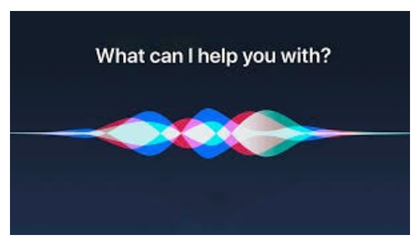
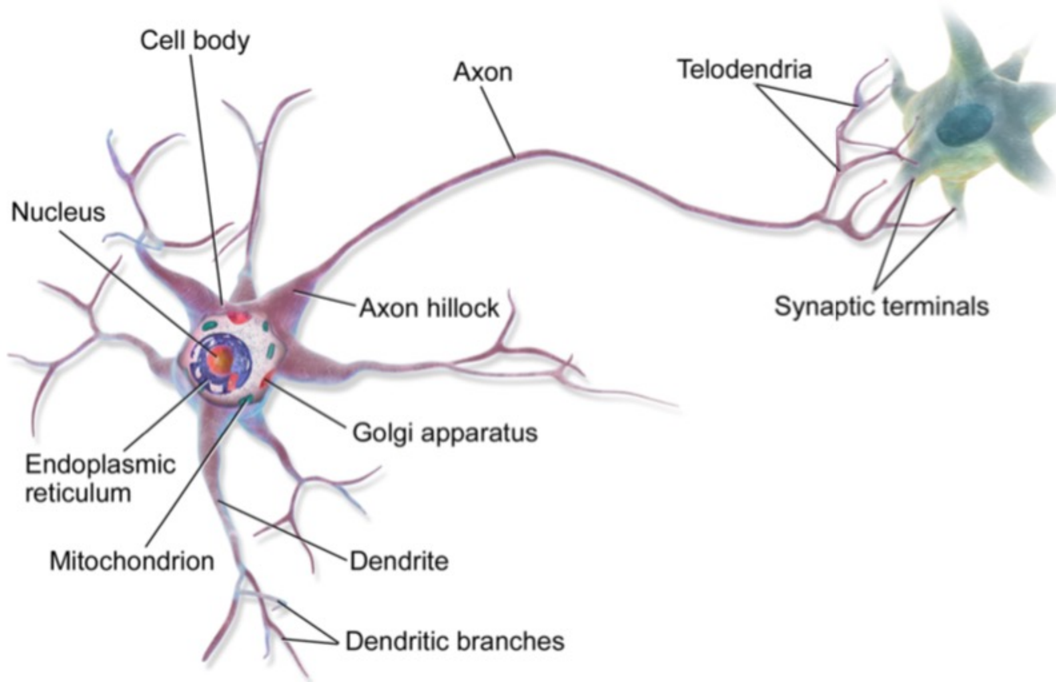
Deep Learning

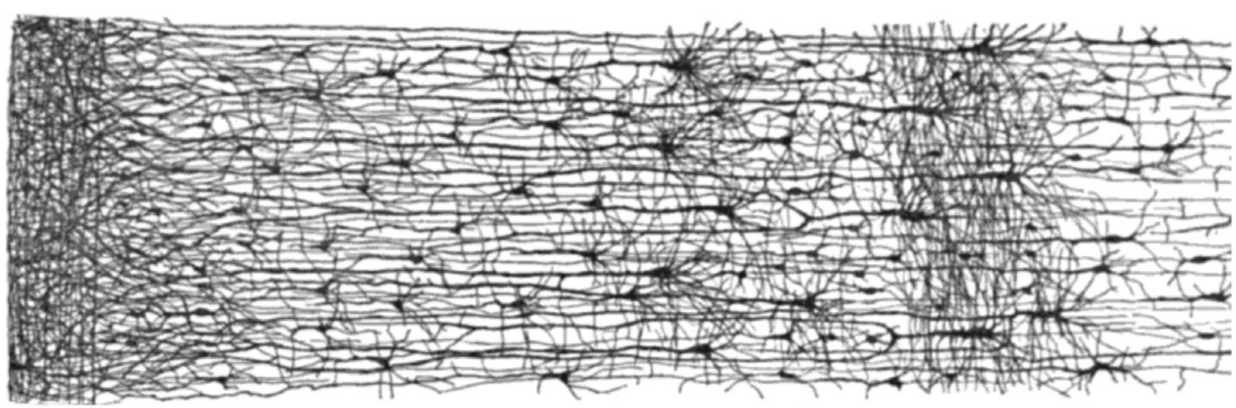
LEADING Bootcamp

Artificial Neural Network (ANN)

- Artificial Neural Network (ANN) is a Machine Learning model inspired by the networks of biological neurons found in our brains.
- ANN is the core of Deep Learning.

