IIT ROORKEE

NPTEL ONLINE CERTIFICATION COURSE

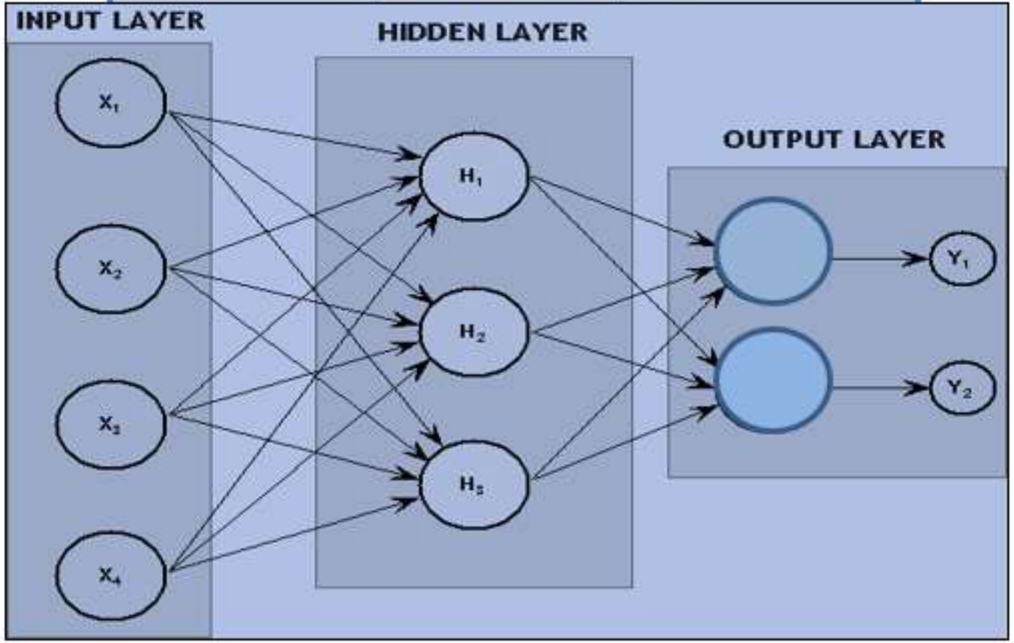
Artificial Neural Network

N.SUKAVANAM

DEPARMENT OF MATHEMATICS

Artificial Neural Network

A 2-layer Feed-Forward Network with 4 Inputs and 2 Outputs



Artificial Neural Network

Consider a neural network with $n - input\{x_1, x_2 \dots \dots \dots x_n\}$,with l neurons in the hidden layer and $m - output\ \{y_1, y_2 \dots \dots \dots y_m\}$ layer.

Let u_{ij} be the weight connecting i^{th} input and j^{th}hidden neuron and
v_{jk}be the weight connecting j^{th}hidden neuron and the k^{th}output neuron.

The value at the j^{th}hidden layer is given by

$$h_j = \sigma\left[\sum_{i=1}^{n} u_{ij}x_i\right] \dots \dots \dots \dots \dots (1)$$

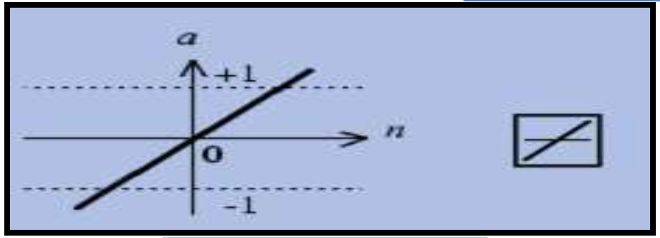
where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

is a sigmoid function which is used here as the transfer function.

