Review

PERCEPTRON

The Perceptron

- A perceptron separates the input space into two halves, positive and negative.
- All the inputs that produce ***true*** lie on one side (positive half) and all the inputs that produce ***false*** lie on the other side (negative half space)

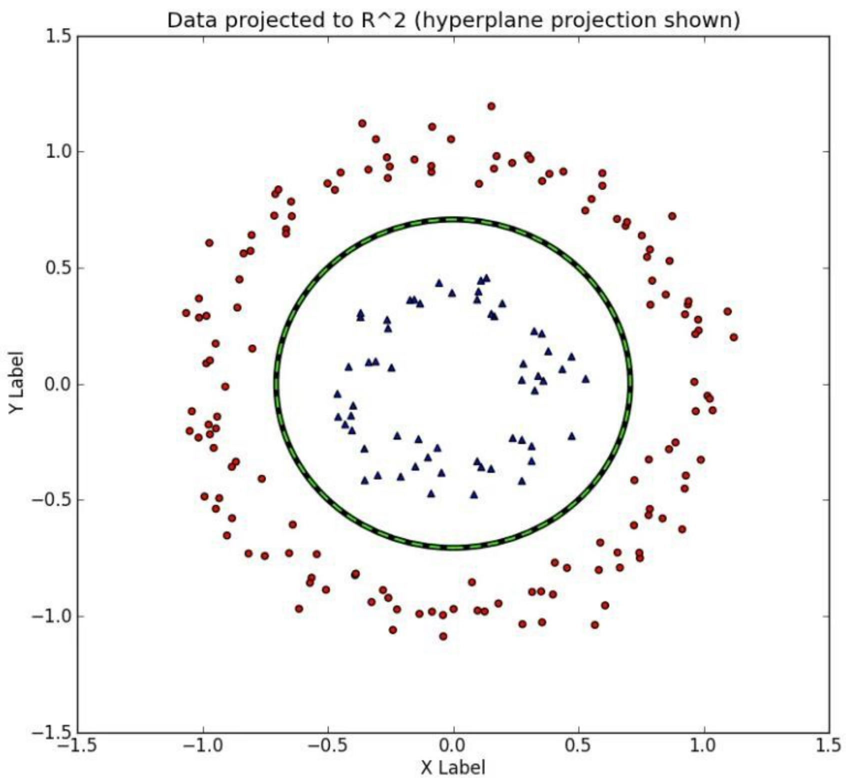
- **A single Perceptron can only be used to implement linearly separable functions**
  - Just like M-P Neuron

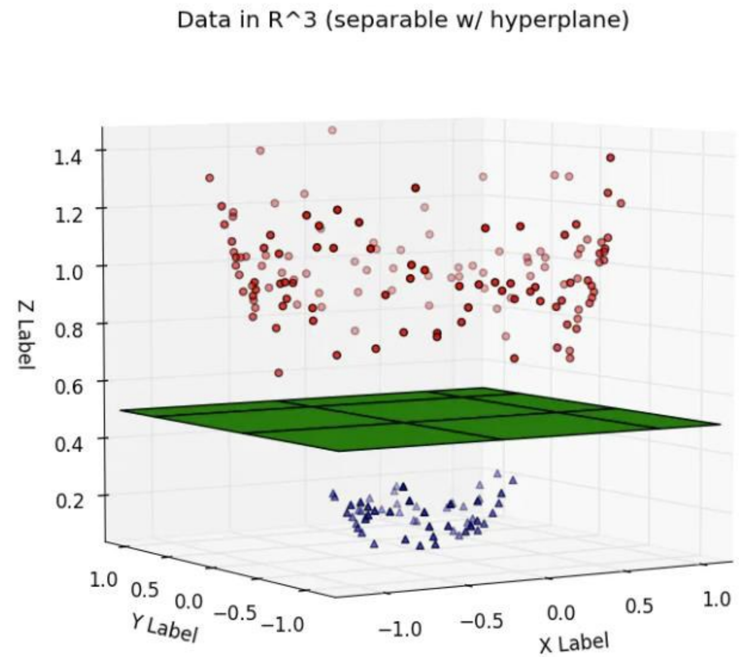
- **How Perceptron is different than M-P Neuron?**
  - The inputs can be assigned different importance
  - The weights and the thresholds can be learned.
  - The inputs can be real values

How to make linearly separable decision boundary? What should be changed in Perceptron?

