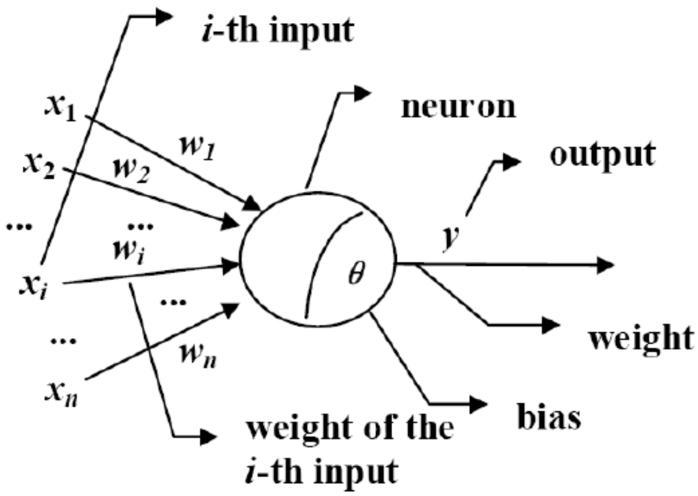
Chapter 9 Neural Networks

The M-P Neuron Model

- Input: $x_i (1 \le i \le n)$
- Weight: $w_i (1 \le i \le n)$
- Bias: θ
- Activation function: $f(\cdot)$
- Output: y

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i - \theta\right)$$



Basic Steps

1. Input to the hidden unit

$$net_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} = \mathbf{w}_j^t \mathbf{x}$$

2. Activation of the hidden unit

$$y_j = f(net_j)$$

3. Activation fuction

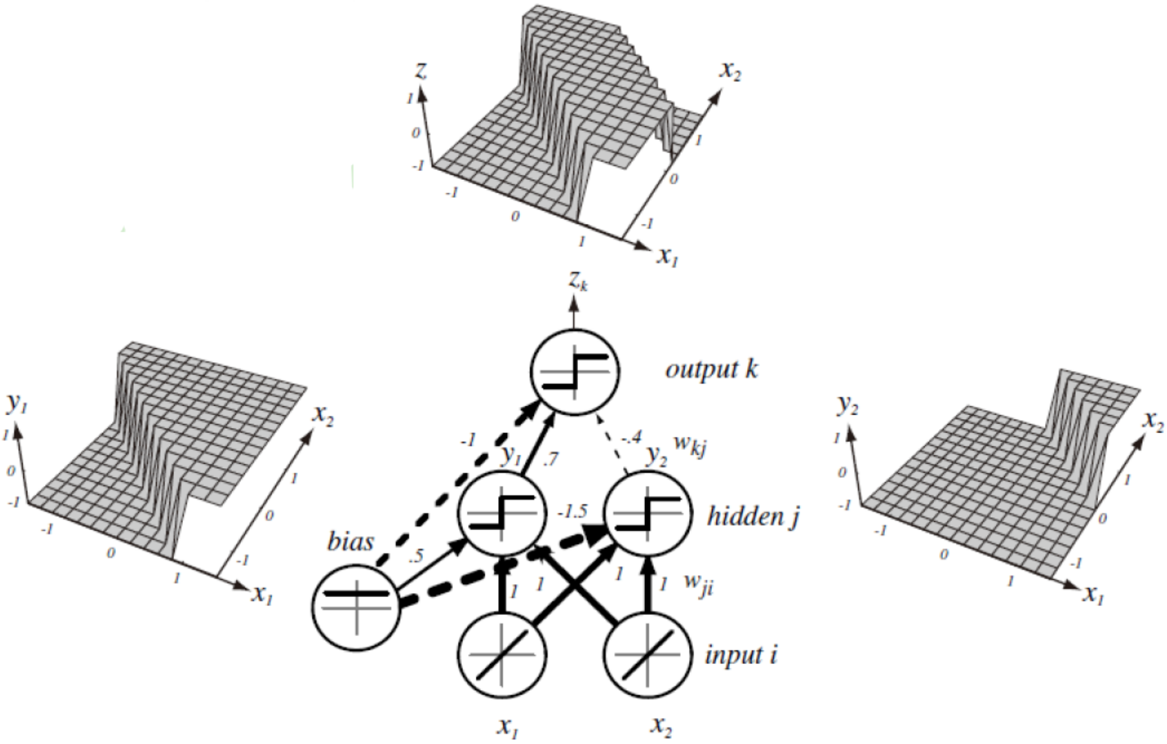
$$f(net) = \mathrm{Sgn}(net) = \begin{cases} 1 & if\ net \ge 0 \\ -1 & if\ net < 0 \end{cases}$$

