PyTorch

Machine Learning & Deep Learning (Barcha uchun)

<07> Deep and wide.

Mansurbek Abdullaev

https://uzbek.gitbook.io/ai/

mansurbek.comchemai@gmail.com

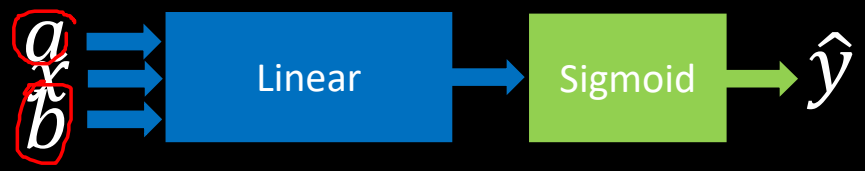
@MansurbekUST

Oxford University ga PhD ga topshirish

a, b

$x w + b = \hat{y}$

GPA (a)	Tajriba (b)	Qabul ?
2.1	0.1	0
4.2	0.8	1
3.1	0.9	0
3.3	0.2	1




```
x_data = torch.Tensor([[2.1, 0.1],
                       [4.2, 0.8],
                       [3.1, 0.9],
                       [3.3, 0.2]])
```

```
y_data = torch.Tensor([[0.],
                       [1.],
                       [0.],
                       [1]])
```

Matritsalrni ko'paytirish

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

2 x 4 4 x 3 2 x 3

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

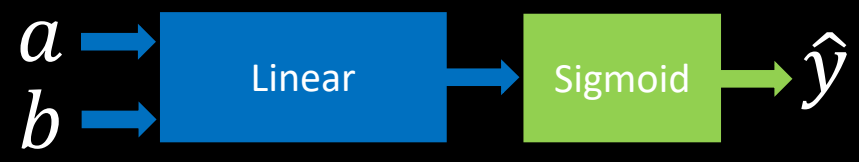
$A \quad B \longrightarrow C$

$m \times r \quad r \times n \qquad m \times n$

Matritsalrni ko'paytirish

```
x_data = torch.Tensor([[2.1, 0.1],
                       [4.2, 0.8],
                       [3.1, 0.9],
                       [3.3, 0.2]])
```

```
y_data = torch.Tensor([[0.],
                       [1.],
                       [0.],
                       [1]])
```

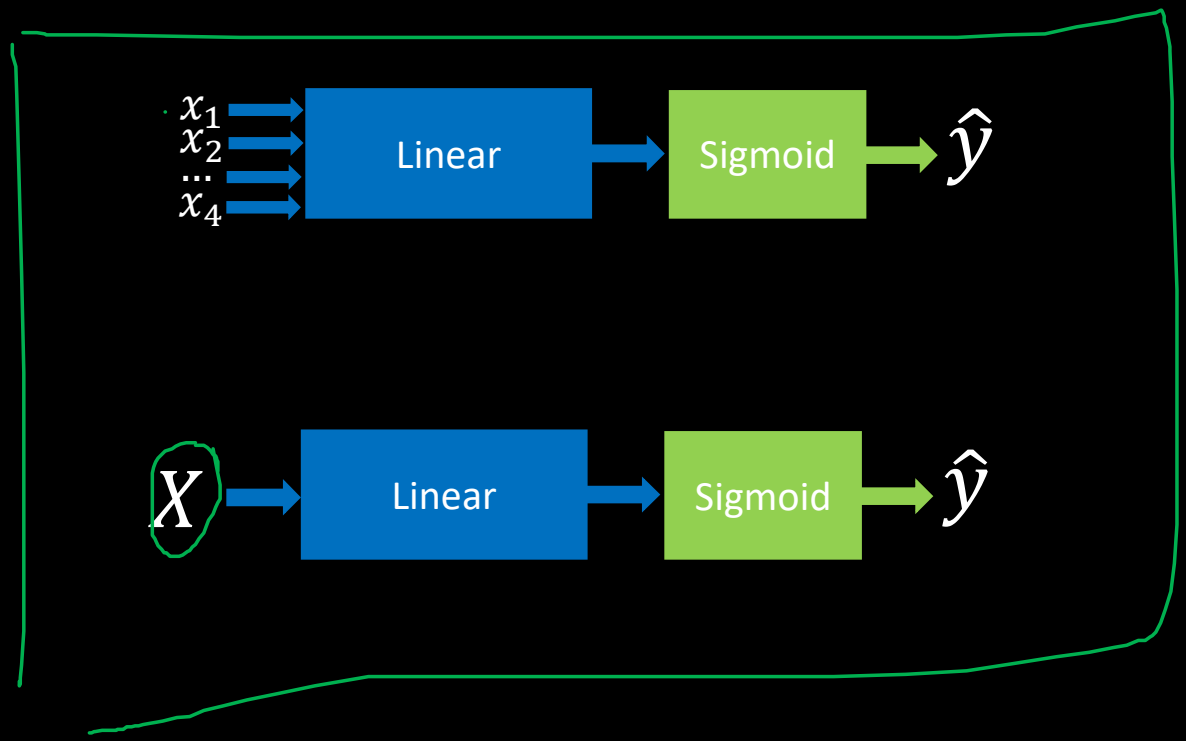


$$\underbrace{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix}}_{x \in \mathbb{R}^{N\times 2}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{w \in \mathbb{R}^{2\times 1}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{y \in \mathbb{R}^{N\times 1}}$$

