# Machine Learning

Patrick Barry

2/28/2020 Group Meeting

# Definition

- “Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.” – Nvidia
- “Machine learning is the science of getting computers to act without being explicitly programmed.” – Stanford
- “Machine learning is based on algorithms that can learn from data without relying on rules-based programming.”- McKinsey & Co.
- “Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.” – University of Washington
- “The field of Machine Learning seeks to answer the question “How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?” – Carnegie Mellon University

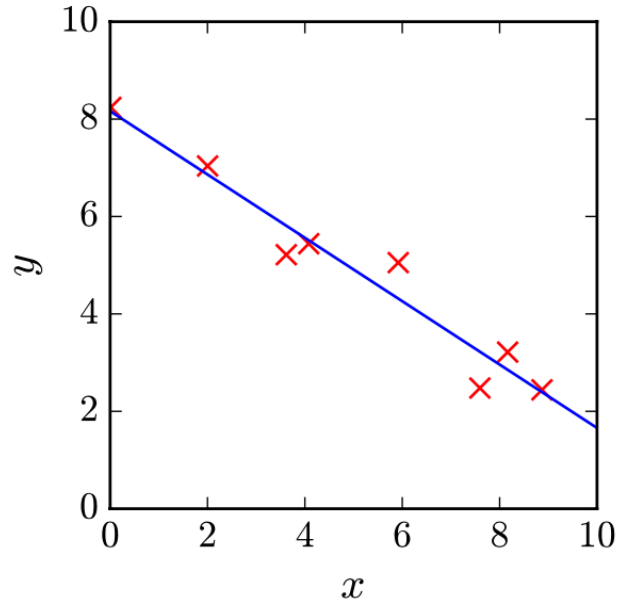
# Categories of Machine Learning

1. Unsupervised
   - Algorithm attempts to learn meaningful structures in the data
   - Examples are k-means/hierarchical clustering
2. Supervised
   - Involves output label associated with each instance in the dataset
   1. Real-Valued
      - Regression Models
   2. Discrete/Categorical
      - Classification models

# Review: Linear Regression

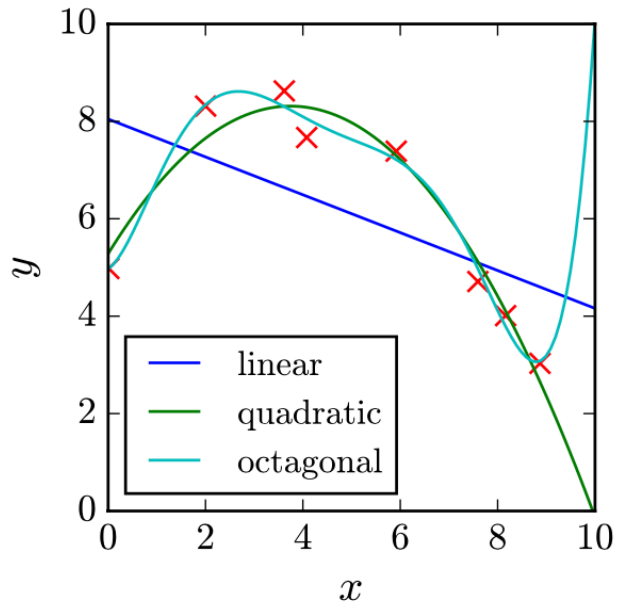
How to fit data

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \vec{a}) - y_i)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a$ when $\left( \frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \Big|_{x,y} = 0 \right)$



# Review: ~~Linear~~ Regression

Quadratic?




## How to fit data

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x + a_2 x^2$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \vec{a}) - y_i)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a$ when $\left( \frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \Big|_{x,y} = 0 \right)$

Is that good enough? How many parameters can we add?

# Logistic Regression

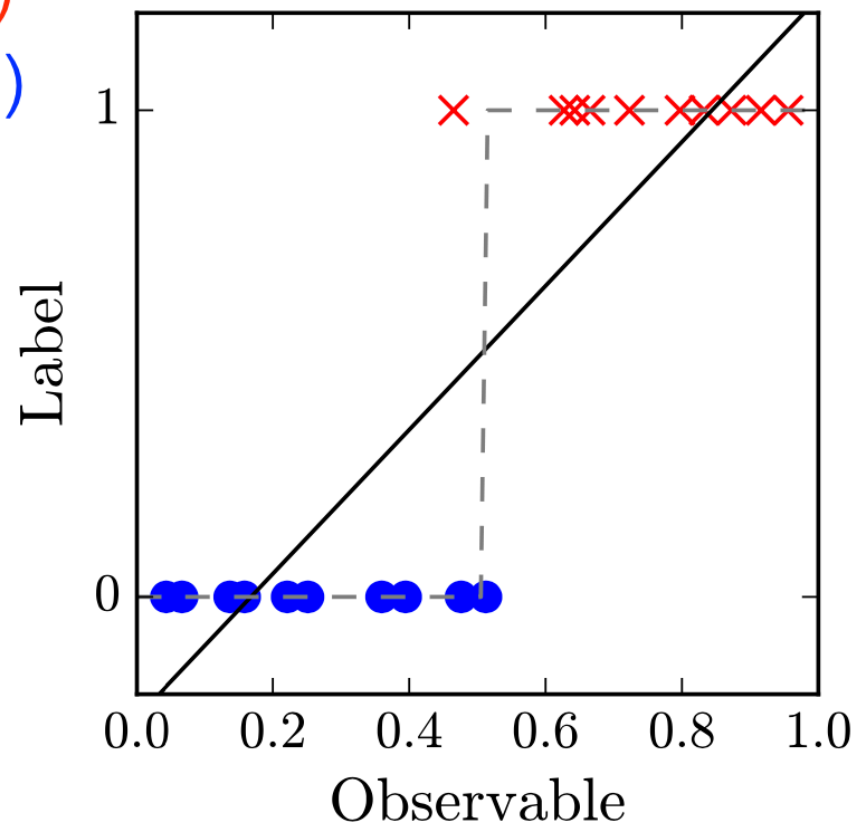
What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)

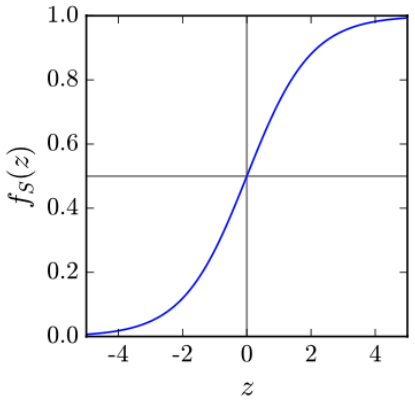
Other class as 0 (Background)

- Linear Fit?
- Round to nearest number?
- How does it generalize to more data?