4. Input to the output unit

$$net_k = \sum_{j=1}^{n_H} w_{kj} y_j + w_{k0} = \mathbf{w}_k^t \mathbf{y}$$

5. Activation of the output unit

$$z_k = f(net_k)$$



Feedforward (前馈) Neural Network

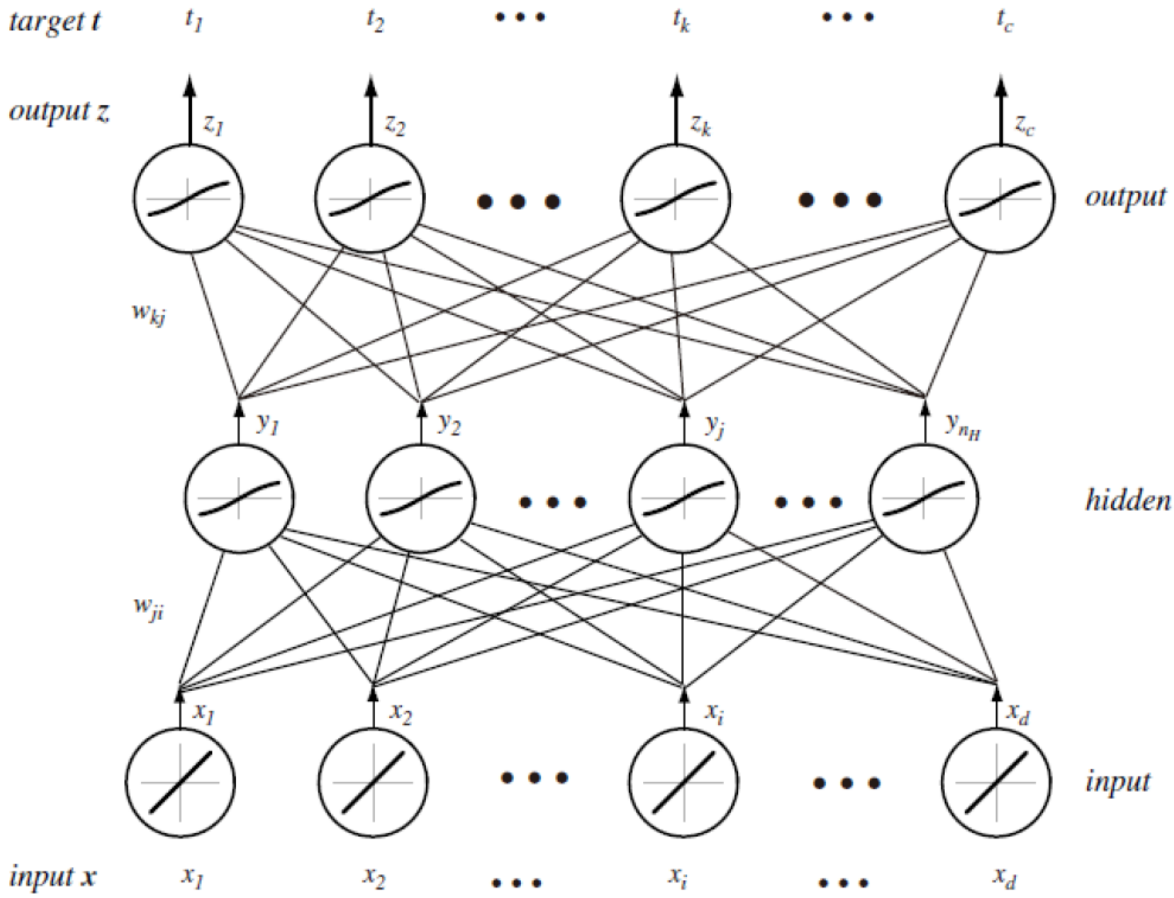
A $d - n_H - c$ fully connected three-layer network

Settings

- d : # features
- n_H : # hidden neurons
- c : # output neurons
- $\mathrm{x} = (x_1, x_2, \ldots, x_d)^t$: **training** pattern
- $\mathrm{t} = (t_1, t_2, \ldots, t_c)^t$: desired **output**
- w_{ji} : **input-to-hidden** layer weight
  - i-th feature to j-th hidden unit
- w_{kj} : **hidden-to-output** layer weight
  - j-th hidden to k-th output unit
- $(1 \leq i \leq d; 1 \leq j \leq n_H; 1 \leq k \leq c)$
- $\mathbf{w} = (w_{11}, \ldots, w_{n_H d}, \ldots, w_{cn_H})^t$
  - # parameters in w : $n_H(d+c)$
- net_i: the input (weighted sum from previous layer) of the i-th neuron of current layer

Feedforward procedure

$$net_j = \sum_{i=1}^{d} w_{ji} x_i \quad (1 \le j \le n_H)$$
$$y_j = f(net_j) \quad (1 \le j \le n_H)$$
$$net_k = \sum_{j=1}^{n_H} w_{kj} y_j \quad (1 \le k \le c)$$
$$z_k = f(net_k) \quad (1 \le k \le c)$$
$$g_k(\mathrm{x}) = z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i\right)\right)$$