Credit: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

One way: Adding Dimensions to Achieve Linear Separability





Second Way: Use Hidden Layers

Credit: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f)

Training a Perceptron

- **Before moving to hidden layers, lets understand why these are useful!**
- The new weights are changed via the equation:

  - Where:

$$\boxed{w_i = w_i + \Delta w_i}$$

$$\boxed{\Delta w_i = \eta(y - \hat{y})x_i}$$

This is known as **Perceptron Training Rule**

This is learning rate which dictates how much the weights should change

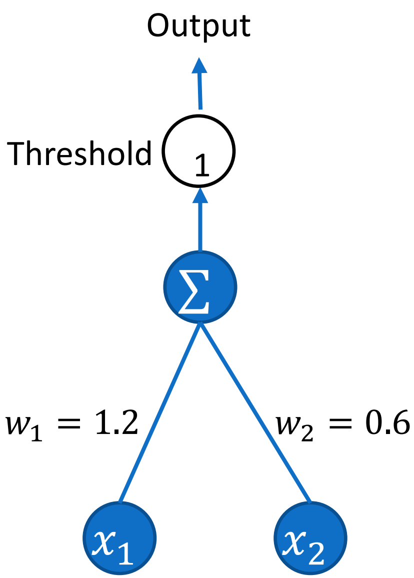
Actual label (Target)

Predicted label (Output of Network)

Combined form of equation $$\boxed{w_i = w_i + \eta(y - \hat{y})x_i}$$

A Simple Example: AND Function

$w_1 = 1.2$ $w_2 = 0.6$ $\eta = 0.5$ $threshold = 1$



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Note: We are not using Bias (x_0) for this example.

A Simple Example: AND Function

$w_1 = 1.2$ $\quad w_2 = 0.6$ $\quad \eta = 0.5$ $\quad threshold = 1$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

For record 1:

$$\sum w_i \times x_i = 0 \times 1.2 + 0 \times 0.6 = 0$$

Sum is not greater than threshold, so output is 0

For record 2:

$$\sum w_i \times x_i = 0 \times 1.2 + 1 \times 0.6 = 0.6$$

Sum is not greater than threshold, so output is 0

For record 3:

$$\sum w_i \times x_i = 1 \times 1.2 + 0 \times 0.6 = 1.2$$

Sum is greater than threshold, so output is 1

The target is 0 and the output is 1, update the weights!

$$\boldsymbol{w_i = w_i + \eta(y - \hat{y})x_i}$$

A Simple Example: AND Function

$w_1 = 1.2$ $\quad w_2 = 0.6$ $\quad \eta = 0.5$ $\quad threshold = 1$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

For record 3:

$$\sum w_i \times x_i = 1 \times 1.2 + 0 \times 0.6 = 1.2$$

Sum is greater than threshold, so output is 1

The target is 0 and the output is 1, update the weights!

$$\boldsymbol{w_i = w_i + \eta(y - \hat{y})x_i}$$

$$\boldsymbol{w_1 = 1.2 + 0.5(0 - 1)1 = 0.7}$$

$$\boldsymbol{w_2 = 0.6 + 0.5(0 - 1)0 = 0.6}$$

A Simple Example: AND Function

$w_1 = 0.7$ $w_2 = 0.6$ $\eta = 0.5$ $threshold = 1$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

For record 1:

$$\sum w_i \times x_i = 0 \times 0.7 + 0 \times 0.6 = 0$$

Sum is not greater than threshold, so output is 0

For record 2:

$$\sum w_i \times x_i = 0 \times 0.7 + 1 \times 0.6 = 0.6$$

Sum is not greater than threshold, so output is 0

For record 3:

$$\sum w_i \times x_i = 1 \times 0.7 + 0 \times 0.6 = 0.7$$

Sum is not greater than threshold, so output is 0

For record 4:

$$\sum w_i \times x_i = 1 \times 0.7 + 1 \times 0.6 = 1.3$$

Sum is greater than threshold, so output is 1

A Simple Example: AND Function

$w_1 = 0.7$ $\quad w_2 = 0.6$ $\quad \eta = 0.5$ $\quad threshold = 1$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Take Aways:

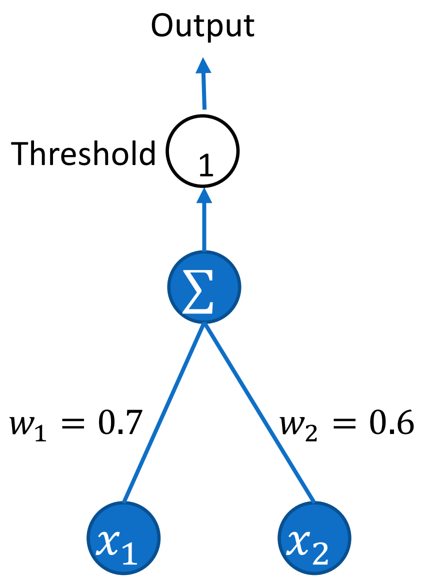
Only update weights when the predicted output is not equal to the actual output.

If all records/training examples are classified correctly with certain set of weights, stop the training.

Save the final weights for future deployments for unseen data.

