

Deep Learning & Neural Networks

Dr. Shahira Ahmed

DS710

Lecture 3



Deep Learning & Neural Networks DS710

Lecture 3
Training the Network-Part(1)



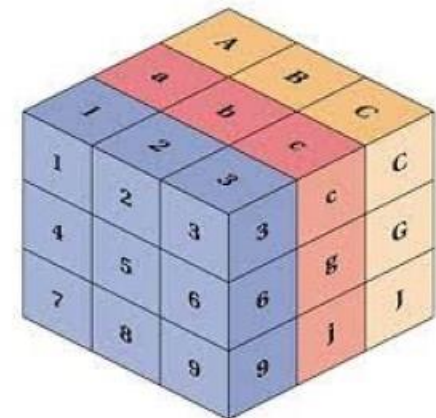
- Linear Algebra Review
- What is Vectorization?
- How Vectorization is important in Machine learning?
- Advantages of Vectorized Implementation
- Demonstration on Colab
- Representing Neural Network weighted sum as vectors
- Training Network

- Scalars is only a Single numbers
- Vectors: A vector is an array of numbers

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

- Matrix:
$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \dots & \dots & \dots & \dots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

- A tensor is a n-dimensional array with $n > 2$



Grayscale Image

6 x 6



6 x 6

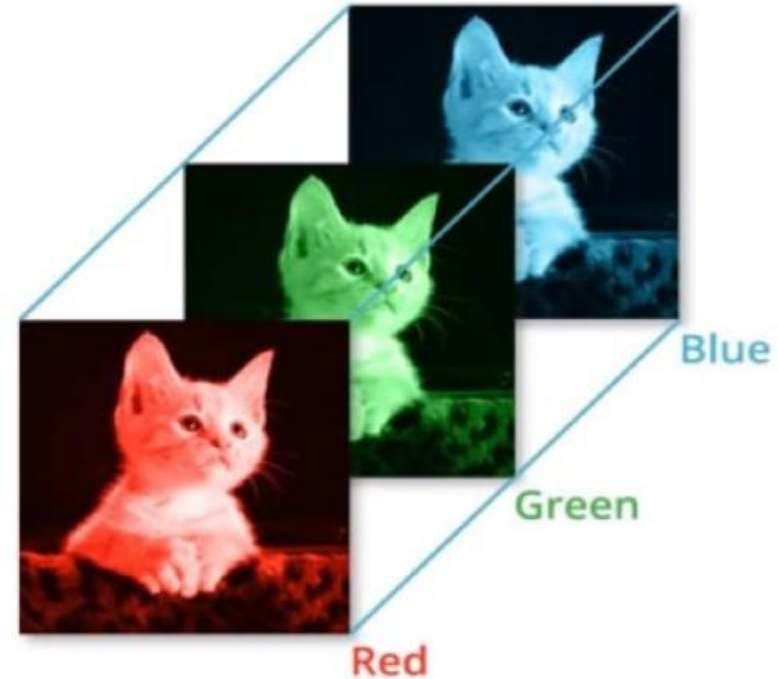
1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

<https://dev.to/sandeepbalachandran/machine-learning-going-further-with-cnn-part-2-41km>

Linear Algebra Review

RGB Image

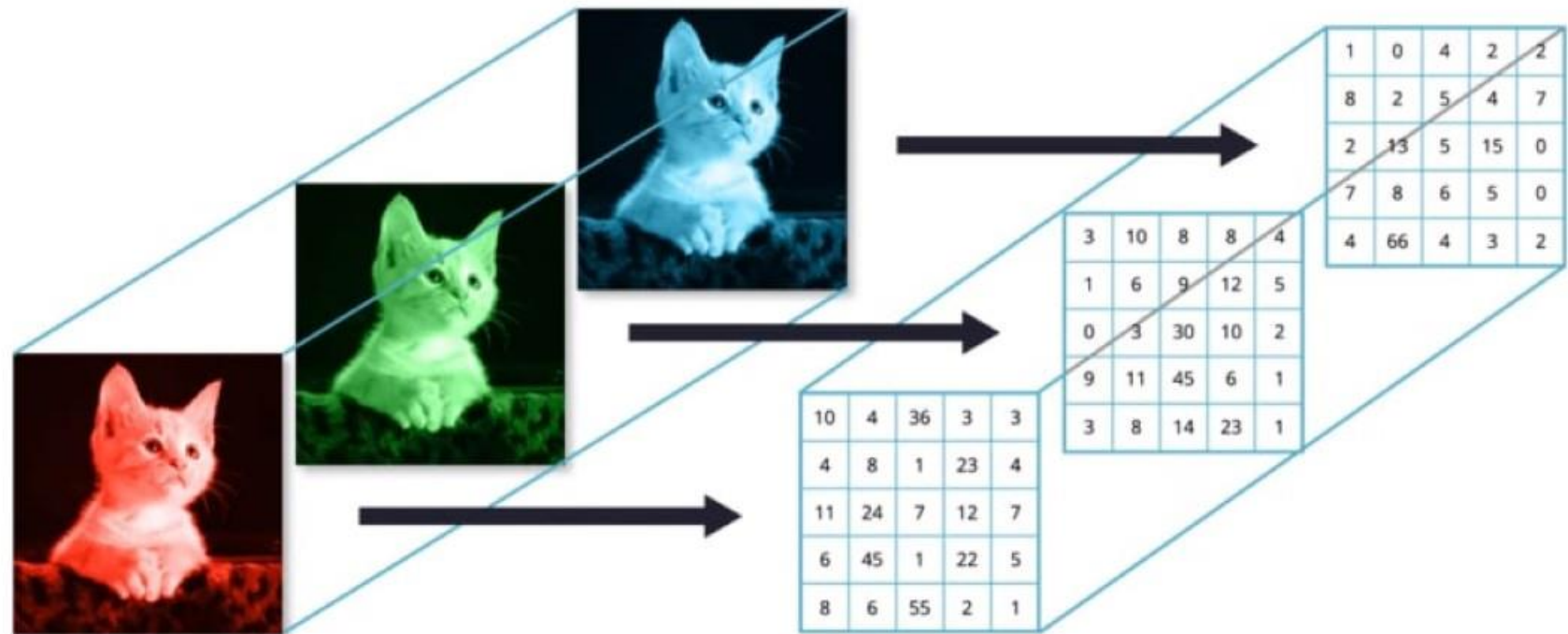
$5 \times 5 \times 3$



<https://dev.to/sandeepbalachandran/machine-learning-going-further-with-cnn-part-2-41km>



RGB Image



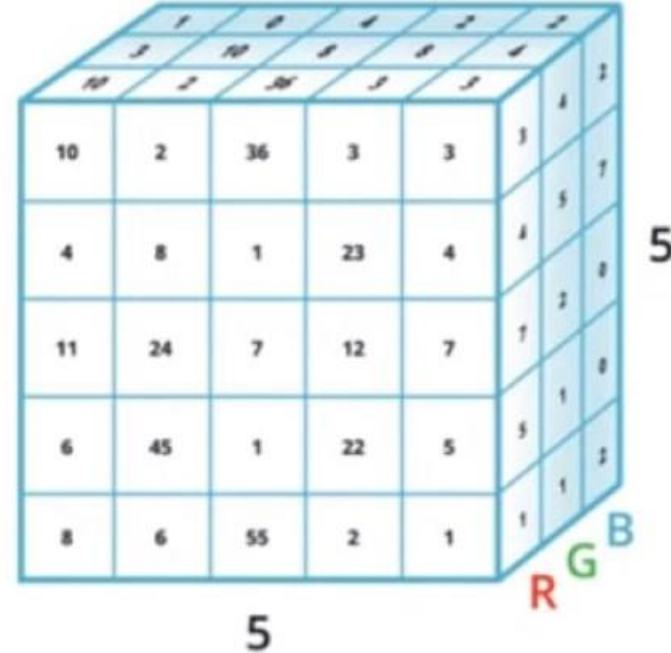
<https://dev.to/sandeepbalachandran/machine-learning-going-further-with-cnn-part-2-41km>

RGB Image

5 x 5 x 3



3D Array

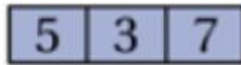


<https://dev.to/sandeepbalachandran/machine-learning-going-further-with-cnn-part-2-41km>

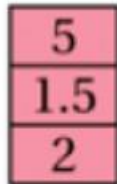
Linear Algebra Review

(11)

SCALAR



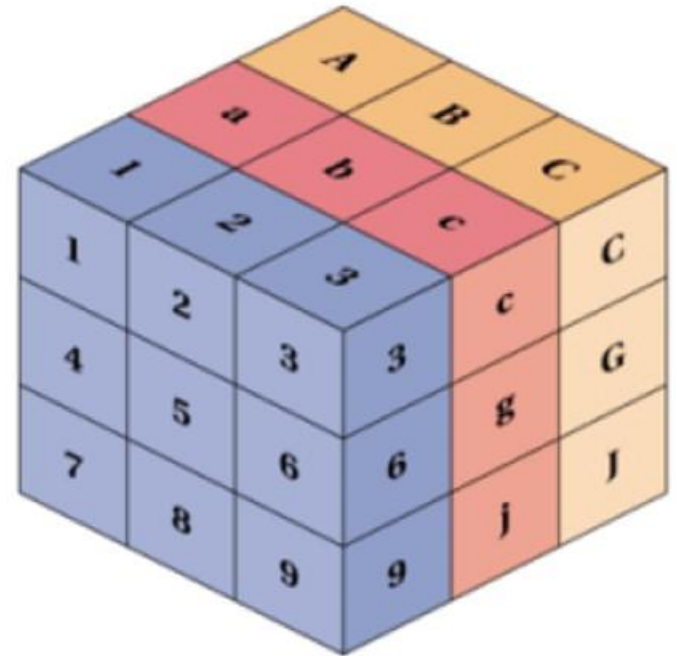
Row Vector
(shape 1x3)



Column Vector
(shape 3x1)



MATRIX



TENSOR

- An algebraic operation in which two equal length vectors are being multiplied such that it produces a single number. Dot Product often called as inner product. This product results in a scalar number

$$a \cdot b = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}_{(1 \times n)} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \left\{ a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5 \right\}$$

Dot Product

Dot Product

$$\begin{aligned} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} &= \begin{bmatrix} (1 \cdot 5) + (2 \cdot 7) & (1 \cdot 6) + (2 \cdot 8) \\ (3 \cdot 5) + (4 \cdot 7) & (3 \cdot 6) + (4 \cdot 8) \end{bmatrix} \\ &= \begin{bmatrix} 5 + 14 & 6 + 16 \\ 15 + 28 & 18 + 32 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix} \end{aligned}$$

