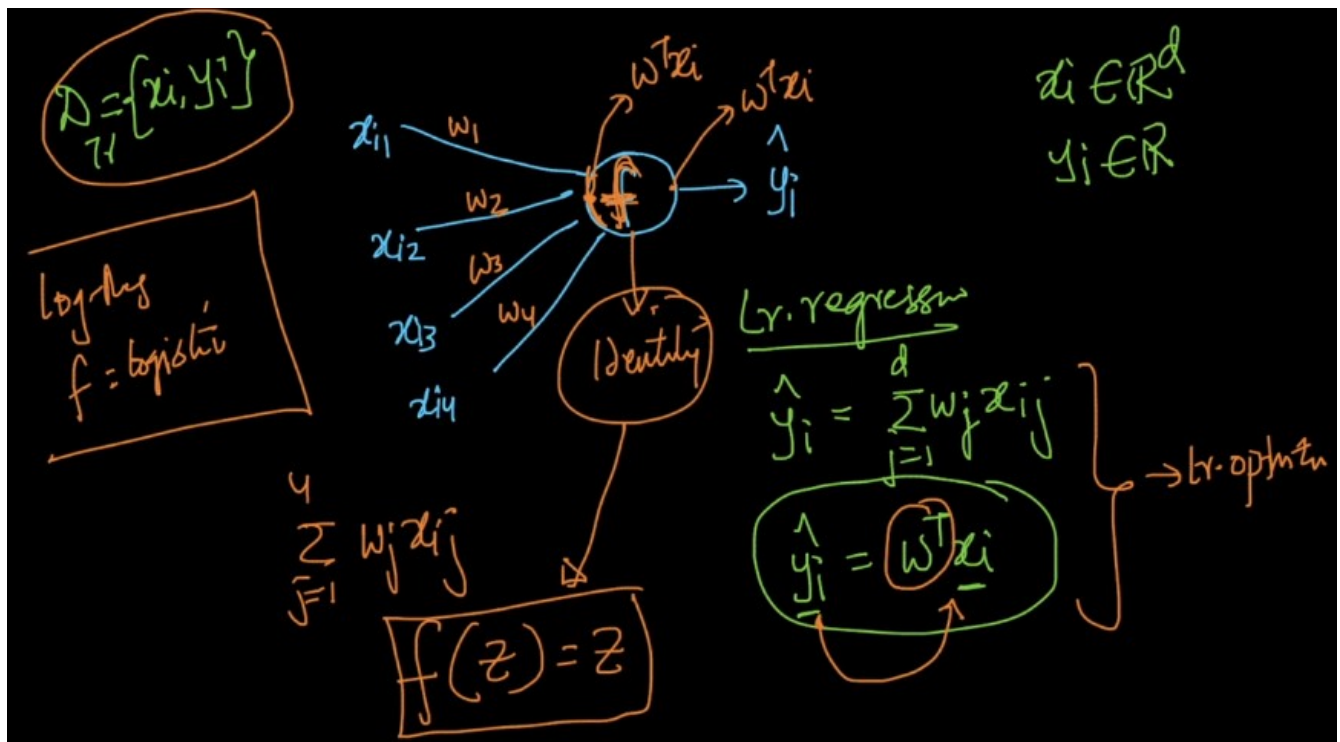
## Neural Networks

Train a single Neuron Model:

To find the best edge weights using Dtrain data, Perception and Log reg are single neuron models for classification.