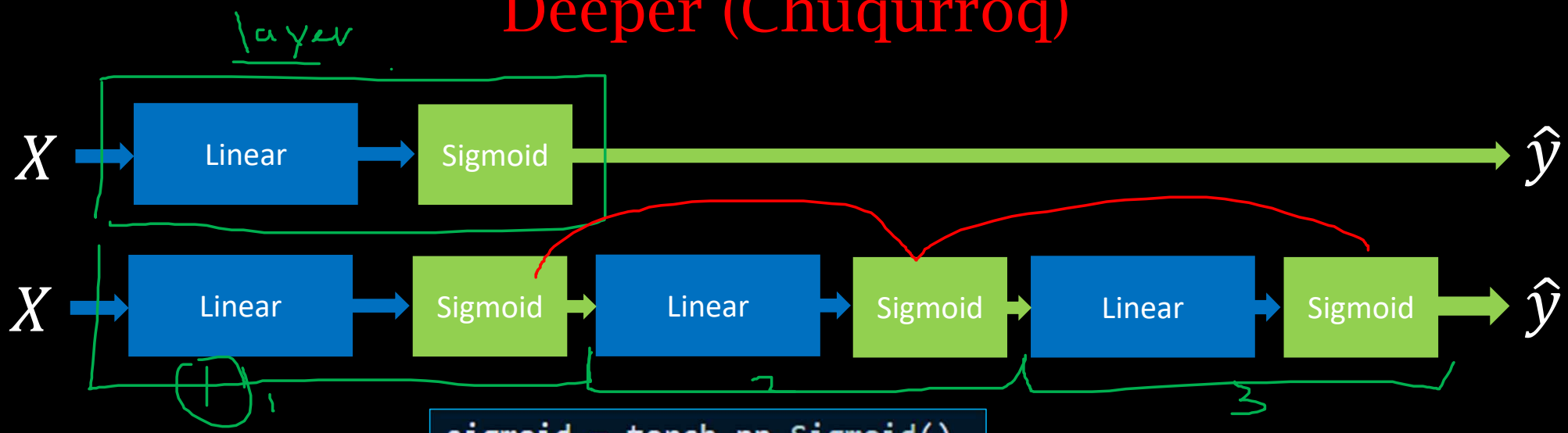
$$XW = \hat{Y}$$

```
linear = torch.nn.Linear(2, 1)
y_pred = linear(x_data)
```

Wider (Kengroq)



Deeper (Chuqurroq)