- Multiply two vectors a and b with dimension $n \times 1$ and $m \times 1$ then the outer product of the vector results in a rectangular matrix of $n \times m$.

$$a \otimes b = ab^T = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}_{(n \times 1)} \begin{bmatrix} b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix}_{(1 \times n)} = \begin{Bmatrix} a_1b_1 & a_1b_2 & a_1b_3 & a_1b_4 & a_1b_5 \\ a_2b_1 & a_2b_2 & a_2b_3 & a_2b_4 & a_2b_5 \\ a_3b_1 & a_3b_2 & a_3b_3 & a_3b_4 & a_3b_5 \\ a_4b_1 & a_4b_2 & a_4b_3 & a_4b_4 & a_4b_5 \\ a_5b_1 & a_5b_2 & a_5b_3 & a_5b_4 & a_5b_5 \end{Bmatrix}_{(n \times n)}$$

Outer Product

Element-wise multiplication of two matrices is the algebraic operation in which each element of first matrix is multiplied by its corresponding element in the later matrix. Dimension of the matrices should be same.

$$a \circ b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}_{(n \times 1)} \circ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \begin{bmatrix} a_1b_1 \\ a_2b_2 \\ a_3b_3 \\ a_4b_4 \\ a_5b_5 \end{bmatrix}_{(n \times 1)}$$

Element wise Product

Transpose

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Input

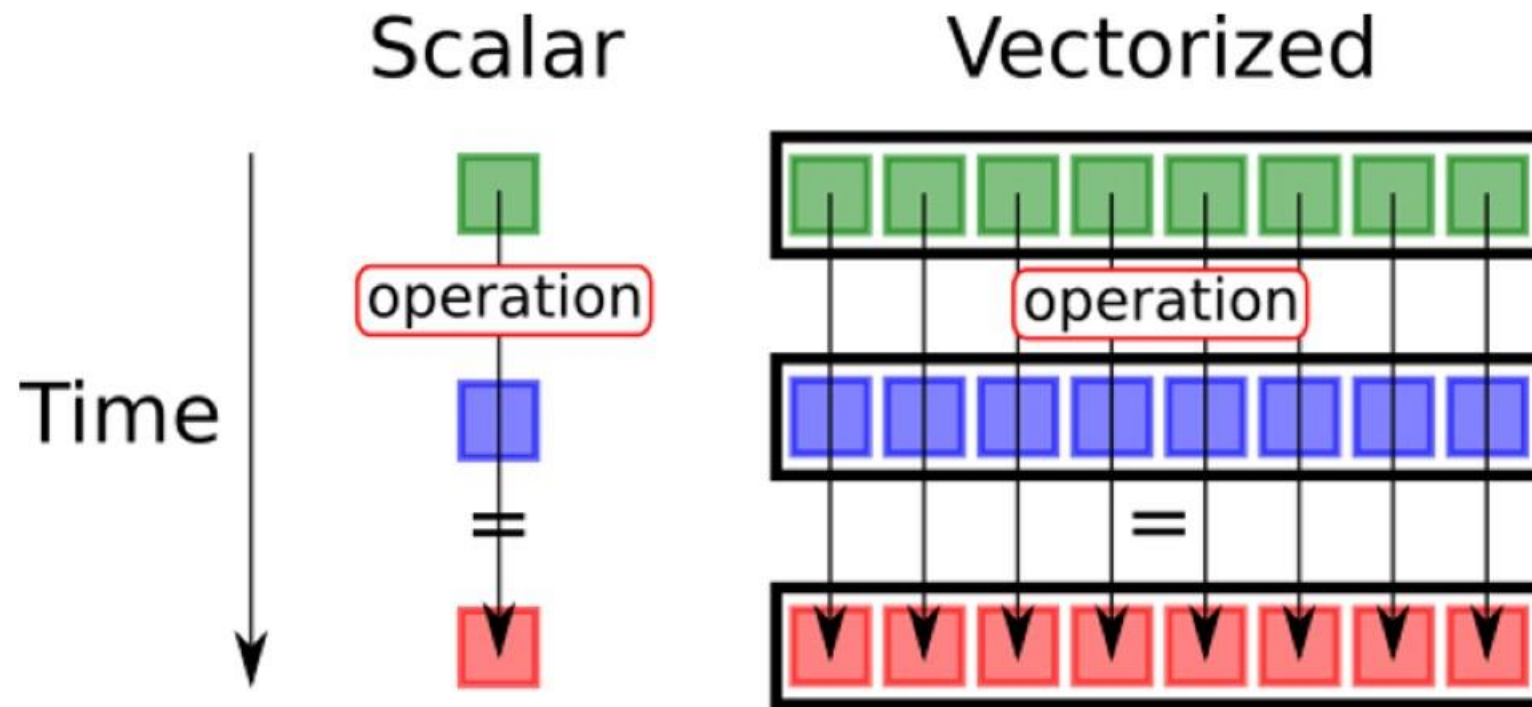
$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Output

What is Vectorization?

- Vectorization in machine learning refers to the process of represents data into Vectors, Matrices, and Tensors.
- Process data without using loops

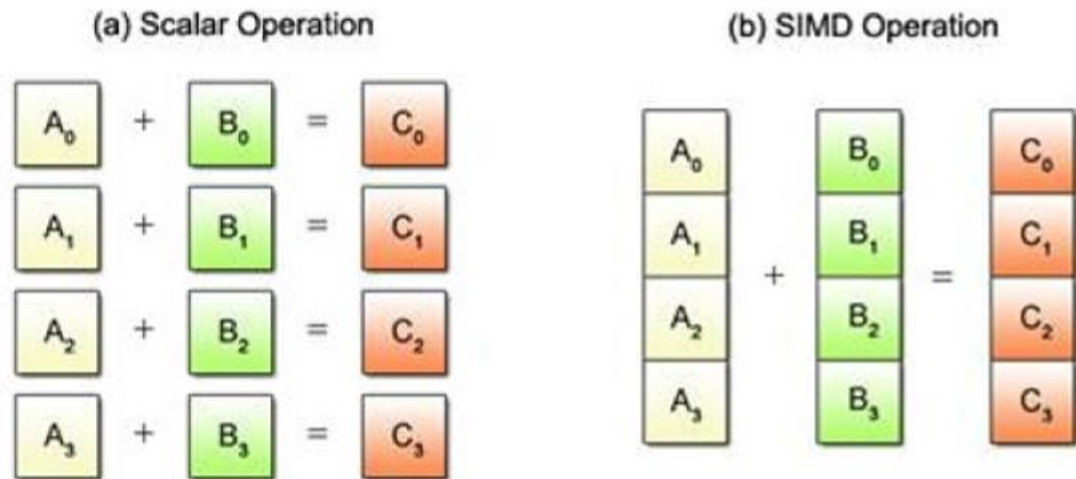
Why Vectorization



<https://medium.com/@jwbtmf/vectorization-in-deep-learning-c47f0d171d0a>

Why Vectorization?

SIMD is short for Single Instruction/Multiple Data, while the term SIMD operations refer to a computing method that enables the processing of multiple data with a single instruction



Why Vectorization?

- Efficiency : Due to the fact that they make use of the parallel processing capability of contemporary CPUs and GPUs, vectorized operations are often performed significantly quicker than their non-vectorized equivalents. This can lead to significant speed-ups in training and inference.
- Memory : Vectorized operations can also reduce memory usage.

Why Vectorization

- Express data as matrices
- Perform computations on multiple data points simultaneously
- Performing parallel computations using matrix operations
- Hardware accelerators like GPUs
- Efficient memory utilization
- Less Time- more speed

Why Vectorization Demo

This notebook is open with p

ple this in [Notebook settings](#).



Vectorization.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment

Share



+ Code + Text

✓ TPU RAM
Disk

```
print("Vectorized version=====>",VecTime,"ms")
#print ("DotProduct Results",C)
print("-----")
C=0
start=time.time()
for i in range(1000000):
    C+= A[i]*B[i]
end=time.time()
LoopTime=(end-start)*1000
#print ("DotProduct Results",C)
print("Loop version =====>",LoopTime,"ms")
print()
print ("loop to victorize=",LoopTime/VecTime)
```

```
Vectorized version=====> 11.303186416625977 ms
DotProduct Results 250246.30477494356
-----
DotProduct Results 250246.30477494773
Loop version =====>= 919.0149307250977 ms

loop to victorize= 81.30582800734038
```

Why Vectorization Demo

This notebook is open with p... | ...ple this in [Notebook](#)