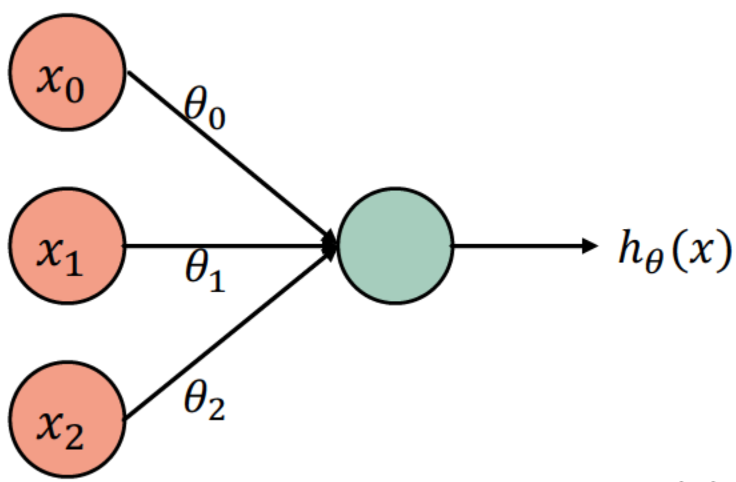



Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches



Perceptron

- Binary classifier that maps input to an output
- One of the simplest ANN architectures
- Based on Linear Threshold unit (LTU)



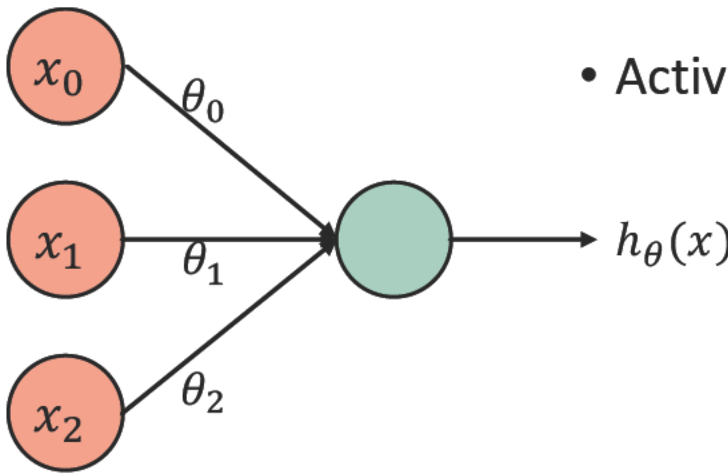
$$h_\theta(x) = \begin{cases} 0 & if \sum_j \theta_j x_j \leq threshold \\ 1 & if \sum_j \theta_j x_j > threshold \end{cases}$$

Perceptron

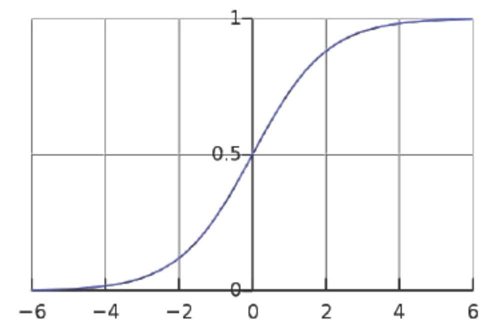
```
>>> from sklearn.linear_model import Perceptron

>>> per_clf = Perceptron(random_state=42)
>>> per_clf.fit(X, y)

>>> per_clf.predict([[2, 0.5]])
```

