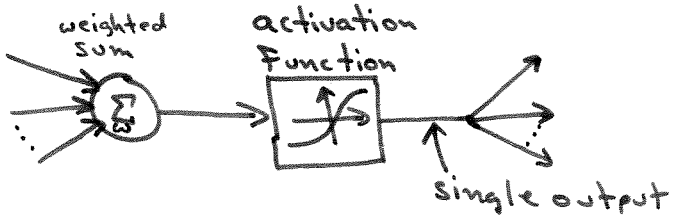
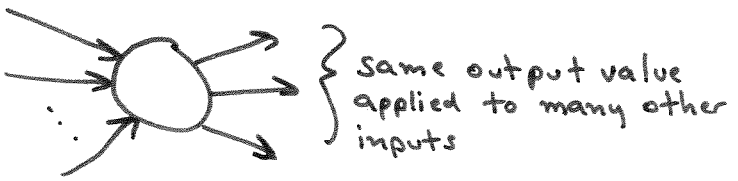
Chapter 13 FeedForward Neural Networks

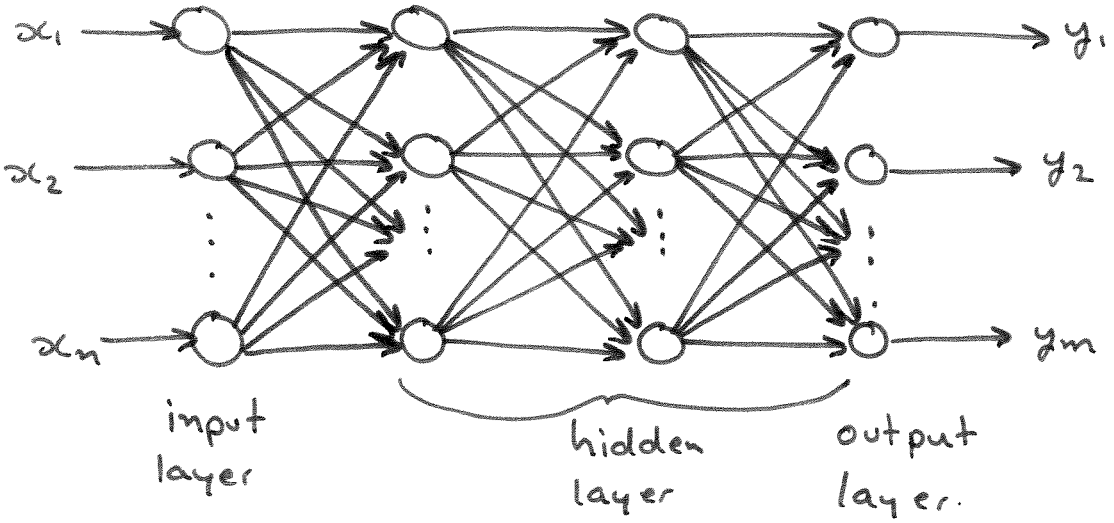
Single neuron model:



multi-input single-output system:



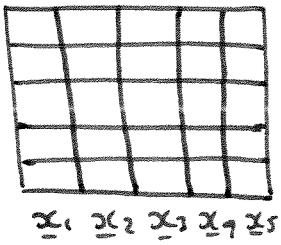
FeedForward neural network



mapping: $F: \mathbb{R}^n \to \mathbb{R}^m$ (possibly nonlinear)

want to approximate F by adjusting the weights in the system. $\Rightarrow$ "distributed representation" of F

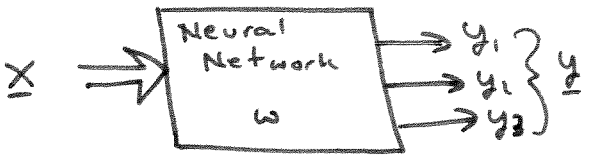
Example: Pattern Recognition



5x5 pixel image

let $\underline{X} = \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \underline{x}_3 \\ \underline{x}_4 \\ x_5 \end{pmatrix} \in \mathbb{R}^{25}$

$\underline{x}_i \in \mathbb{R}^5 \quad i = 1(1)5$



training set: "picture" dog $\underline{X}_d$ $\underline{y}_d = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

cat $\underline{X}_c$ $\underline{y}_c = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

bird $\underline{X}_b$ $\underline{y}_b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

} set weights to achieve this

this is called supervised learning.

now input picture of Fish $\underline{X}_F$ ⇒ [ ] ⇒ $\underline{y}_F$? $\begin{pmatrix} .5 \\ 0 \\ .5 \end{pmatrix}$

interpretation ??