Vectorization.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
print(
C=0
start=time.time()
for i in range(1000000):
    C+= A[i]*B[i]
end=time.time()
LoopTime=(end-start)*1000
#print ("DotProduct Results",C)
print("Loop version =====>=",LoopTime,"ms")
print()
print ("loop to victorize=",LoopTime/VecTime)
```

Vectorized version=====> 11.303186416625977 ms
DotProduct Results 250246.30477494356

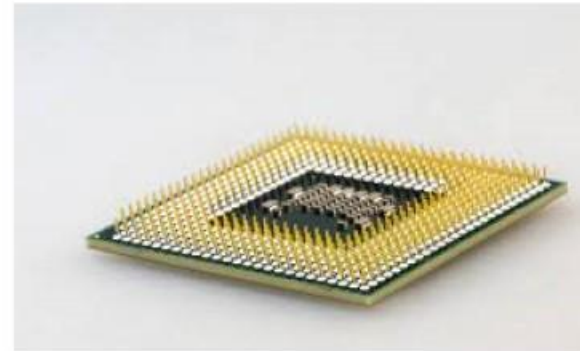
DotProduct Results 250246.30477494773
Loop version =====>= 919.0149307250977 ms

loop to victorize= 81.30582800734038

Machine Learning - Going Furth...

- **CPU (Central Processing Unit):**

- A CPU is the primary processing unit in a computer system. CPUs excel at sequential processing and are optimized for tasks that require high single-threaded performance. They are commonly found in desktop and laptop computers, servers, and mobile devices.



File Home Insert Draw Design Transitions Animations Sli

Pause 00:00:00 Select Area Audio Record Pointer

New Slide Reuse Slides Table Pictures Screenshot Photo Album Cameo Shapes Icons SmartArt Power BI Forms Links Comment Text Box & Footer Tools Media

Slides Tables Images Camera Illustrations Power BI Forms Comments Text

Mute Cancel Noise ON Talk Time 00:00 (0%) Teams

CPU

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

- Click to add text

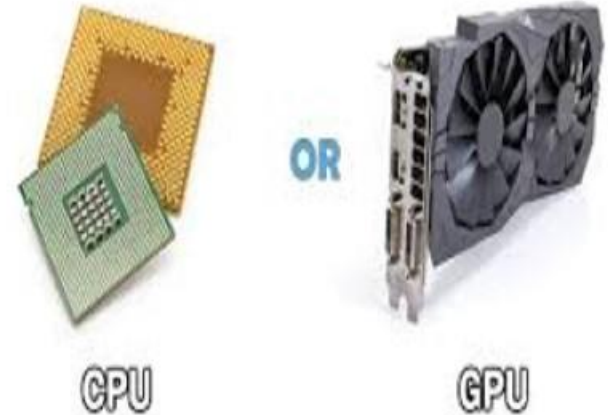
www.midocean.ae

ie 24 of 65 English (United States) Accessibility: Investigate