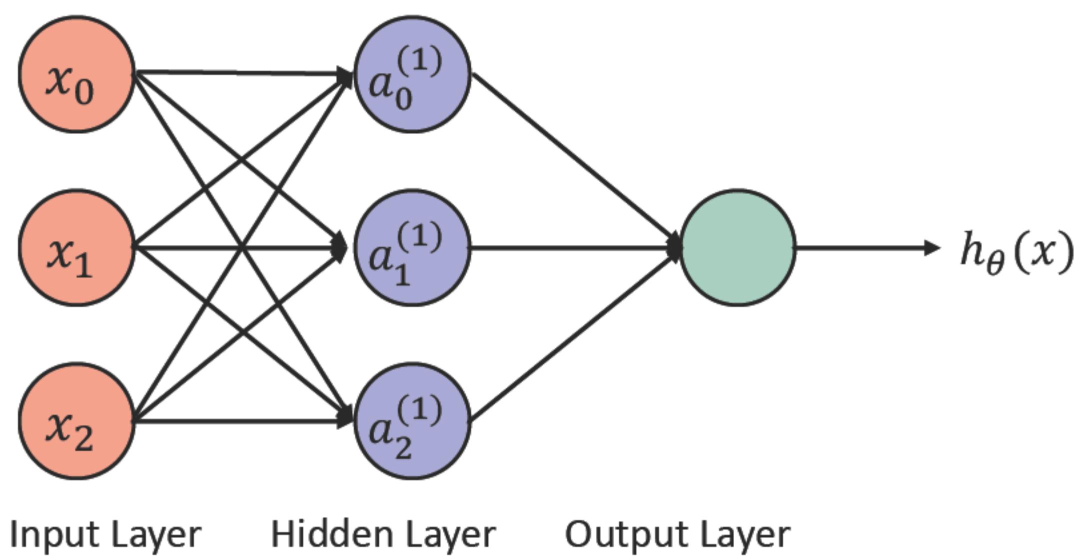
- Decision boundary is linear, incapable of learning complex patters
- This is the weakness of any other linear classifiers

Architecture

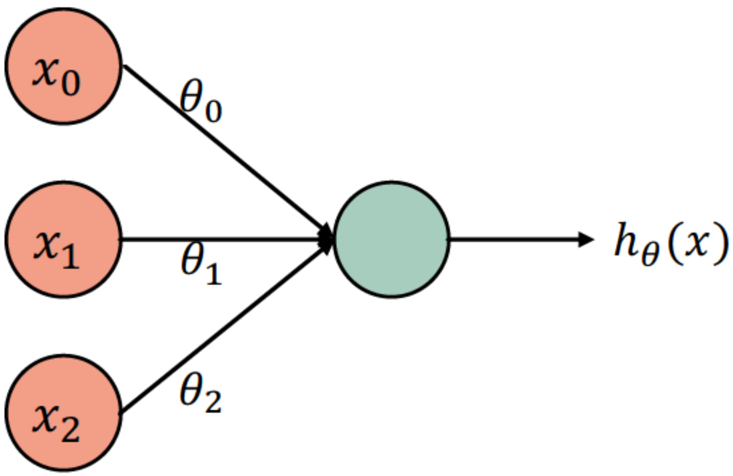


- Activation function: $g(z) = \frac{1}{1+e^{-z}}$





Logical computation

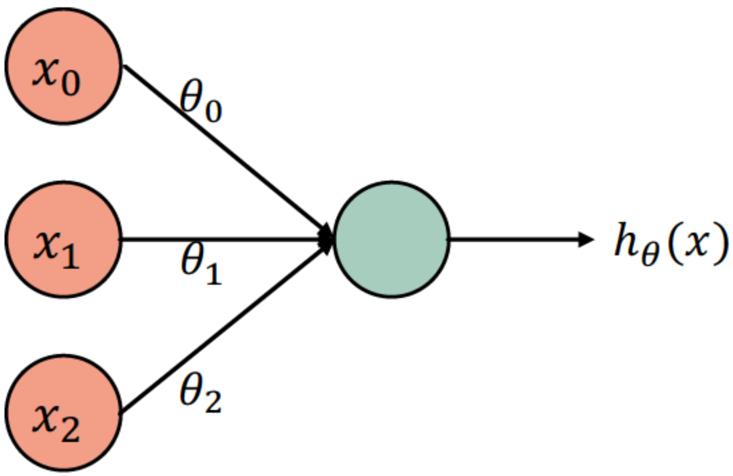


- $x_0 = 1$
- $x_1, x_2 \in \{0, 1\}$

- $\theta_0 = -30$
- $\theta_1 = 20$
- $\theta_2 = 20$
- $y = x_1 \ AND \ x_2$
- $h_\theta(x) = g(-30 + 20x_1 + 20x_2)$

