Deep Learning for Computer Vision

Neural Networks: A Review

Vineeth N Balasubramanian

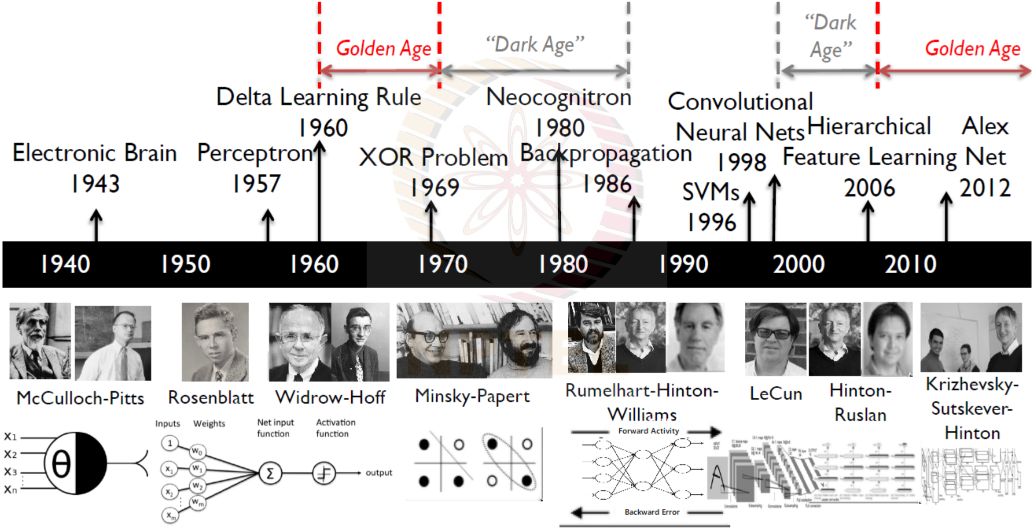
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Acknowledgements

- Most content of this lecture are based on **Lecture 2** of **CS7015** course taught by Mitesh Khapra at IIT-Madras

History of Neural Networks

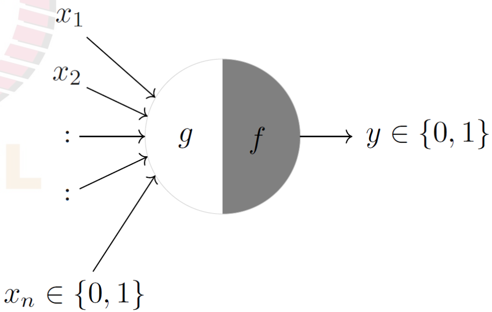


McCulloch-Pitts Neuron

- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- g aggregates the inputs and the function f takes a decision based on this aggregation
- The inputs can be excitatory or inhibitory
- $y = 0$ if any x_i is inhibitory, else

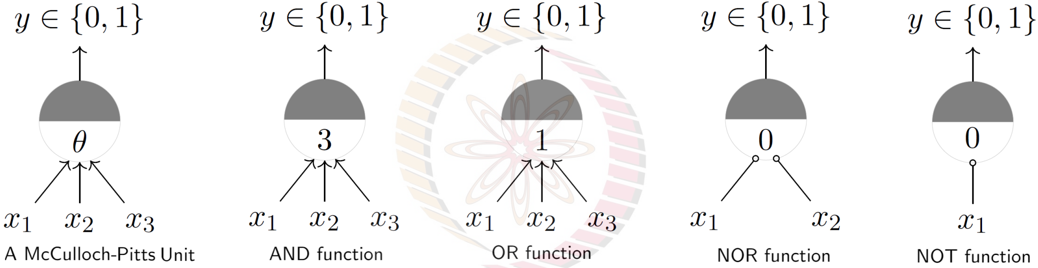
$$g(x_1, x_2, \ldots, x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \; \textit{if} \; g(\mathbf{x}) \geq \theta \\ &= 0 \; \textit{if} \; g(\mathbf{x}) < \theta \end{aligned}$$



- θ is a thresholding parameter

McCulloch-Pitts Neuron



- Feedforward MP networks can compute any Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$
- Recursive MP networks can simulate any Deterministic Finite Automaton (DFA) (See this paper[1] for more information)