# Logistic Regression

## What if we are trying to predict a class, not a number?

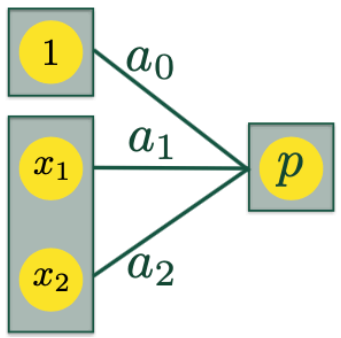



$$L(\vec{x},\vec{y},\vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(f_S(p(x,a))\right) + (1-y_i)\log\left(1-f_S(p(x,a))\right)\right)$$

$$f_S(z) = \frac{1}{1+e^{-z}} \qquad z = p(x,a)$$

What is $p(x,a)$ ?

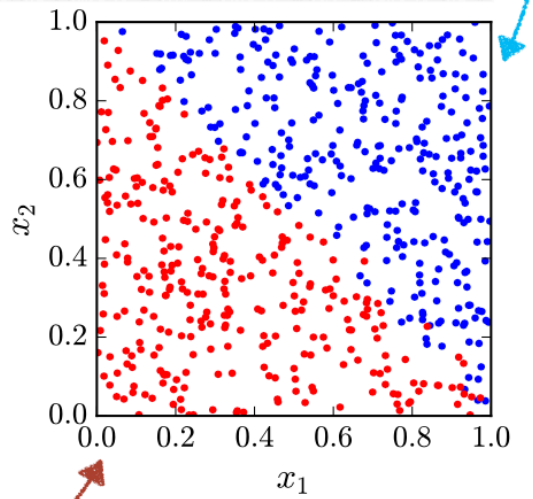
$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$




