Regression and Classification

SC4001 – Tutorial 2

1. Train a linear neuron to learng the mapping from input $\boldsymbol{x} \in \boldsymbol{R}^3$ to output y from the following examples:

$\boldsymbol{x} = (x_1, x_2, x_2)$	y
$(0.09 \quad -0.44 \quad -0.15)$	-2.57
$(0.69 \quad -0.99 \quad -0.76)$	-2.97
$(0.34 \quad 0.65 \quad -0.73)$	0.96
$(0.15 \quad 0.78 \quad -0.58)$	1.04
$(-0.63 \quad -0.78 \quad -0.56)$	-3.21
$(0.96 \quad 0.62 \quad -0.66)$	1.05
$(0.63 \quad -0.45 \quad -0.14)$	-2.39
$(0.88 \quad 0.64 \quad -0.33)$	0.66

(a) Show one iteration of learning of the neuron with

i. Stochastic gradient descent learning

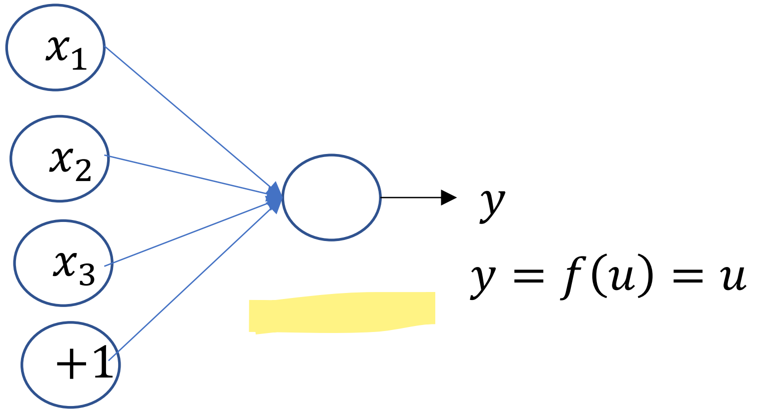
ii. Gradient descent learning

Initialize the weights as $\begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix}$ and biases to 0.0, and use a leaning factor $\alpha = 0.01$.

(b) Plot the learning curves (mean square error vs. epochs) until convergence. Determine the learned weights and biases.

(c) Find the predicted values y of training inputs after the training.

$\boldsymbol{x} = (x_1, x_2, x_3)$	d
$(0.09, -0.44, -0.15)$	-2.57
$(0.69, -0.99, -0.76)$	-2.97
$(0.34, 0.65, -0.73)$	0.96
$(0.15, , 0.78, -0.58)$	1.04
$(-0.63, -0.78, -0.56)$	-3.21
$(0.96, 0.62, -0.66)$	1.05
$(0.63, -0.45, -0.14)$	-2.39
$(0.88, 0.64, -0.33)$	0.66



initial weights and biases: $\boldsymbol{w} = \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix}$, $b = 0.0$

Learning factor $\alpha = 0.01$

(a) SGD for the linear neuron:

Given a training dataset $\{(\boldsymbol{x}_p, d_p)\}_{p=1}^{P}$

Set learning parameter α

Initialize $\boldsymbol{w}$ and b

Repeat until convergence:

For every training pattern $(\boldsymbol{x}_p, d_p)$:

$$y_p = \boldsymbol{x}_p{}^T\boldsymbol{w} + b$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(d_p - y_p)\boldsymbol{x}_p$$

$$b \leftarrow b + \alpha(d_p - y_p)$$

SGD:
learning factor $\alpha = 0.01$

Epoch 1

$p = 1$

Apply $\boldsymbol{x}_p = \begin{pmatrix} 0.34 \\ 0.65 \\ -0.73 \end{pmatrix}$, $d_p = 0.96$

$$y_p = \boldsymbol{x_p}^T \boldsymbol{w} + b = (0.34 \quad 0.65 \quad -0.73) \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix} + 0.0 = -0.19$$

$$s.e. = (d_p - y_p)^2 = (0.96 + 0.19)^2 = 1.31$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(d_p - y_p)\boldsymbol{x_p} = \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix} + 0.01 \times (0.96 + 0.19) \begin{pmatrix} 0.34 \\ 0.65 \\ -0.73 \end{pmatrix} = \begin{pmatrix} 0.78 \\ 0.03 \\ 0.62 \end{pmatrix}$$

$$b \leftarrow b + \alpha(d_p - y_p) = 0.0 + 0.01 \times (0.96 + 0.19) = 0.01$$

$p = 2$

Apply $\boldsymbol{x_p} = \begin{pmatrix} 0.63 \\ -0.45 \\ -0.14 \end{pmatrix}$, $d_p = -2.39$

$$y_p = \boldsymbol{x_p}^T \boldsymbol{w} + b = (0.63 \quad -0.45 \quad -0.14) \begin{pmatrix} 0.78 \\ 0.03 \\ 0.63 \end{pmatrix} + 0.01 = 0.4$$

$$s.e. = \left(d_p - y_p\right)^2 = (-2.39 - 0.4)^2 = 7.78$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\left(d_p - y_p\right)\boldsymbol{x_p} = \begin{pmatrix} 0.78 \\ 0.03 \\ 0.63 \end{pmatrix} + 0.01 \times (-2.39 - 0.4) \begin{pmatrix} 0.63 \\ -0.45 \\ -0.14 \end{pmatrix} = \begin{pmatrix} 0.76 \\ 0.04 \\ 0.63 \end{pmatrix}$$

$$b \leftarrow b + \alpha\left(d_p - y_p\right) = 0.01 + 0.01 \times (-2.39 - 0.4) = -0.02$$

Continue apply other patterns ....

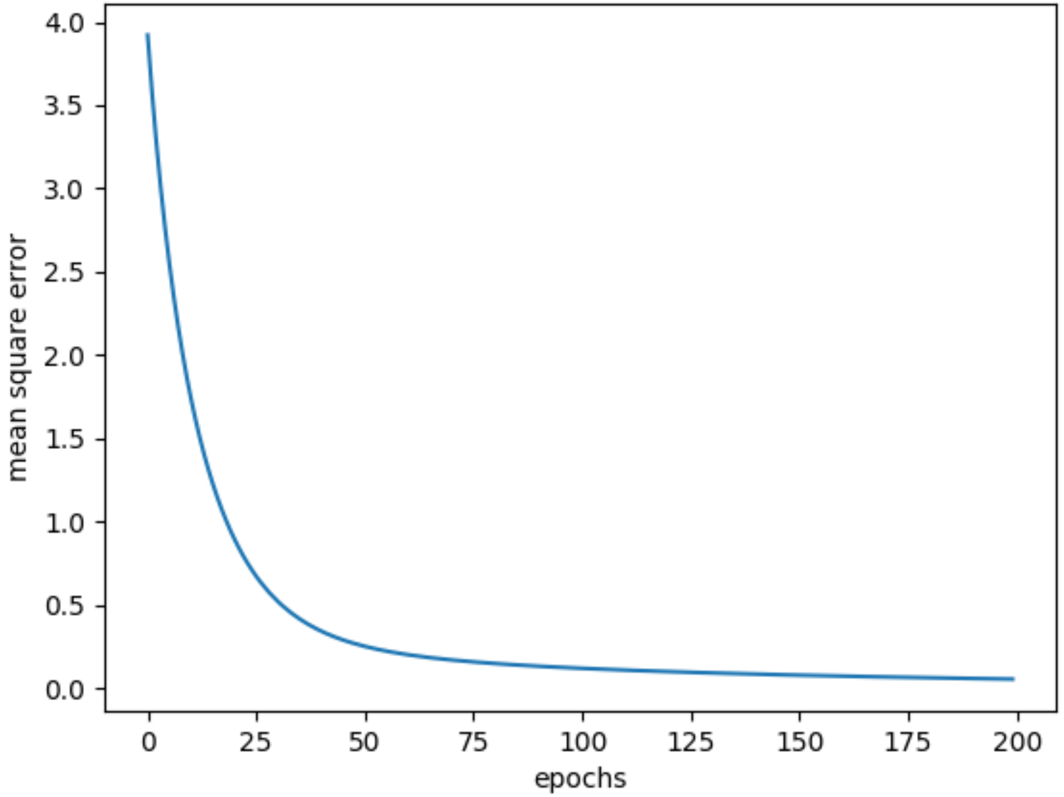
Then continue epochs 2, 3, ... until convergence.

Epoch 1

Epoch 1 involves all training patterns

$\boldsymbol{x}$	d	y	s.e.	$\boldsymbol{w}$	b
$\begin{pmatrix}0.34\\0.65\\-0.73\end{pmatrix}$	0.96	-0.19	1.31	$\begin{pmatrix}0.78\\0.03\\0.63\end{pmatrix}$	0.01
$\begin{pmatrix}0.63\\-0.45\\-0.14\end{pmatrix}$	-2.39	0.4	7.78	$\begin{pmatrix}0.76\\0.04\\0.63\end{pmatrix}$	-0.02
$\begin{pmatrix}0.88\\0.64\\-0.33\end{pmatrix}$	0.66	0.47	0.04	$\begin{pmatrix}0.76\\0.04\\0.63\end{pmatrix}$	-0.01
$\begin{pmatrix}0.96\\0.62\\-0.66\end{pmatrix}$	1.05	0.33	0.52	$\begin{pmatrix}0.77\\0.05\\0.62\end{pmatrix}$	-0.01
$\begin{pmatrix}0.09\\-0.44\\-0.15\end{pmatrix}$	-2.57	-0.05	6.34	$\begin{pmatrix}0.76\\0.06\\0.63\end{pmatrix}$	-0.03
$\begin{pmatrix}0.69\\-0.99\\-0.76\end{pmatrix}$	-2.97	-0.04	8.59	$\begin{pmatrix}0.74\\0.09\\0.65\end{pmatrix}$	-0.06
$\begin{pmatrix}-0.63\\-0.78\\-0.56\end{pmatrix}$	-3.21	-0.96	5.05	$\begin{pmatrix}0.76\\0.10\\0.66\end{pmatrix}$	-0.08
$\begin{pmatrix}0.15\\0.78\\-0.58\end{pmatrix}$	1.04	-0.27	1.73	$\begin{pmatrix}0.76\\0.11\\0.65\end{pmatrix}$	-0.07

m.s.e = 3.92


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
mean square error
0
25
50
75
100
125
150
175
200
epochs