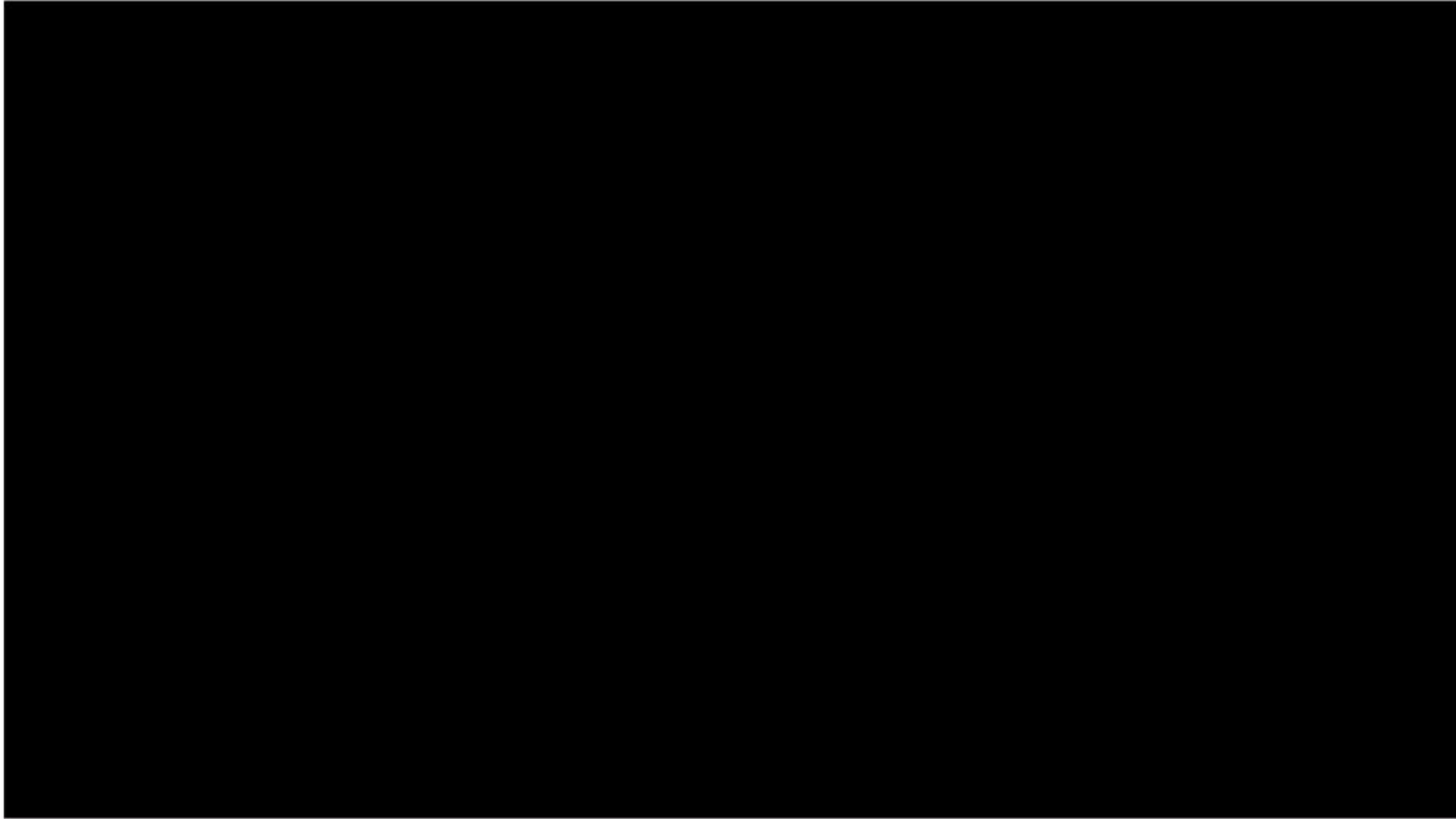
Notes

- **GPU (Graphics Processing Unit):**

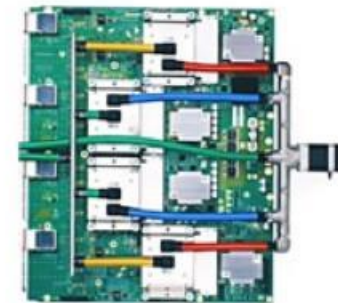
- A GPU is a specialized processor that is designed to handle computationally intensive tasks related to computer graphics and parallel processing. Originally developed for rendering graphics in video games, GPUs have evolved into highly parallel processors capable of performing complex mathematical calculations. They excel at tasks that can be broken down into many smaller, independent tasks that can be executed simultaneously. GPUs are commonly used in graphics-intensive applications, such as gaming, image processing, scientific simulations, and machine learning.



- <https://www.youtube.com/watch?v=-P28LKWTzrl>

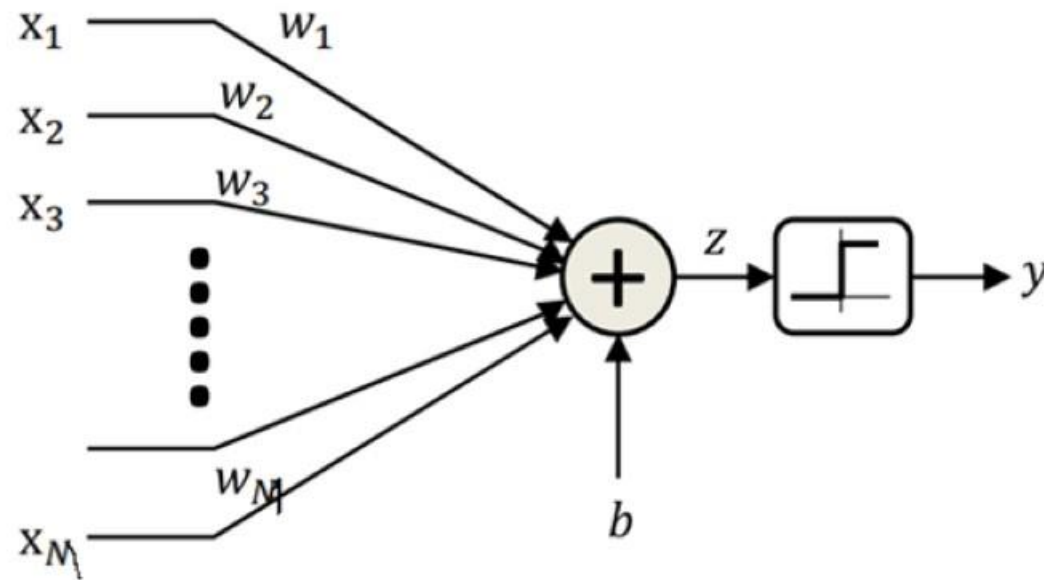


- **TPU (Tensor Processing Unit):**
 - A TPU is a specialized application-specific integrated circuit (ASIC) developed by Google for accelerating machine learning workloads.
 - It is specifically designed to perform tensor operations, which are fundamental to many machine learning algorithms.
 - TPUs are highly optimized for matrix multiplication and are capable of delivering significantly higher performance compared to CPUs and GPUs for deep learning tasks.
 - They are commonly used in data centers and cloud computing platforms to accelerate machine learning training and inference.



TPU^{v4}

Recap: the perceptron



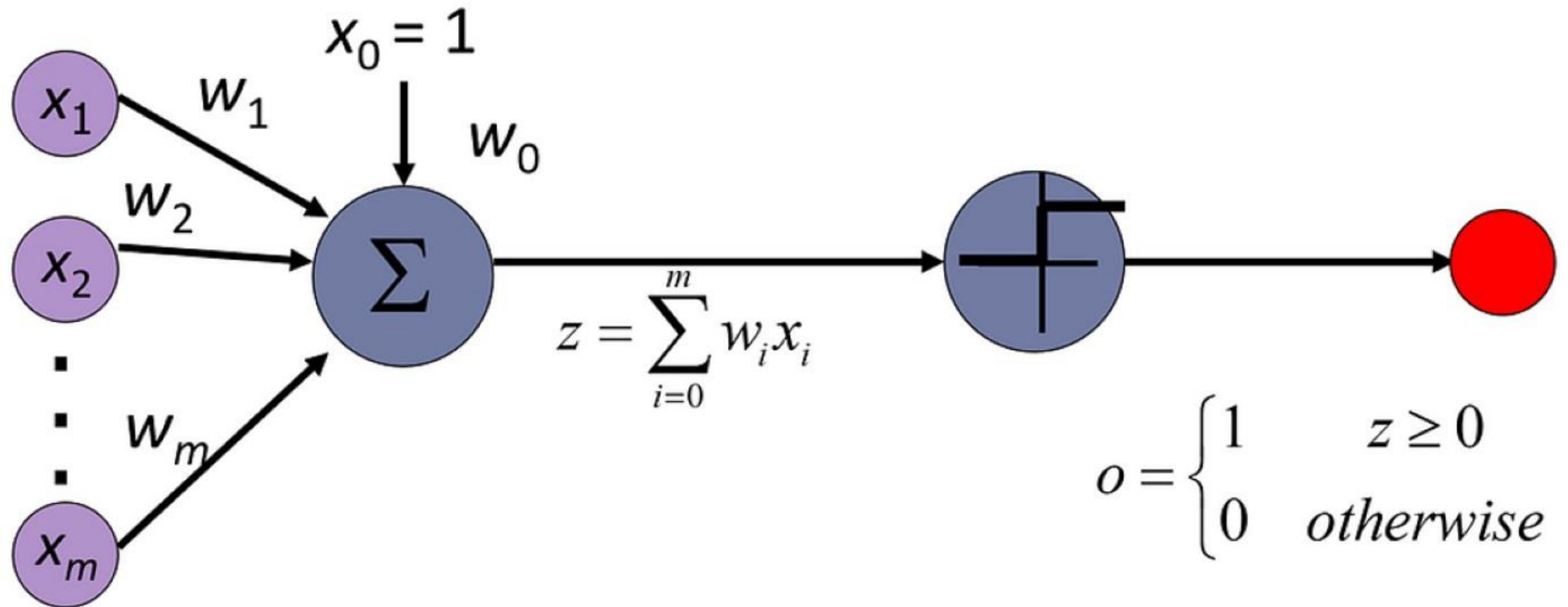
Linear vs Affine?

$$z = \sum_i w_i x_i + b$$