Minimize the loss with respect to $\vec{a}$

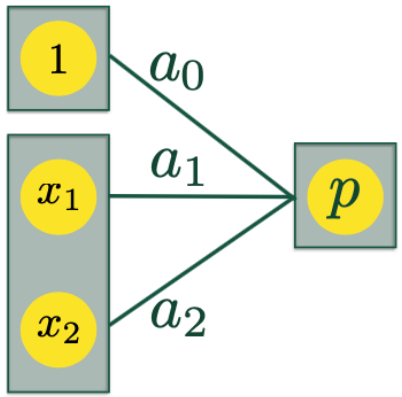
Boundary at $p(x,a) = 0$

Large + values of $p$



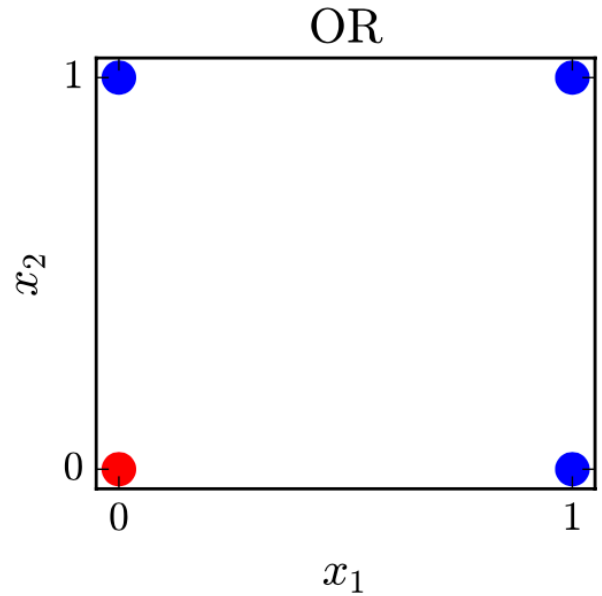

Large - values of $p$

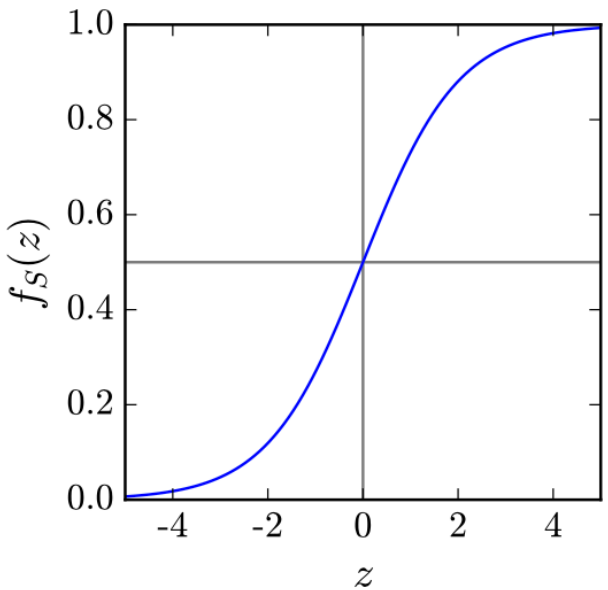
# Neural Networks



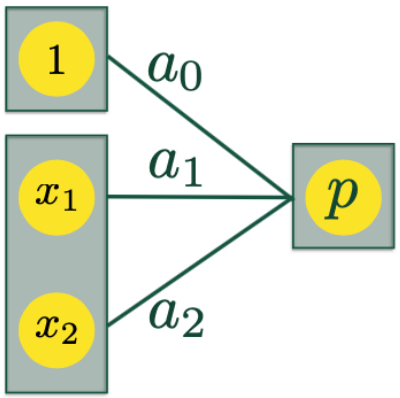

OR

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



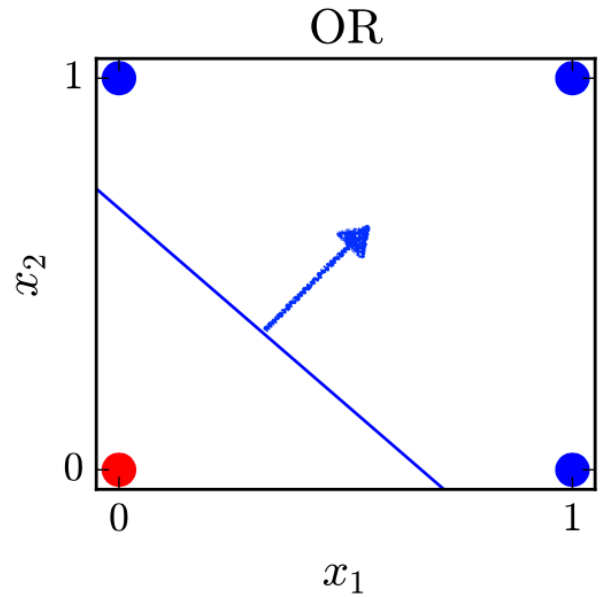



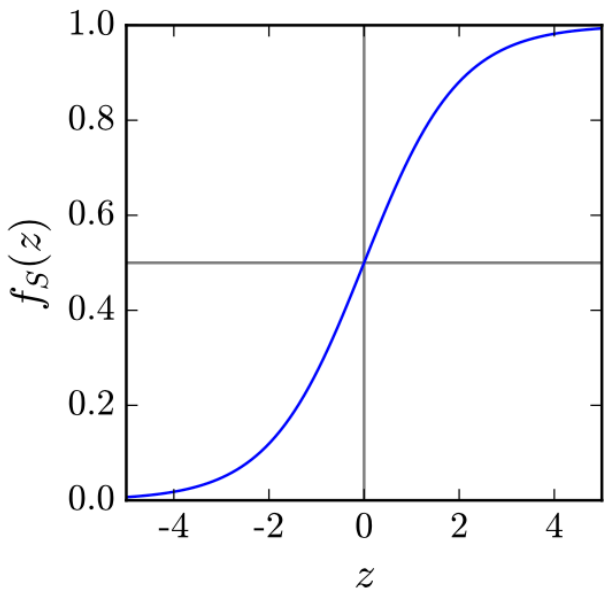
# Neural Networks




OR

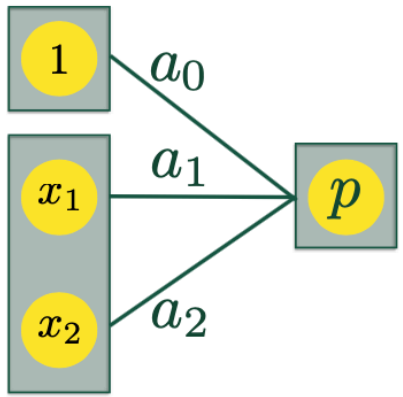
| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