At convergence:

$$\boldsymbol{w} = \begin{pmatrix} 0.37 \\ 2.57 \\ -0.21 \end{pmatrix}, b = -1.17$$

mse = 0.054

If $\boldsymbol{x} = (x_1, x_2, x_3)^T$, the learned function by the linear neuron:

$$y = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$y = (0.37 \quad 2.57 \quad -0.21) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - 1.17$$

$$y = 0.37x_1 + 2.57x_2 - 0.21x_3 - 1.17$$

Predicted values:

$$y = 0.37x_1 + 2.57x_2 - 0.21x_3 - 1.17$$

x: [-0.63 -0.78 -0.56], d: -3.21, y: -3.28035
x: [0.96 0.62 -0.66], d: 1.05, y: 0.919454
x: [0.09 -0.44 -0.15], d: -2.57, y: -2.22926
x: [0.88 0.64 -0.33], d: 0.66, y: 0.871378
x: [0.34 0.65 -0.73], d: 0.96, y: 0.782616
x: [0.15 0.78 -0.58], d: 1.04, y: 1.01435
x: [0.69 -0.99 -0.76], d: -2.97, y: -3.29005
x: [0.63 -0.45 -0.14], d: -2.39, y: -2.0579

(b) GD for a linear neuron

Given a training dataset$(\boldsymbol{X}, \boldsymbol{d})$

Set the learning parameterα

Initialize $\boldsymbol{w}$ and b

Repeat until convergence:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + b\mathbf{1}_P$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\boldsymbol{X}^T(\boldsymbol{d} - \boldsymbol{y})$$

$$b \leftarrow b + \alpha\mathbf{1}_P{}^T(\boldsymbol{d} - \boldsymbol{y})$$

GD for a linear neuron

$$\boldsymbol{X} = \begin{pmatrix} 0.09 & -0.44 & -0.15 \\ 0.69 & -0.99 & -0.76 \\ 0.34 & 0.65 & -0.73 \\ 0.15 & 0.78 & -0.58 \\ -0.63 & -0.78 & -0.56 \\ 0.96 & 0.62 & -0.66 \\ 0.63 & -0.45 & -0.14 \\ 0.88 & 0.64 & -0.33 \end{pmatrix}, \boldsymbol{d} = \begin{pmatrix} -2.57 \\ -2.97 \\ 0.96 \\ 1.04 \\ -3.21 \\ 1.05 \\ -2.39 \\ 0.66 \end{pmatrix}$$

initial weights and biases: $\boldsymbol{w} = \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix}, b = 0.0$

Learning factor $\alpha = 0.01$

$$\text{Output } \boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + b\mathbf{1}_P = \begin{pmatrix} 0.09 & -0.44 & -0.15 \\ 0.69 & -0.99 & -0.76 \\ 0.34 & 0.65 & -0.73 \\ 0.15 & 0.78 & -0.58 \\ -0.63 & -0.78 & -0.56 \\ 0.96 & 0.62 & -0.66 \\ 0.63 & -0.45 & -0.14 \\ 0.88 & 0.64 & -0.33 \end{pmatrix} \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix} + 0.0 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.03 \\ 0.03 \\ -0.18 \\ -0.24 \\ -0.85 \\ 0.34 \\ 0.39 \\ 0.48 \end{pmatrix}$$

$$\boldsymbol{d} = \begin{pmatrix} -2.57 \\ -2.97 \\ 0.96 \\ 1.04 \\ -3.21 \\ 1.05 \\ -2.39 \\ 0.66 \end{pmatrix}$$

$$\text{m.s.e.} = \frac{1}{8}\sum_{p=1}^{8}\left(d_p - y_p\right)^2$$
$$= \frac{1}{8}\left((-2.57 + 0.03)^2 + (-2.97 - 0.03)^2 + \cdots\cdots + (0.66 - 0.48)^2\right)$$
$$= 4.02$$

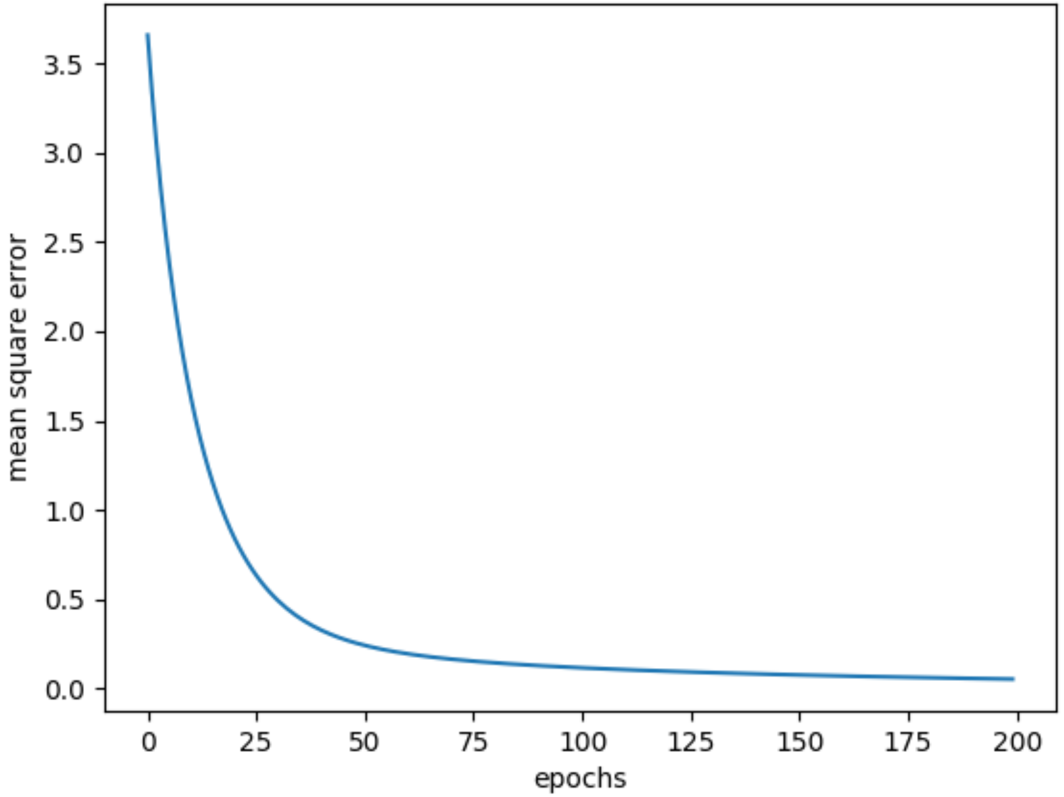
$$\boldsymbol{w} = \boldsymbol{w} + \alpha \boldsymbol{X}^T(\boldsymbol{d} - \boldsymbol{y})$$

$$= \begin{pmatrix} 0.77 \\ 0.02 \\ 0.63 \end{pmatrix} + 0.01 \times \begin{pmatrix} 0.09 & 0.69 & 0.34 & 0.15 & -0.63 & 0.96 & 0.63 & 0.88 \\ -0.44 & -0.99 & 0.65 & 0.78 & -0.78 & 0.62 & -0.45 & 0.64 \\ -0.15 & -0.76 & -0.73 & -0.58 & -0.56 & -0.66 & -0.14 & -0.33 \end{pmatrix} \left(\begin{pmatrix} -2.57 \\ -2.97 \\ 0.96 \\ 1.04 \\ -3.21 \\ 1.05 \\ -2.39 \\ 0.66 \end{pmatrix} - \begin{pmatrix} -0.03 \\ 0.03 \\ -0.18 \\ -0.24 \\ -0.85 \\ 0.34 \\ 0.39 \\ 0.48 \end{pmatrix} \right)$$

$$= \begin{pmatrix} 0.76 \\ 0.11 \\ 0.65 \end{pmatrix}$$

$$b = b + \alpha \mathbf{1}_P{}^T(\boldsymbol{d} - \boldsymbol{y}) = 0.0 + 0.01 \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \left(\begin{pmatrix} -2.57 \\ -2.97 \\ 0.96 \\ 1.04 \\ -3.21 \\ 1.05 \\ -2.39 \\ 0.66 \end{pmatrix} - \begin{pmatrix} -0.03 \\ 0.03 \\ -0.18 \\ -0.24 \\ -0.85 \\ 0.34 \\ 0.39 \\ 0.48 \end{pmatrix} \right) = -0.07$$

epoch	y	mse	w	b
2	$\begin{pmatrix} -0.15 \\ -0.16 \\ -0.22 \\ -0.25 \\ -1.01 \\ 0.29 \\ 0.26 \\ 0.45 \end{pmatrix}$	3.66	$\begin{pmatrix} 0.75 \\ 0.21 \\ 0.67 \end{pmatrix}$	-0.14
200	$\begin{pmatrix} -2.23 \\ -3.29 \\ 0.78 \\ 1.01 \\ -3.28 \\ 0.92 \\ -2.06 \\ 0.87 \end{pmatrix}$	0.05	$\begin{pmatrix} 0.368 \\ 2.56 \\ -0.21 \end{pmatrix}$	-1.164



3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
mean square error
0
25
50
75
100
125
150
175
200
epochs

At convergence:

$$\boldsymbol{w} = \begin{pmatrix} 0.37 \\ 2.57 \\ -0.21 \end{pmatrix}, b = -1.16$$

mse = 0.054

If $\boldsymbol{x} = (x_1, x_2, x_3)^T$, the learned function by the linear neuron:

$$y = \boldsymbol{w}^T\boldsymbol{x} + b$$

$$y = \begin{pmatrix} 0.37 & 2.57 & -0.21 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - 1.16$$

$$y = 0.37x_1 + 2.57x_2 - 0.21x_3 - 1.16$$

After learning …

x	$\boldsymbol{d}$	y **(SGD)**	y **(GD)**
$(0.09, -0.44, -0.15)$	-2.57	-2.23	-2.23
$(0.69, -0.99, -0.76)$	-2.97	-3.29	-3.29
$(0.34, 0.65, -0.73)$	0.96	0.78	0.78
$(0.15, , 0.78, -0.58)$	1.04	1.01	1.01
$(-0.63, -0.78, -0.56)$	-3.21	-3.28	-3.28
$(0.96, 0.62, -0.66)$	1.05	0.92	0.92
$(0.63, -0.45, -0.14)$	-2.39	-2.06	-2.06
$(0.88, 0.64, -0.33)$	0.66	0.87	0.88
m.s.e.		0.055	0.054
$\boldsymbol{w}$		$\begin{pmatrix} 0.369 \\ 2.566 \\ -0.212 \end{pmatrix}$	$\begin{pmatrix} 0.368 \\ 2.567 \\ -0.207 \end{pmatrix}$
b		-1.165	-1.163
y		$0.37x_1 + 2.57x_2 - 0.21x_3 - 1.1$	$0.37x_1 + 2.57x_2 - 0.21x_3 - 1.16$

2. Two-dimensional training patterns (inputs) to design a dichotomizer are given as:

$$\mathbf{X}_1 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}; \quad \mathbf{X}_2 = \begin{bmatrix} 7 \\ 3 \end{bmatrix}; \quad \mathbf{X}_3 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}; \quad \mathbf{X}_4 = \begin{bmatrix} 5 \\ 4 \end{bmatrix}; \qquad \mathit{Class}\ 1$$

$$\mathbf{X}_5 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \quad \mathbf{X}_6 = \begin{bmatrix} -1 \\ -3 \end{bmatrix}; \quad \mathbf{X}_7 = \begin{bmatrix} -2 \\ 3 \end{bmatrix}; \quad \mathbf{X}_8 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}; \qquad \mathit{Class}\ 2$$

(a) Determine whether the two classes of patterns are linearly separable.