A Simple Example: AND Function

$w_1 = 0.7$ $w_2 = 0.6$ $\eta = 0.5$ $threshold = 1$



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Trained Perceptron

A Simple Example: OR Function

$w_1 = 0.6$ $w_2 = 0.6$ $\eta = 0.5$ $threshold = 1$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

For record 1:

$$\sum w_i \times x_i = 0 \times 0.6 + 0 \times 0.6 = 0$$

Sum is not greater than threshold, so output is 0

For record 2:

$$\sum w_i \times x_i = 0 \times 0.6 + 1 \times 0.6 = 0.6$$

Sum is not greater than threshold, so output is 0

The target is 1 and the output is 0, update the weights!

$$\boldsymbol{w_i = w_i + \eta(y - \hat{y})x_i}$$

$$\boldsymbol{w_1 = 0.6 + 0.5(1 - 0)0 = 0.6}$$

$$\boldsymbol{w_2 = 0.6 + 0.5(1 - 0)1 = 1.1}$$

A Simple Example: OR Function

$w_1 = 0.6$ $w_2 = 1.1$ $\eta = 0.5$ $threshold = 1$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

For record 1:

$$\sum w_i \times x_i = 0 \times 0.6 + 0 \times 1.1 = 0$$

Sum is not greater than threshold, so output is 0

For record 2:

$$\sum w_i \times x_i = 0 \times 0.6 + 1 \times 1.1 = 1.1$$

Sum is greater than threshold, so output is 1

For record 3:

$$\sum w_i \times x_i = 1 \times 0.6 + 0 \times 1.1 = 0.6$$

Sum is not greater than threshold, so output is 0

The target is 1 and the output is 0, update the weights!

$$\boldsymbol{w_i = w_i + \eta(y - \hat{y})x_i}$$

$$\boldsymbol{w_1 = 0.6 + 0.5(1 - 0)1 = 1.1}$$

$$\boldsymbol{w_2 = 0.6 + 0.5(1 - 0)0 = 1.1}$$

A Simple Example: OR Function

$w_1 = 1.1$ $\quad w_2 = 1.1$ $\quad \eta = 0.5$ $\quad threshold = 1$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

For record 1:

$$\sum w_i \times x_i = 0 \times 1.1 + 0 \times 1.1 = 0$$

Sum is not greater than threshold, so output is 0

For record 2:

$$\sum w_i \times x_i = 0 \times 1.1 + 1 \times 1.1 = 1.1$$

Sum is greater than threshold, so output is 1

For record 3:

$$\sum w_i \times x_i = 1 \times 1.1 + 0 \times 1.1 = 1.1$$

Sum is not greater than threshold, so output is 1

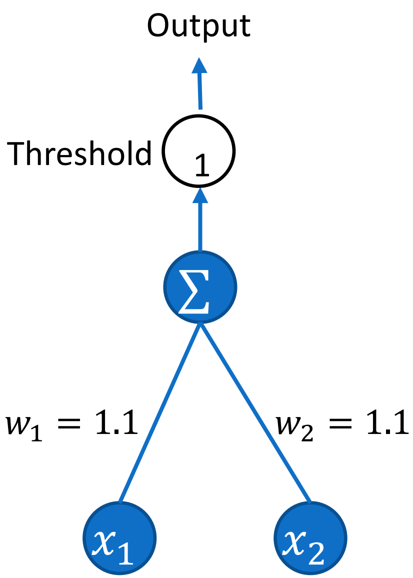
For record 4:

$$\sum w_i \times x_i = 1 \times 1.1 + 1 \times 1.1 = 2.2$$

Sum is not greater than threshold, so output is 1

A Simple Example: OR Function

$w_1 = 1.1$ $w_2 = 1.1$ $\eta = 0.5$ $threshold = 1$



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Trained Perceptron

These were few of infinite possible solutions…

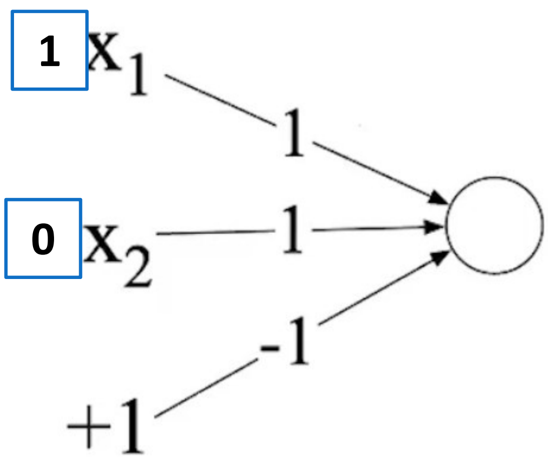
- Why infinite solutions?

- The solution is nothing but a set of “weights and biases” which gives us the right output.
  - As weights can be any continuous value, depending on the initial random initialization, we have infinite possibilities.

Other Solutions to OR and AND Problems

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



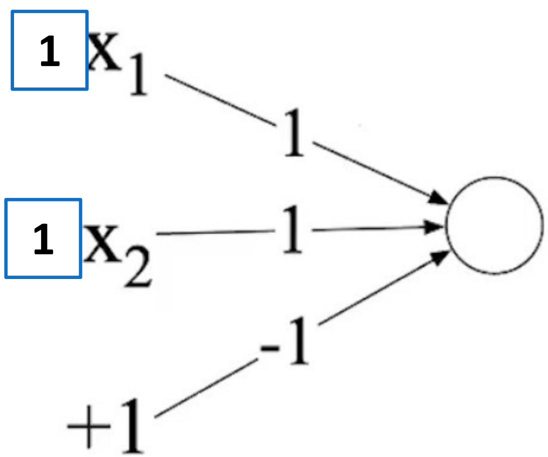
$$\boxed{\mathbf{1 \times 1 + 0 \times 1 + (-1) = 0}}$$

Solution for AND

Note: We are using Bias for this solution.