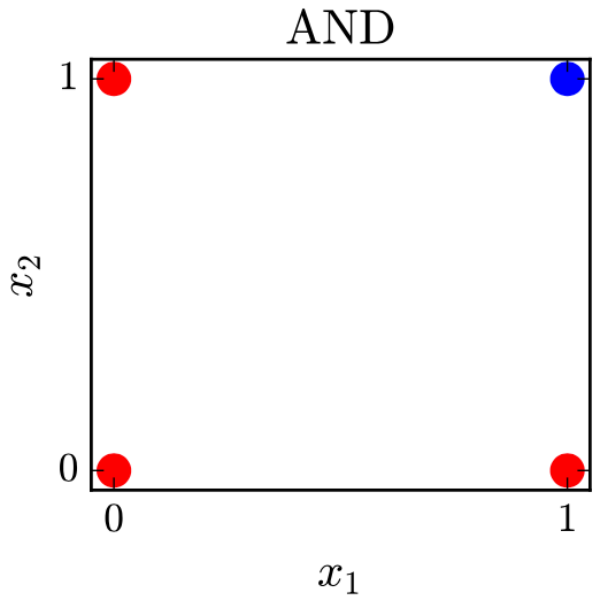



$$a_0 = -10, \quad a_1 = 15, \quad a_2 = 15$$

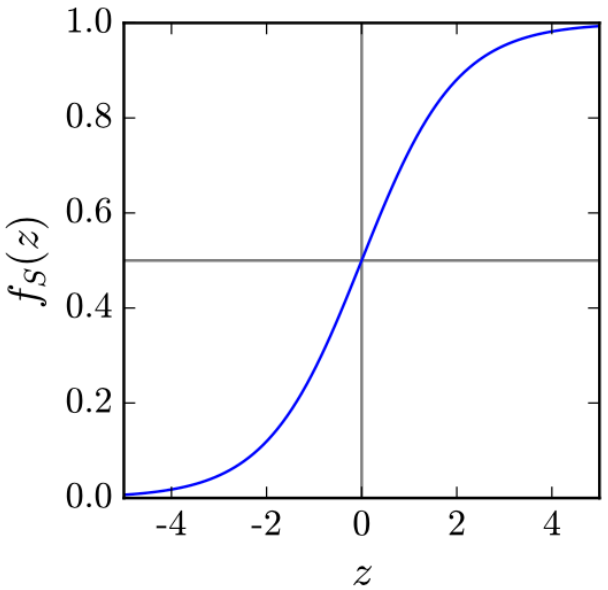
# Neural Networks



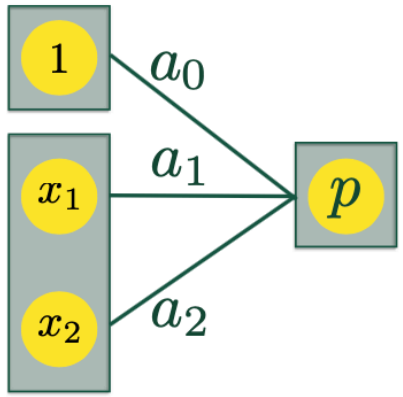



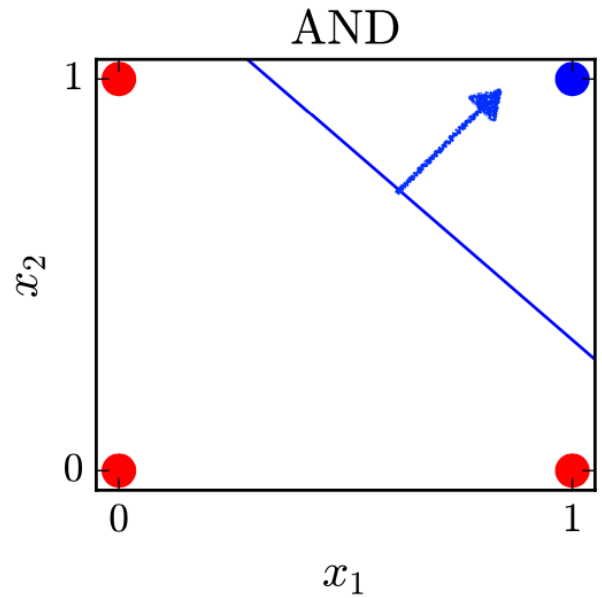

AND

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

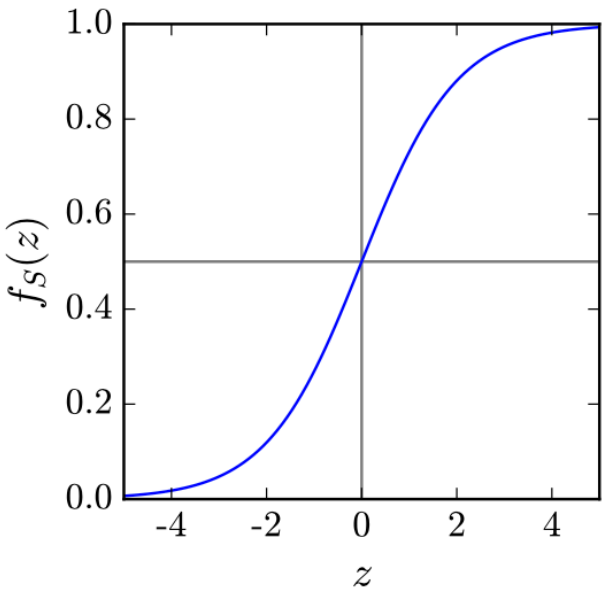


# Neural Networks



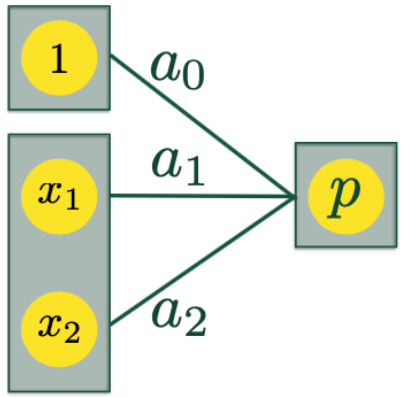




| AND | | |
|---|---|---|
| $x_1$ | $x_2$ | y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




$$a_0 = -20, \quad a_1 = 15, \quad a_2 = 15$$

# Neural Networks




**OR**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$a_0 = -10, \quad a_1 = 15, \quad a_2 = 15$

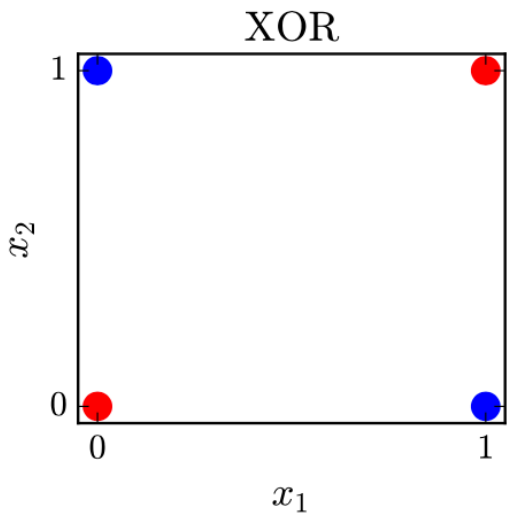
**AND**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$a_0 = -20, \quad a_1 = 15, \quad a_2 = 15$

**XOR**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



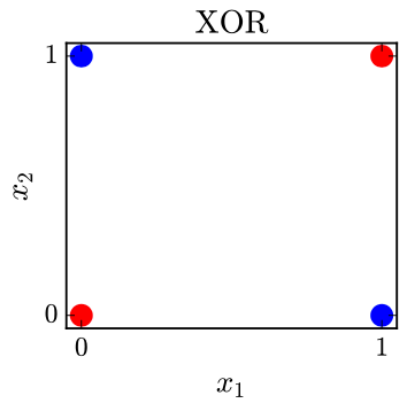

This system cannot produce XOR

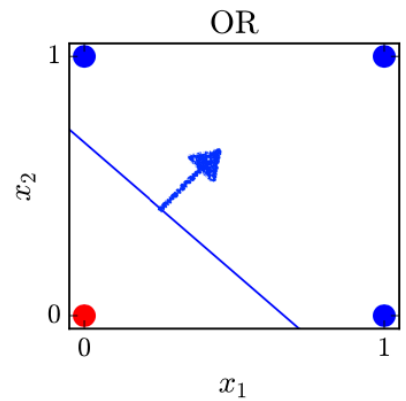
(cannot make a two sided cut)