(b) Find the center of gravity of patterns in each class. Show that a linear decision boundary passing perpendicularly through the middle point of the line joining the two centroids is given by:

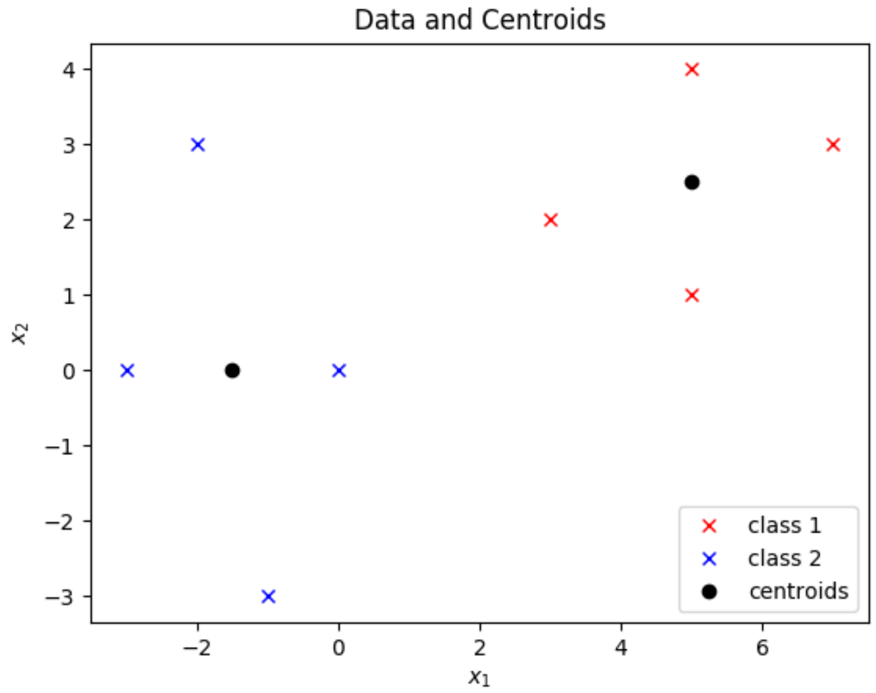
$$6.5x_1 + 2.5x_2 - 14.5 = 0$$

$$\boldsymbol{x}_1 = \begin{pmatrix}5\\1\end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix}7\\3\end{pmatrix}, \boldsymbol{x}_3 = \begin{pmatrix}3\\2\end{pmatrix}, \boldsymbol{x}_4 = \begin{pmatrix}5\\4\end{pmatrix} \rightarrow \text{class 1}$$

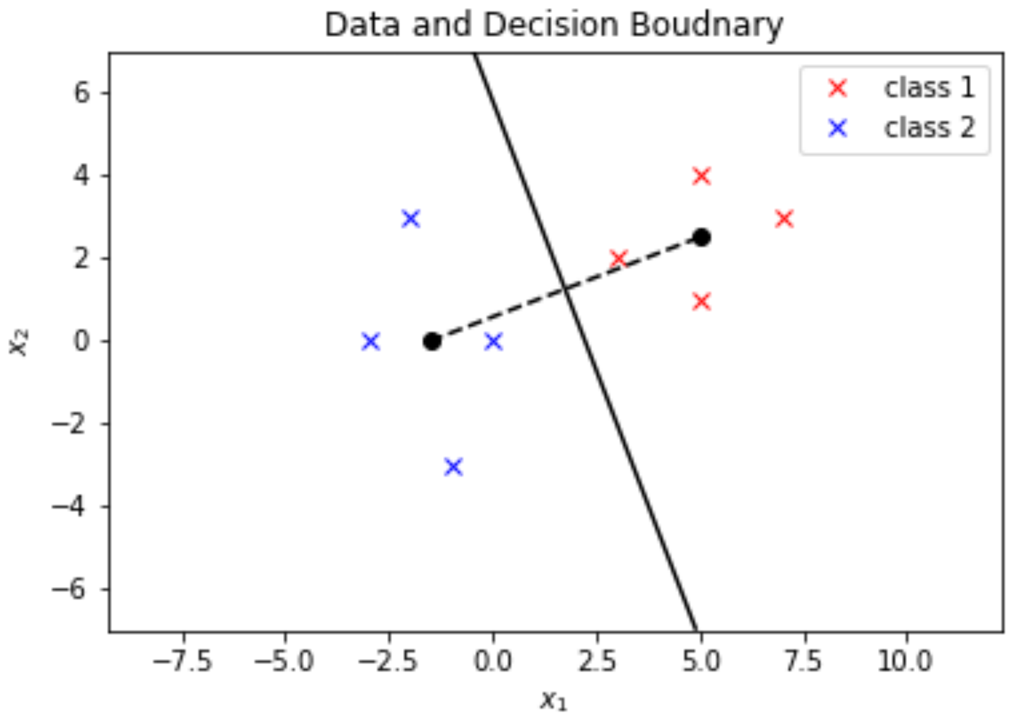
$$\boldsymbol{x}_5 = \begin{pmatrix}0\\0\end{pmatrix}, \boldsymbol{x}_6 = \begin{pmatrix}-1\\-3\end{pmatrix}, \boldsymbol{x}_7 = \begin{pmatrix}-2\\3\end{pmatrix}, \boldsymbol{x}_8 = \begin{pmatrix}-3\\0\end{pmatrix} \rightarrow \text{class 2}$$

Center of class-1: $\mu_1 = \frac{1}{4}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3 + \boldsymbol{x}_4) = \begin{pmatrix}5.0\\2.5\end{pmatrix}$

For class-2: $\mu_2 = \frac{1}{4}(\boldsymbol{x}_5 + \boldsymbol{x}_6 + \boldsymbol{x}_7 + \boldsymbol{x}_8) = \begin{pmatrix}-1.5\\0.0\end{pmatrix}$



The two classes are linearly separable!



Data and Decision Boudnary
class 1
class 2
6
4
2
0
−2
−4
−6
x_2
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x_1

The vector connecting two centroids = $\mu_1 - \mu_2$ = $\begin{pmatrix} 6.5 \\ 2.5 \end{pmatrix}$ (1)

The middle point connecting two centroids = $\frac{1}{2}(\mu_1 + \mu_2)$ = $\frac{1}{2}\left[\begin{pmatrix} 5.0 \\ 2.5 \end{pmatrix} + \begin{pmatrix} -1.5 \\ 0.0 \end{pmatrix}\right]$ = $\begin{pmatrix} 1.75 \\ 1.25 \end{pmatrix}$

If any point $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ on the boundary line,

the vector connecting that point to the middle point = $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 1.75 \\ 1.25 \end{pmatrix}$ = $\begin{pmatrix} x_1 - 1.75 \\ x_2 - 1.25 \end{pmatrix}$ (2)

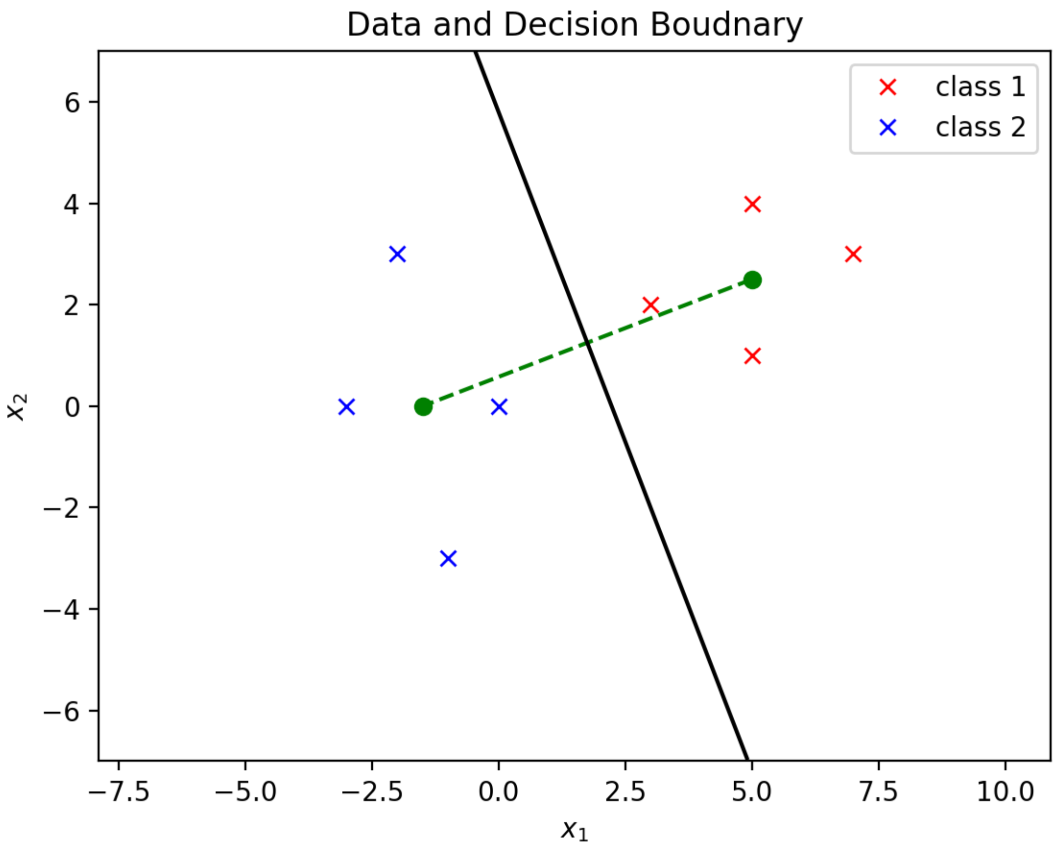
Since the vectors (1) and (2) are normal, their **inner product** should be zero.

$$\begin{pmatrix} 6.5 & 2.5 \end{pmatrix} \begin{pmatrix} x_1 - 1.75 \\ x_2 - 1.25 \end{pmatrix} = 0$$
$$6.5(x_1 - 1.75) + 2.5(x_2 - 1.25) = 0$$
$$6.5x_1 + 2.5x_2 - 14.5 = 0$$

(c) Design a discrete perceptron having the decision boundary as in part (b) for the classification.