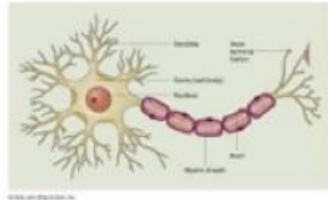
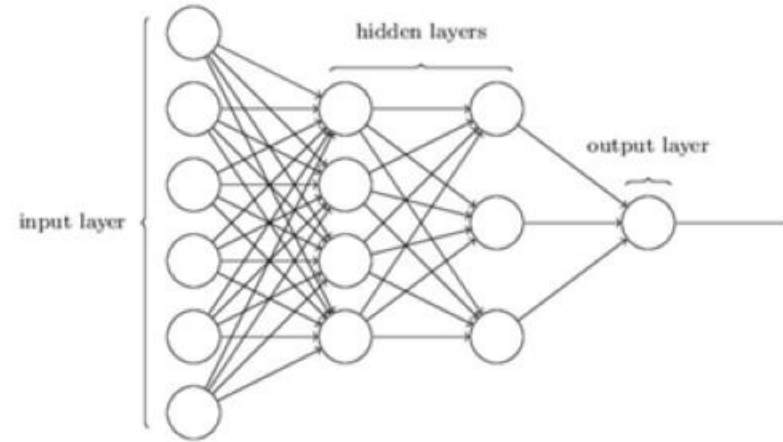
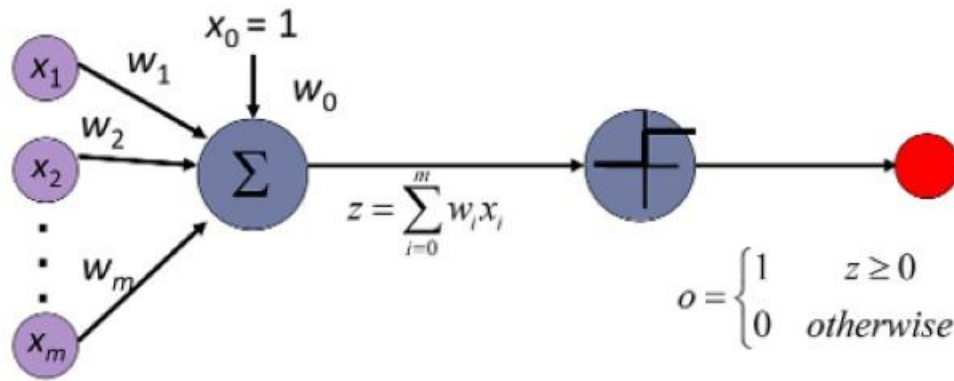
$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{else} \end{cases}$$

- “Fires” if the affine function of inputs is positive

Recap: the perceptron

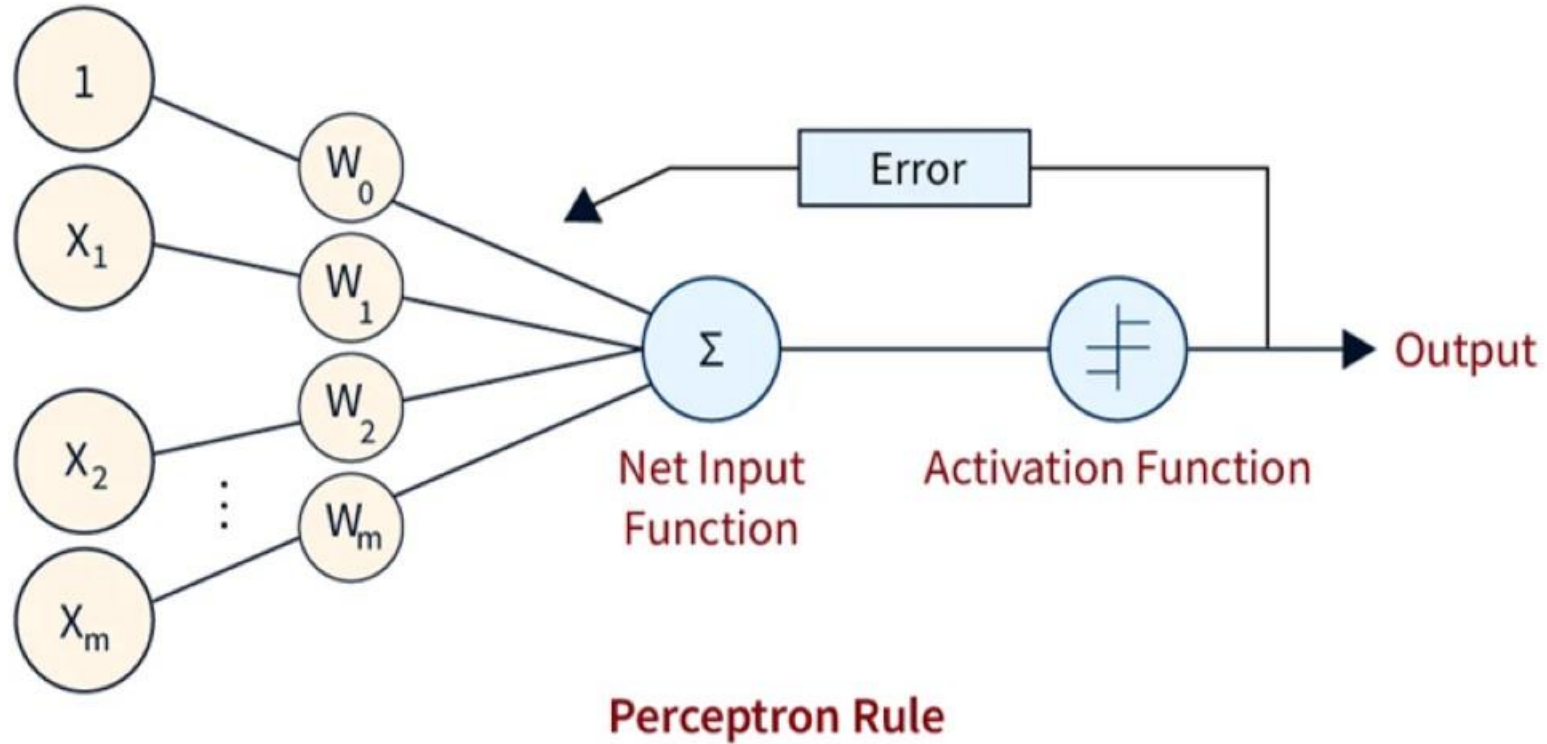


Recap : Nnets and the brain



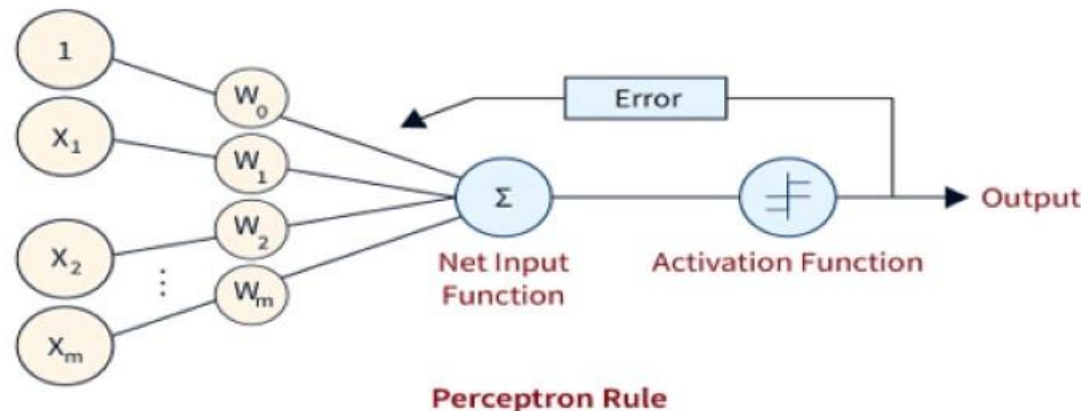
- Neural nets are composed of networks of computational models of neurons called perceptrons

Learning the Network



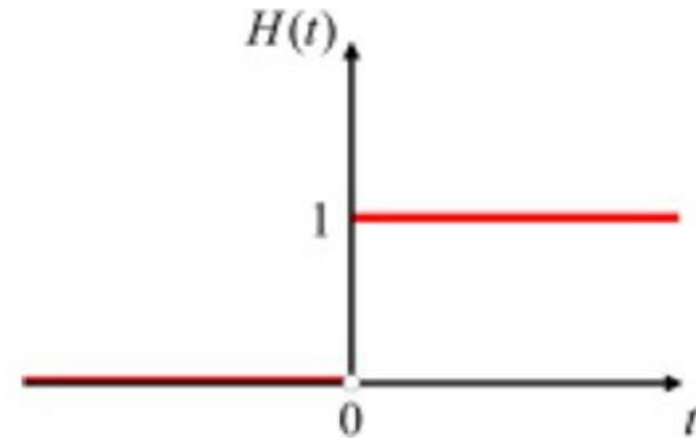
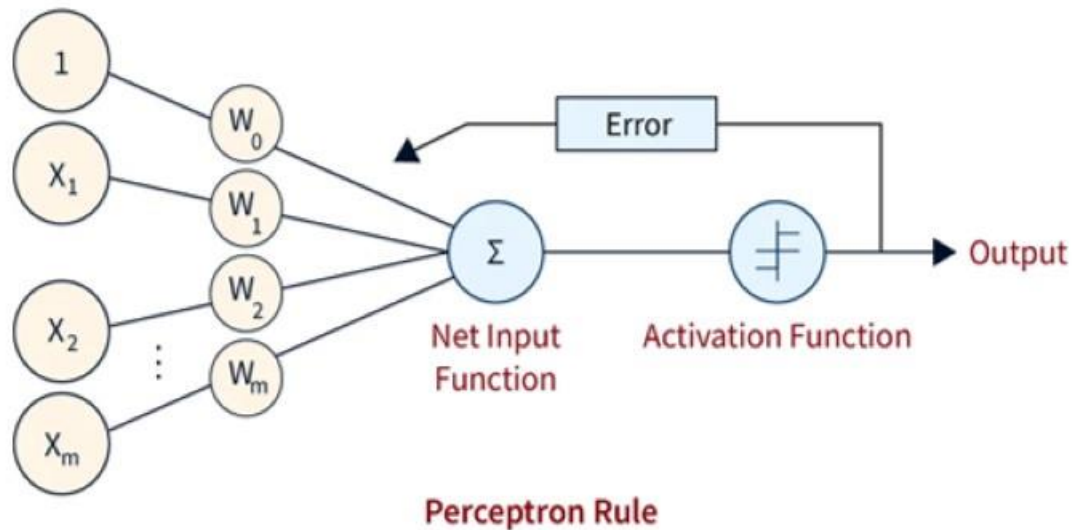
Calculate Weights

- (1) Calculate Weighted Sum: multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:
- $\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \dots + w_m * x_m$. Add another term called bias 'b' to the weighted sum to improve the model performance.



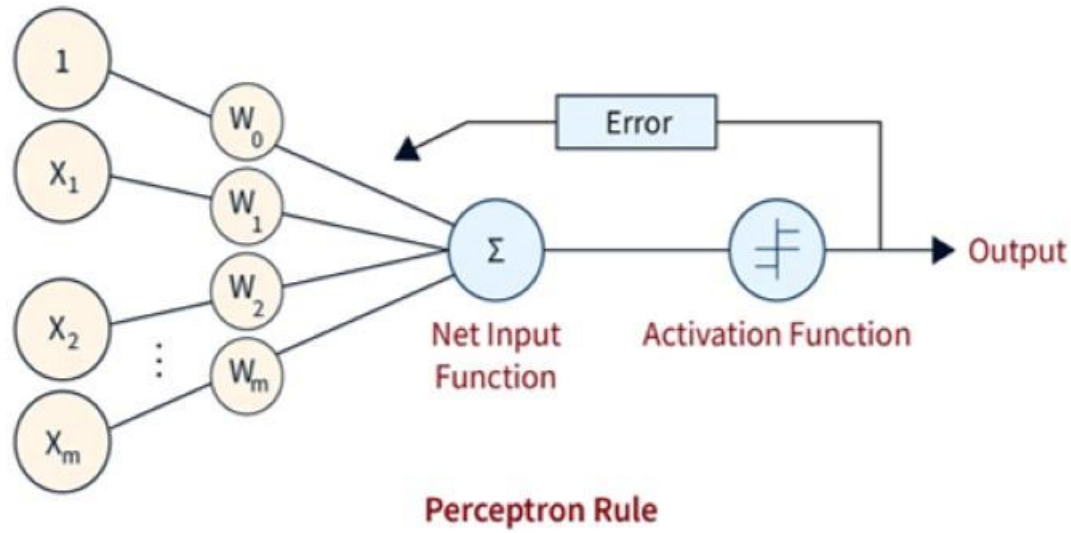
$$= \sum_{i=0}^m w_i x_i$$

- (2) an activation function is applied to this weighed sum, producing a binary or a continuous-valued output. $Y = f(\sum_{i=0}^m w_i x_i)$

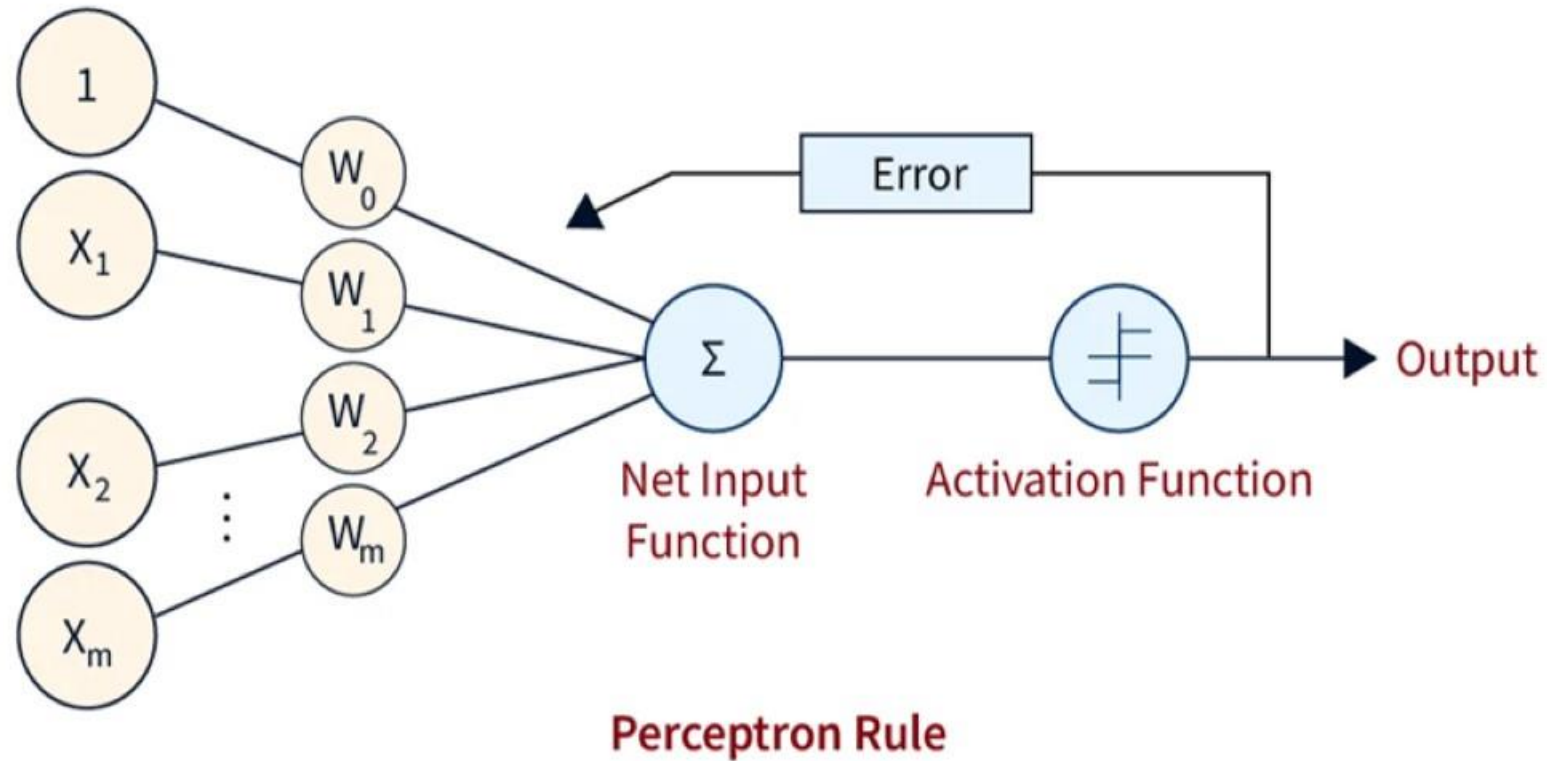


Perceptron learning algorithm

- (3) Evaluate
 - The difference between this output and the actual target value is computed to get the error term, E , generally in terms of mean squared error. The steps up to this form the forward propagation part of the algorithm $(\text{Predicted} - \text{Actual})^2$.



Perceptron learning algorithm



