CS-568 Deep Learning

Nazar Khan

PUCIT

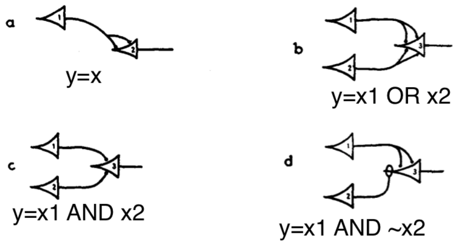
History of Neural Computation

Mimicing the neuron

- Having established the neuron as the central element of animal systems, early research mimiced its function.
- Artificial neuron models were extremely simplified abstractions of the real neuron.
- To this day, they are extremely simplified.

> *All models are wrong, but some are useful. -George Box*

McChulloch & Pitts Model[1]



- For Boolean inputs and outputs.
- Assumed neurons to either fire or stay silent.
- This allowed them to model neurons using propositional logic.
- No mechanism for learning.

[1] McCulloch and Pitts, 'A logical calculus of the ideas immanent in nervous activity'.

Hebbian Learning[2]

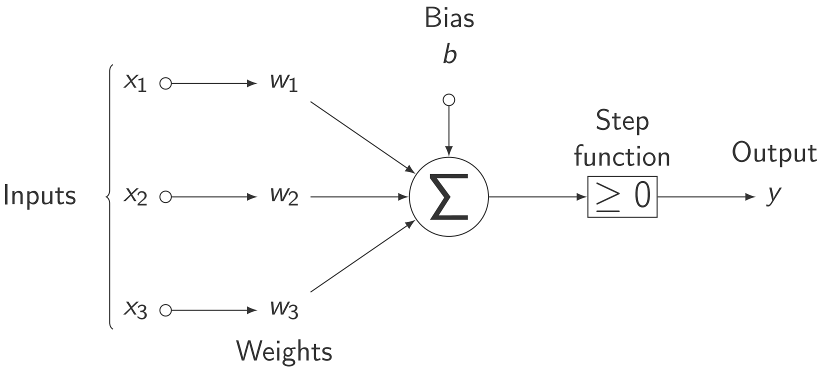
- Donald Hebb exploited the observation that metabolic changes take place between neuronal connections when one neuron frequently causes another neuron to fire.
- If x_i triggers y, then increase the weight w_i of their connection as

$$w_i \leftarrow w_i + \eta x_i y$$

- Association determines strength of connection.
- Problems: Unbounded learning.

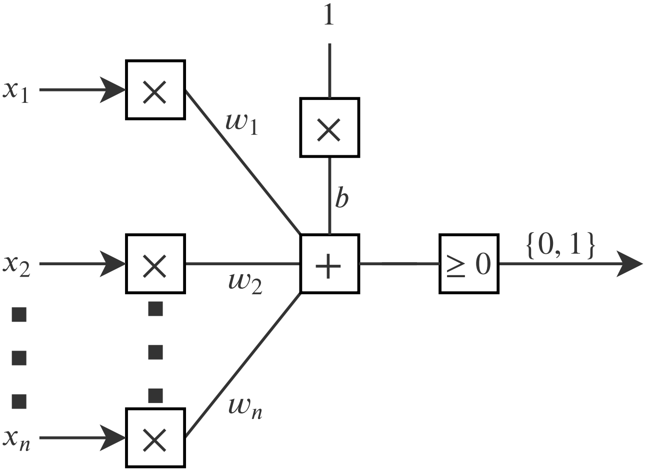
[2] Hebb, 'The organization of behavior; a neuropsychological theory.'

Perceptron[3]



[3]Rosenblatt, 'The perceptron: a probabilistic model for information storage and organization in the brain.'

Perceptron



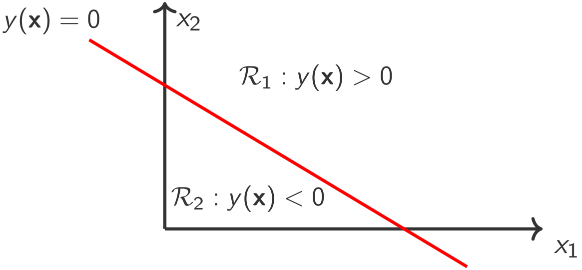
Perceptron

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \sum w_i x_i + b \geq 0 \\ 0 & \sum w_i x_i + b < 0 \end{cases}$$

- For real-valued inputs and weights.
- Also models Boolean logic.
  - AND gate: set all w_is to 1 and $b = -n$.
  - OR gate: set all w_is to 1 and $b = -1$.
  - NOT gate: for just a single input x_1, set w_1 to -1 and $b = 0$.
  - *Can't model XOR gate.*
  - Since {AND, OR, NOT} form a basis for all logic gates, *combinations of perceptrons* can model *any* logic function.