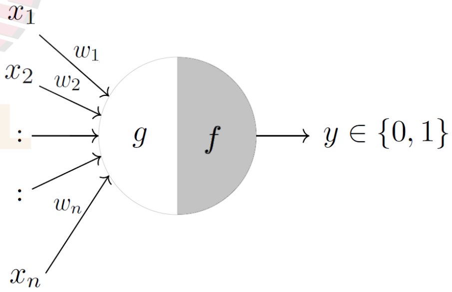
[1] Forcada, Neural Networks: Automata and Formal Models of Computation, 2002

Perceptrons

- Frank Rosenblatt, an American psychologist, proposed the perceptron model (1958)
- Later refined and carefully analyzed by Minsky and Papert (1969)
- A more general computational model than McCulloch-Pitts neurons
- Inputs are no longer limited to boolean values

$$g(x_1, x_2, \ldots, x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} w_i * x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \; if \; g(\mathbf{x}) \geq \theta \\ &= 0 \; if \; g(\mathbf{x}) < \theta \end{aligned}$$



Perceptrons

- Frank Rosenblatt, an American psychologist, proposed the perceptron model (1958)
- Later refined and carefully analyzed by Minsky and Papert (1969)
- A more general computational model than McCulloch-Pitts neurons
- Inputs are no longer limited to boolean values

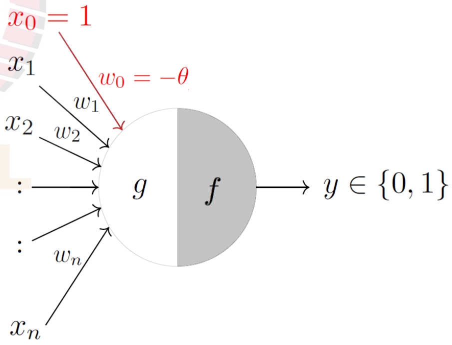
- A more accepted convention

$$g(x_1, x_2, \ldots, x_n) = g(\mathbf{x}) = \sum_{i=0}^{n} w_i * x_i$$

$$y = f(g(\mathbf{x})) = 1 \; if \; g(\mathbf{x}) \geq 0$$

$$= 0 \; if \; g(\mathbf{x}) < 0$$

$$where \; x_0 = 1 \; and \; w_0 = -\theta$$



Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random $\mathbf{x} \in P \cup N$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random $\mathbf{x} \in P \cup N$

  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random $\mathbf{x} \in P \cup N$
    if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random $\mathbf{x} \in P \cup N$

  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$

  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- Why would this work?

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- Why would this work?
- We are interested in finding the line $\sum_{i=0}^{n} w_i * x_i = 0$ or $\mathbf{w}^\intercal \mathbf{x} = 0$ which divides the input space into two halves (binary classifier)

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- Why would this work?
- We are interested in finding the line $\sum_{i=0}^{n} w_i * x_i = 0$ or $\mathbf{w}^\intercal \mathbf{x} = 0$ which divides the input space into two halves (binary classifier)
- Every point ($\mathbf{x}$) on this line satisfies the equation $\mathbf{w}^\intercal \mathbf{x} = 0$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end

```
/* algorithm converges when all the inputs
   are classified correctly                  */
```

- Why would this work?
- We are interested in finding the line $\sum_{i=0}^{n} w_i * x_i = 0$ or $\mathbf{w}^\intercal \mathbf{x} = 0$ which divides the input space into two halves (binary classifier)
- Every point $(\mathbf{x})$ on this line satisfies the equation $\mathbf{w}^\intercal \mathbf{x} = 0$
- What can you tell about the angle (α) between w and any point $(\mathbf{x})$ which lies on this line?