$w_1, w_2, w_3, \dots \dots \dots w_m$

Samples

Variables

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6		Variable m
Sample 1	29	31	34	36	39	48		48
Sample 2	27	29	31	34	36	45		45
Sample 3	25	27	29	31	34	42		42
Sample 4	23	25	27	29	31	39		39
...	...	...	...	...	...	...		...
Sample n	18	20	21	23	24	26		20

$$w_1, w_2, w_3, \dots \dots \dots w_m$$

Samples

Variables

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6		Variable m
Sample 1	29	31	34	36	39	48		48
Sample 2	27	29	31	34	36	45		45
Sample 3	25	27	29	31	34	42		42
Sample 4	23	25	27	29	31	39		39
...	...	...	...	...	...	...		...
Sample n	18	20	21	23	24	26		20

- The **dimensions** of a matrix give the number of rows and columns of the matrix *in that order*.

3 columns

↓ ↓ ↓

$$A = \begin{bmatrix} -2 & 5 & 6 \\ 5 & 2 & 7 \end{bmatrix}$$

← ← 2 rows

2*3
Matrix

Matrix Operation

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 2 & 5 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 3 & 2 & 2 \\ 2 & 4 & 5 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 2 & 5 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 3 & 2 & 2 \\ 2 & 4 & 5 & 1 \end{bmatrix}$$

2 row entries
= 2 columns



$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 2 & 5 \end{bmatrix}$$

2 column entries
= 2 rows



$$B = \begin{bmatrix} 1 & 3 & 2 & 2 \\ 2 & 4 & 5 & 1 \end{bmatrix}$$

$$(m \times n) \cdot (n \times k)$$



product is defined

Is CD defined?

$$C = \begin{bmatrix} 5 & 3 & 6 & 1 \\ 6 & 8 & 5 & 3 \end{bmatrix} \text{ and } D = \begin{bmatrix} 2 & 1 & 8 \\ 7 & 5 & 5 \end{bmatrix}.$$