Linear regression problem: Linear regression and optimization. Here  $F$  is an identity function,  $F(x) = x$



Logistic regression: But in case of logistic regression  $F$  is a logistic function.  
 The optimization problem for linear regression is as follows.

lv. regress<sup>n</sup>

$\min_{w_i} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \text{reg}$   $\rightarrow \# \text{pts Train}$

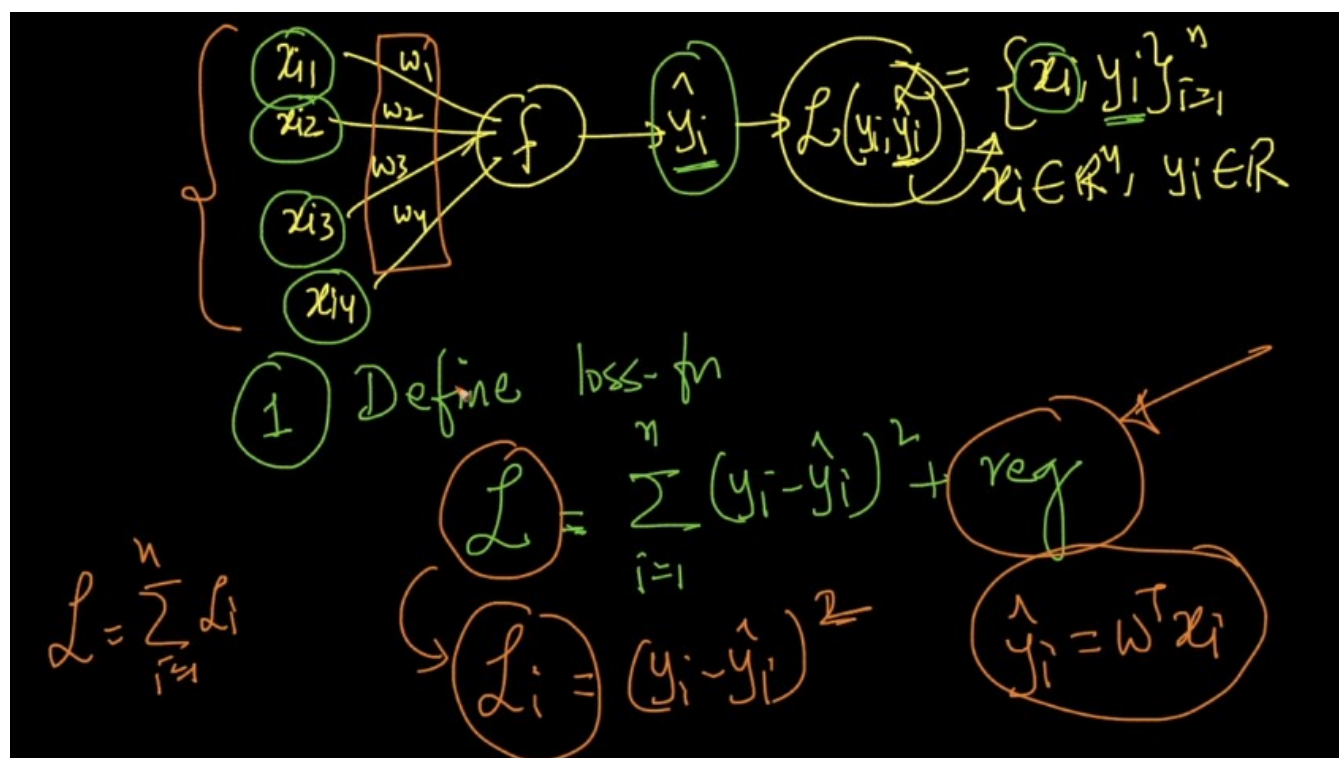
$\hat{y}_i = w^T x_i$

---

Optimization

$\min_{w_i} \sum_{i=1}^n (y_i - w^T x_i)^2 + \|w\|_2^2$

Neural networks in Depth -



Optimization problem:

Find  $W_i$  that minimizes the optimization problem.

$F$  is identity in case of Linear regression.

$F$  is Threshold in case of perception.

$F$  is Sigmoid in case of Logistic regression.