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random $\mathbf{x} \in P \cup N$
    if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- Why would this work?
- We are interested in finding the line $\sum_{i=0}^{n} w_i * x_i = 0$ or $\mathbf{w}^\intercal \mathbf{x} = 0$ which divides the input space into two halves (binary classifier)
- Every point $(\mathbf{x})$ on this line satisfies the equation $\mathbf{w}^\intercal \mathbf{x} = 0$
- What can you tell about the angle (α) between w and any point $(\mathbf{x})$ which lies on this line?
- The angle is 90° $(\because \cos\alpha = \frac{\mathbf{w}^\intercal \mathbf{x}}{||\mathbf{w}||||\mathbf{x}||})$
- Since the vector w is perpendicular to every point on the line it is actually perpendicular to the line itself

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random x $\in P \cup N$
  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$?

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than $90°$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than 90°
- What will be the angle between vector $\mathbf{x} \in N$ and $\mathbf{w}$?

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than 90°
- What will be the angle between vector $\mathbf{x} \in N$ and $\mathbf{w}$? Greater than 90°
- Ponder and convince yourself this is the case

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random x $\in P \cup N$

  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$

  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**

    $\mathbf{w} = \mathbf{w} - \mathbf{x}$

end

`/* algorithm converges when all the inputs are classified correctly */`

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than 90°
- What will be the angle between vector $\mathbf{x} \in N$ and $\mathbf{w}$? Greater than 90°
- Ponder and convince yourself this is the case
- For $\mathbf{x} \in P$ if $\mathbf{w}^\intercal \mathbf{x} < 0$ then it means that the angle (α) between this $\mathbf{x}$ and the current $\mathbf{w}$ is greater than 90

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random x $\in P \cup N$
  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than 90°
- What will be the angle between vector $\mathbf{x} \in N$ and $\mathbf{w}$? Greater than 90°
- Ponder and convince yourself this is the case
- For $\mathbf{x} \in P$ if $\mathbf{w}^\intercal\mathbf{x} < 0$ then it means that the angle (α) between this $\mathbf{x}$ and the current $\mathbf{w}$ is greater than 90
- But we want it to be less than 90

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What will be the angle between vector $\mathbf{x} \in P$ and $\mathbf{w}$? Less than 90°
- What will be the angle between vector $\mathbf{x} \in N$ and $\mathbf{w}$? Greater than 90°
- Ponder and convince yourself this is the case
- For $\mathbf{x} \in P$ if $\mathbf{w}^\intercal \mathbf{x} < 0$ then it means that the angle (α) between this $\mathbf{x}$ and the current $\mathbf{w}$ is greater than 90
- But we want it to be less than 90
- How is adding $\mathbf{x}$ to $\mathbf{w}$ helping us?

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random x $\in P \cup N$

  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$

  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**

    $\mathbf{w} = \mathbf{w} - \mathbf{x}$

end

```
/* algorithm converges when all the inputs
   are classified correctly                  */
```

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\cos \alpha_{new} \propto \mathbf{w_{new}}^\intercal \mathbf{x}$$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned} \cos \alpha_{new} &\propto \mathbf{w_{new}}^\intercal \mathbf{x} \\ &\propto (\mathbf{w} + \mathbf{x})^\intercal \mathbf{x} \end{aligned}$$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end

```
/* algorithm converges when all the inputs
   are classified correctly                  */
```

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned}\cos \alpha_{new} &\propto \mathbf{w_{new}}^\intercal \mathbf{x} \\ &\propto (\mathbf{w} + \mathbf{x})^\intercal \mathbf{x} \\ &\propto \mathbf{w}^\intercal \mathbf{x} + \mathbf{x}^\intercal \mathbf{x}\end{aligned}$$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
/* algorithm converges when all the inputs are classified correctly */

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$
\begin{aligned}
\cos \alpha_{new} &\propto \mathbf{w_{new}}^\intercal \mathbf{x} \\
&\propto (\mathbf{w} + \mathbf{x})^\intercal \mathbf{x} \\
&\propto \mathbf{w}^\intercal \mathbf{x} + \mathbf{x}^\intercal \mathbf{x} \\
&\propto \cos \alpha + \mathbf{x}^\intercal \mathbf{x}
\end{aligned}
$$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$

$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$

$P \leftarrow$ input with labels 1;

$N \leftarrow$ input with labels 0;

Initialize w randomly;

while *!convergence* **do**

  Pick random $\mathrm{x} \in P \cup N$

  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal\mathbf{x} < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$

