Artificial Neural Network Introduction

Introduction :

- **ANNs**: Artificial neural networks are at the core of state-of-the-art approaches to a variety of visual recognition tasks. It's a pattern recognizer.
  The concept of ANNs is based on four psychological keys:
  1. A dense and complicated network of neurons.
  2. The propagation of "all-or-none" electrochemical signals between cells.
  3. Neurons are classifiers, which specialize with training.
  4. the brain learns patterns by adjusting the efficacy of communication between pairs of neurons.

- Guidelines for building an artificial pattern recognizer:
  - A network of nearly-identical units.
  - Communicate between units in binary.
  - The recognizer should be a trainable classifier.
  - Adjusting weights on the connections of the network for the learning task.

- A variety of neural network algorithms were developed but they faced two major critical bottlenecks.
  1. Computational : for network with hundreds of units small tasks
  2. The lack of training data.

Mathematical Implementation:

- Decomposition of a complicated function to multiple simple functions: the functions will be our neurons. The unitary function is a neuron.
- The function of Neurons : Inputs from other neurons are summed, and this sum is passed through a threshold function, denoted by σ. The output of this function is sent to other neurons. σ is the inner function of neurons.
- The simplest inner function : The threshold
  $\sigma(x)=\{0 \text{ if } x<\text{threshold}, 1 \text{ if } x>=\text{threshold}$
- Other inner functions : the logistic sigmoid function, the tanh function, The hinge loss function.
- Flexibility : The input will be au weighted sum rather than a simple sum,The weights represent "connection strengths" between pairs of neurons.

  Case of one input :
  $O=\sigma(\phi w+b)$, ϕ *i*s the actual input, w *is* a weight term, b is a bias
  Case of multiple inputs :

$O=\sigma(\phi_1 w_1+\ldots+\phi_n w_n+b)$.

$\longrightarrow$ The inputs are approximately stagnated and the parameters $w_1,\ldots,w_n$ and b can be adjusted to guide the output O to any value within $(-1,1)$.

- Organization : Neural network organized with layers of nodes,

  Every pair of neighbouring layers is fully connected.
  Every node has a bias (b), and a set of incoming weighted edges ($w_1,w_2,\ldots,w_n$). Using these parameters, each node processes the previous layer outputs (denoted $\phi_1,\phi_2,\ldots,\phi_n$), and sends its own output O along weighted edges to the next layer.

Notation : $O_{lq}=\sigma(O_{l-1q}w_{l1q}+\ldots+O_{l-1n}w_{lnq}+b_{lq})$, l is the the order of the current layer, q is the order of the current node, w_{pq} means that this weight connects the pth node in the previous layer to node q on the current layer.

Training Part :

- Since we have a target value for every final layer node, we can define the error of our network as <u>the mean squared error of those nodes.</u>

$$E=1/2\sum_{k\in K}(O_k-t_k)^2$$

- K is the set of nodes in the final layer of the network, O_k is the output of a final layer node k, and t_k is the target value of that node.
- We are searching to minimize E in function of the parameters of f. We can image E as au surface in a high dimensional space and f as a ball on this surface.

$\longrightarrow$ We want to find the set of parameters that put f on the surface's lowest point.

- Final layers nodes :

$$\partial E/\partial w_{pk}=d_k O_p$$

$$d_k=(O_k-t_k)O_k(1-O_k).$$

k is the current node and p is the previous node.

$$\partial E/\partial b_k=d_k.$$

Interpretation:
$\partial E/\partial w_{pk}$ = (derivative of $\sigma(x_k)$)$*$(error at w_{pk}'s destination)$*$(data passed through w_{pk}).

- Hidden layer nodes :

  For any node not in the final layer, we do not know from the outset what that node's target should be. Because the targets are unknown, or "hidden," we call the nodes on these layers "hidden nodes."

$$\partial E/\partial w_{pq}=d_q O_p.$$

$d_q = O_q(1 - O_q)\sum_{k \in K} d_k w_{qk}.$

$\partial E / \partial b_q = d_q.$

Backpropagation :

- The algorithm operates on the fact that the derivatives on one layer contain partial solutions to the derivatives on the layer below.
- The algorithm's name comes from how it propagates partial solutions backward through the network (from the final layer to the first layer).