(2) Optimization

$$\min_{w_i} \sum_{i=1}^n (y_i - \underbrace{w^T x_i}_{\hat{y}_i = f(w^T x_i)})^2 + \text{reg}$$

$\hookrightarrow \text{lr. reg } f \text{ is } I$

---


$$\min_{w_i} \sum_{i=1}^n (y_i - f(w^T x_i))^2 + \text{reg}$$

percept  $\leftarrow$  Thr  $\leftarrow$   $I = \text{lr. reg}$   
 $\hookrightarrow$  Sigmoid:  $-\log \text{ reg}$

Solve the optimization problem:

$$w^* = \arg \min_w \sum_{i=1}^n (y_i - f(w^T x_i))^2 + \text{reg}$$

(3) Solve the optimization problem

(a) initialization of  $\underline{w_i}$   $\rightarrow$  random ✓

(b)  $\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial w_3} \\ \frac{\partial L}{\partial w_4} \end{bmatrix}$

$w \in \mathbb{R}^4$   
 $x_i \in \mathbb{R}^4$

SGD


Updating weights -


(c)  $W_{\text{new}} = W_{\text{old}} - \eta \left[ \nabla_w L \right]_{W_{\text{old}}}$

$(w_i)_{\text{new}} = (w_i)_{\text{old}} - \eta \left[ \frac{\partial L}{\partial w_i} \right]_{(w_i)_{\text{old}}}$

for iter = 1 to  $\infty$

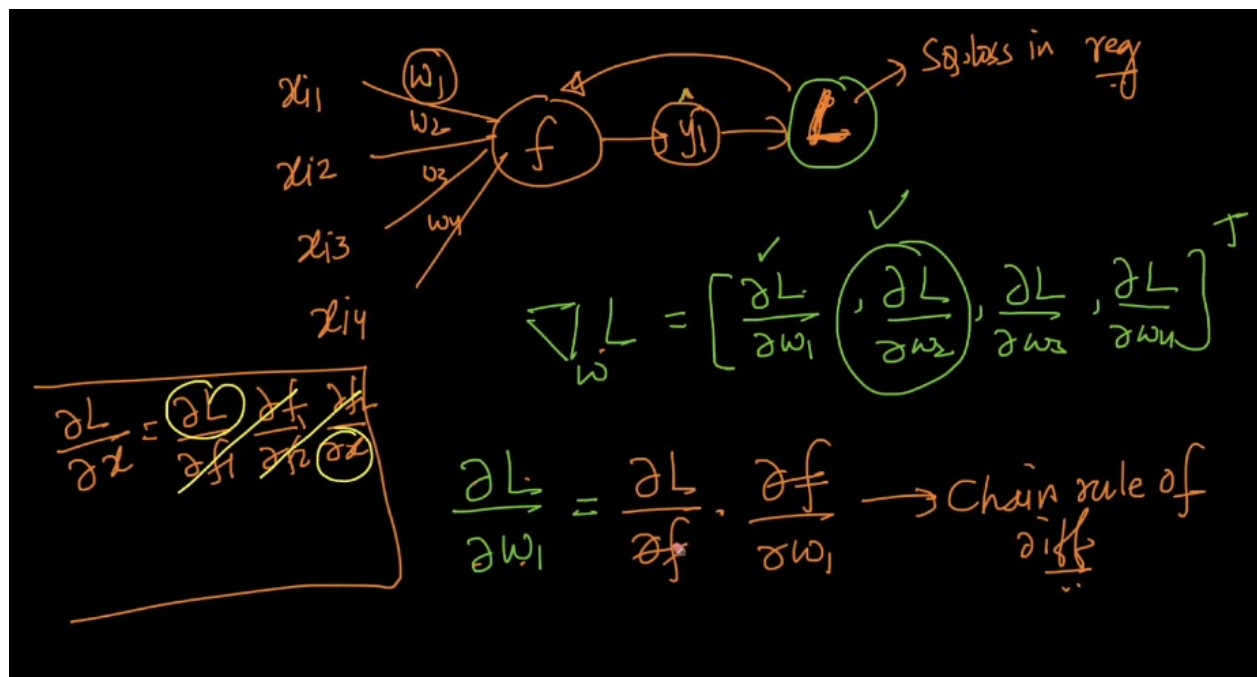
The key part is computing gradient descent we will use all the data for computing the new weights. In SGD we will use the small batch of points and approximate the weight vector.