x_1	x_2	$h_\theta(x)$
0	0	0
0	1	0
1	0	0
1	1	1

Logical computation

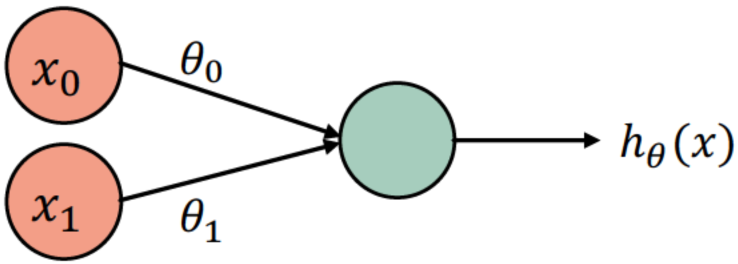


- $x_0 = 1$
- $x_1, x_2 \in \{0, 1\}$

- $\theta_0 = -10$
- $\theta_1 = 20$
- $\theta_2 = 20$
- $y = x_1 \ OR \ x_2$
- $h_\theta(x) = g(-10 + 20x_1 + 20x_2)$

x_1	x_2	$h_\theta(x)$
0	0	0
0	1	1
1	0	1
1	1	1

Logical computation

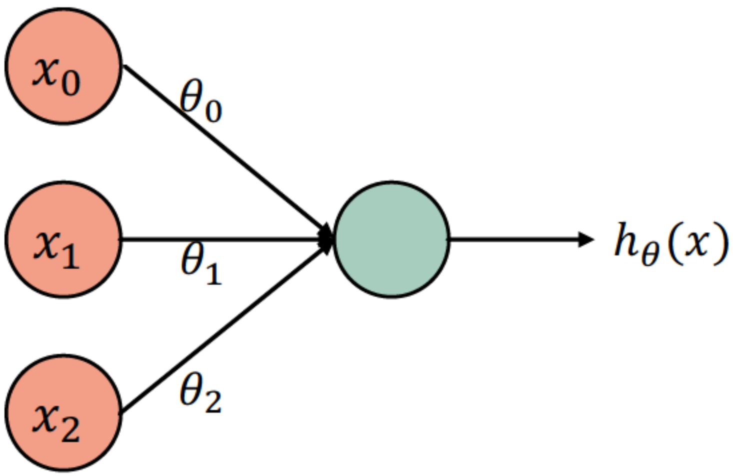


- $x_0 = 1$
- $x_1 \in \{0, 1\}$

- $\theta_0 = 10$
- $\theta_1 = -20$
- $y = NOT\ x_1$
- $h_\theta(x) = g(10 - 20x_1)$

x_1	$h_\theta(x)$
0	1
1	0

Logical computation



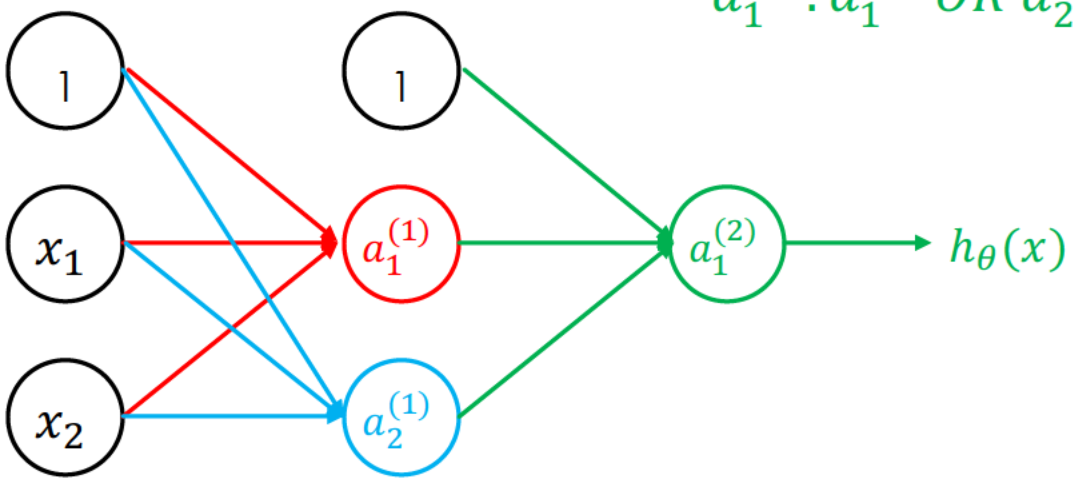
- $x_0 = 1$
- $x_1, x_2 \in \{0, 1\}$

- $\theta_0 = 10$
- $\theta_1 = -20$
- $\theta_2 = -20$
- $y = (\text{NOT } x_1)\ AND\ (NOT\ x_2)$
- $h_\theta(x) = g(10 - 20x_1 - 20x_2)$