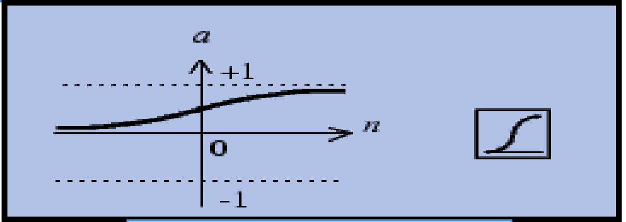
Artificial Neural Network

The Transfer Functions



Linear transfer function

$$Y^{linear} = X$$



log-sigmoid transfer function

$$Y^{sigmoid} = \frac{1}{1+e^{-X}}$$

Some other Kinds of sigmoid used in perceptron's

Rational $$f(s) = \frac{s}{|s| + \alpha}$$

Hyperbolic tangent $$f(s) = \tanh\frac{s}{\alpha} = \frac{e^{\frac{s}{\alpha}} - e^{-\frac{s}{\alpha}}}{e^{\frac{s}{\alpha}} + e^{-\frac{s}{\alpha}}}$$

Artificial Neural Network

The value at the output $y_k : k = 1,2, \ldots \ldots \ldots m$ is given by:

$$y_k = \sum_{j=1}^{l} v_{jk} h_j$$

$$y_k = \sum_{j=1}^{l} v_{jk}\, \sigma \left[\sum_{i=1}^{n} u_{ij} x_i \right] \ldots \ldots \ldots \ldots \ldots (2)$$

Denoting $X = [x_1\ x_2 \ldots \ldots \ldots x_n]'$ and Y $= [y_1\ y_2 \ldots \ldots \ldots y_m]'$

$U_{n\times l} = \{u_{ij}\}_{\substack{i=1,n \\ j=1,l}}$ and $V_{l\times m} = \{v_{jk}\}_{\substack{j=1,l \\ k=1,m}}$

Equation (2) can be written as:

$$Y = V'\sigma(U'X) \ldots \ldots \ldots \ldots \ldots (3)$$

Function Approximation using Neural Network

Let $f: R^n \to R^m$ be a continuous function defined on a closed and bounded set $\Omega \subset R^n$, then there exists weight matrices *U* and *V* such that $f(X)$ *is approximated* using neural network as in equation (3), i.e. ,for any given $\epsilon > 0$,

thereexists matrices *U* and *V* such that:

$$||f(X) - Y|| < \epsilon$$

where Y is given by Equation (3).

Weight Updating Algorithm

Let f be a given function and

$$E(V,U) = \|f(X) - Y\|^2$$

be the error in the approximation.

Hence E is a function of U and V: $i = 1,2,\ldots\ldots n,\ \ j = 1,2,\ldots\ldots l,\ \ k = 1,2,\ldots\ldots m$

Gradient Descent Iteration and Algorithm

$$U^{\gamma+1} = U^{\gamma} - \alpha.(DU)^{\gamma} \quad : \gamma = 0,1,2, \dots \dots \dots (4)$$

where

$$(DU)^{\gamma} = \left\{\frac{\partial E}{\partial u_{ij}}\right\}^{\gamma}_{\substack{i=1,n \\ j=1,l}}$$

and

$$V^{\gamma+1} = V^{\gamma} - \alpha.(DV)^{\gamma} \quad : \gamma = 0,1,2, \dots \dots$$

where

$$(DV)^{\gamma} = \left\{\frac{\partial E}{\partial v_{jk}}\right\}^{\gamma}_{\substack{j=1,l \\ k=1,m}}$$

Initialize the weights V and W as zero matrices. α is called the learning rate which can be chosen to be a small positive real number less than 1(e.g. say $\alpha = 0.01$)

Example

We have one input x, two neurons in the hidden layer, one output y.
Let

$$y = v_1\sigma(u_1x) + v_2\sigma(u_2x)$$

be the neural network.
Let a be the required constant output.
To approximate a using neural network y, define:

$$E = |a - y|^2$$

be the error.

$$E = [a - v_1\sigma(u_1x) - v_2\sigma(u_2x)]^2$$

Here the sigmoid function σ is:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Example

$$E = \left[a - \frac{v_1}{1 + e^{-u_1 x}} - \frac{v_2}{1 + e^{-u_2 x}}\right]^2 = N^2$$

$$\frac{\partial E}{\partial V_1} = 2N\left(-\frac{1}{1 + e^{-u_1 x}}\right)$$

$$\frac{\partial E}{\partial w_1} = 2N\left(\frac{v_1}{(1 + e^{-u_1 x})^2}\right)(-x)e^{-u_1 x}$$

Example

One input x, Two hidden neurons, one output y.