Other Solutions to OR and AND Problems

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$



$$\mathbf{1 \times 1 + 1 \times 1 + (-1) = 1}$$

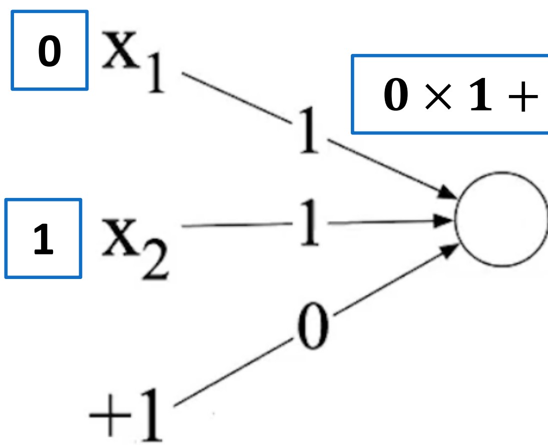
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Solution for AND

Note: No threshold required.

Other Solutions to OR and AND Problems

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

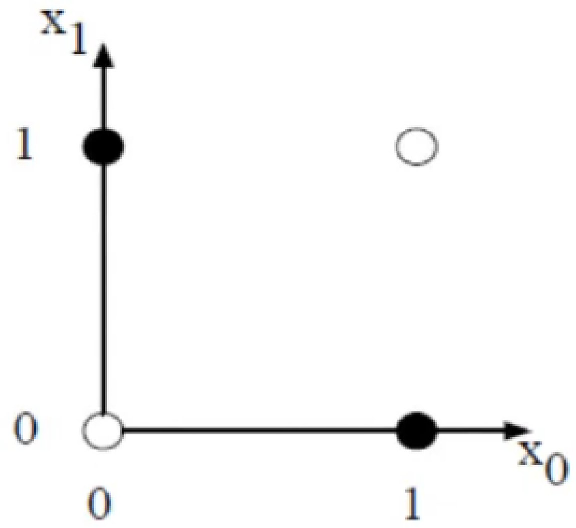


$$\mathbf{0 \times 1 + 1 \times 1 + 1 \times 0 = 1}$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Solution for OR

Not So Simple Example: XOR Function

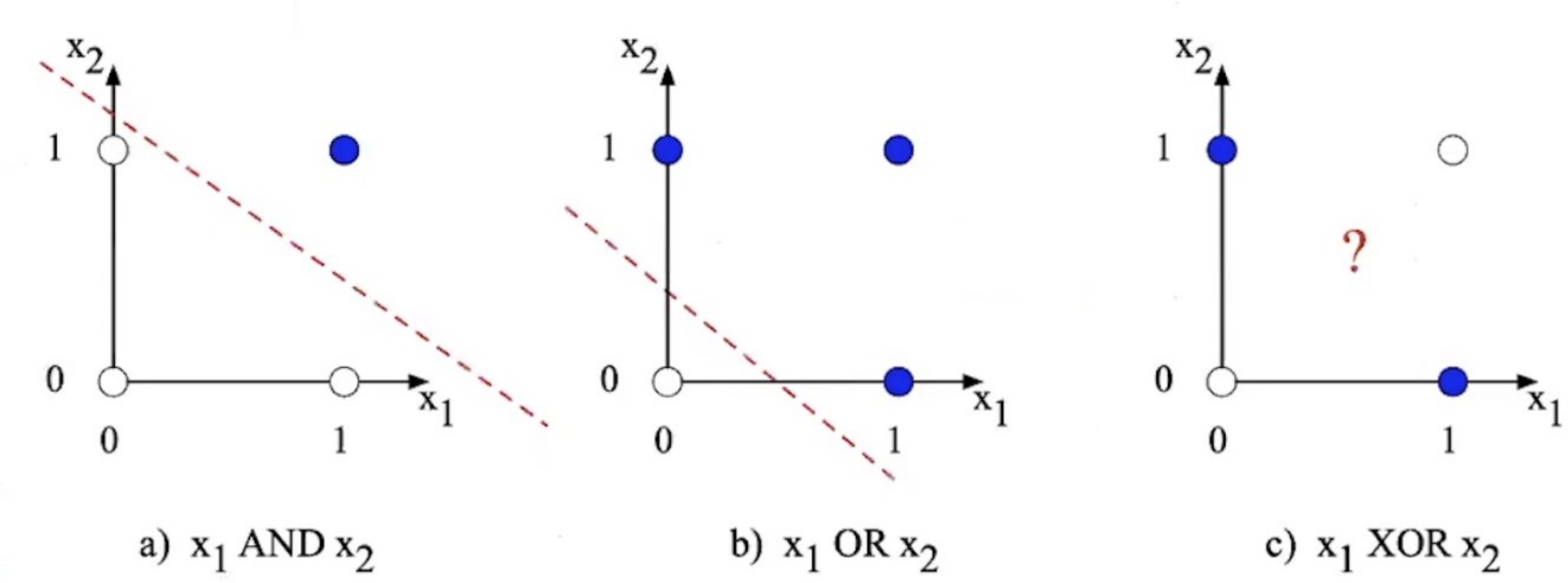


x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

No weights would satisfy all the target labels.