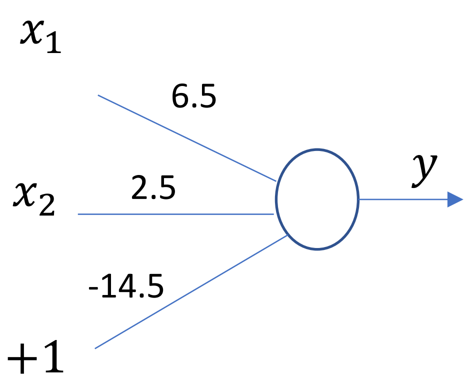
(d) Determine the classes identified by the neuron for following input patterns:

$$\begin{pmatrix}4\\2\end{pmatrix}, \begin{pmatrix}0\\5\end{pmatrix}, \begin{pmatrix}36/13\\0\end{pmatrix}$$



$$u = 6.5x_1 + 2.5x_2 - 14.5 = 0$$

Discrete perceptron:



Weights:

$$\boldsymbol{w} = \begin{pmatrix} 6.5 \\ 2.5 \end{pmatrix}$$

Bias:

$$b = -14.5$$

$$u = 6.5x_1 + 2.5x_2 - 14.5$$

$$u = \boldsymbol{w}^T\boldsymbol{x} + b = (6.5 \quad 2.5)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 14.5$$

Training patterns

$\boldsymbol{x}$	$\begin{pmatrix} 5 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 7 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 5 \\ 4 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -1 \\ -3 \end{pmatrix}$	$\begin{pmatrix} -2 \\ 3 \end{pmatrix}$	$\begin{pmatrix} -3 \\ 0 \end{pmatrix}$
$\boldsymbol{u}$	20.5	38.5	10.0	28.0	-14.5	-28.5	-20.0	-34.0
class	1	1	1	1	2	2	2	2

$$u > 0.0 \rightarrow class\ 1$$
$$u \leq 0.0 \rightarrow class\ 2$$

Perfectly classified!

Test patterns:

$$\boldsymbol{x} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}, u = 16.5 > 0 \rightarrow class\ 1$$

$$\boldsymbol{x} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}, u = -2 \leq 0 \rightarrow class\ 2$$

$$\boldsymbol{x} = \begin{pmatrix} 36/13 \\ 0 \end{pmatrix}, u = 3.5 > 0 \rightarrow class\ 1$$

3. Use ‘make_blobs’ function from **sklearn**.datasets to create 100 samples of two Gaussian distributed classes for 3-dimensional inputs:

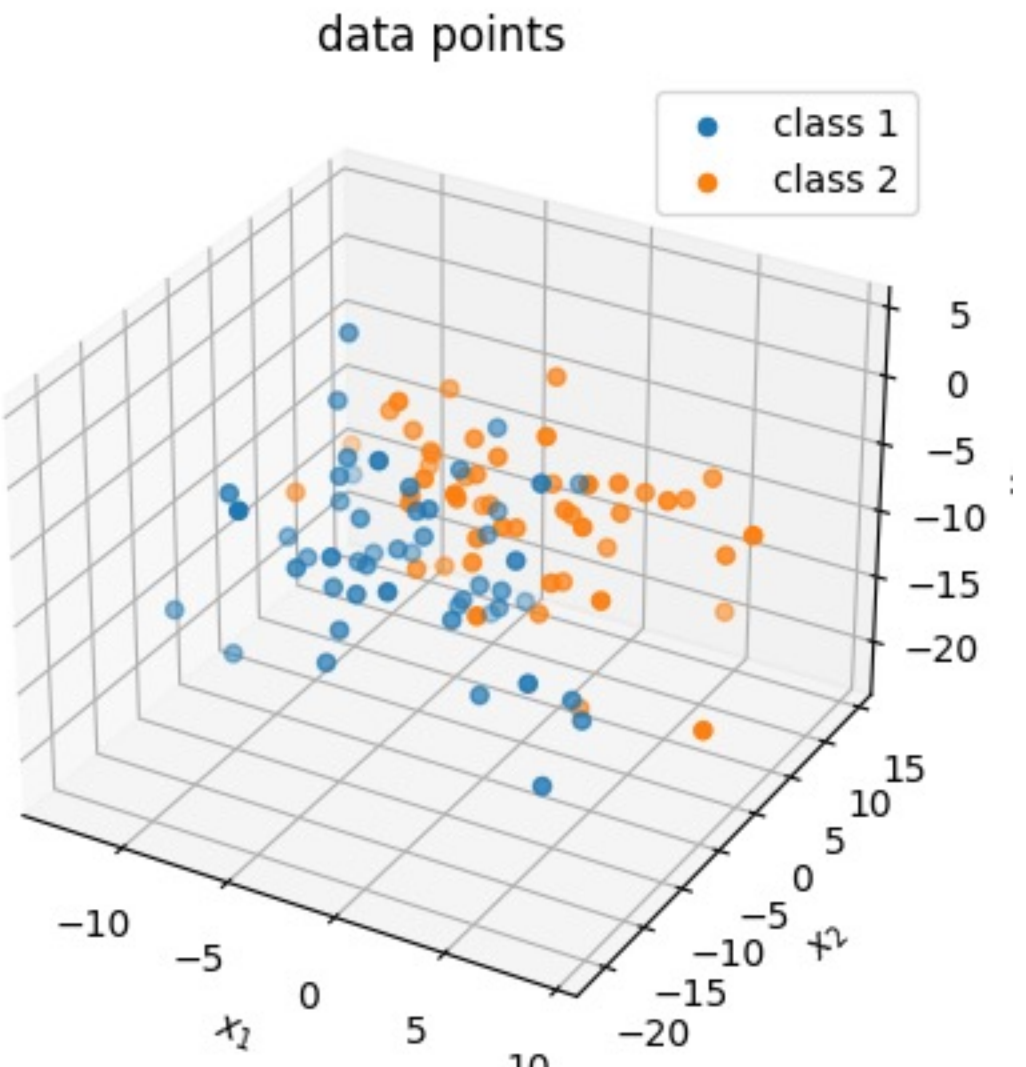
   from **sklearn**.datasets import make_blobs

   Assume each class has a standard deviation = 5.0.

   Use torch.nn.BCELoss() and torch.autograd() functions to train a logistic neuron to separate the two classes.
   Find the classification error at convergence and plot the decision boundary.

```
from sklearn.datasets import make_blobs

# define dataset
X, y = make_blobs(n_samples=100, n_features=3, centers=2, cluster_std=5, random_state=1)

# print first few examples
for i in range(5):
	print(X[i], y[i])
```

```
[ 5.82704597 -8.69737967 -14.86660705] 1
[ -5.95713961 6.15921976 -16.55912956] 0
[-7.16265579 10.13010842 -5.4897589 ] 0
[-5.19652244 -6.84653722 -9.28479932] 1
[ 0.9050892 2.91602569 -7.55512177] 0
```

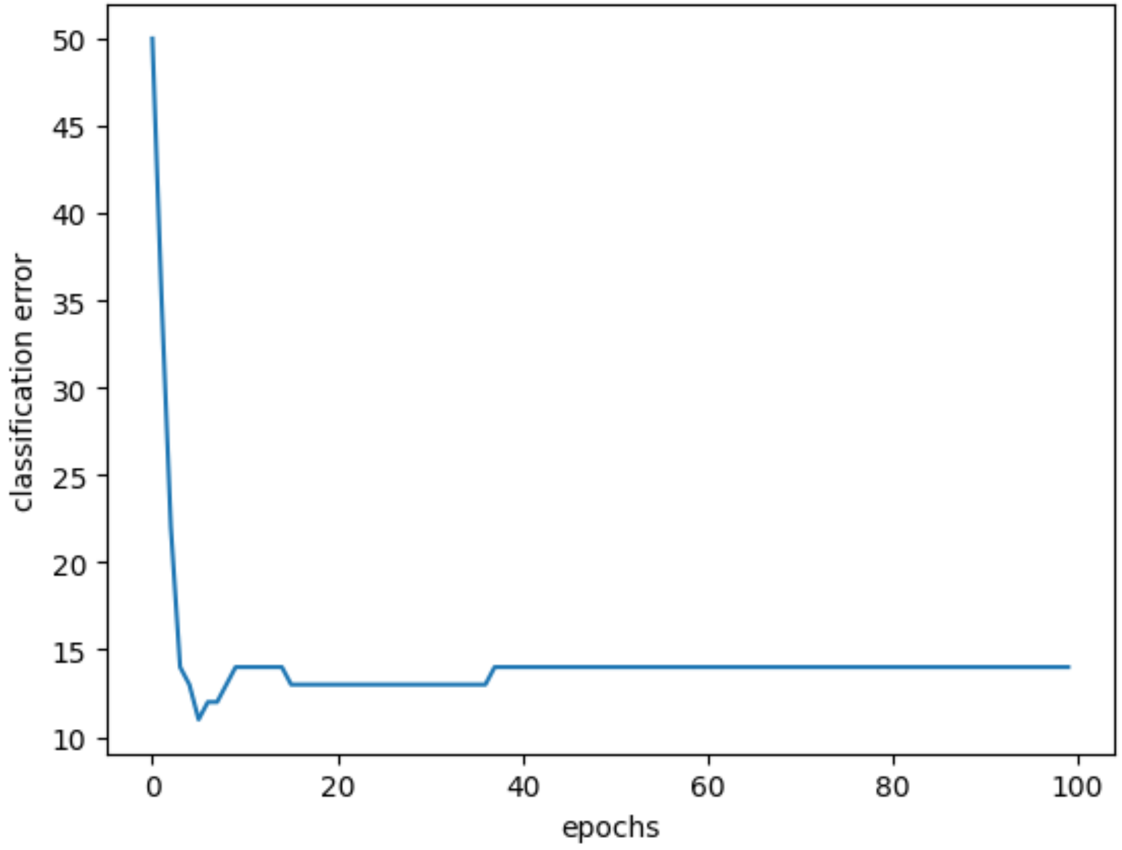

data points
class 1
class 2
5
0
−5
−10
−15
−20
15
10
5
0
−5
−10
−15
−20
x_2
−10
−5
0
5
10
x_1