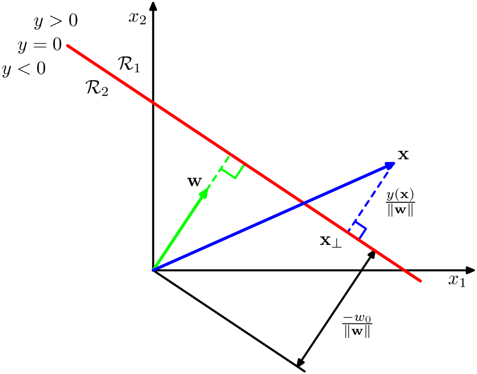
Detour – Linear Classifier

- Consider the function $y(\mathbf{x}) = \sum w_i x_i + w_0 = \mathbf{w}^T\mathbf{x} + w_0$.
- Thresholding $y(\mathbf{x})$ against 0 divides input space into two regions.
  - $\mathcal{R}_1$: where $y(\mathbf{x}) > 0$, and
  - $\mathcal{R}_2$: where $y(\mathbf{x}) < 0$.
- The line $y(\mathbf{x}) = 0$ is called the *linear decision boundary*.



Detour – Linear Classifier

- Weight vector $\mathbf{w}$ is always orthogonal to the decision boundary.
  - Proof: For *any* two points $\mathbf{x}_A$ and $\mathbf{x}_B$ on the boundary, $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0 \Rightarrow \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$. Since vector $\mathbf{x}_A - \mathbf{x}_B$ is along the boundary, $\mathbf{w}$ must be orthogonal to it.



Detour – Linear Classifier

- Normal distance of any point $\mathbf{x}$ from decision boundary can be computed as $d = \frac{y(\mathbf{x})}{||\mathbf{w}||}$.
  - Proof:

$$\mathbf{x} = \mathbf{x}_\perp + d\frac{\mathbf{w}}{||\mathbf{w}||}$$

$$\Rightarrow \underbrace{\mathbf{w}^T\mathbf{x} + w_0}_{y(\mathbf{x})} = \underbrace{\mathbf{w}^T\mathbf{x}_\perp + w_0}_{y(\mathbf{x}_\perp)=0} + d\underbrace{\mathbf{w}^T\frac{\mathbf{w}}{||\mathbf{w}||}}_{||\mathbf{w}||}$$

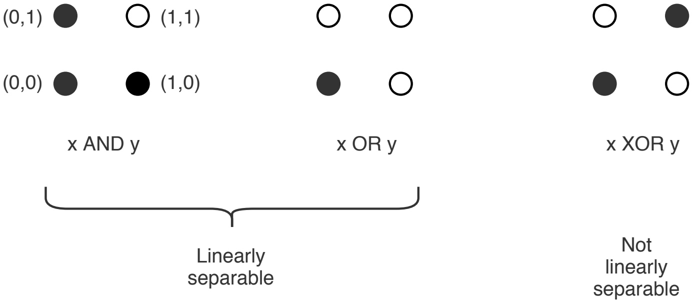
$$\Rightarrow d = \frac{y(\mathbf{x})}{||\mathbf{w}||}$$

- Note that this distance is a signed distance.
- Normal distance to boundary from origin $(\mathbf{x} = \mathbf{0})$ is $\frac{w_0}{||\mathbf{w}||}$.

$$\mathbf{w}^T\mathbf{w} = w_1^2 + w_2^2 = ||\mathbf{w}||^2$$

Perceptron

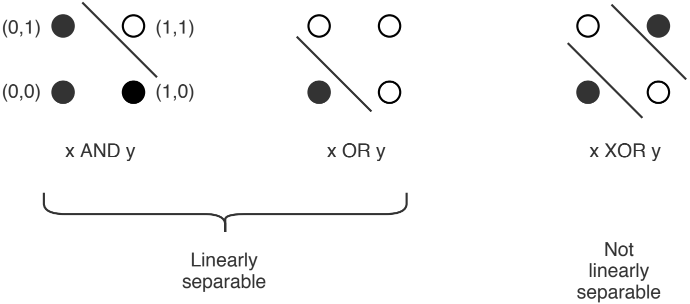
- A perceptron is actually a linear classifier.
- It's weights w_i and b represent a line that divides input space into 2 regions.



A perceptron cannot model the XOR problem because *XOR is not a linear classification problem*. No single line can separate the 0s (black) from the 1s (white).

Perceptron

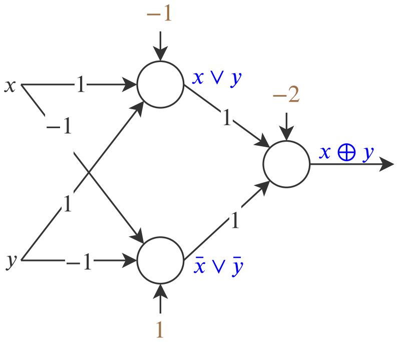
- A perceptron is actually a linear classifier.
- It's weights w_i and b represent a line that divides input space into 2 regions.



A perceptron cannot model the XOR problem because *XOR is not a linear classification problem*. No single line can separate the 0s (black) from the 1s (white). *But combination of two lines can*.

Multilayer Perceptrons

- Combination of perceptrons can solve the XOR problem.
- Two lines can divide the XOR space into 0-region and 1-region.



3 perceptrons can model XOR. Two perceptrons for the two lines and a final perceptron to combine them into a final decision.

> A *network* of 3 neurons is more powerful than 1 neuron.
> Just like the brain!

Perceptron

- *Importantly*, the weights of a perceptron can be learned.
- Perceptron learning rule:

$$w_i \leftarrow w_i + \eta(y - t)x_i$$

if output y and desired target t are different.