CD Undefined

A is a 4×2 matrix and B is a 2×3 matrix.

- [illegible]

Matrix Operations

$$w_1, w_2, w_3, \dots, w_m$$

$$1 \times m$$

Samples

Variables

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6		Variable m
Sample 1	29	31	34	36	39	48		48
Sample 2	27	29	31	34	36	45		45
Sample 3	25	27	29	31	34	42		42
Sample 4	23	25	27	29	31	39		39
...	...	...	...	...	...	...		...
Sample n	18	20	21	23	24	26		20

$$n \times m$$

Matrix Operations

$w_1, w_2, w_3, \dots \dots \dots w_m$

w_1
 w_2
 w_3

 w_m

$1 \times m$

$m \times 1$

Samples

Variables

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6		Variable m
Sample 1	29	31	34	36	39	48		48
Sample 2	27	29	31	34	36	45		45
Sample 3	25	27	29	31	34	42		42
Sample 4	23	25	27	29	31	39		39
...	...	...	...	...	...	...		...
Sample n	18	20	21	23	24	26		20

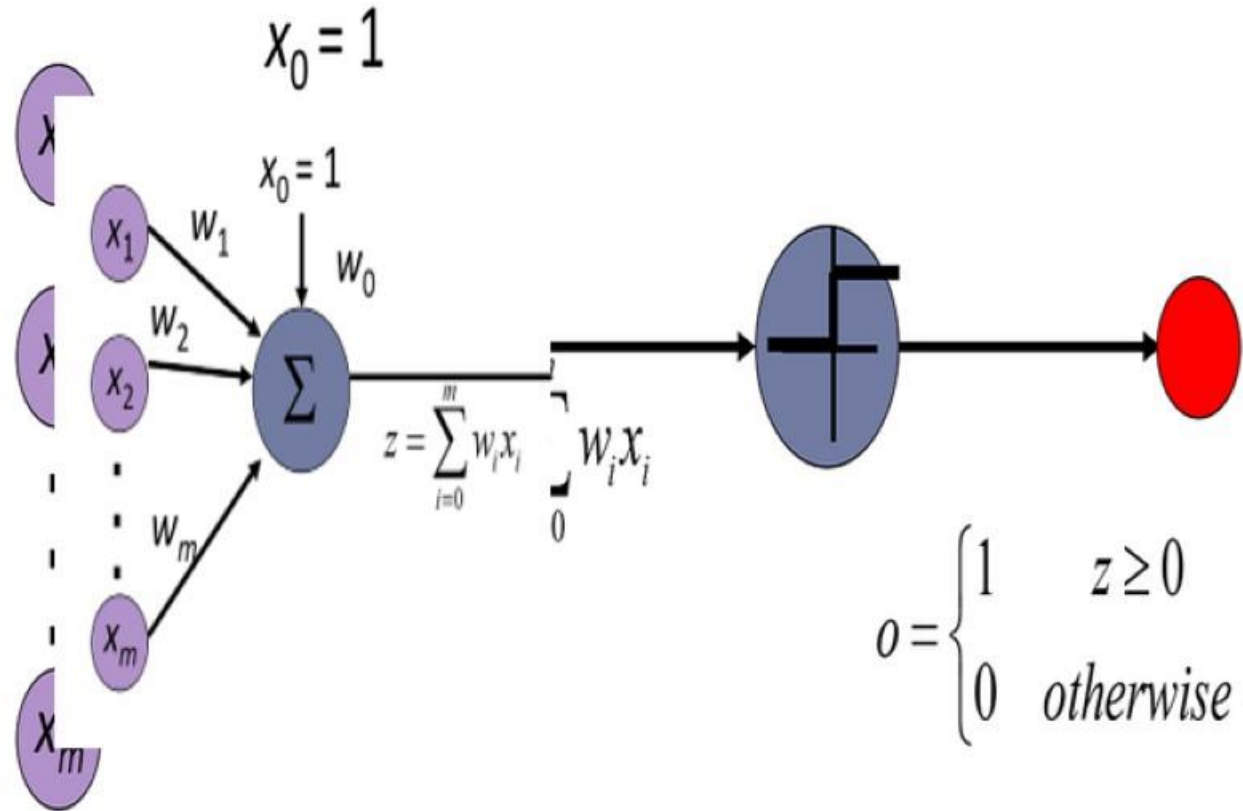
$n \times m$

- Weighted Sum

- $\sum_{i=1}^m w_i x_i + b$

- $\sum_{i=0}^m w_i x_i$

- $\mathbf{w}^T \mathbf{x}$

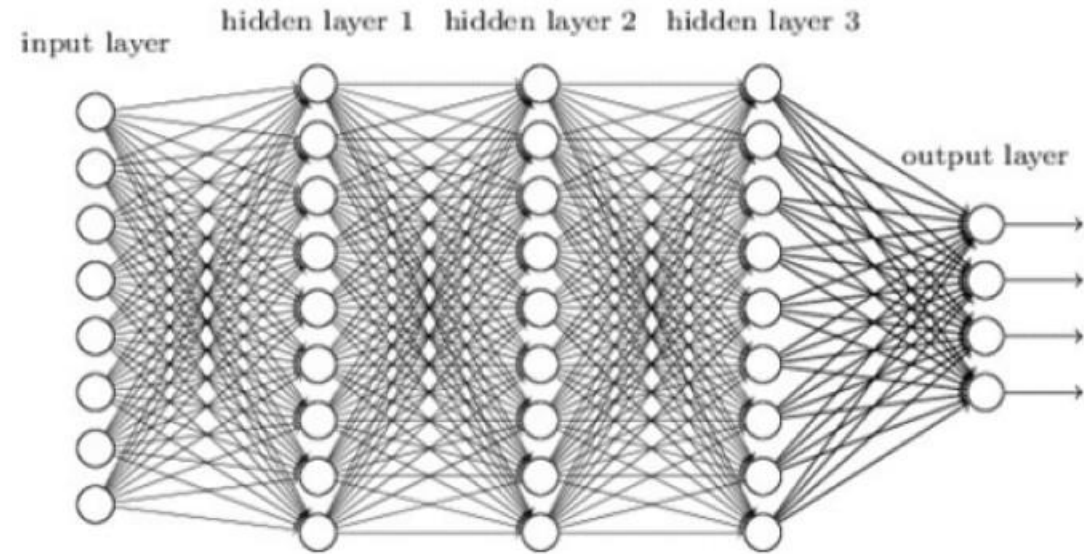


Learning the Network

Dr. Shahira Ahmed

- A perceptron model's output can only be a binary number i.e. "0" or "1" because of activation function.
- The perceptron model can only be used to categorize the input vectors' linearly separable sets. If input vectors are non-linear, they can't be properly classified.

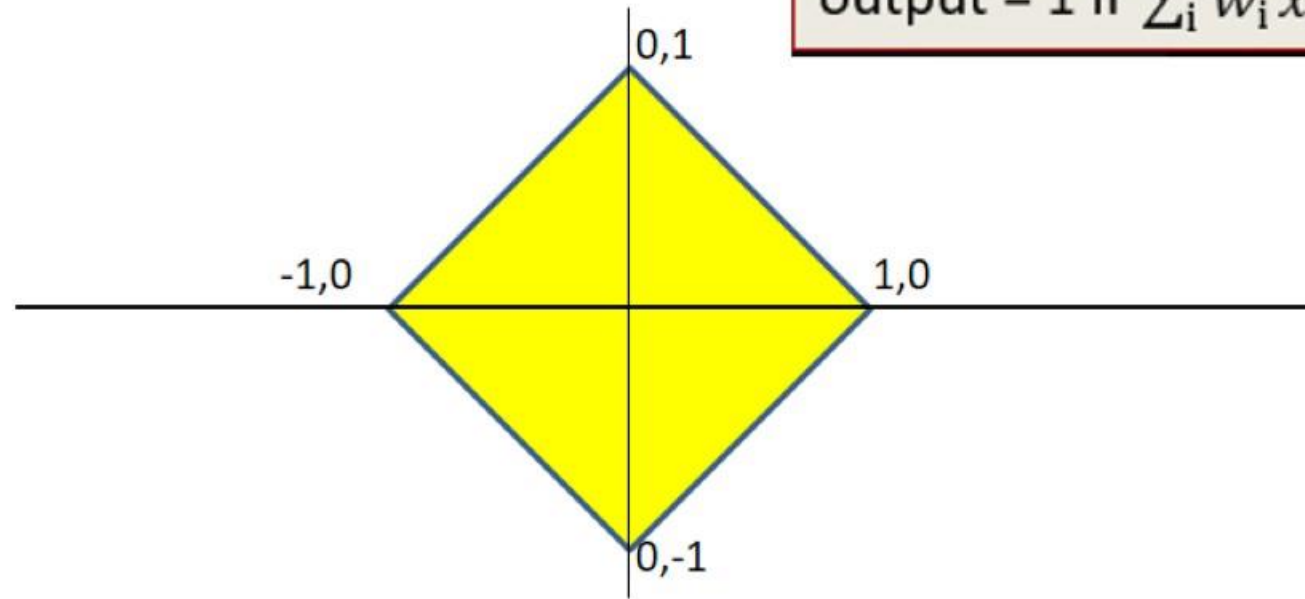
The multi-layer perceptron



- A network of perceptrons