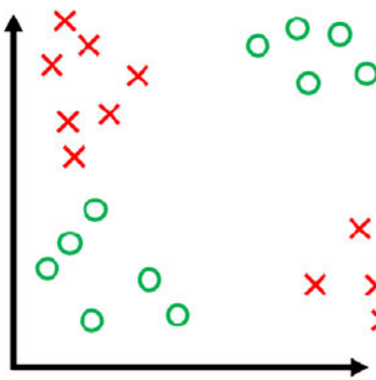
x_1	x_2	$h_\theta(x)$
0	0	1
0	1	0
1	0	0
1	1	0

$y = x_1 \ XNOR \ x_2$

- $a_1^{(1)}: x_1 \ AND \ x_2$
- $a_2^{(1)}: (\text{NOT } x_1) \ AND \ (NOT \ x_2)$
- $a_1^{(2)}: a_1^{(1)} \ OR \ a_2^{(1)}$



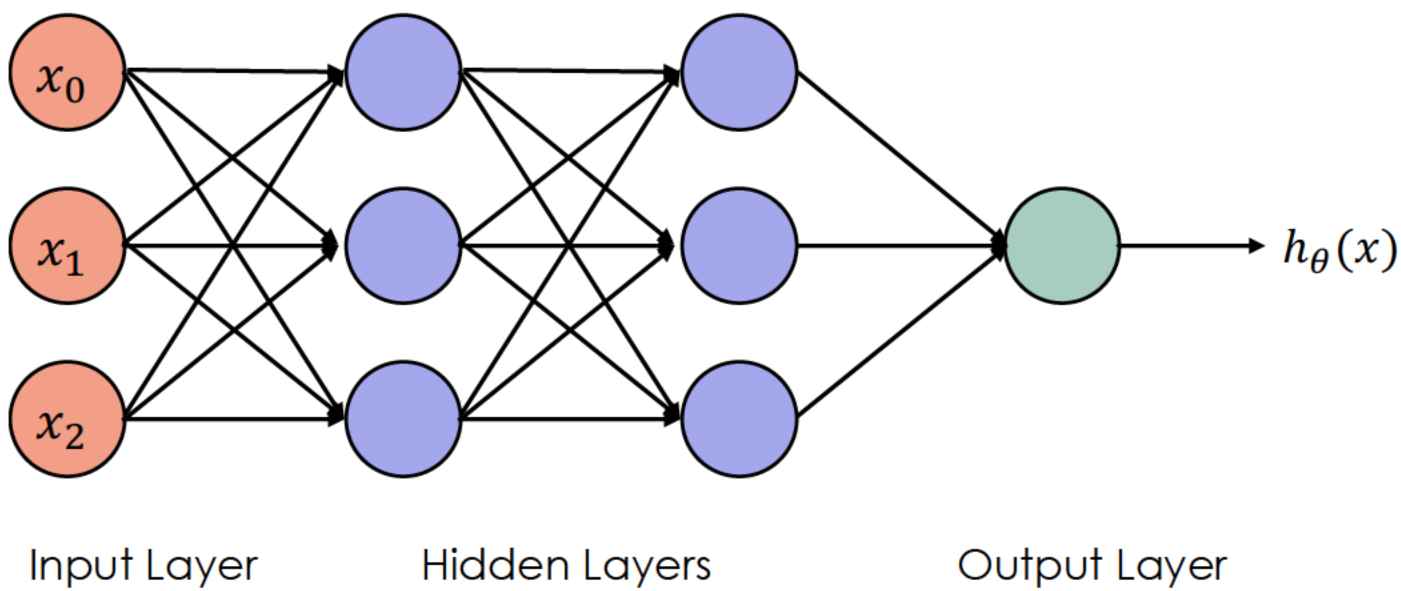
x_1	x_2	$a_1^{(1)}$	$a_2^{(1)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1