GD:   $\rightarrow x_i$ 's &  $y_i$ 's

SGD:   $\approx$  one pt  $\{x_i, y_i\}$   $\leftarrow$  SGD  
Small batch of pts

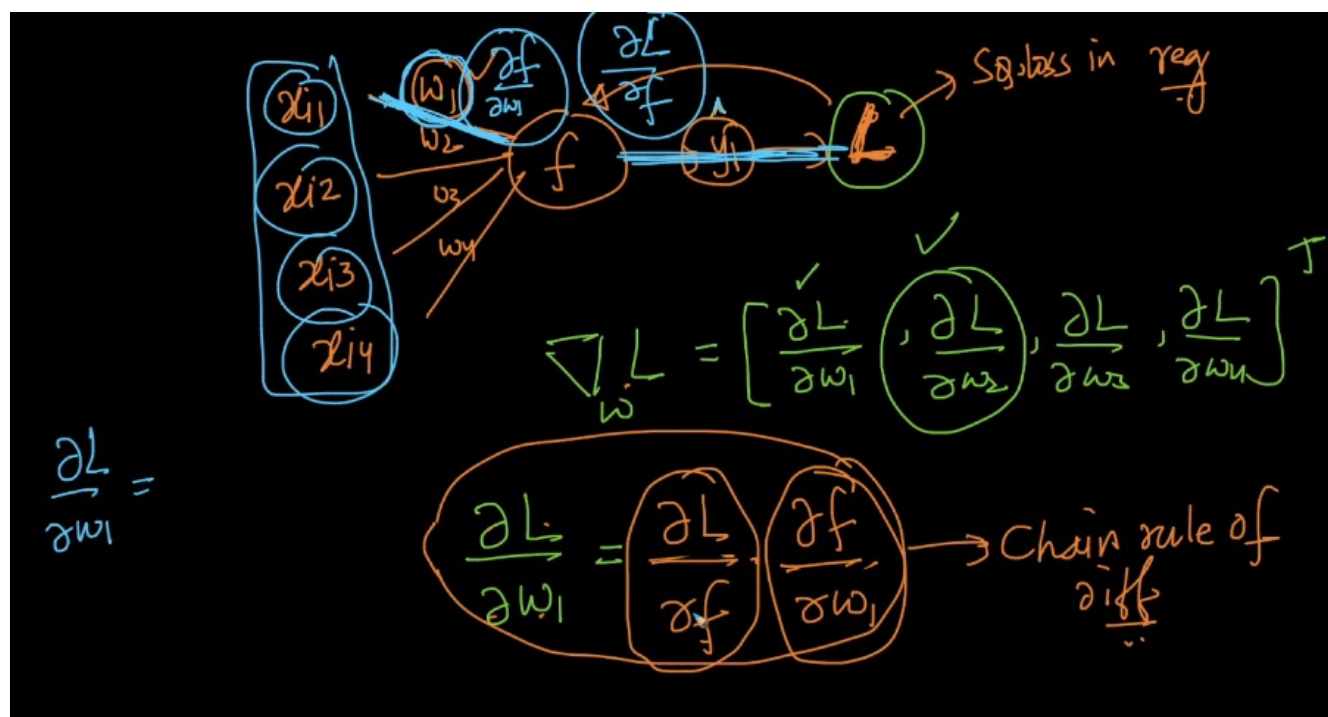


In regression we have squared loss for the network.



These are not ratios, they are derivatives :P.

Pictorial representation of derivatives:



$$\begin{aligned}
 f(w^T x_i) &= w^T x_i \\
 L &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \cancel{\text{reg}} \\
 &= \sum_{i=1}^n (y_i - f)^2 \\
 \frac{\partial L}{\partial f} &= \sum_{i=1}^n 2(y_i - f(w^T x_i))
 \end{aligned}
 \quad \left| \quad \frac{\partial f}{\partial w_1} = x_i
 \right.$$

Training MLP : Chain Rule