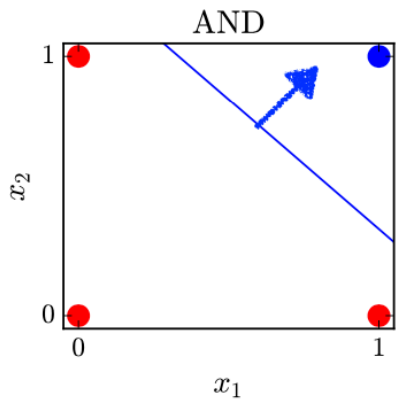
# Neural Networks

XOR

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

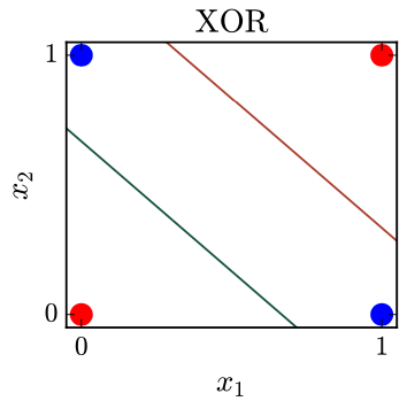


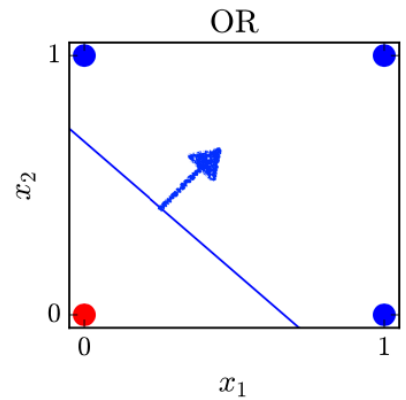



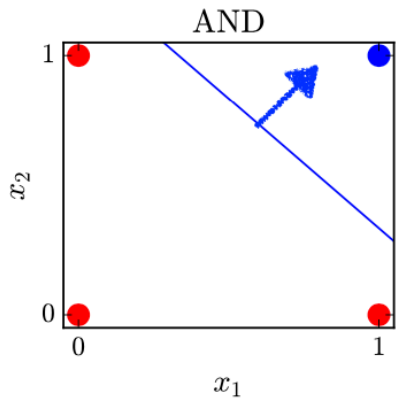




# Neural Networks

| XOR | | |
|---|---|---|
| $x_1$ | $x_2$ | y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

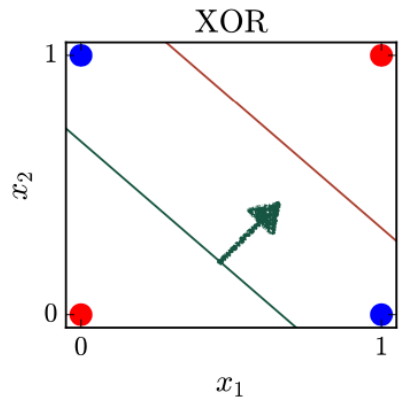


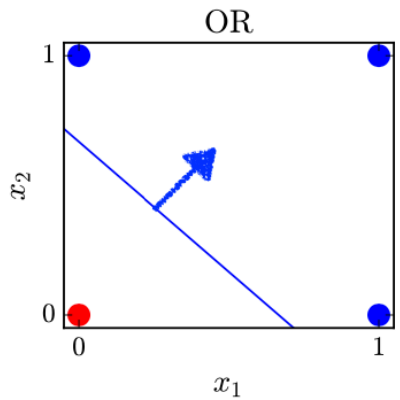



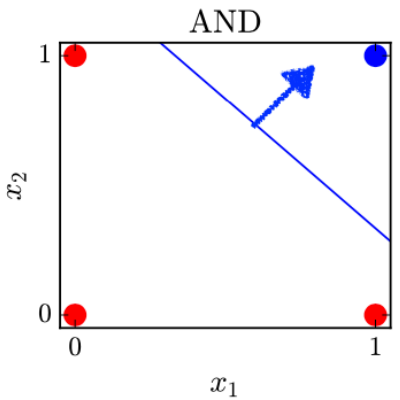



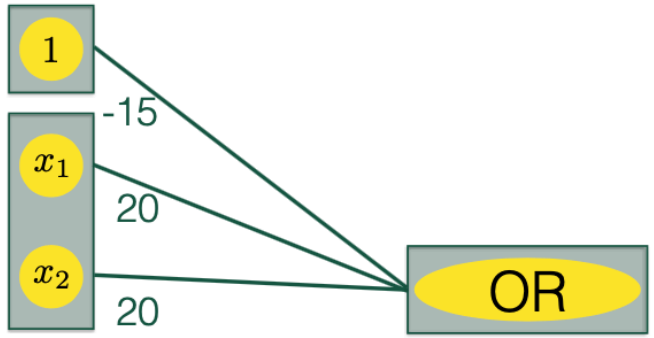

# Neural Networks

| XOR | | |
|---|---|---|
| $x_1$ | $x_2$ | y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