a, b

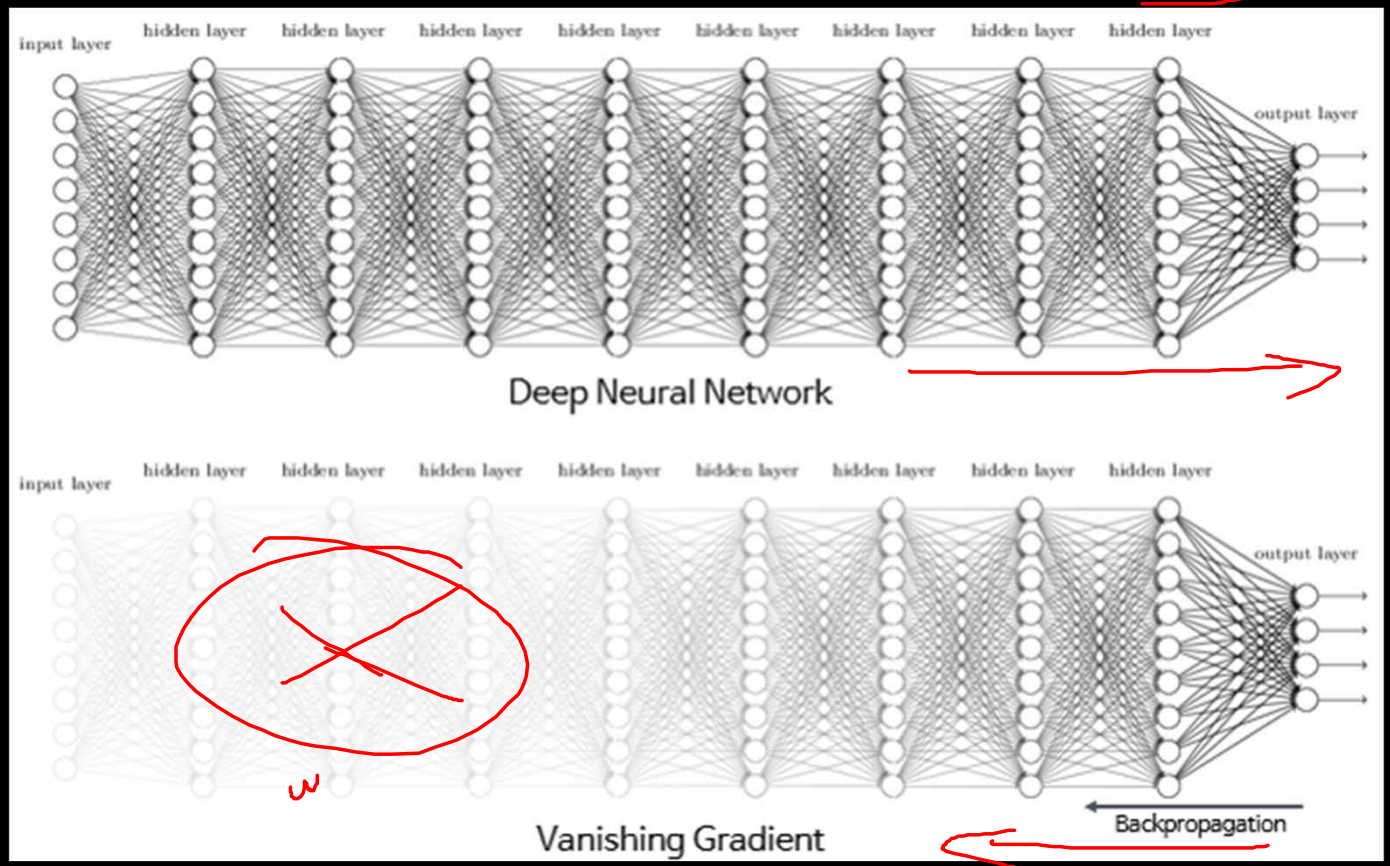
```
sigmoid = torch.nn.Sigmoid()

lin1 = torch.nn.Linear(2, 4)
lin2 = torch.nn.Linear(4, 3)
lin3 = torch.nn.Linear(3, 1)

rv1 = sigmoid(lin1(x_data))
rv2 = sigmoid(lin2(rv1))
y_pred = sigmoid(lin3(rv2))
```

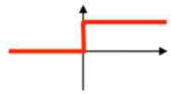
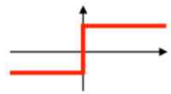
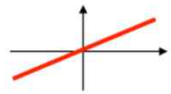
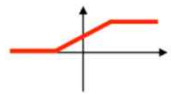
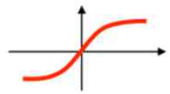
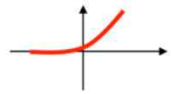
Sigmoid : Gradient Yo'qolish Muammosi
(Vanishing Gradient Problem)