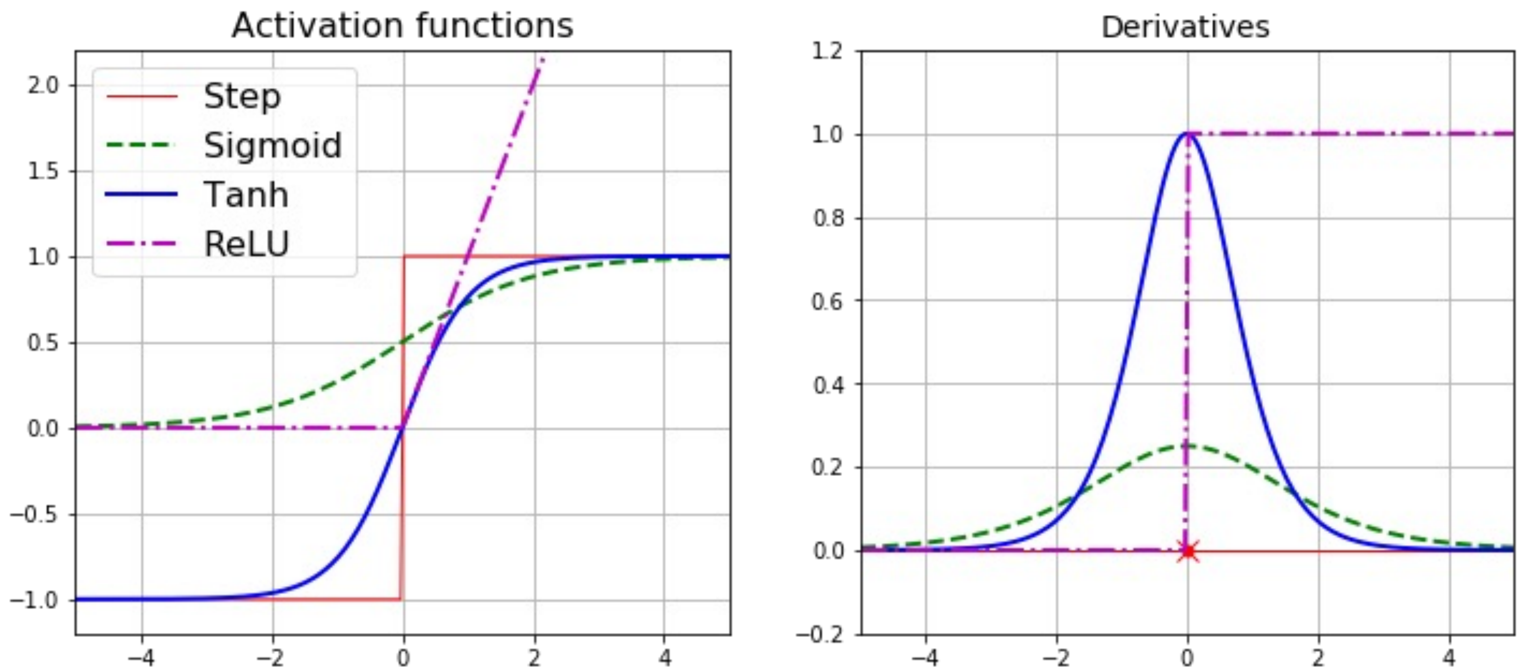
[Minsky 1969]

Multi-layer Perceptron (MLP)

- An ANN contains a deep stack of hidden layers is called Deep Neural Network (DNN)



Non-Linear Activation Function

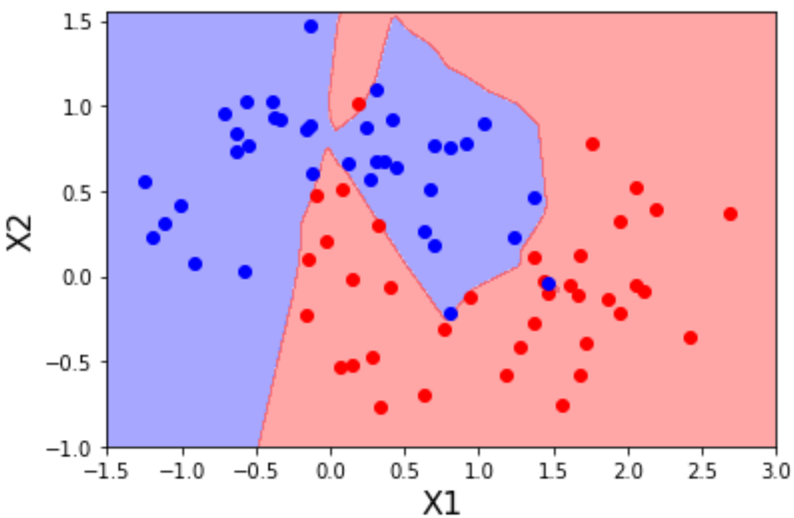


- Tanh (The hyperbolic tangent function): $\tanh(\mathrm{z}) = \frac{2}{1+e^{-2z}} - 1$
  - Continuous and differentiable
  - Range:[-1, 1], help speed up convergence
- ReLU (Rectified Linear Unit function):
  - Continuous and differentiable (except for z = 0)
  - Fast to compute, usually set as default