  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal\mathbf{x} \geq 0$ **then**

    $\mathbf{w} = \mathbf{w} - \mathbf{x}$

end

/* algorithm converges when all the inputs are classified correctly */

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned}
\cos\alpha_{new} &\propto \mathbf{w_{new}}^\intercal\mathbf{x} \\
&\propto (\mathbf{w} + \mathbf{x})^\intercal\mathbf{x} \\
&\propto \mathbf{w}^\intercal\mathbf{x} + \mathbf{x}^\intercal\mathbf{x} \\
&\propto \cos\alpha + \mathbf{x}^\intercal\mathbf{x} \\
\cos\alpha_{new} &> \cos\alpha
\end{aligned}$$

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$
\begin{aligned}
\cos \alpha_{new} &\propto \mathbf{w_{new}}^\intercal \mathbf{x} \\
&\propto (\mathbf{w} + \mathbf{x})^\intercal \mathbf{x} \\
&\propto \mathbf{w}^\intercal \mathbf{x} + \mathbf{x}^\intercal \mathbf{x} \\
&\propto \cos \alpha + \mathbf{x}^\intercal \mathbf{x} \\
\cos \alpha_{new} &> \cos \alpha
\end{aligned}
$$

- Thus α_{new} will be less than α and this is exactly what we want

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
    Pick random x $\in P \cup N$
    if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal \mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$
    if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal \mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned}\cos \alpha_{new} &\propto \mathbf{w_{new}}^\intercal \mathbf{x} \\ &\propto (\mathbf{w} + \mathbf{x})^\intercal \mathbf{x} \\ &\propto \mathbf{w}^\intercal \mathbf{x} + \mathbf{x}^\intercal \mathbf{x} \\ &\propto \cos \alpha + \mathbf{x}^\intercal \mathbf{x} \\ \cos \alpha_{new} &> \cos \alpha\end{aligned}$$

- Thus α_{new} will be less than α and this is exactly what we want
- We can work out the math for the other case, when $\mathbf{x} \in N$ and $\mathbf{w}^\intercal \mathbf{x} \geq 0$ quite similarly

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning

$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]$
$\mathbf{x} = [1, x_1, x_2, \ldots, x_n]$
$P \leftarrow$ input with labels 1;
$N \leftarrow$ input with labels 0;
Initialize w randomly;
while *!convergence* **do**
  Pick random x $\in P \cup N$
  if $\mathbf{x} \in P$ *and* $\mathbf{w}^\intercal\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  if $\mathbf{x} \in N$ *and* $\mathbf{w}^\intercal\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
end
`/* algorithm converges when all the inputs are classified correctly */`

- What happens to the new angle (α_{new}) when $\mathbf{w_{new}} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned}\cos\alpha_{new} &\propto \mathbf{w_{new}}^\intercal\mathbf{x} \\ &\propto (\mathbf{w}+\mathbf{x})^\intercal\mathbf{x} \\ &\propto \mathbf{w}^\intercal\mathbf{x} + \mathbf{x}^\intercal\mathbf{x} \\ &\propto \cos\alpha + \mathbf{x}^\intercal\mathbf{x} \\ \cos\alpha_{new} &> \cos\alpha\end{aligned}$$

- Thus α_{new} will be less than α and this is exactly what we want
- We can work out the math for the other case, when $\mathbf{x} \in N$ and $\mathbf{w}^\intercal\mathbf{x} \geq 0$ quite similarly
- For a formal convergence proof, please see this link

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

- The fourth condition contradicts conditions 2 and 3

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

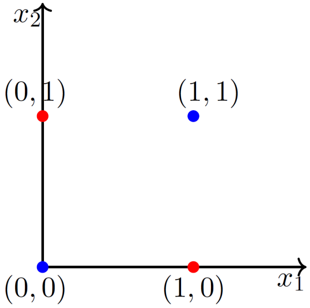
- The fourth condition contradicts conditions 2 and 3
- No solution possible satisfying this set of inequalities

The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

- The fourth condition contradicts conditions 2 and 3
- No solution possible satisfying this set of inequalities