Perceptrons are linear classifiers….

Decision Boundaries

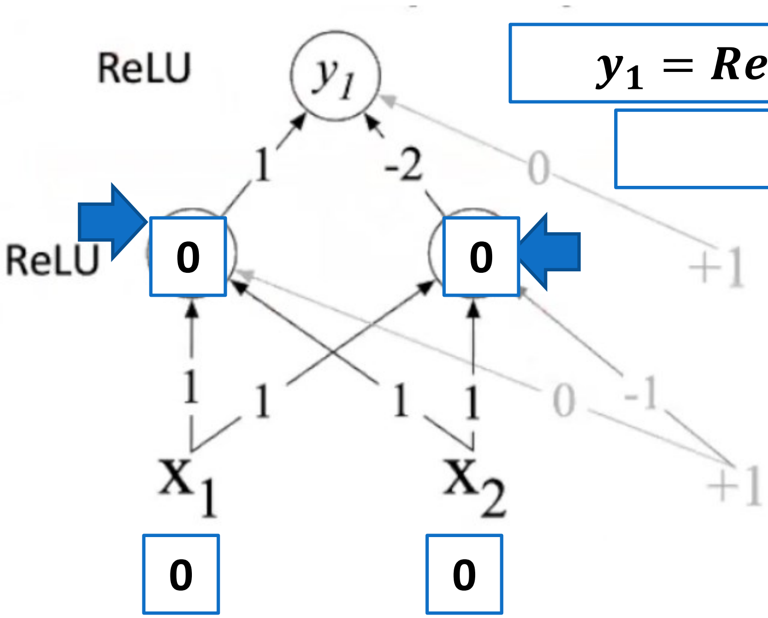


How to learn such complex decision boundaries?

XOR Solution: Multilayer Perceptron

- XOR **can't** be calculated by a single perceptron
- XOR **can** be calculated by a layered network of units

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$y_1 = ReLU(0 \times 1 + 0 \times -2 + 1 \times 0)$$

$$y_1 = ReLU(0) = 0$$

$$h_1 = ReLU(0 \times 1 + 0 \times 1 + 1 \times 0)$$

$$h_1 = ReLU(0) = 0$$

$$h_2 = ReLU(0 \times 1 + 0 \times 1 + 1 \times (-1))$$

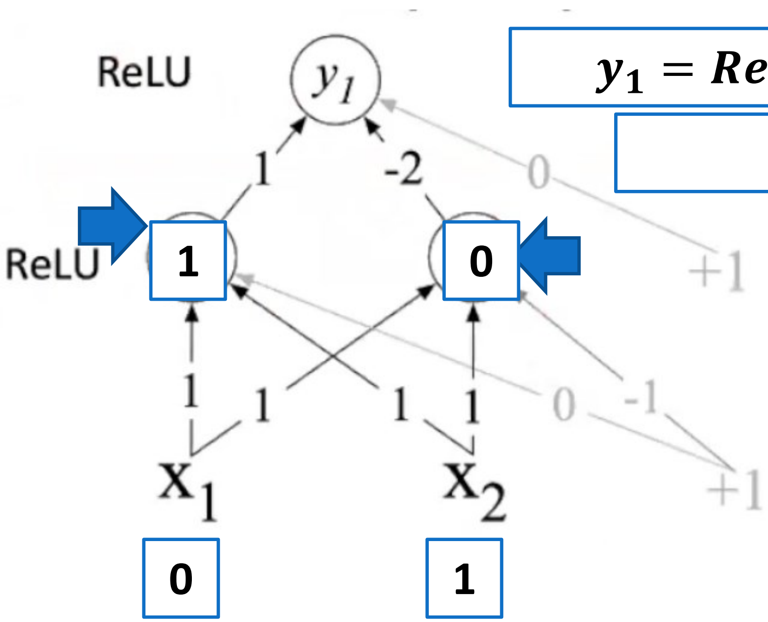
$$h_2 = ReLU(-1) = 0$$

Solution Credit: Goodfellow et al. (2016)

XOR Solution: Multilayer Perceptron

- ❑XOR **can't** be calculated by a single perceptron
- ❑XOR **can** be calculated by a layered network of units