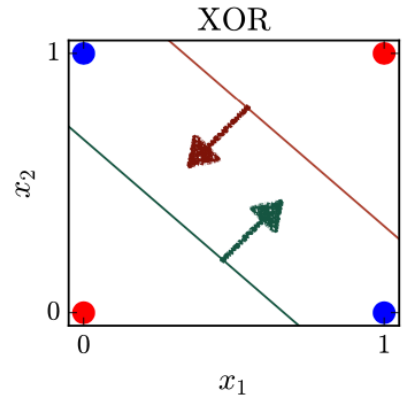



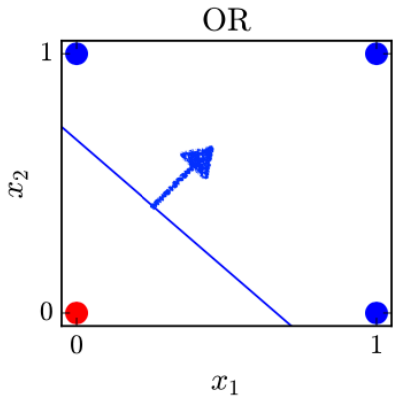



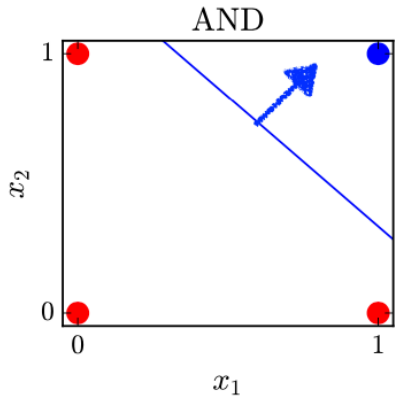
# Neural Networks

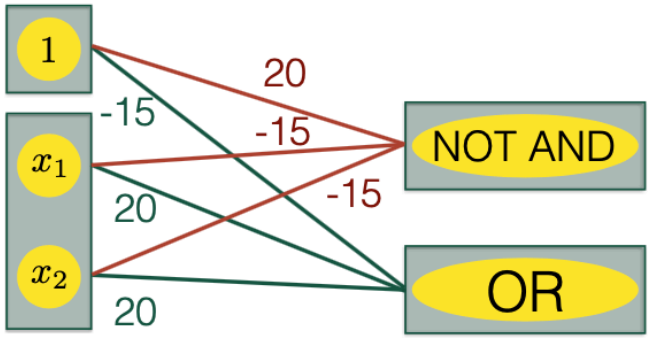
## XOR

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



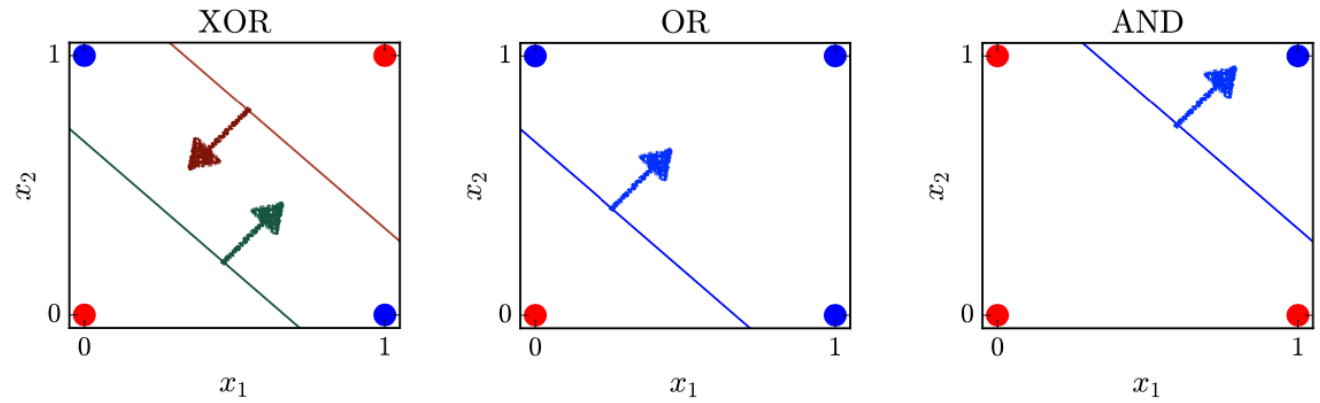



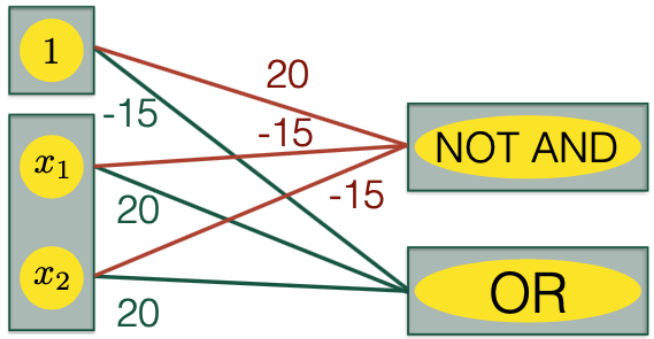



# Neural Networks

| XOR | | |
|---|---|---|
| $x_1$ | $x_2$ | y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

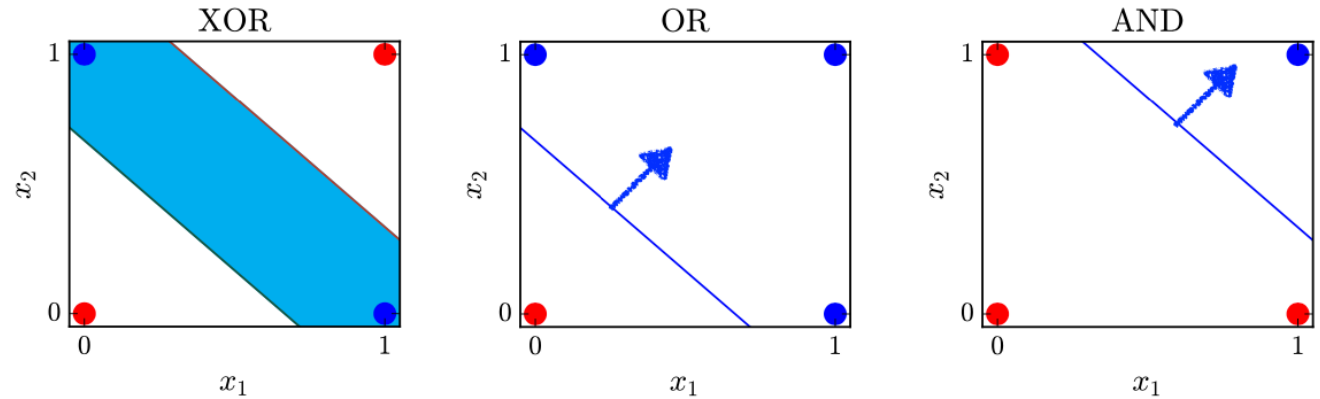


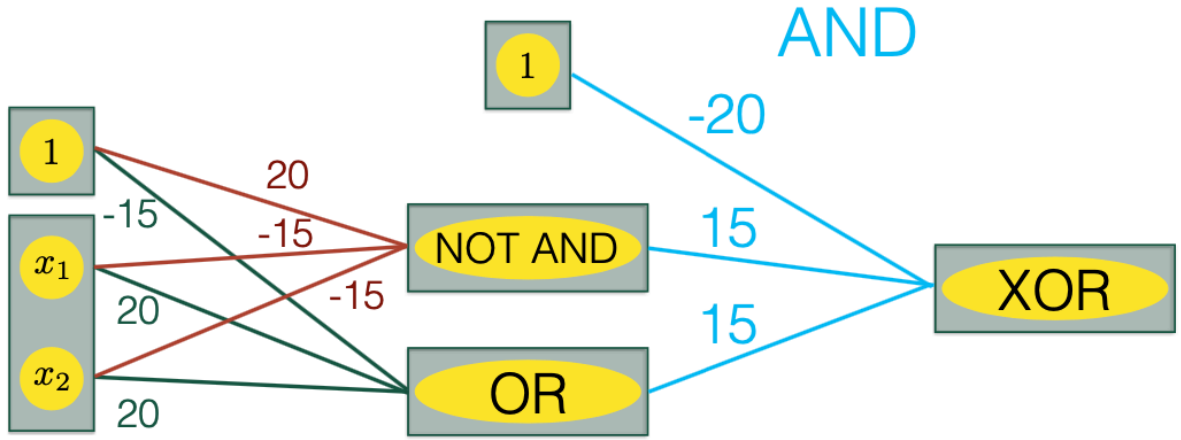




| $x_1$ | $x_2$ | OR | NOT AND | XOR |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