Let $x = 2, y = 5$

Then

$$y = v_1\sigma(u_1x) + v_2\sigma(u_2x)$$

$$5 = \frac{v_1}{1+e^{-2u_1}} - \frac{v_2}{1+e^{-2u_2}}$$

$$\begin{pmatrix} v_1 \\ v_2 \\ u_1 \\ u_2 \end{pmatrix}^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^0 - (0.1).2(5-0)\begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Example

$$\begin{pmatrix} v_1 \\ v_2 \\ u_1 \\ u_2 \end{pmatrix}^1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} v_1 \\ v_2 \\ u_1 \\ u_2 \end{pmatrix}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} - (0.1).2.(5 - \frac{1}{2} - \frac{1}{2}) \begin{pmatrix} -1 \\ -1 \\ -2 \\ -2 \end{pmatrix}$$

$$(w)^2 = \begin{pmatrix} v_1 \\ v_2 \\ u_1 \\ u_2 \end{pmatrix}^2 = \begin{pmatrix} 1.8 \\ 1.8 \\ 1.6 \\ 1.6 \end{pmatrix} \Rightarrow \frac{1.8}{1 + e^{-(1.6).2}} + \frac{1.8}{1 + e^{-3.2}} \approx 3$$

Proceeding as above, we can see that the weights are converging.

Example

Let $f(x) = x^2 : x \in [0,1]$ be a given (one variable) function. To approximate it using Neural Network;
Consider a partition $[0.1, 0.2, \ldots \ldots \ldots 0.9, 1]$
and the corresponding values $[0.01, 0.04, \ldots \ldots \ldots 0.8, 1]$
Let $x_k = (0.1)k; k = 1,2, \ldots \ldots \ldots 10$

$$a_k = (0.1)^2 k^2; k = 1,2, \ldots \ldots \ldots 10$$

Consider a hidden layer with 5 neurons.
Let

$$y(x) = \sum_{j=1}^{5} v_j \, \sigma\left(\sum_{i=1}^{5} u_i . x\right)$$

be the Neural Network as a function of x.
Let

$$y_k = \sum_{j=1}^{5} v_j \, \sigma\left(\sum_{i=1}^{5} u_i . x_k\right)$$

Example

Define

$$E = \sum_{k=1}^{10} (a_k - y_k)^2$$

Then E is a function of $u_i; i = 1,2 \ldots \ldots 5 \; and \; v_j; j = 1,2, \ldots \ldots 5$

By minimizing $E \; using \; the \; gradient \; method$, we can get $u_i \; and \; v_j$ and the approximation of $\mathrm{f}(x)$ as a Neural Network.

Solving Inverse Kinematics using Neural Network

Consider the kinematics of a 2-link manipulator as

$$x_1 = l_1 \cos\theta_1 + l_2\cos(\theta_1+\theta_2)$$
$$x_2 = l_1 \sin\theta_1 + l_2\sin(\theta_1+\theta_2)$$