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$\boldsymbol{y_1 = ReLU(1 \times 1 + 0 \times -2 + 1 \times 0)}$$

$$\boldsymbol{y_1 = ReLU(1) = 1}$$

$$\boldsymbol{h_1 = ReLU(0 \times 1 + 1 \times 1 + 1 \times 0)}$$

$$\boldsymbol{h_1 = ReLU(1) = 1}$$

$$\boldsymbol{h_2 = ReLU(0 \times 1 + 1 \times 1 + 1 \times (-1))}$$

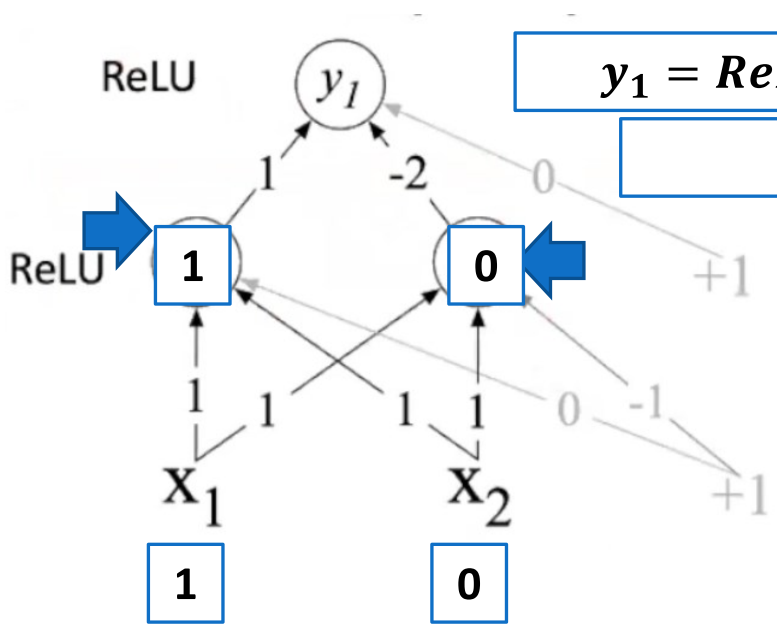
$$\boldsymbol{h_2 = ReLU(0) = 0}$$

Solution Credit: Goodfellow et al. (2016)

XOR Solution: Multilayer Perceptron

- XOR **can't** be calculated by a single perceptron
- XOR **can** be calculated by a layered network of units

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$\boldsymbol{y_1 = ReLU(1 \times 1 + 0 \times -2 + 1 \times 0)}$$

$$\boldsymbol{y_1 = ReLU(1) = 1}$$

$$\boldsymbol{h_1 = ReLU(1 \times 1 + 0 \times 1 + 1 \times 0)}$$

$$\boldsymbol{h_1 = ReLU(1) = 1}$$

$$\boldsymbol{h_2 = ReLU(1 \times 1 + 0 \times 1 + 1 \times (-1))}$$

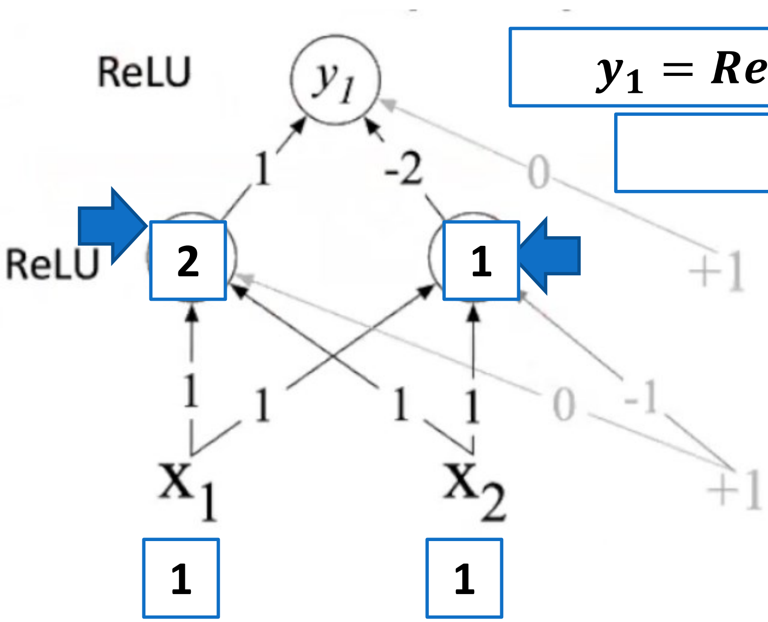
$$\boldsymbol{h_2 = ReLU(0) = 0}$$

Solution Credit: Goodfellow et al. (2016)

XOR Solution: Multilayer Perceptron

- XOR **can't** be calculated by a single perceptron
- XOR **can** be calculated by a layered network of units

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$y_1 = ReLU(2 \times 1 + 1 \times -2 + 1 \times 0)$$

$$y_1 = ReLU(0) = 0$$

$$h_1 = ReLU(1 \times 1 + 1 \times 1 + 1 \times 0)$$

$$h_1 = ReLU(2) = 2$$

$$h_2 = ReLU(1 \times 1 + 1 \times 1 + 1 \times (-1))$$

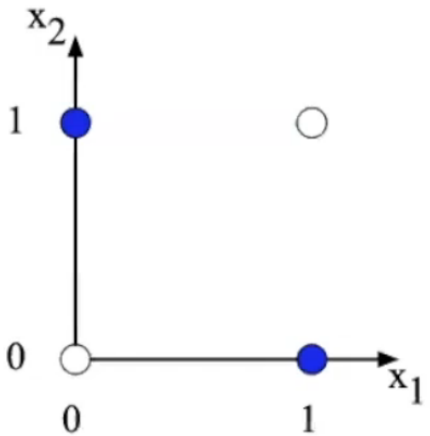
$$h_2 = ReLU(1) = 1$$

Solution Credit: Goodfellow et al. (2016)