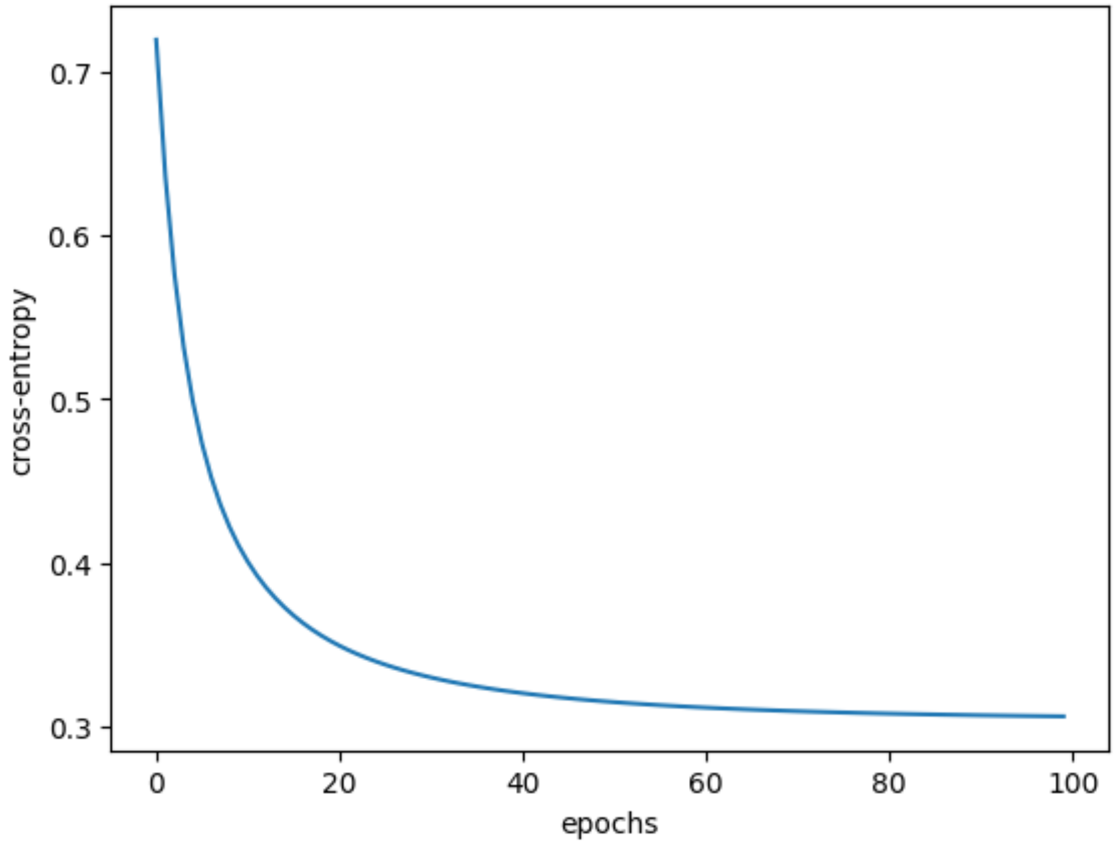
```python
import torch
import numpy as np


# create a class for a logistic neuron
class Logistic():
  def __init__(self):
    self.w = torch.tensor(0.05*np.random.rand(3), dtype=torch.double, requires_grad=True)
    self.b = torch.tensor(0., dtype=torch.double, requires_grad=True)

  def __call__(self, x):
    u = torch.matmul(torch.tensor(x), self.w) + self.b
    logits = torch.sigmoid(u)

    return logits
```

```python
import torch.nn as nn
loss = nn.BCELoss()

# create a logistic neuron object
model = Logistic()

# one iteration of training
def train(model, inputs, targets, learning_rate):

  logits = model(inputs)
  loss_ = loss(logits, torch.tensor(targets, dtype=torch.double))
  err = torch.sum(torch.not_equal(logits > 0.5, torch.tensor(targets)))

  loss_.backward()

  with torch.no_grad():
    model.w -= learning_rate * model.w.grad
    model.b -= learning_rate * model.b.grad

  model.w.grad = None
  model.b.grad = None

  return loss_, err
```

```python
# training begins
entropy, err = [], []
for epoch in range(100):
entropy_, err_ = train(model, X, y, lr)

    entropy.append(entropy_.detach().numpy())
    err.append(err_.detach().numpy())
```

Learning curves





At convergence:

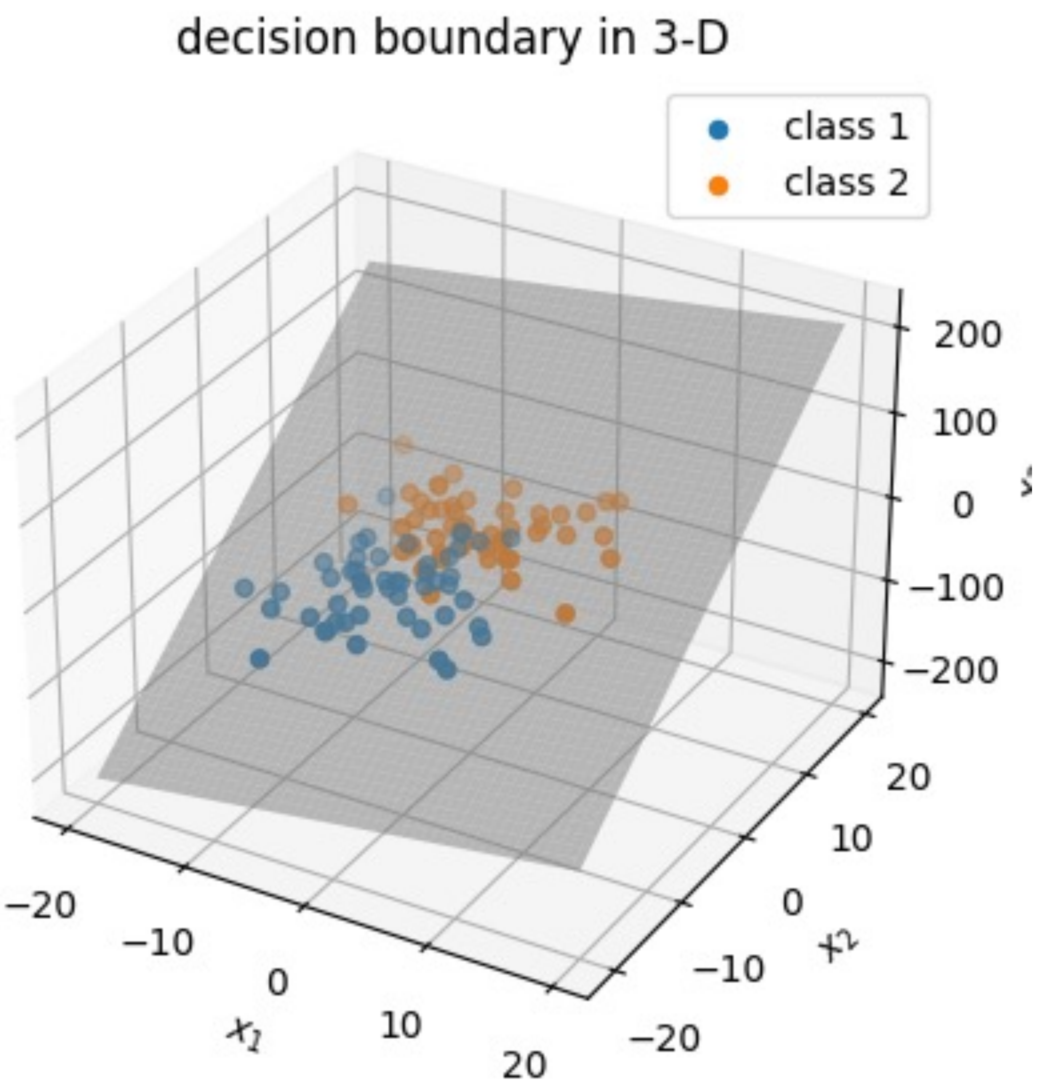
w: [-0.07863051 -0.38637461 0.04367386],
b: -0.0053
entropy: 0.30624,
error: 14

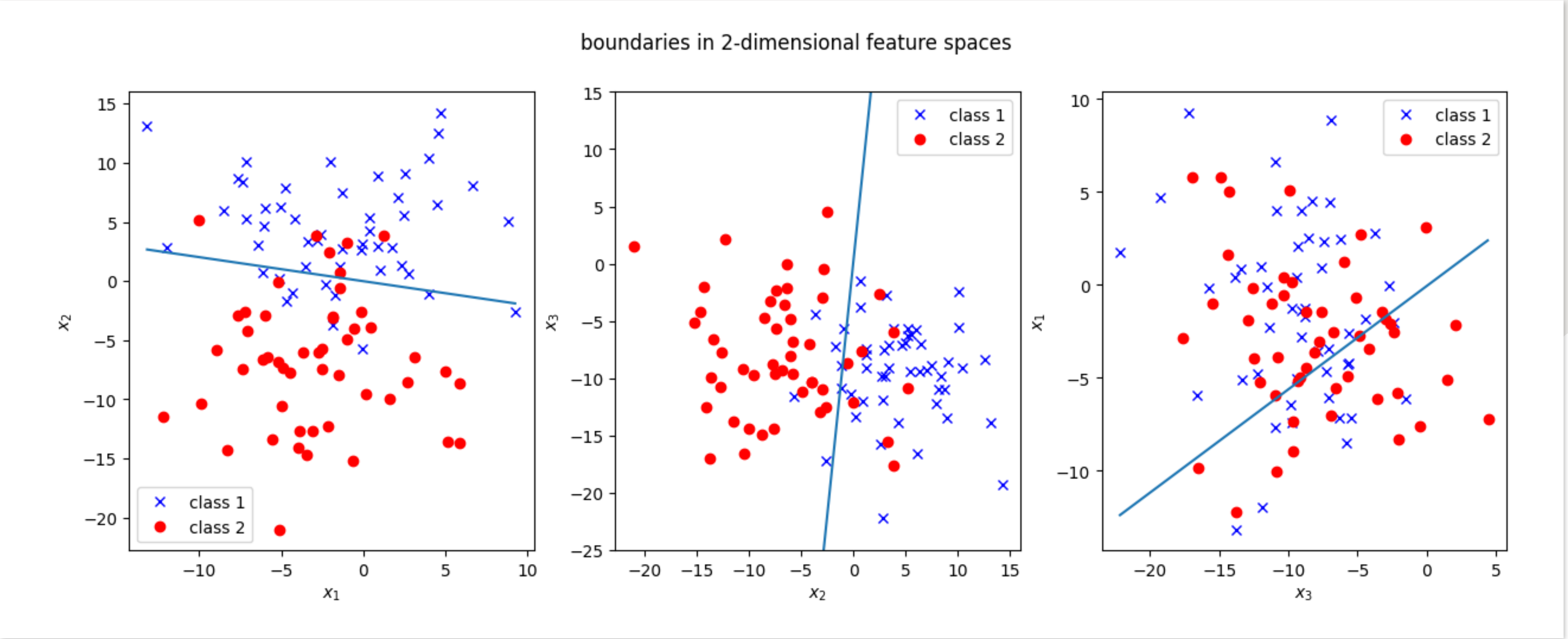
Decision boundary:

