Artificial Neural Networks (ANNs)

Biologically inspired computational model:

(1) **Simple** computational units (neurons).
(2) **Highly interconnected** - connectionist view
(3) Vast **parallel** computation, consider:
   • Human brain has $\sim 10^{11}$ neurons
   • Slow computational units, switching time $\sim 10^{-3}$ sec
     (compared to the computer $>10^{-10}$ sec)
   • Yet, you can recognize a face in $\sim 10^{-1}$ sec
   • This implies only about 100 sequential, computational neuron steps - this seems too low for something as complicated as recognizing a face
   ⇒ **Parallel processing**

ANNs are naturally parallel - each neuron is a self-contained computational unit that depends only on its inputs.
The Perceptron

- A simple, single layered neural “network” - only has a single neuron.
- However, even this simple neural network is already powerful enough to perform classification tasks.
The Architecture

Transfer Function:
\[ \text{sgn}(k) = \begin{cases} +1 & \text{if } k \geq 0 \\ -1 & \text{otherwise} \end{cases} \]

Perceptron Computation:
\[ y = \text{sgn} \left( b + \sum_{i=1}^{m} w_i x_i \right) \]

Note: \( y \in \{ +1, -1 \} \)

Binary Classification
A perceptron computes the value,

\[ y = \text{sgn}\left(b + \sum_{i=1}^{m} w_i x_i\right) \]

Ignoring the activation function \( \text{sgn} \) and setting \( m = 1 \), we obtain,

\[ y' = b + w_1 x_1 \]

But this is the equation of a line with slope \( w \) and offset \( b \).

Observation: For the general case the perceptron computes a hyperplane in order to accomplish its classification task,

\[ y' = b + \sum_{i=1}^{m} w_i x_i = b + \vec{w} \cdot \vec{x} \]
In order for the hyperplane to become a classifier we need to find $b$ and $w$ => learning!
Learning Algorithm

Let \( D = \{ (\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n) \} \subset H \times \{-1, +1\} \)

\[
\begin{align*}
\vec{w} &\leftarrow 0 \\
b &\leftarrow 0 \\
R &\leftarrow \max_{1 \leq i \leq n} | \vec{x}_i | \\
\eta &\leftarrow 0 < \eta < 1 \\
\text{repeat} \\
\quad \text{for } i = 1 \text{ to } n \\
\quad \quad \text{if } \text{sign}(\vec{w} \cdot \vec{x}_i + b) \neq y_i \text{ then} \\
\quad \quad \quad \vec{w} &\leftarrow \vec{w} + \eta y_i \vec{x}_i \\
\quad \quad \quad b &\leftarrow b + \eta y_i R^2 \\
\quad \quad \text{end if} \\
\quad \text{end for} \\
\text{until no mistakes made in the for-loop} \\
\text{return } (\vec{w}, b)
\end{align*}
\]

Note: learning is very different here compared to decision trees...here we have many passes over the data until the perceptron converges on a solution.
Demo


R perceptron demo
Observations

- The learned information is represented as weights and the bias ⇒ sub-symbolic learning
- In order to apply this learned information we need a neural network structure
- The learned information is not directly accessible to us ⇒ non-transparent model