Let (x_1, x_2) be the input and

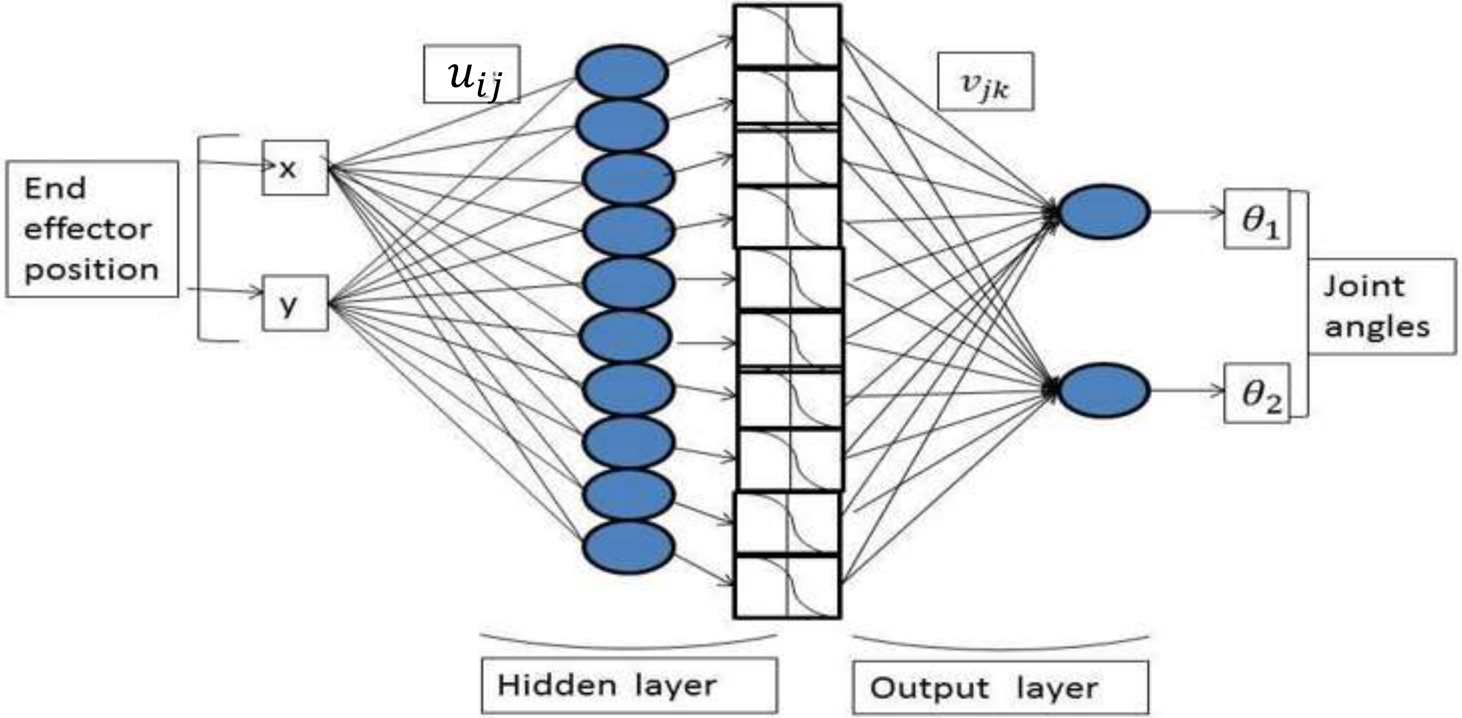
$$\theta_k^N = \sum_{j=1}^{l} v_{jk}\,\sigma\left(\sum_{i=1}^{2} u_{ij}x_i\right) k\,;k=1,2$$

be the Neural Network for θ_k.
Let

$$x_1^N = l_1\cos\theta_1^N + l_2\cos(\theta_1^N+\theta_2^N)$$
$$x_2^N = l_1\sin\theta_1^N + l_2\sin(\theta_1^N+\theta_2^N)$$

Be the Neural Network for x_1 and x_2
To find the weights v_{jk} and u_{ij}, we define the error E which is minimized using the learning algorithm as given by steepest descent iterative formula (4).
Define the error E as

$$E = \sum_{i=1}^{2}\left(x_i - x_i^N\right)^2$$



End effector position
x
y
u_{ij}
v_{jk}
θ_1
θ_2
Joint angles
Hidden layer
Output layer

Neural Network Trajectory Generation

- For 4 constrains:

$$x_A(t) = \left(\frac{t - t_0}{t_f - t_0}\right)^2 N_1(t, U_1, V_1) + \left(\frac{t_f - t_0}{t_f - t_0}\right)^2 N_2(t, U_2, V_2)$$

- For 6constrains:

$$x_A(t) = \left(\left(\frac{t_2 - t}{t_2}\right) \left(\left(\frac{t - t_0}{t_f - t_0}\right)^2 N_1(t, U_1, V_1) + \left(\frac{t_f - t}{t_f - t_0}\right)^2 N_2(t, U_2, V_2) \right) + \left(\frac{(t_f - t)t}{t_2(t_f - t_2)}\right)^2 N_3(t, U_3, V_3) \right)$$

Thanks!