$$u = \boldsymbol{w}^T \boldsymbol{x} + b = 0$$

$$(-0.079 \quad -0.386 \quad 0.044)\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - 0.005 = 0$$

$$-0.079x_1 - 0.386x_2 + 0.044x_3 + 0.005 = 0$$





boundaries in 2-dimensional feature spaces
class 1
class 2
x_1
x_2
x_3

4. Train a perceptron to learn the following function ϕ:

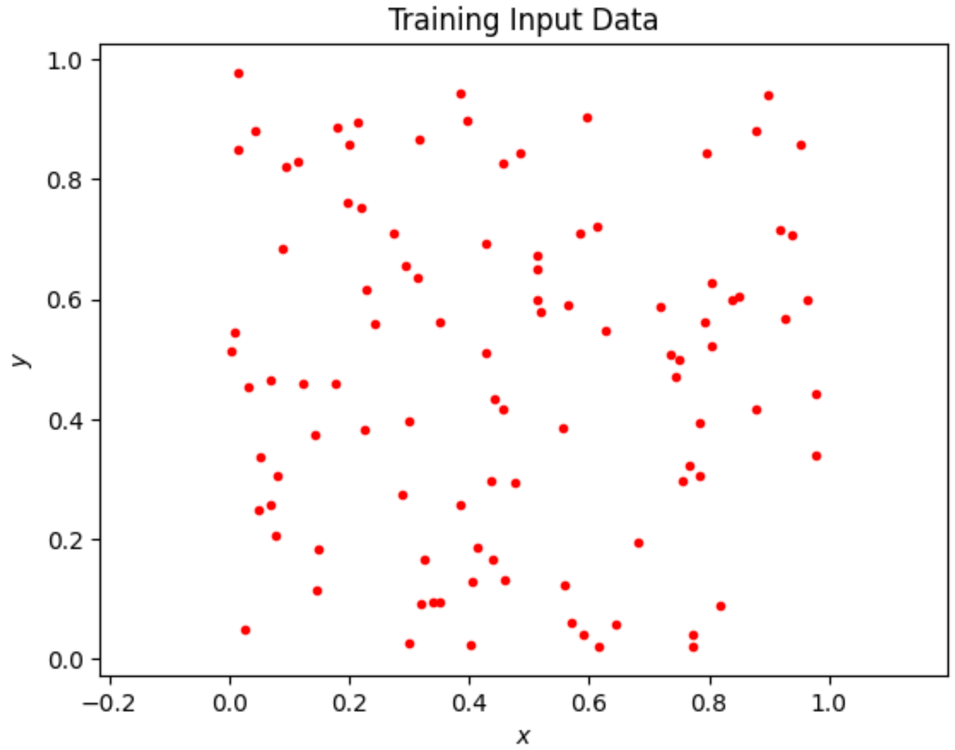
$$\phi(x, y) = 1.5 + 3.3x - 2.5y + 1.2xy$$

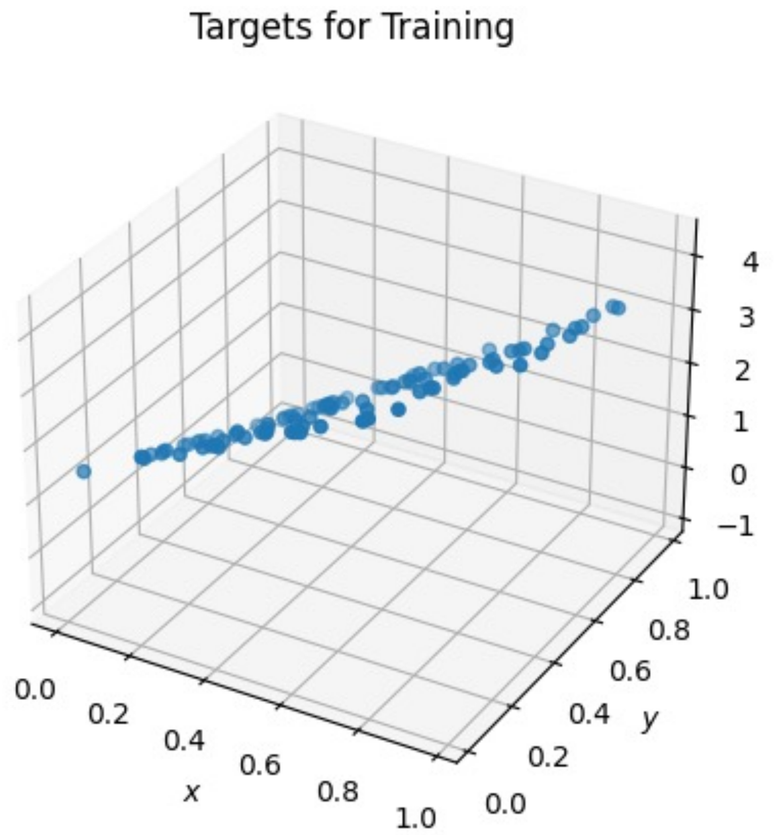
for $0 \leq x, y \leq 1$.

(a) Sample 100 data points randomly from the input space to create a training dataset.
(b) Use the gradient descent algorithm to train the perceptron
(c) Compute the training error and plot the function approximated by the perceptron.
(d) Show how a linear neuron can be used to predict the above function
(e) Compare the results of approximations above by the linear neuron and the perceptron

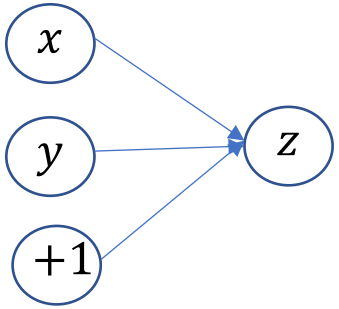
$$z = \phi(x, y) = 1.5 + 3.3x - 2.5y + 1.2xy \quad \text{for every } 0 \le x, y \le 1$$

Training dataset was created by randomly sampling the input feature space and computing the corresponding labels.





For **perceptron**, the activation function is a sigmoidal and the range of output should be known.



Let:

$$z = \phi(x, y) = 1.5 + 3.3x - 2.5y + 1.2xy$$

where $0 \le x, y \le 1$

Differentiating the function to find the maximum and minimum points:

.

$$\frac{\partial z}{\partial x} = 3.3 + 1.2y = 0 \ \rightarrow y = -2.75$$

$$\frac{\partial z}{\partial y} = -2.5 + 1.2x = 0 \ \rightarrow x = 2.08$$

That is, maximum/minimum occurs outside the given region.
The maximum/minimum occur at boundary points for the given input region.

(x, y)	$(\mathbf{0}, \mathbf{0})$	$(\mathbf{1}, \mathbf{0})$	$(\mathbf{0}, \mathbf{1})$	$(\mathbf{1}, \mathbf{1})$
z	1.5	4.8	-1.0	3.5

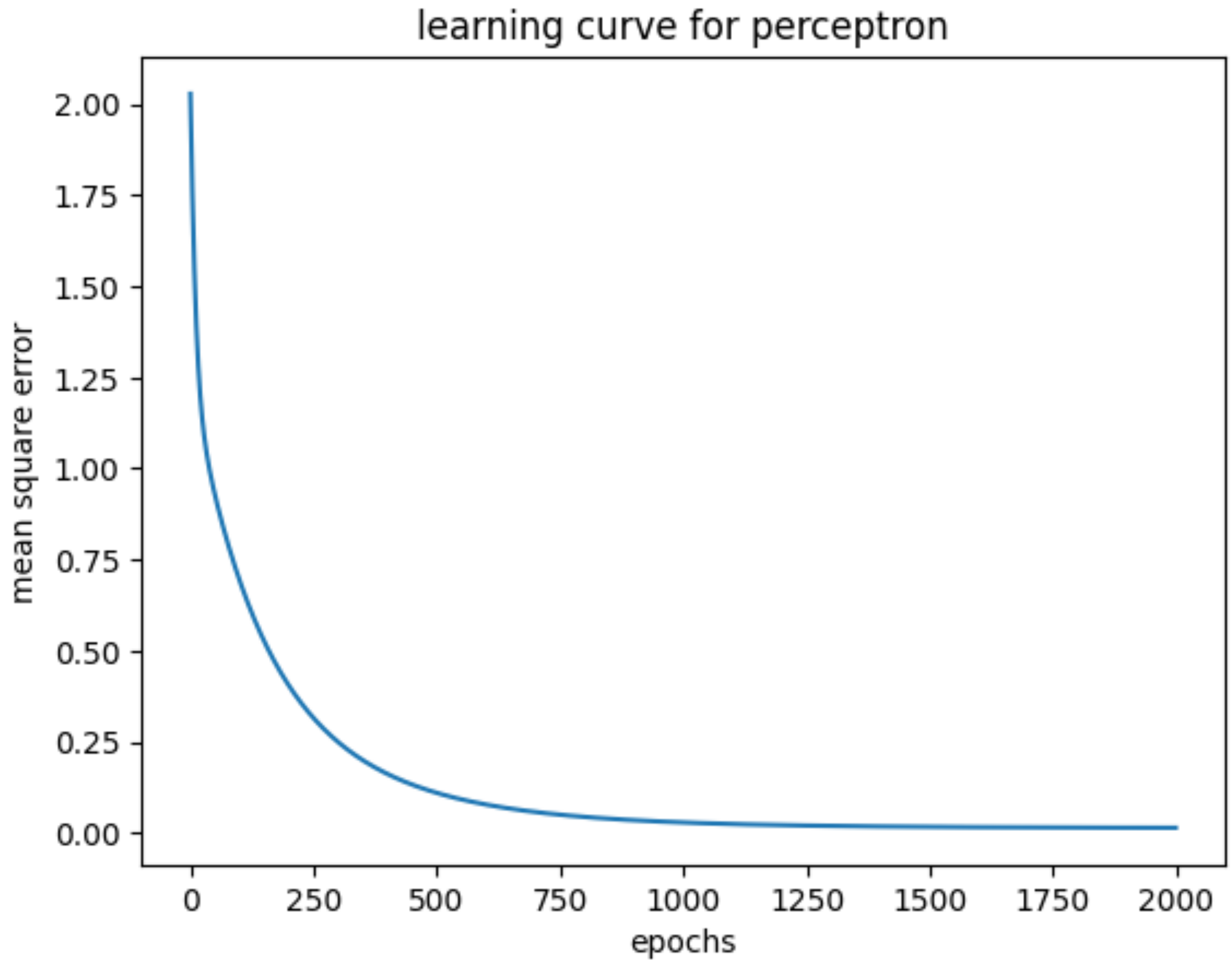
Therefore, $z \in [-1.0, 4.8]$

For the perceptron, activation function:

$$f(u) = \frac{5.8}{1 + e^{-u}} - 1.0$$

```python
# a class for the preceptron
class Perceptron():
  def __init__(self):
    self.w = torch.tensor(np.random.rand(2,1), dtype=torch.double, requires_grad=True)
    self.b = torch.tensor(0., dtype=torch.double, requires_grad=True)

  def __call__(self, x):
    u = torch.matmul(torch.tensor(x), self.w) + self.b
    y = 5.8*torch.sigmoid(u)-1.0
    return y

 # mean squared error as the loss function
 def loss_fn(y_pred, d):
    return torch.mean(torch.square(y_pred - d))


 # create a linear neuron
 model = Perceptron()
```

```python
# training

idx = np.arange(no_data)

for epoch in range(no_epochs):

    np.random.shuffle(idx)
    XX, YY = X[idx], Y[idx]

    y_ = model(XX)
    loss_ = loss_fn(y_, torch.tensor(YY))

    loss_.backward()

    with torch.no_grad():
        model.w -= lr * model.w.grad
        model.b -= lr * model.b.grad

    model.w.grad = None
    model.b.grad = None
```


learning curve for perceptron
mean square error
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
250
500
750
1000
1250
1500
1750
2000
epochs