x → Linear → Sigmoid → $\hat{y}$



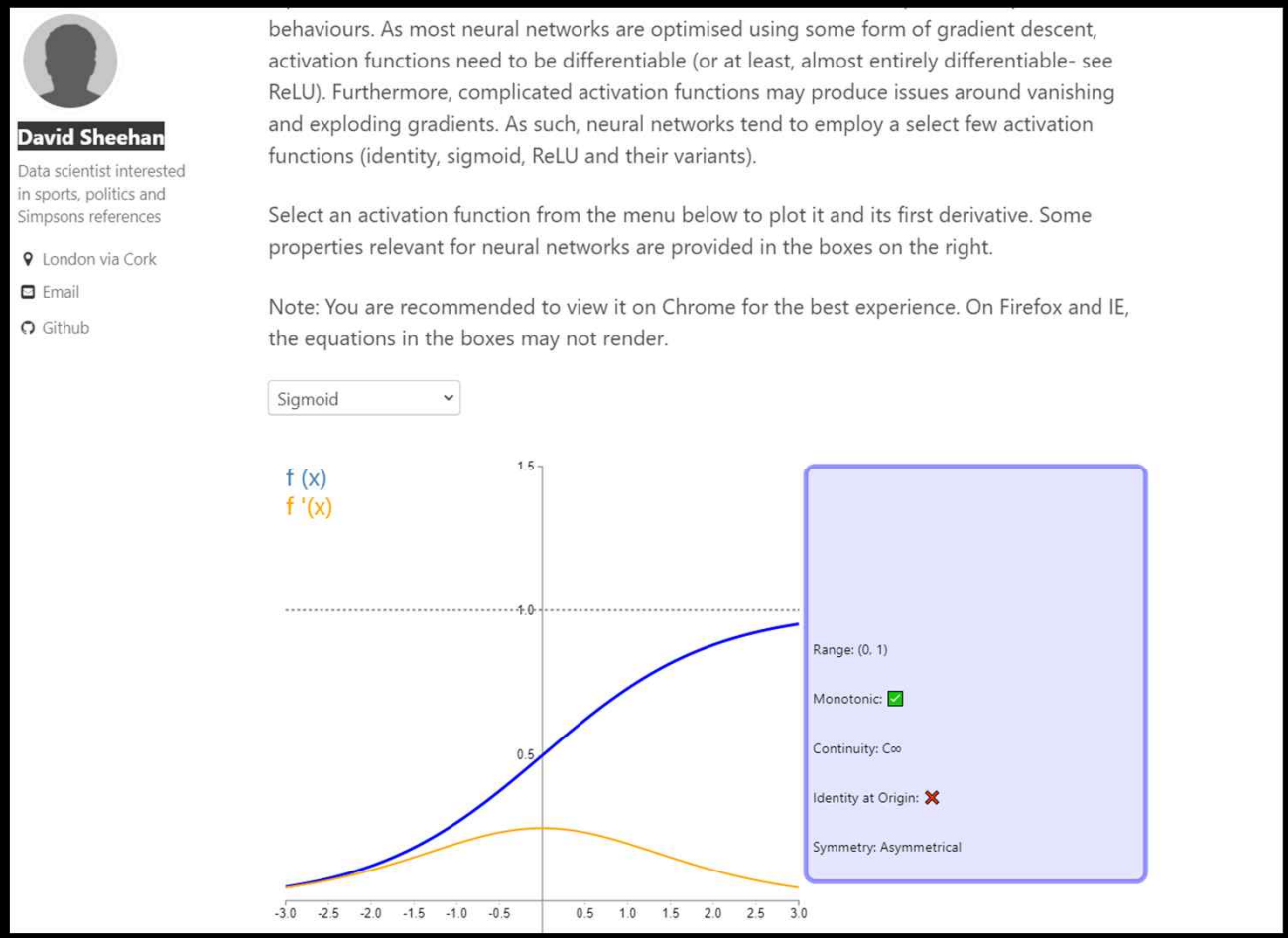
https://en.wikipedia.org/wiki/Vanishing_gradient_problem

Aktivlashtiruvchi funksiya
(Activation function)

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

http://rasbt.github.io/mlxtend/user_guide/general_concepts/activation-functions/

Aktivlashtiruvchi funksiya
(Activation function)



https://dashee87.github.io/deep%20learning/visualising-activation-functions-in-neural-networks/

Diabet klassifikatsiyasi (misol)

759 rows x 9 columns

-0.29412	0.487437	0.180328	-0.29293	0	0.00149	-0.53117	-0.03333	0
-0.88235	-0.14573	0.081967	-0.41414	0	-0.20715	-0.76687	-0.66667	1
-0.05882	0.839196	0.04918	0	0	-0.30551	-0.49274	-0.63333	0
-0.88235	-0.10553	0.081967	-0.53535	-0.77778	-0.16244	-0.924	0	1
0	0.376884	-0.34426	-0.29293	-0.60284	0.28465	0.887276	-0.6	0
-0.41177	0.165829	0.213115	0	0	-0.23696	-0.89496	-0.7	1
-0.64706	-0.21608	-0.18033	-0.35354	-0.79196	-0.07601	-0.85483	-0.83333	0
0.176471	0.155779	0	0	0	0.052161	-0.95218	-0.73333	1
-0.76471	0.979899	0.147541	-0.09091	0.283688	-0.09091	-0.93168	0.066667	0
-0.05882	0.256281	0.57377	0	0	0	-0.86849	0.1	0
-0.52941	0.105528	0.508197	0	0	0.120715	-0.9035	-0.7	1
0.176471	0.688442	0.213115	0	0	0.132638	-0.60803	-0.56667	0
0.176471	0.396985	0.311475	0	0	-0.19225	0.163962	0.2	1
-0.88235	0.899497	-0.01639	-0.53535	1	-0.10283	-0.72673	0.266667	0
-0.17647	0.005025	0	0	0	-0.10581	-0.65329	-0.63333	0
0	0.18593	0.377049	-0.05051	-0.45627	0.365127	-0.59607	-0.66667	0
-0.17647	0.075377	0.213115	0	0	-0.11774	-0.8497	-0.66667	0

```python
xy_data = np.loadtxt('../Data/diabetes.csv', delimiter=',', dtype = np.float32)

x_data = torch.from_numpy(xy_data[:, 0:-1])
y_data = torch.from_numpy(xy_data[:, [-1]])

print(x_data.shape)   # torch.Size([759, 8])
print(y_data.shape)   # torch.Size([759, 1])
```