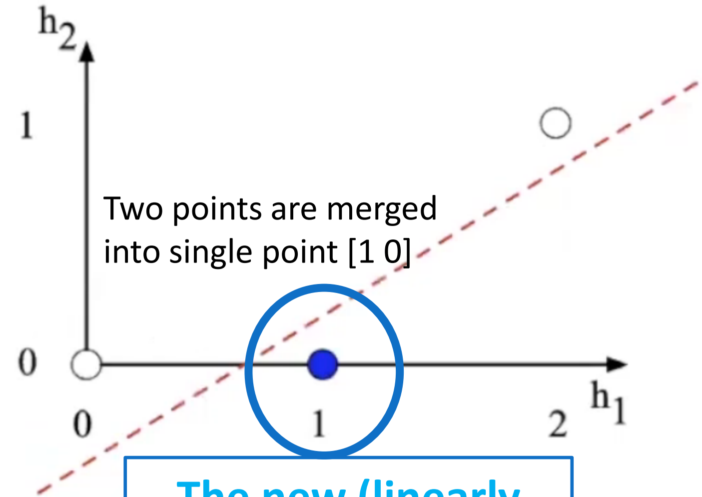
The Hidden Representation h

- Did you notice what happened to the **hidden space h for the inputs** where **only one input was 1?**

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0



The original x space



The new (linearly separable) h space

Lesson: Hidden layer learns a useful representation of the input

Solution Credit: Goodfellow et al. (2016)

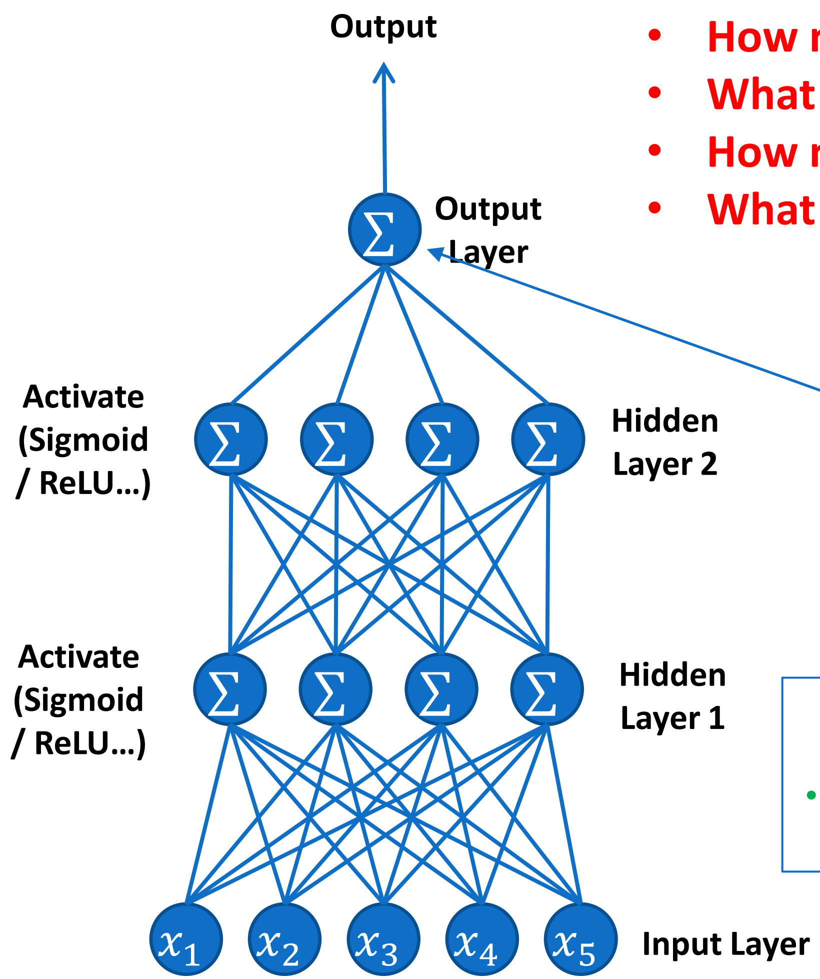
The Hidden Representation h

- In this example, we used predefined weights
- But in actual, the weights are learned using **Backpropagation Algorithm**
- Which means, the hidden layer **learns useful representation of the input during the training**
- **This intuition that Multilayer NNs can learn useful representation of the inputs automatically is one of their key advantages**

Implementation Details

- Can multilayer Perceptron/NNs only be used for classification tasks?
  - No! The output can also be a continuous label (i.e., regression)
  - The difference is, you will compute SSE as an error and use Gradient Descent just like we did in linear regression!