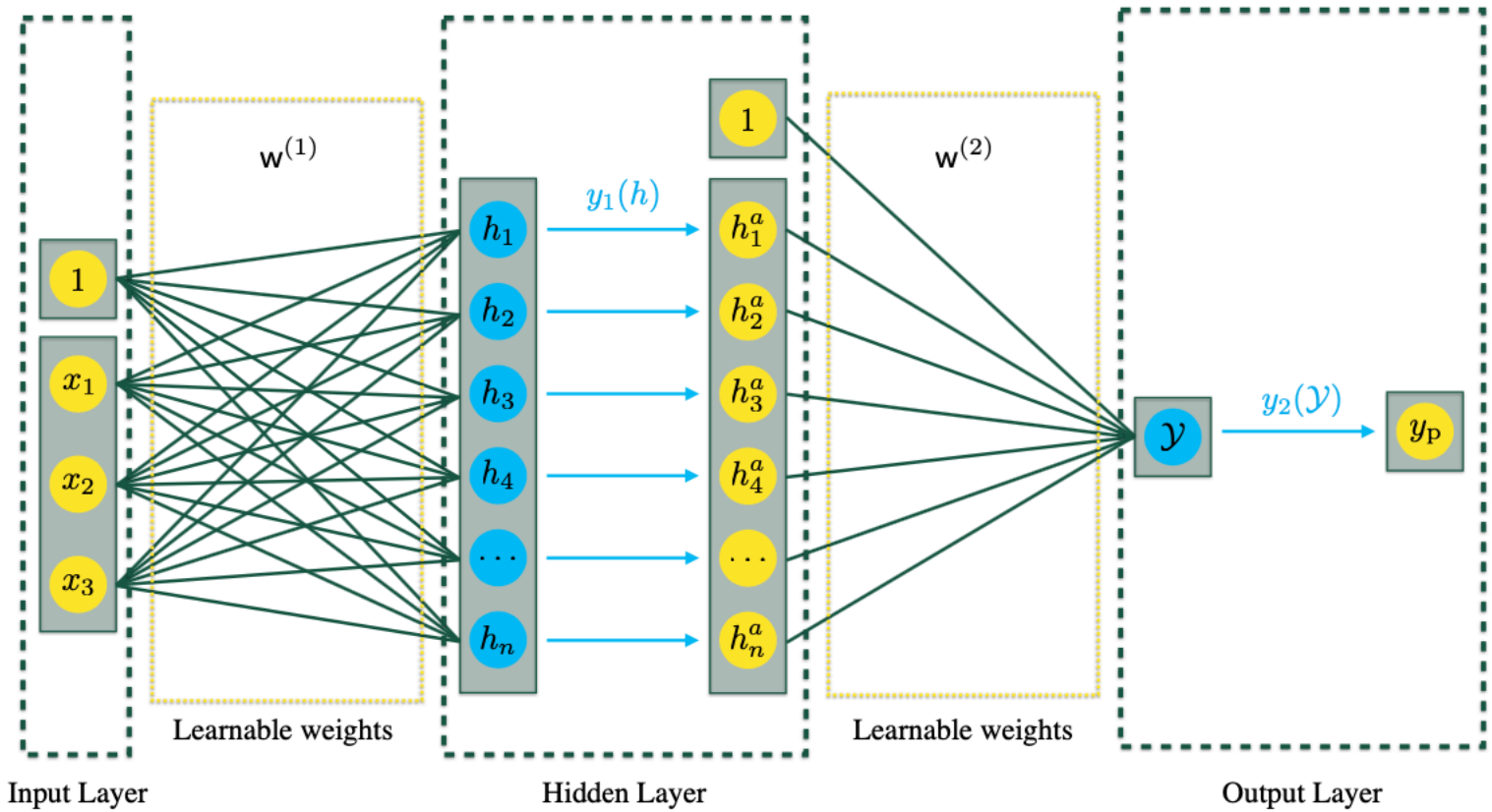
# Neural Networks

| XOR | | |
|---|---|---|
| $x_1$ | $x_2$ | y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