Activation Fuction

- Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Tanh

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- ReLU

$$f(x) = \max(0, x)$$

Backpropagation Algorithm

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\sum_{j=1}^{n_H} w_{kj} y_j}{\partial w_{kj}} = -\delta_k y_j$$

where

$$
\begin{aligned}
\delta_k &= -\frac{\partial J}{\partial net_k} \\
&= -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} \\
&= -\frac{\partial\left(\frac{1}{2}\sum_{k=1}^{c}\left(t_k - z_k\right)^2\right)}{\partial z_k}\frac{\partial f\left(net_k\right)}{\partial net_k} \\
&= \left(t_k - z_k\right) f'\left(net_k\right)
\end{aligned}
$$

so that

$$
\Delta w_{kj} = -\eta\frac{\partial J}{\partial w_{kj}} = \eta\delta_k y_j = \eta\left(t_k - z_k\right) f'\left(net_k\right) y_j
$$

where

$$
f'\left(net_k\right) = f' = \begin{cases} f(1-f) & if\ f\ is\ Sigmoid \\ 1 - f^2 & if\ f\ is\ Tanh \end{cases}
$$

and

$$
\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial\left(\sum_{i=1}^{d} w_{ji}x_i\right)}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} f'\left(\mathrm{net}_j\right) x_i = -\delta_j x_i
$$

where

$$
\begin{aligned}
\frac{\partial J}{\partial y_j} &= \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}\left(t_k - z_k\right)^2\right] \\
&= -\sum_{k=1}^{c}\left(t_k - z_k\right)\frac{\partial z_k}{\partial y_j} \\
&= -\sum_{k=1}^{c}\left(t_k - z_k\right)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} \\
&= -\sum_{k=1}^{c}\left(t_k - z_k\right) f'\left(\mathrm{net}_k\right) w_{kj} \\
&= -\sum_{k=1}^{c} w_{kj}\delta_k
\end{aligned}
$$

so that

$$
\Delta w_{ji} = -\eta\frac{\partial J}{\partial w_{ji}} = \eta\delta_j x_i = -\eta\frac{\partial J}{\partial y_j} f'\left(net_j\right) x_i = \eta\left[\sum_{k=1}^{c} w_{kj}\delta_k\right] f'\left(\mathrm{net}_j\right) x_i
$$

Note that δ_k and δ_j is called **the sensitivity of neuron's unit's**

Stochastic training

One pattern is **randomly selected** from the training set, and **the weights are updated by presenting the chosen pattern to the network**

1. **begin initialize** n_H, $\mathbf{w}$, criterion θ, η, $m \leftarrow 0$
2.   **do** $m \leftarrow m + 1$
3. $\mathbf{x}^m \leftarrow$ randomly chosen training pattern
4.   Invoke the forward and backpropagation procedures on $\mathbf{x}^m$ to obtain $\delta_k\,(1 \leq k \leq c)$, y_j and $\delta_j\,(1 \leq j \leq n_H)$
5. $w_{ji} \leftarrow w_{ji} + \eta \delta_j x_i; \quad w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$
6.   **until** $\|\nabla J(\mathbf{w})\| \leq \theta$
7.   **return** $\mathbf{w}$
8. **end**

Stochastic backpropagation

Batch training

All patterns in the training set are presented to the network at once, and the weights are **updated** in one **epoch**

- $\mathcal{D} = \{(\mathbf{x}^m, \mathbf{t}^m) \mid 1 \leq m \leq n\}$: training set consisting of n patterns
- $\mathrm{x}^m = (x_1, x_2, \ldots, x_d)^t$: training pattern
- $\mathrm{t}^m = (t_1, t_2, \ldots, t_c)^t$: desired output

replace $\mathrm{J}(\mathbf{w})$

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2 \longrightarrow J(\mathbf{w}) = \frac{1}{2}\sum_{m=1}^{n}\|\mathbf{t}^m - \mathbf{z}^m\|^2$$

1. **begin initialize** n_H, $\mathbf{w}$, criterion θ, η, $r \leftarrow 0$
2.   **do** $r \leftarrow r + 1$ *(increment epoch)*
3. $m \leftarrow 0$
4.   **do** $m \leftarrow m + 1$
5. $\mathbf{x}^m \leftarrow$ the m-th pattern in the training set
6.   Invoke the forward and backpropagation procedures on $\mathbf{x}^m$ to obtain $\delta_k\,(1 \leq k \leq c)$, y_j and $\delta_j\,(1 \leq j \leq n_H)$
7. $w_{ji} \leftarrow w_{ji} + \eta \delta_j x_i; \quad w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$
8.   **until** $m = n$
9.   **until** $\|\nabla J(\mathbf{w})\| \leq \theta$
10.   **return** $\mathbf{w}$
11. **end**

Batch backpropagation