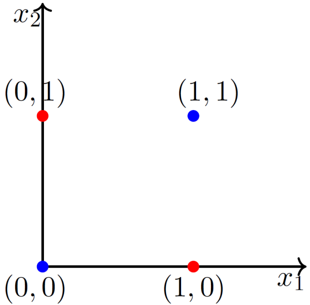


The XOR Conundrum

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

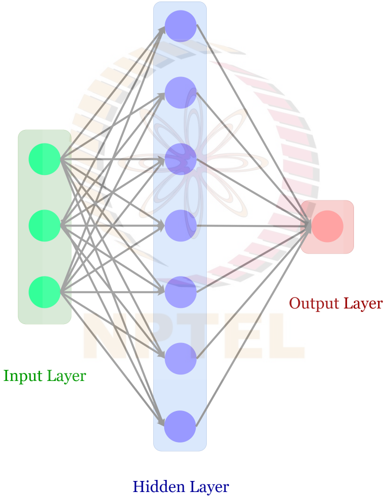
$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

- The fourth condition contradicts conditions 2 and 3
- No solution possible satisfying this set of inequalities

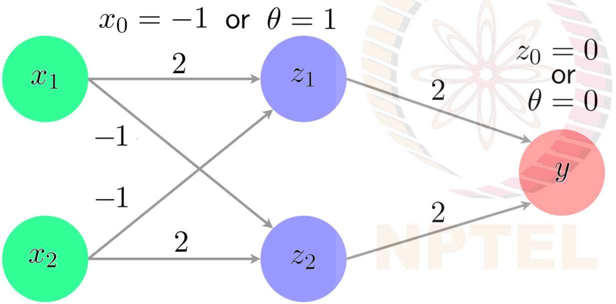


- Indeed you can see that it is impossible to draw a line which separates the red points from the blue points

Multi-Layer Perceptrons



Solving XOR with Multi-Layer Perceptrons



(x_1,x_2)	(z_1,z_2)	y
(0,0)	(0,0)	0
(0,1)	(0,0)	1
(1,0)	(1,0)	1
(1,1)	(0,0)	0

Multi-Layer Perceptrons

Theorem: Any boolean function of n inputs can be represented exactly by a network of perceptrons containing 1 hidden layer with 2^n perceptrons and one output layer containing 1 perceptron

Proof (Informal): How?

Multi-Layer Perceptrons

Theorem: Any boolean function of n inputs can be represented exactly by a network of perceptrons containing 1 hidden layer with 2^n perceptrons and one output layer containing 1 perceptron

Proof (Informal): How? Each of the 2^n hidden layer perceptrons can model (or can be fired by) one combination of n inputs.

Note: A network of $2^n + 1$ perceptrons is not necessary but sufficient

But why does this matter?

Multi-Layer Perceptrons

Theorem: Any boolean function of n inputs can be represented exactly by a network of perceptrons containing 1 hidden layer with 2^n perceptrons and one output layer containing 1 perceptron

Proof (Informal): How? Each of the 2^n hidden layer perceptrons can model (or can be fired by) one combination of n inputs.

Note: A network of $2^n + 1$ perceptrons is not necessary but sufficient

But why does this matter? As n increases, the number of perceptrons in the hidden layers increases exponentially. We'd ideally want this to be as few as possible.

Going Beyond Binary Inputs and Outputs

Question

- What about arbitrary functions of the form $y = f(x)$ where $x \in \mathbf{R}^n$ (instead of $\{0, 1\}^n$) and $y \in \mathbf{R}$ (instead of $\{0, 1\}$)?
- Can we use the same perceptron model to represent such functions?

Need for Activation Functions

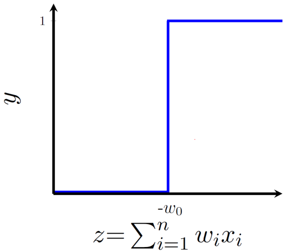
- A perceptron only fires if weighted sum of its inputs is greater than threshold $-w_0$

Need for Activation Functions

- A perceptron only fires if weighted sum of its inputs is greater than threshold $-w_0$
- Thresholding logic could be harsh at times
- E.g., when $-w_0 = 0.5$, though output values 0.49 and 0.51 are very close to each other, the perceptron would assign different labels to them.