MLP Classification

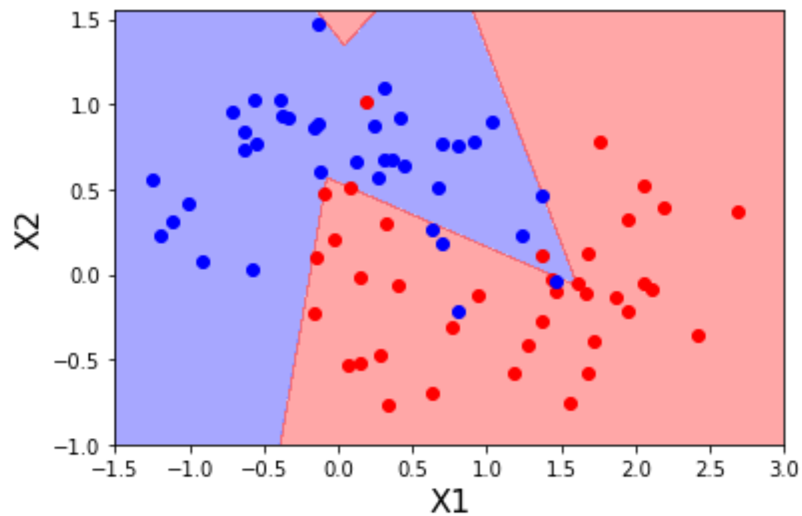
```
>>> from sklearn.neural_network import MLPClassifier
>>> mlp = MLPClassifier().fit(X_train, y_train)
```

hidden_layer_sizes=(100,)



Train score: 0.9867
Test score: 0.9200

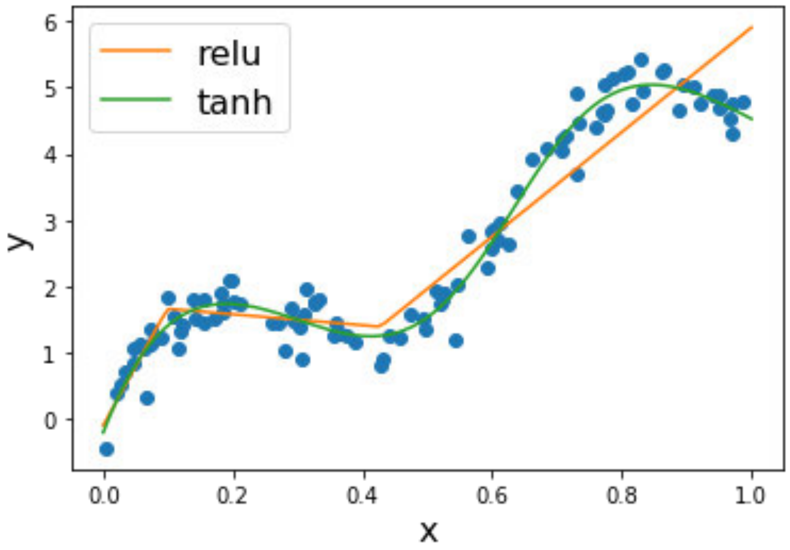
hidden_layer_sizes=(10,)



Train score: 0.9467
Test score: 0.9600

MLP Regression

```
>>> from sklearn.neural_network import MLPRegressor

>>> mlp_relu = MLPRegressor(solver="lbfgs").fit(X, y)

>>> mlp_tanh = MLPRegressor(solver="lbfgs",
activation='tanh').fit(X, y)
```



MLP summary

- Can be formed into advanced architecture: RNN, CNN…
- Capable of capture complex features
- Work well with large datasets
- Problems:
  - Longer training time – require GPUs
  - Prone to overfitting
  - Careful pre-processing is needed
  - Similarly to SVMs and other linear models, they work best with "homogeneous" data, where all the features have similar meanings.
  - For data that has very different kinds of features, tree-based models might work better.