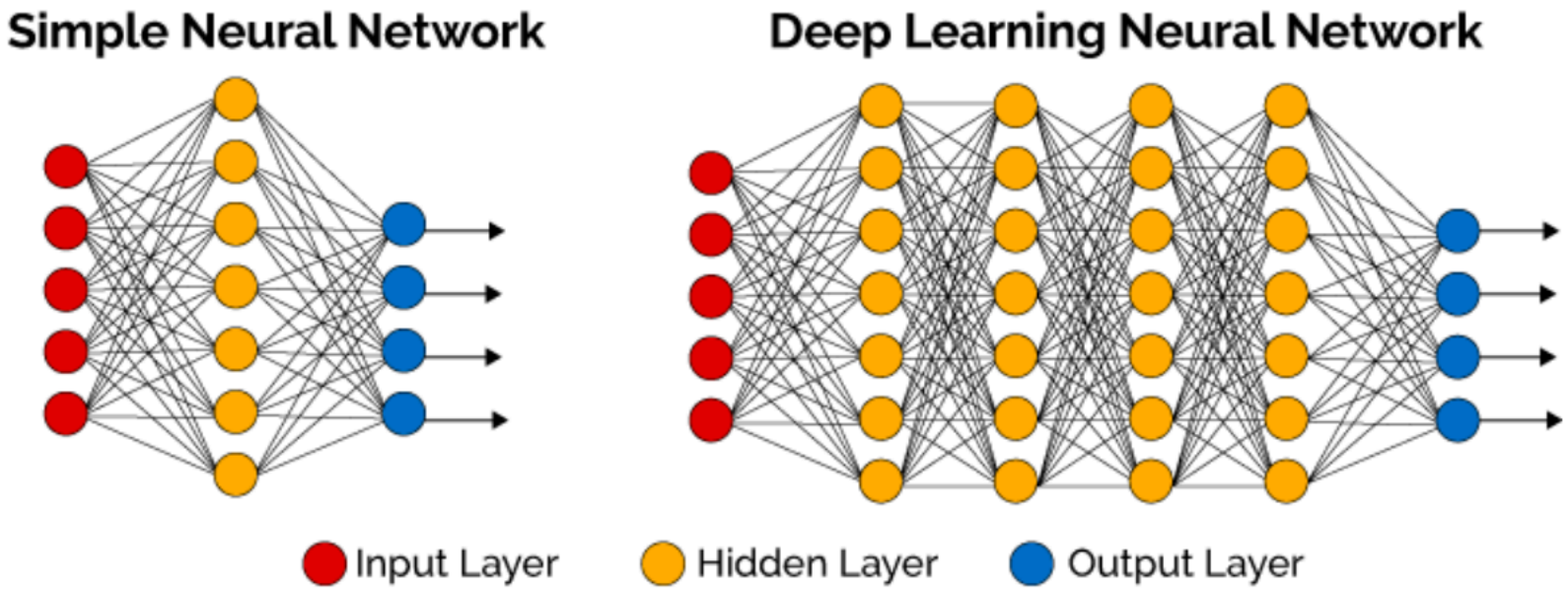



| $x_1$ | $x_2$ | OR | NOT AND | XOR |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |


$w^{(1)}$
$w^{(2)}$
1
$x_1$
$x_2$
$x_3$
$h_1$
$h_2$
$h_3$
$h_4$
...
$h_n$
$y_1(h)$
1
$h_1^a$
$h_2^a$
$h_3^a$
$h_4^a$
...
$h_n^a$
$y$
$y_2(y)$
$y_p$
Learnable weights
Learnable weights
Input Layer
Hidden Layer
Output Layer

# Deep Learning

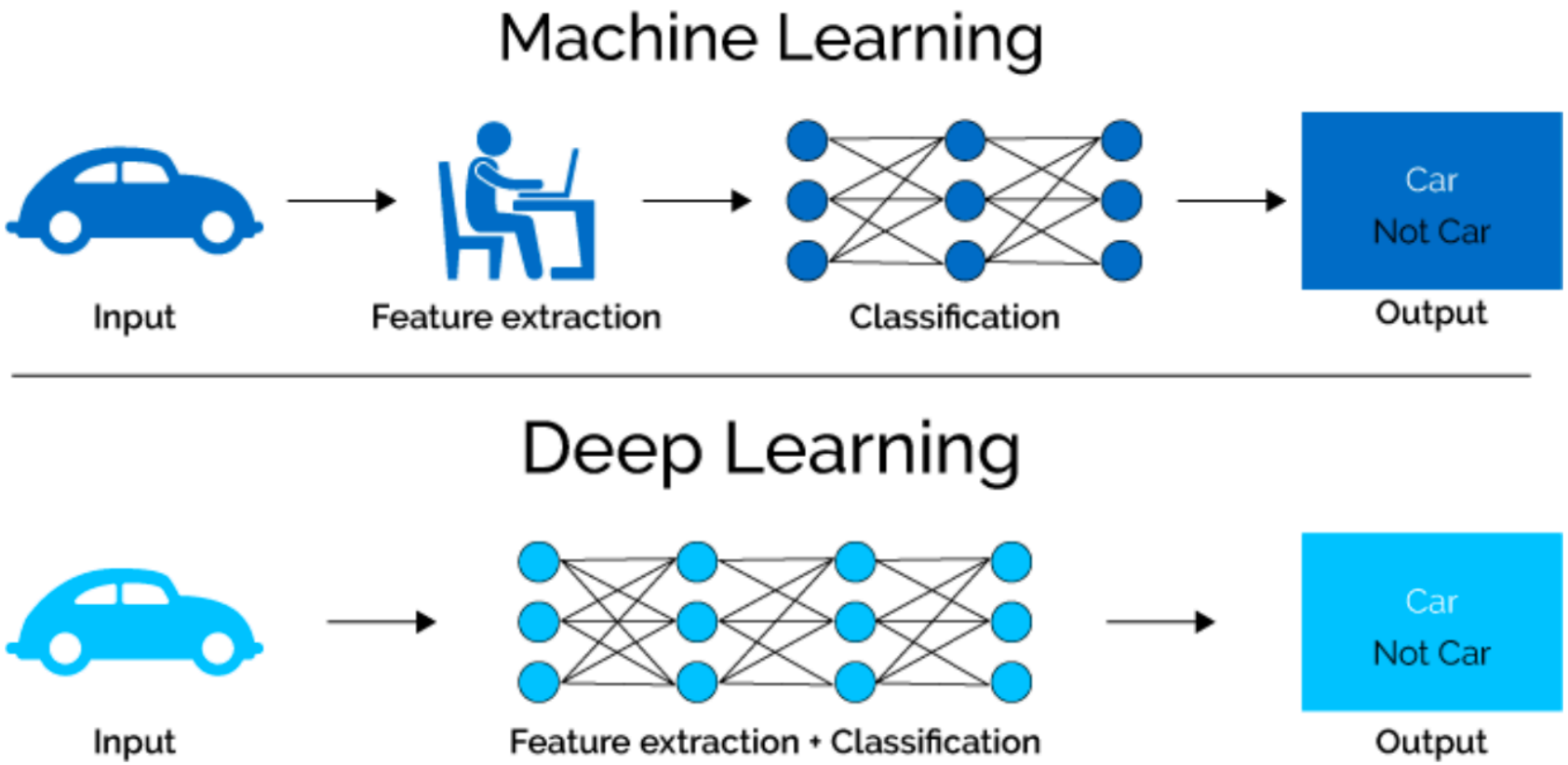



Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network

- In theory, one can fit ***any*** function with infinite data and infinite nodes with only 1 hidden layer
- However, going deeper rather than wider learns non-linearities with fewer parameters

# Deep Learning

Part of the deep learning revolution is end-to-end learning




Pic Credit: Xenonstack | Machine Learning vs Deep Learning

tensorflow.org