At convergence:

$\boldsymbol{w} = \begin{pmatrix} 2.95 \\ -1.91 \end{pmatrix}$, b = −0.48

mse = 0.014

If $\boldsymbol{x} = (x, y)^T$

$$u = \boldsymbol{w}^T \boldsymbol{x} + b$$

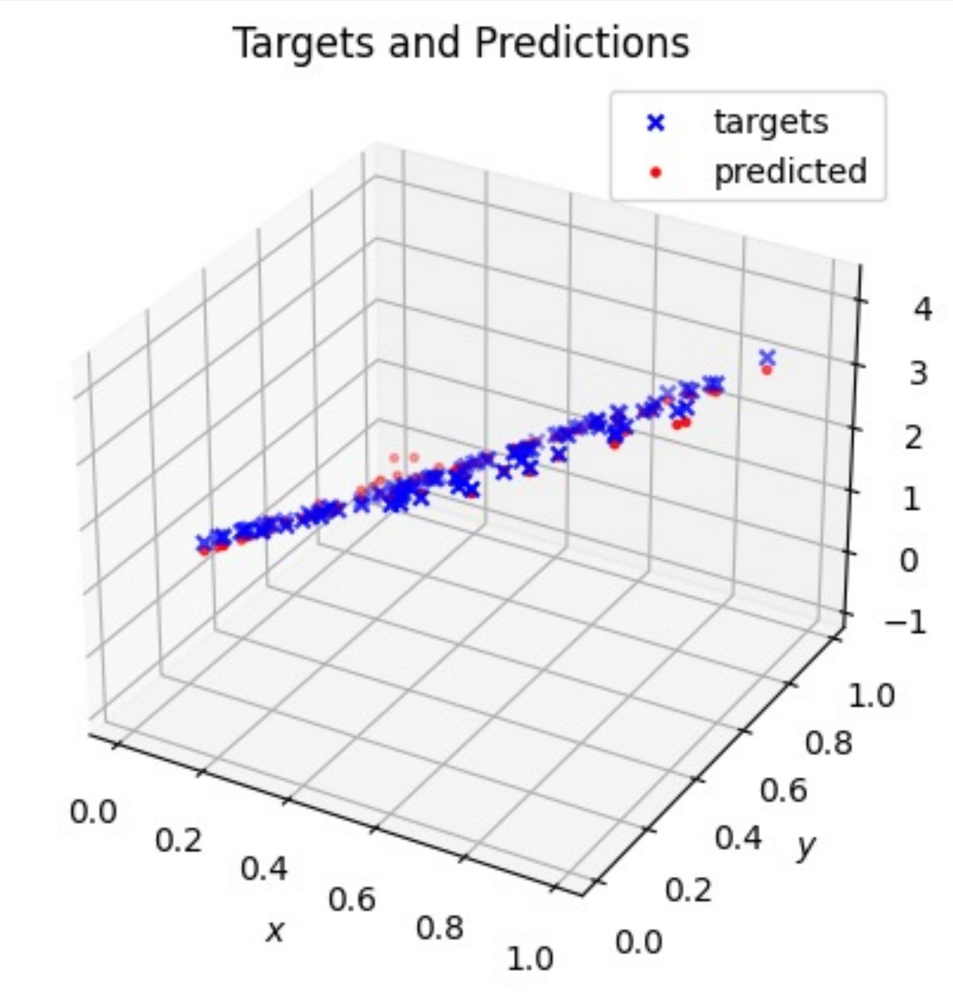
$$u = (2.95 \quad -1.91) \begin{pmatrix} x \\ y \end{pmatrix} - 0.48$$

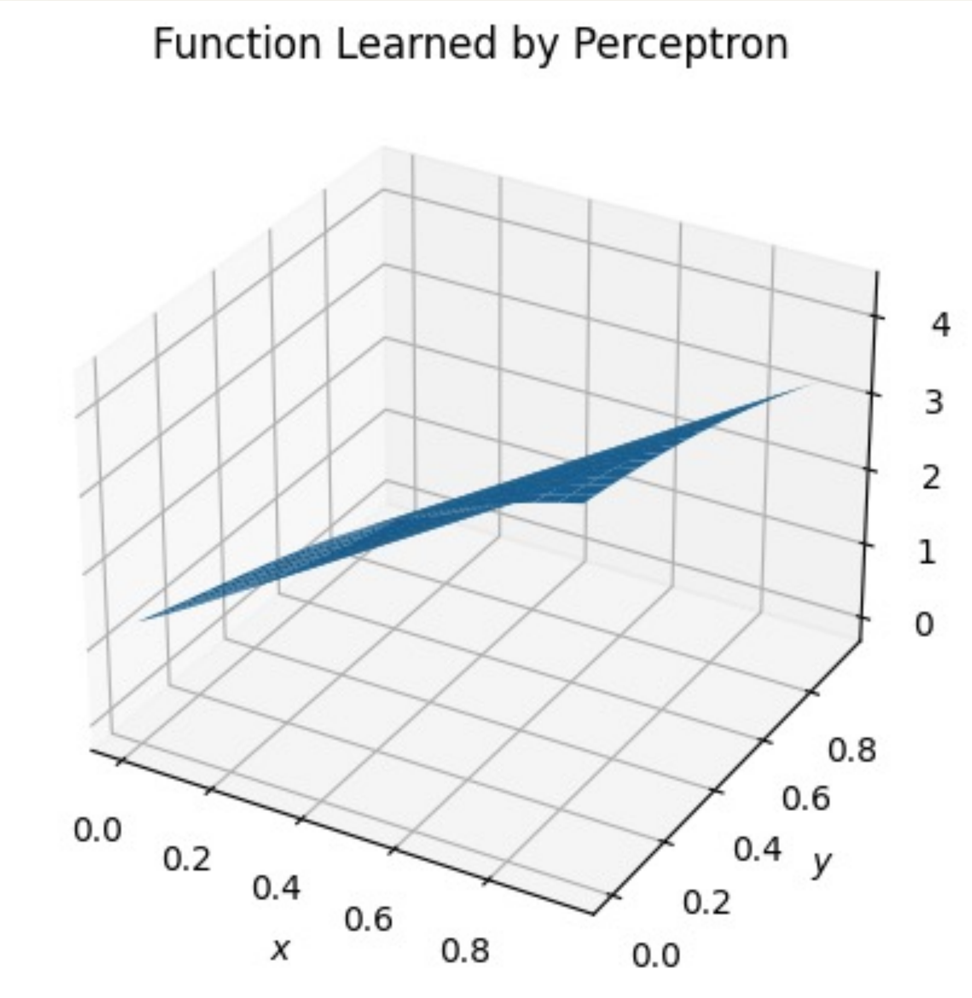
$$u = 2.95x - 1.91y - 0.48$$

The learned function by the perceptron:

$$z = \frac{5.8}{1 + e^{-u}} - 1.0$$

$$z = \frac{5.8}{1 + e^{0.48 - 2.95x + 1.91y}} - 1.0$$


Targets and Predictions
targets
predicted
x
y
0.0
0.2
0.4
0.6
0.8
1.0
−1
0
1
2
3
4

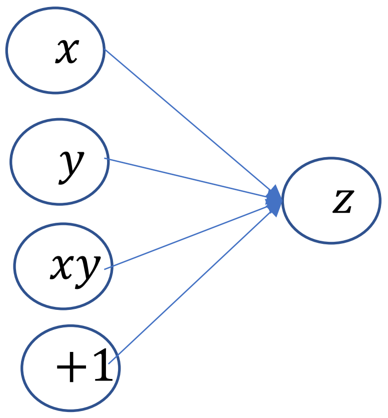

Function Learned by Perceptron
x
y
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4