 - Perceptrons “feed” other perceptrons.

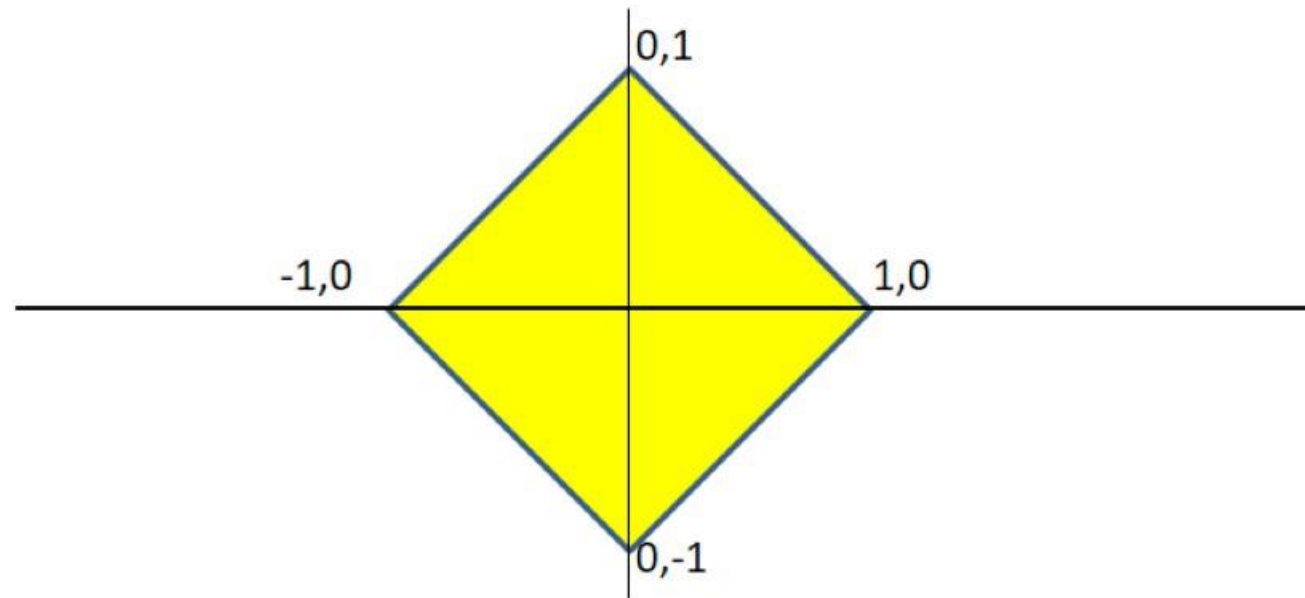
Option 1: Construct by hand

Assuming simple perceptrons:
output = 1 if $\sum_i w_i x_i + b_i \geq 0$, else 0



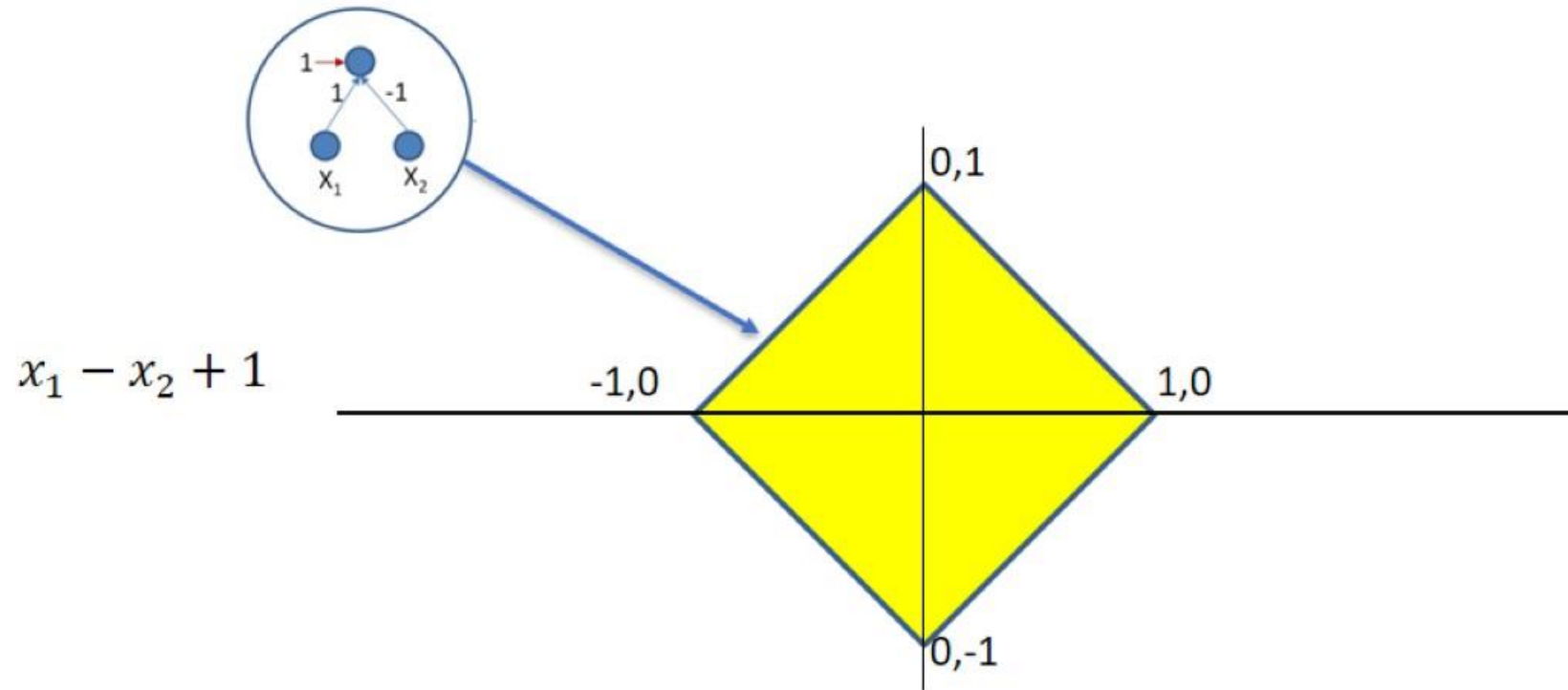
- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

Option 1: Construct by hand



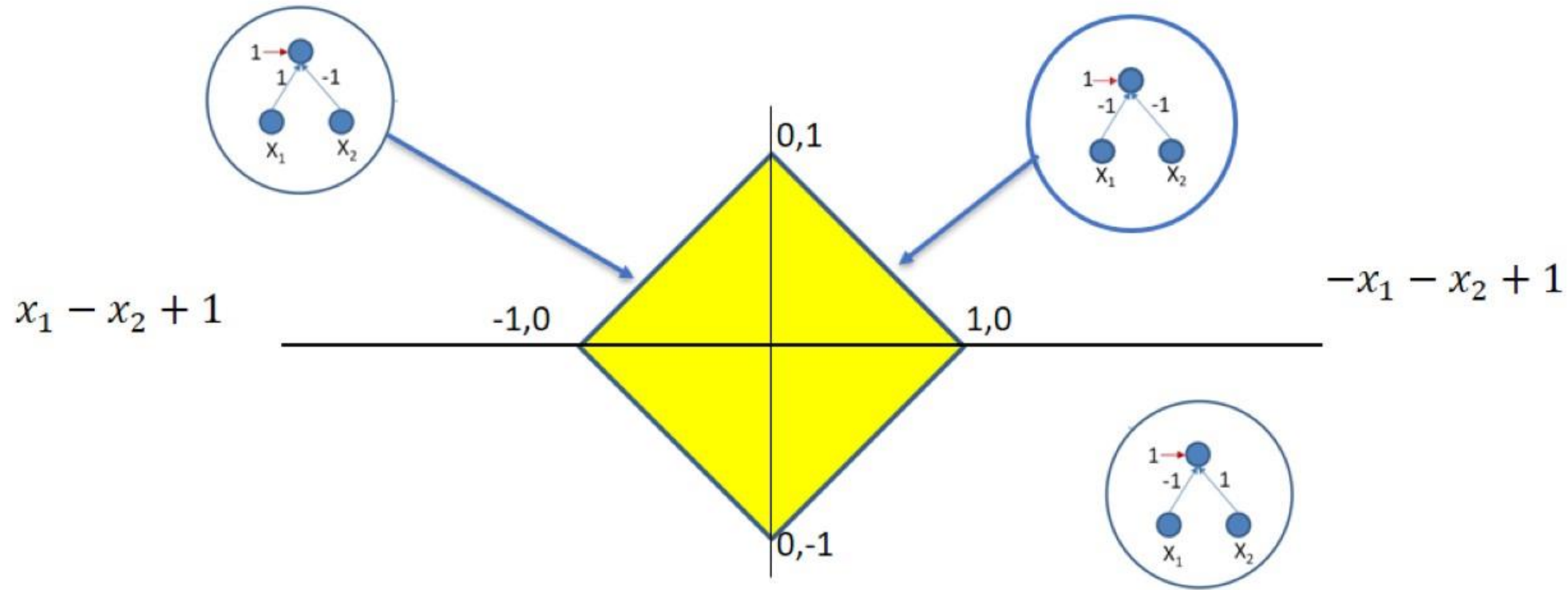
- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

Option 1: Construct by hand



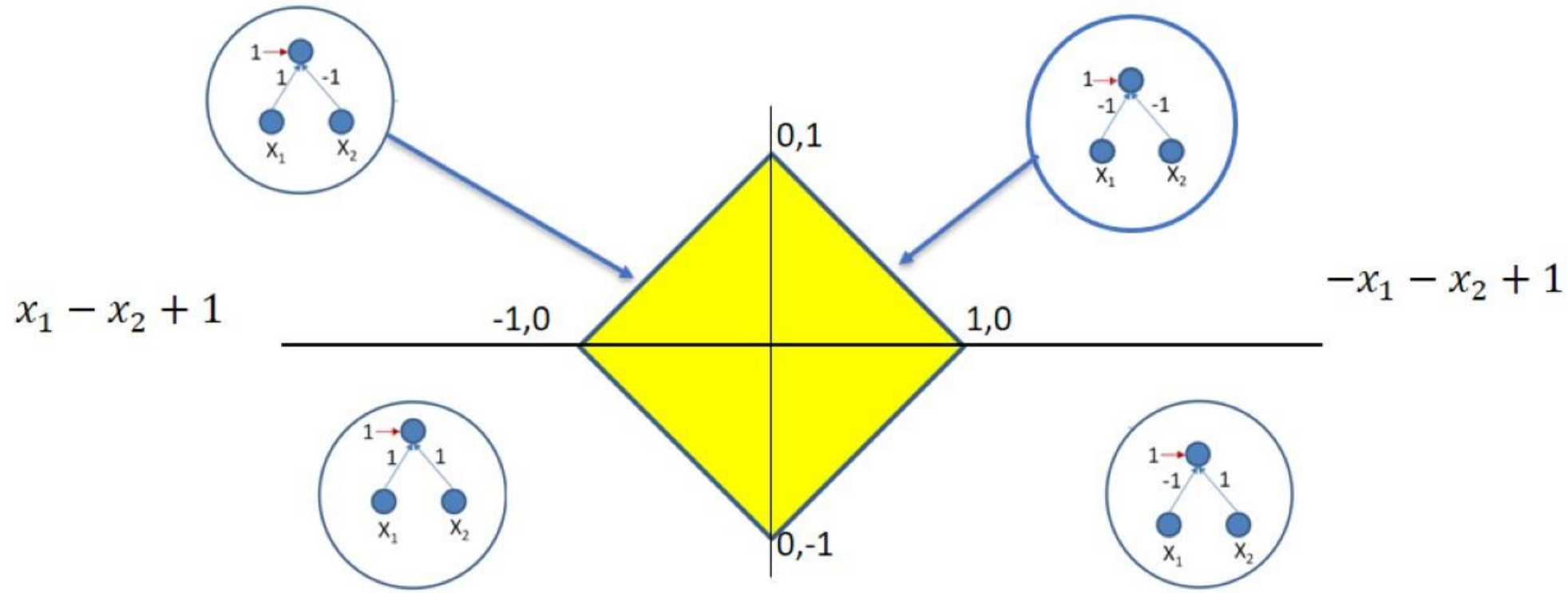
- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

Option 1: Construct by hand



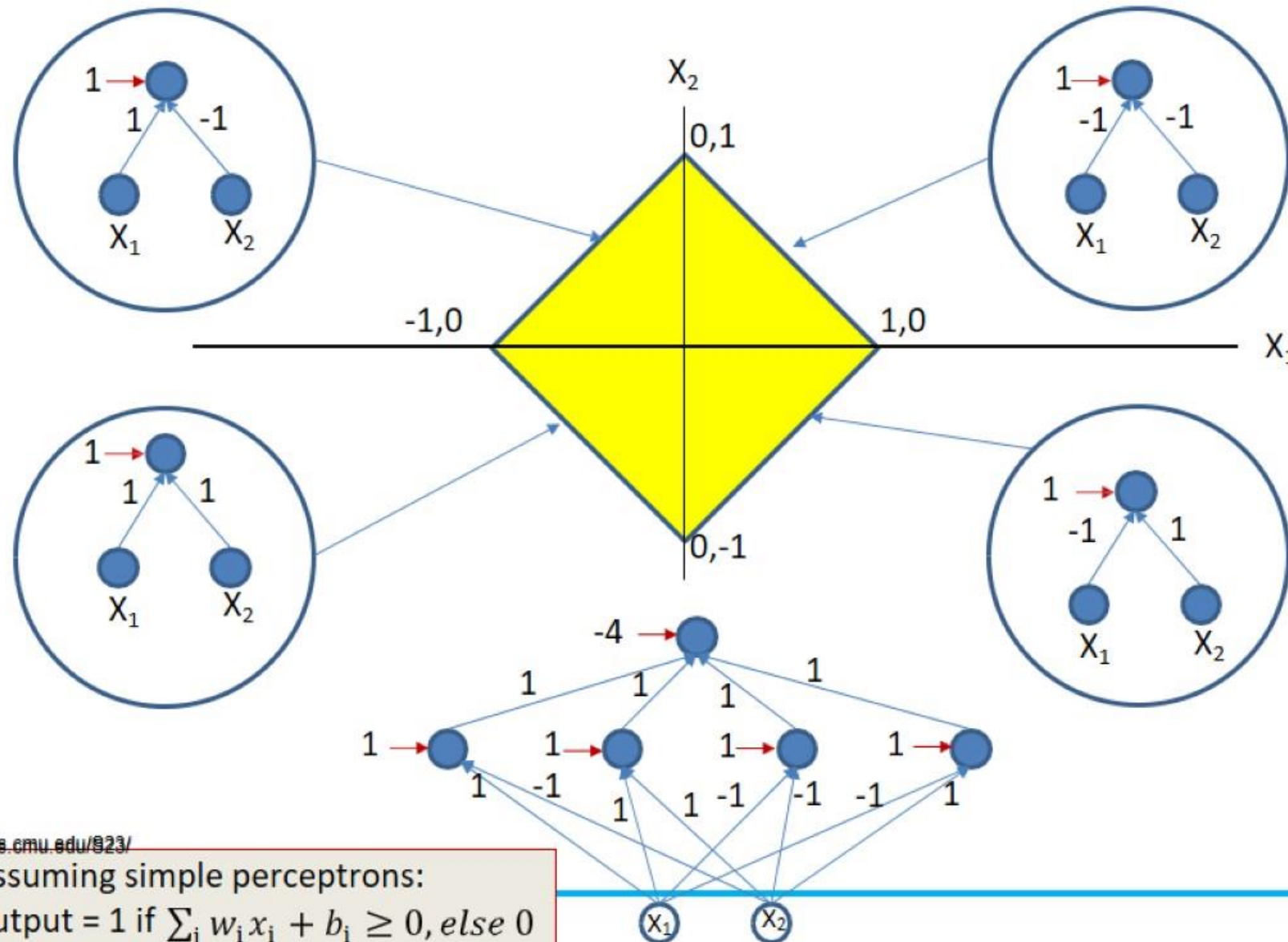
- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

Option 1: Construct by hand



- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary

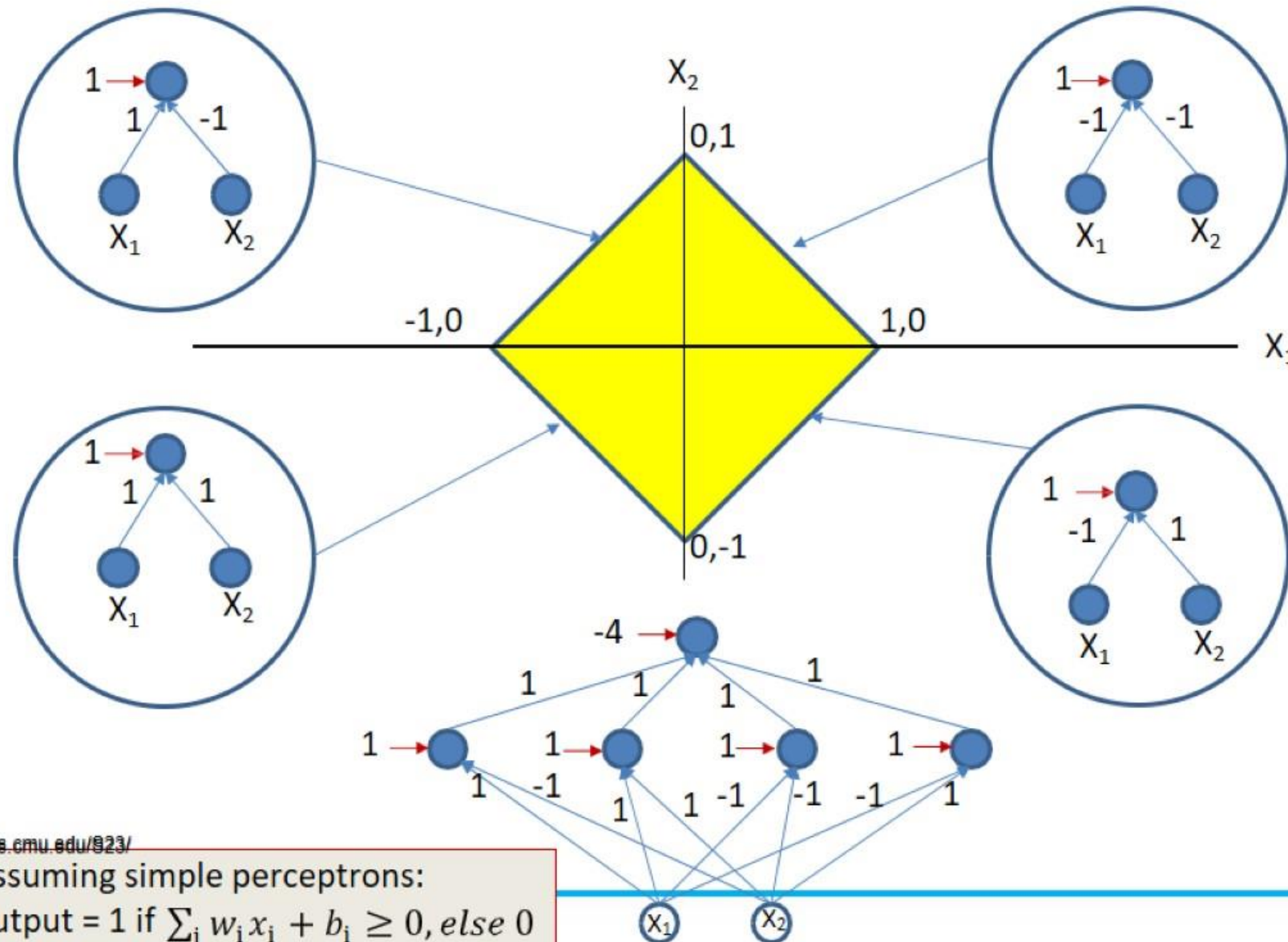
Option 1: Construct by hand



<https://deeplearning.cs.cmu.edu/923/>

Assuming simple perceptrons:
output = 1 if $\sum_i w_i x_i + b_i \geq 0$, else 0

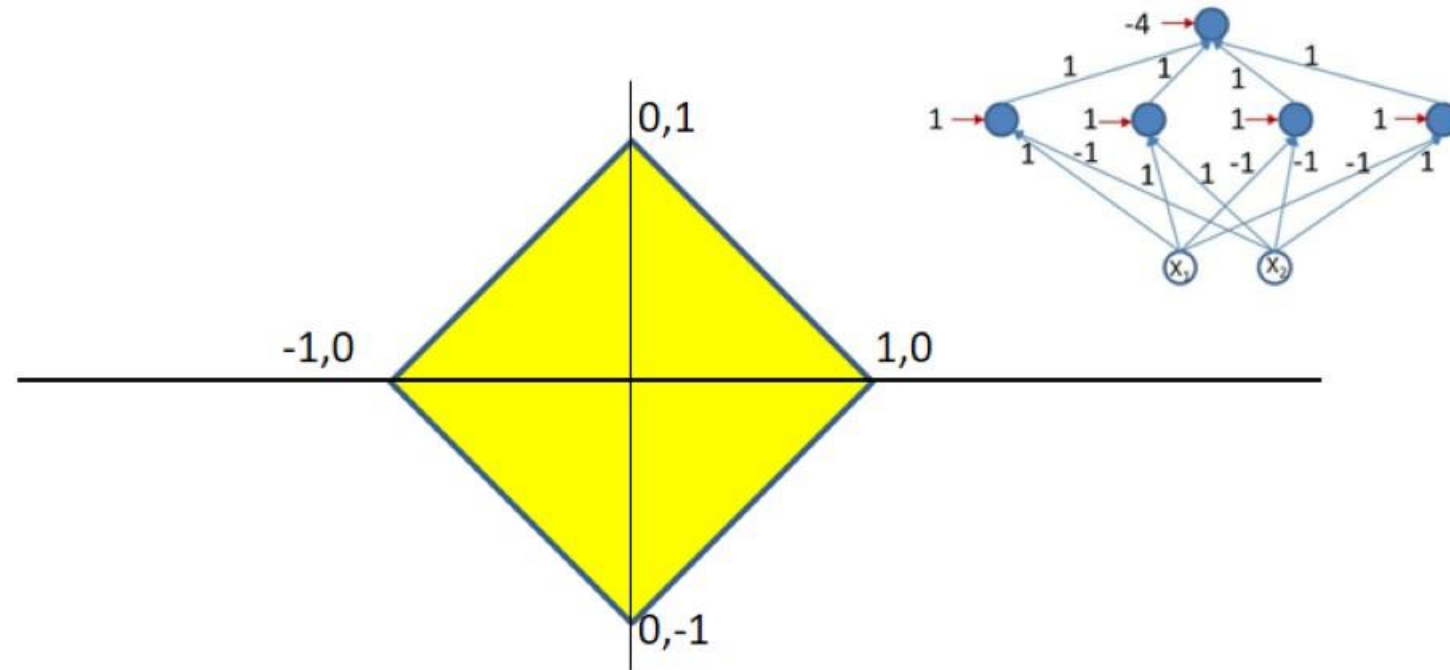
Option 1: Construct by hand



<https://deeplearning.cs.cmu.edu/923/>

Assuming simple perceptrons:
output = 1 if $\sum_i w_i x_i + b_i \geq 0$, else 0

Option 1: Construct by hand



- Given a function, *handcraft* a network to satisfy it
- E.g.: Build an MLP to classify this decision boundary
- **Not possible for all but the simplest problems..**

- Vectorization
- Advantages of Vectorized Implementation
- Single vs Multilayer perceptron
- Training Network by hand



Thank You