MLP to Deep Neural Network



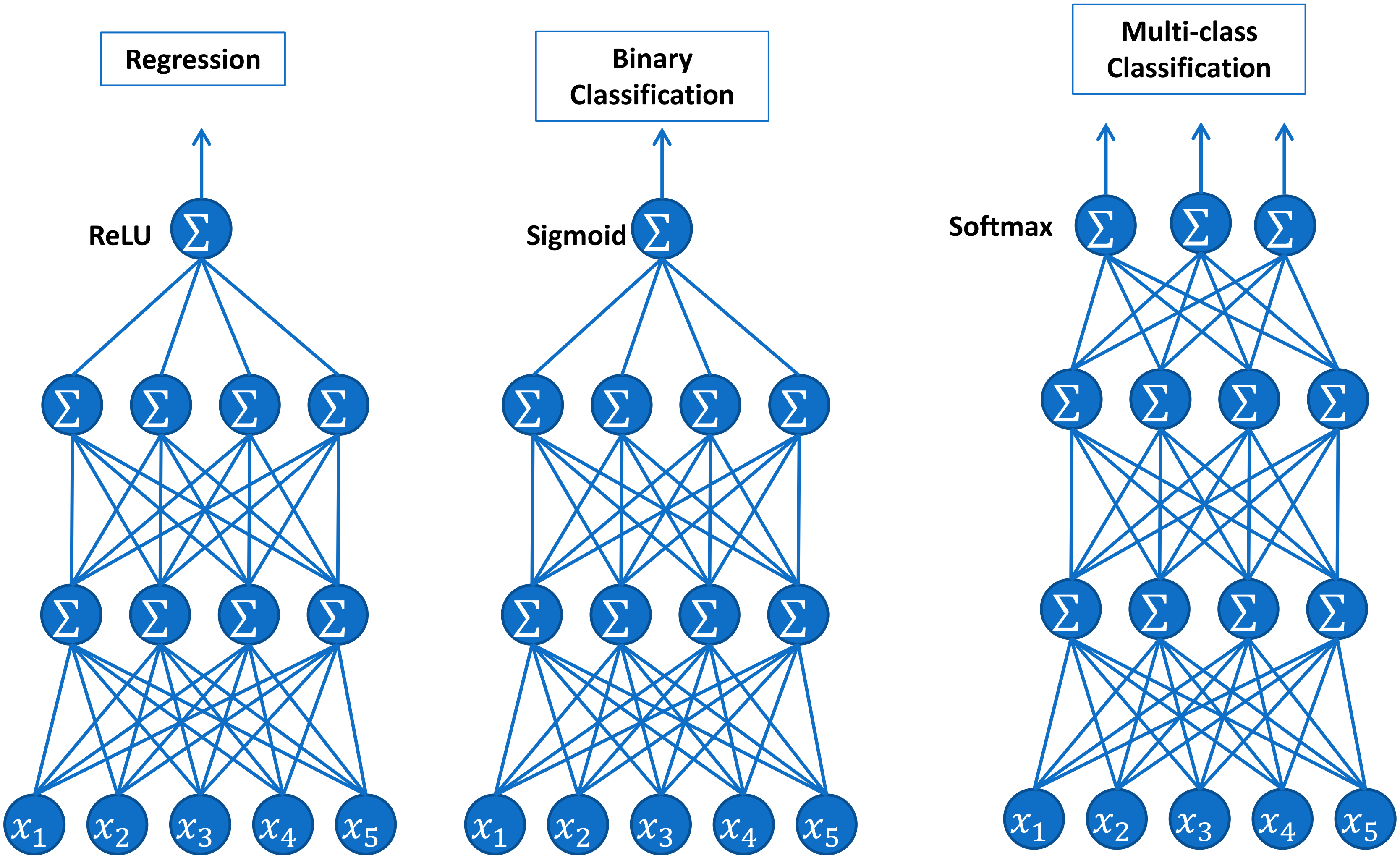
- How many inputs?
- How many neurons in each hidden layer?
- How many hidden layers?
- What activation function to use for hidden layer neurons?
- How many neurons in output layer?
- What activation function to use for output layer?

What activation function should be used at output layer?

It depends!!

- None if you want a regression output.
- ReLU if you want regression output but only positive values.
- Sigmoid if you want a probability $(between\ 0-1)$ for a class (i.e., binary classification.

Deep Neural Network For Different Tasks



Summary

- If your problem is a **regression** problem, you should use a **linear activation** function (i.e., multiply sum by 1 aka no activation).
  - **Regression:** One output node, linear activation.

- If your problem is a **classification** problem, then there are three main types of classification problems and each may use a different activation function.
  - If there are two mutually exclusive classes (binary classification).
    - **Binary Classification:** One output node, sigmoid activation.
  - If there are more than two mutually exclusive classes (multiclass classification).
    - **Multiclass Classification:** One output node per class, softmax activation.
  - If there are two or more mutually inclusive classes (multilabel classification),
    - **Multilabel Classification:** One output node per class, sigmoid activation.

What is multilabel classification?

Credit: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Book Reading

- Murphy – Chapter 8
- Jurafsky – Chapter 5, Chapter 4, Chapter 7
- Tom Mitchel – Chapter 4