Need for Activation Functions

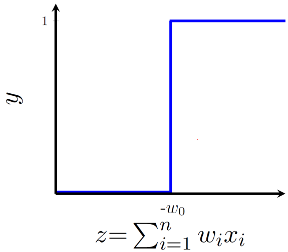
- A perceptron only fires if weighted sum of its inputs is greater than threshold $-w_0$
- Thresholding logic could be harsh at times
- E.g., when $-w_0 = 0.5$, though output values 0.49 and 0.51 are very close to each other, the perceptron would assign different labels to them.
- This behavior is not a characteristic of the problem, the weights or the threshold; it is a characteristic of the perceptron function itself which behaves like a step function



Need for Activation Functions

- A perceptron only fires if weighted sum of its inputs is greater than threshold $-w_0$
- Thresholding logic could be harsh at times
- E.g., when $-w_0 = 0.5$, though output values 0.49 and 0.51 are very close to each other, the perceptron would assign different labels to them.

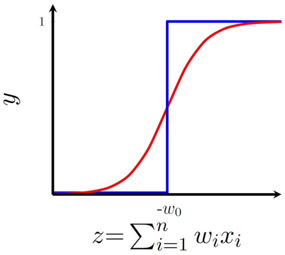
- This behavior is not a characteristic of the problem, the weights or the threshold; it is a characteristic of the perceptron function itself which behaves like a step function
- There will always be a sudden change in decision (from 0 to 1) when $\sum_{i=1}^{n} w_i x_i$ crosses the threshold ($-w_0$)



Need for Activation Functions

- A perceptron only fires if weighted sum of its inputs is greater than threshold $-w_0$
- Thresholding logic could be harsh at times
- E.g., when $-w_0 = 0.5$, though output values 0.49 and 0.51 are very close to each other, the perceptron would assign different labels to them.

- This behavior is not a characteristic of the problem, the weights or the threshold; it is a characteristic of the perceptron function itself which behaves like a step function
- There will always be a sudden change in decision (from 0 to 1) when $\sum_{i=1}^{n} w_i x_i$ crosses the threshold ($-w_0$)
- For most real-world applications, we'd expect a smoother decision function which gradually changes from 0 to 1



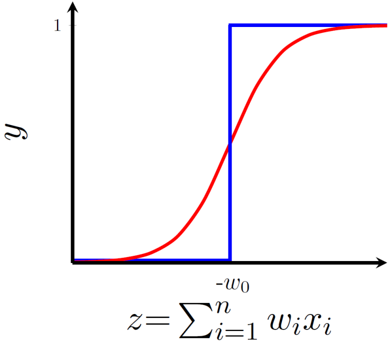
Sigmoid Neurons

- We could use any logistic function to obtain a smoother output function than a step function

Sigmoid Neurons

- We could use any logistic function to obtain a smoother output function than a step function
- One form is the sigmoid function:

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=0}^{n} w_i x_i)}}$$

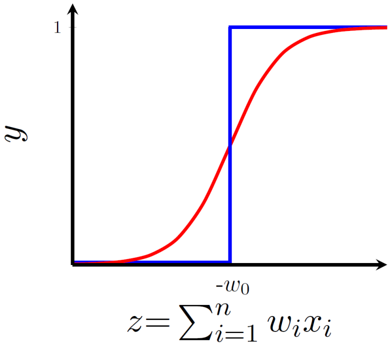


Sigmoid Neurons

- We could use any logistic function to obtain a smoother output function than a step function
- One form is the sigmoid function:

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=0}^{n} w_i x_i)}}$$

- No longer a sharp transition at the threshold $-w_0$

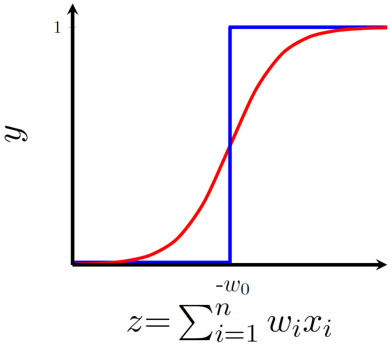


Sigmoid Neurons

- We could use any logistic function to obtain a smoother output function than a step function
- One form is the sigmoid function:

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=0}^{n} w_i x_i)}}$$

- No longer a sharp transition at the threshold $-w_0$
- Also, output is no longer binary but a real value between 0 and 1 which can be interpreted as a probability

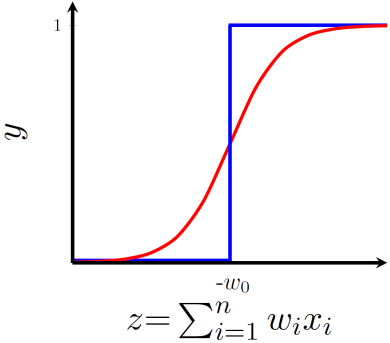


Sigmoid Neurons

- We could use any logistic function to obtain a smoother output function than a step function
- One form is the sigmoid function:

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=0}^{n} w_i x_i)}}$$

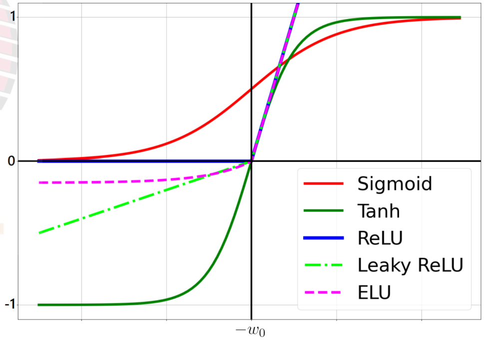
- No longer a sharp transition at the threshold $-w_0$
- Also, output is no longer binary but a real value between 0 and 1 which can be interpreted as a probability
- Unlike the step function, this one is smooth, continuous at $-w_0$ and most importantly **differentiable**



Other Popular Activation Functions

Considering $z = \sum_{i=0}^{n} w_i x_i$

- **Sigmoid:** $y = \frac{1}{1+e^{-z}}$
- **Tanh:** $y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- **Rectified Linear Unit (ReLU):** $y = \max(0, z)$
- **Leaky ReLU:** $y = \max(\alpha z, z), \alpha \in (0, 1)$
- **Exponential Linear Unit (ELU):** $y = \max(\alpha(e^z - 1), z)$, where $\alpha > 0$



Representation Power of MLPs

Theorem: A multilayer network of sigmoid neurons with a single hidden layer can be used to approximate any continuous function to any desired precision. (Also known as **Universal Approximation Theorem**)

Proof: Cybenko, 1989 [2] and Hornik et al, 1989[3]

Note: Also, refer to Chapter 4 of Michael Nielsen's online book for visual explanation of Universal Approximation Theorem

[2] Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems, Vol 2, pp 303-314, 1989

[3] Hornik et al, Multilayer feedforward networks are universal approximators, Neural Networks Vol 2:5, pp 359-366, 1989

Homework

Homework

- Solve XOR using an MLP with 4 hidden units.

Readings

- Please refer to Mitesh Khapra's original lecture slides (and videos) for more detailed explanation of some of these topics, available at CS7015: Deep Learning.
- Other good resources:
  - Deep Learning Book: Chapter 6 for Multilayer Perceptrons
  - Stanford CS231n Notes
  - Stanford UFLDL tutorial on Multilayer Neural Networks
  - Neural Networks: A Systematic Introduction by Raúl Rojas

References

George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314.

Tianping Chen, Hong Chen, and Ruey-wen Liu. "Approximation Capability in by Multilayer Feedforward Networks and Related Problems". In: *Neural Networks, IEEE Transactions on* 6 (Feb. 1995), pp. 25 –30.

Mikel L. Forcada. *Neural Networks: Automata and Formal Models of Computation.* *https://www.dlsi.ua.es//~mlf/nnafmc/pbook.pdf*. 2002.

Michael Collins. *Convergence Proof for the Perceptron Algorithm.* *http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf*. 2012.