$$z = 1.5 + 3.3x - 2.5y + 1.2xy$$

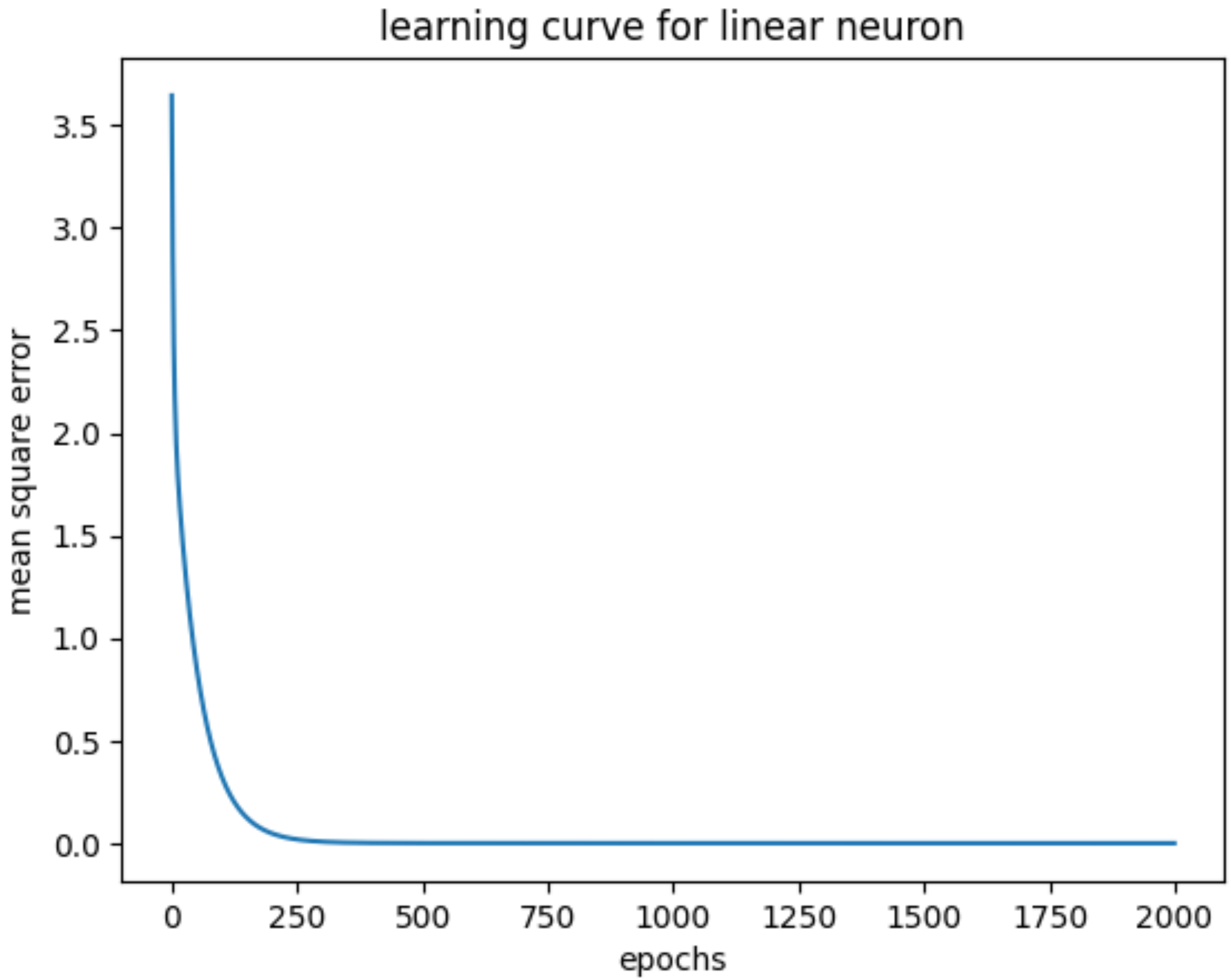
Linear neuron learns a linear function. The above equation can be written as a linear equation:

$$z = 1.5 + 3.3x_1 - 2.5x_2 + 1.2x_3$$

where the linear neuron receives 3 inputs: $x_1 = x$, $x_2 = y$, and $x_3 = xy$.



$$f(u) = u$$


learning curve for linear neuron
mean square error
epochs
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
250
500
750
1000
1250
1500
1750
2000

At convergence,

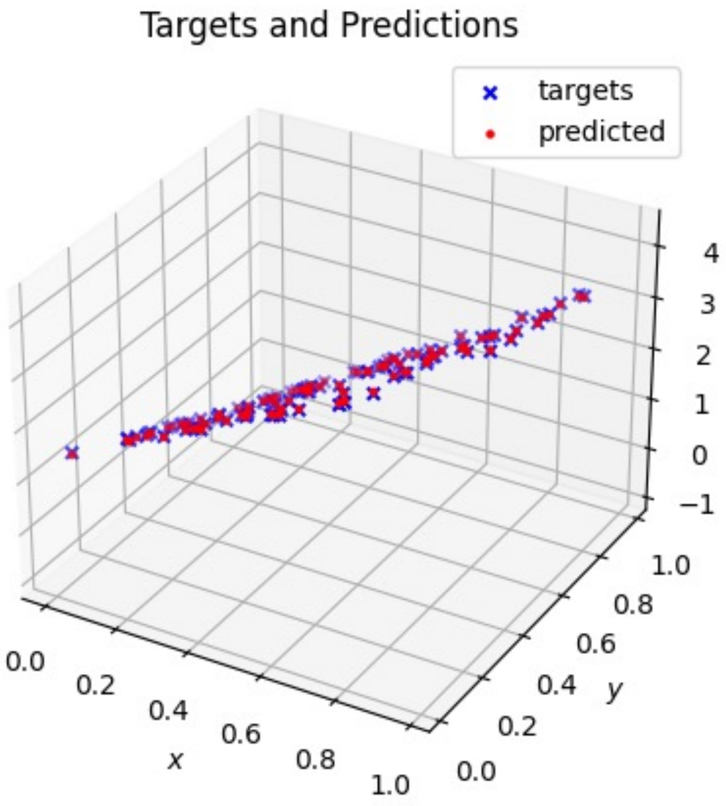
Weights $w = \begin{pmatrix} 3.40 \\ -2.41 \\ 1.02 \end{pmatrix}$ and bias $b = 1.45$

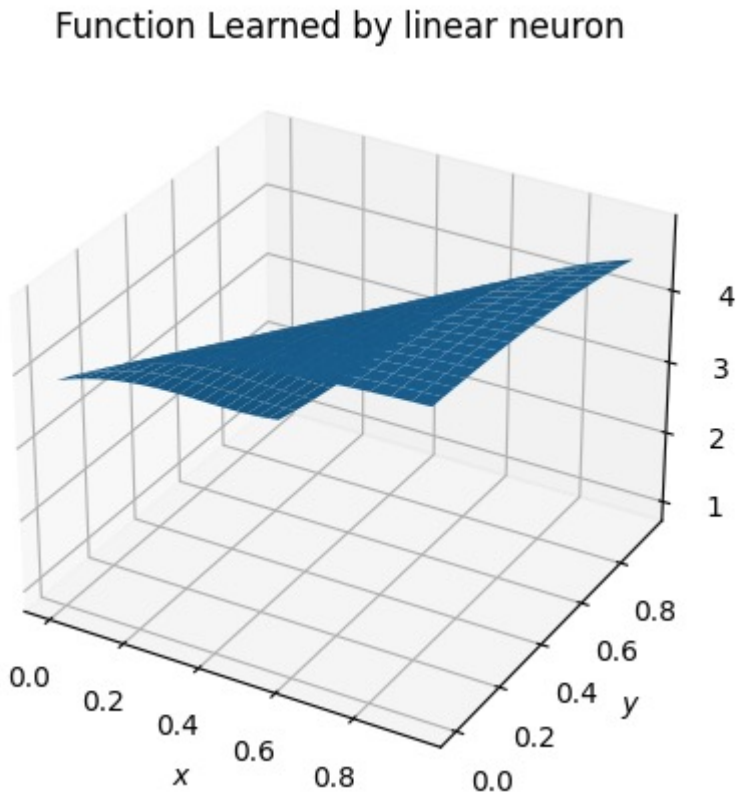
Mean square error = 1.8 x 10^{-4}

The function learned by the linear neuron:

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

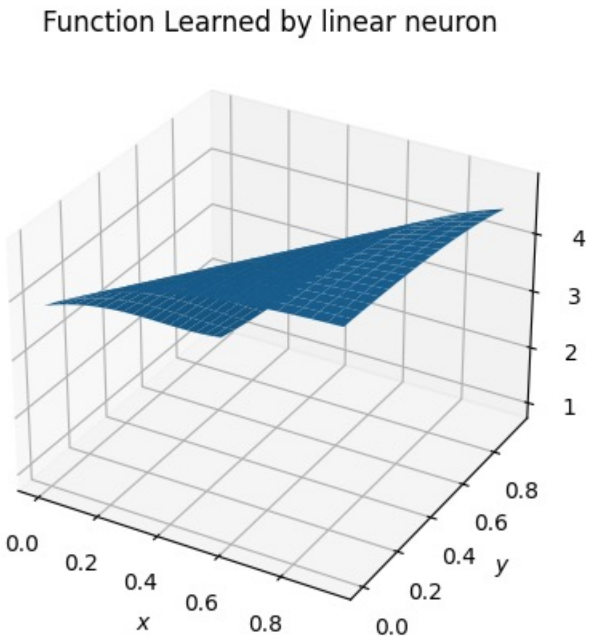
$$z = 3.40x - 2.41y + 1.02xy + 1.45$$


Targets and Predictions
targets
predicted
x
y
0.0
0.2
0.4
0.6
0.8
1.0
−1
0
1
2
3
4

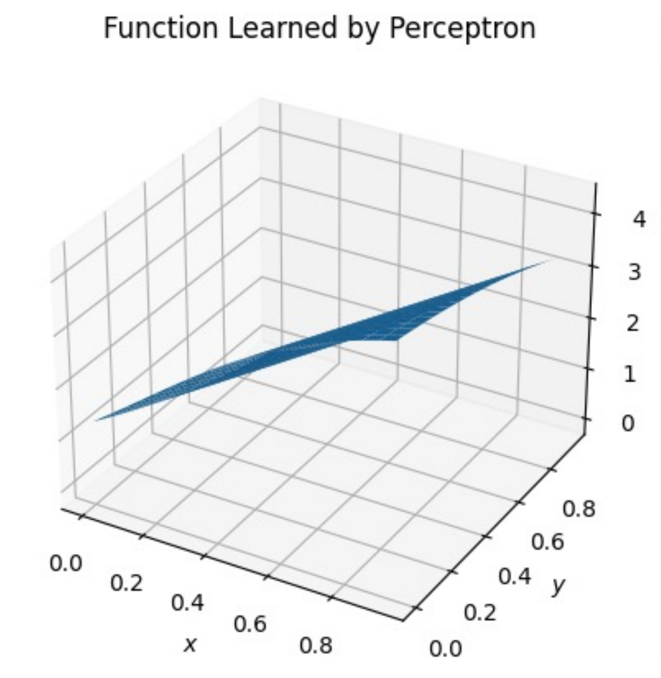

Function Learned by linear neuron
x
y
0.0
0.2
0.4
0.6
0.8
1
2
3
4

$$z = 1.5 + 3.3x - 2.5y + 0.2xy \qquad \text{for } 0 \leq x, y \leq 1$$

	Linear Neuron	Perceptron
$\boldsymbol{w}$	$\begin{pmatrix} 3.40 \\ -2.41 \\ 1.02 \end{pmatrix}$	$\begin{pmatrix} 2.95 \\ -1.91 \end{pmatrix}$
b	1.45	-0.48
m.s.e.	1. 8x 10^{-4}	0.014
function	$z = 3.4x - 2.41y + 1.02xy + 1.45$	$z = \dfrac{5.8}{1 + e^{0.48 - 2.95x + 1.91y}} - 1.0$



$$z = 3.4x - 2.41y + 1.02xy + 1.45$$



$$z = \frac{5.8}{1 + e^{0.48 - 2.95x + 1.91y}} - 1.0$$

- $z = 1.5 + 3.3x - 2.5y + 0.2xy$ for $0 \le x, y \le 1$