---

in general, for a neural network:

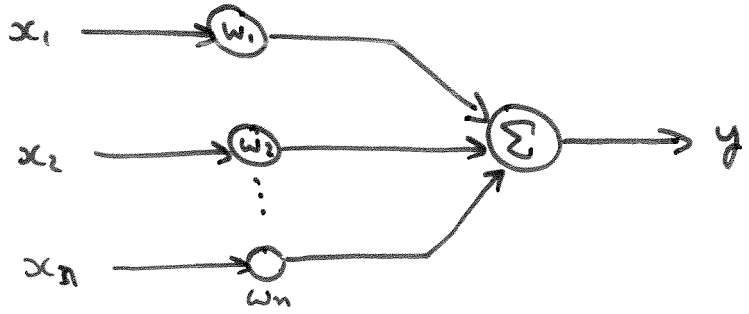
$\underline{x}_d \in \mathbb{R}^n$ ⇒ N.N. ⇒ $\underline{y}_d \in \mathbb{R}^m$

training set: p input/output pairs.

$(\underline{x}_{d,1}, \underline{y}_{d,1}), (\underline{x}_{d,2}, \underline{y}_{d,2}) \ldots (\underline{x}_{d,p}, \underline{y}_{d,p}) \in \mathbb{R}^n \times \mathbb{R}^m$

where: $\underline{y}_{d,i} = F(\underline{x}_{d,i})$ "desired" output

Single - Neuron Training



single linear neuron $\mathbb{R}^n \to \mathbb{R}$

$$y = \sum_{i=1}^{n} w_i x_i = \underline{x}^T \underline{w}$$

input $\underline{x} \in \mathbb{R}^n$ $w \in \mathbb{R}^n$ weights
$y \in \mathbb{R}$ output

given a map $F : \mathbb{R}^n \to \mathbb{R}$ find weights $\underline{w}$ s.t. neuron approximates F as closely as possible.

use a training set of p pairs: $\{(\underline{x}_{d,1}, y_{d,1}) \dots (\underline{x}_{d,p}, y_{d,p})\}$
$y_{d,i} = F(\underline{x}_{d,i})$ $i = 1(1)p$

we want to optimize $\underline{w}$:

$$\min_{\underline{w}} \frac{1}{2} \sum_{i=1}^{p} (y_{d,i} - \underline{x}_{d,i}^T \underline{w})^2$$

or in matrix form, let $X_d = [\underline{x}_{d,1} \cdots \underline{x}_{d,p}] \in \mathbb{R}^{n \times p}$

$$\underline{y}_d = \begin{bmatrix} y_{d,1} \\ \vdots \\ y_{d,p} \end{bmatrix}$$

$$\min_{\underline{w}} \frac{1}{2} \| \underline{y}_d - X_d^T \underline{w} \|^2$$

Case 1: $p < n$

assuming that the p inputs of the training set are independent $\Rightarrow$ rank $X_d^T = p$

we're actually trying to solve $\underline{y}_d = X_d^T \underline{w}$ $\quad X_d^T \in \mathbb{R}^{p \times n}$

if $p < n \Rightarrow$ fewer training pairs, p, than the number of weights, n.
$\Rightarrow$ infinitely many possible solutions for weights, $\underline{w}$ i.e. $\| \underline{y}_d - X_d^T \underline{w} \| = 0$

the solution with the minimum norm $\|\underline{w}\|$ is given by $\quad (A^T (AA^T)^{-1})$

$$\boxed{\underline{w}^* = X_d \left(X_d^T X_d\right)^{-1} \underline{y}_d}$$

by Kaczmarz's algorithm

$$\begin{cases} \underline{w}^{(k+1)} = \underline{w}^{(k)} + \mu \dfrac{e_k \underline{x}_{d,R(k)}}{\| \underline{x}_{d,R(k)} \|^2} & \underline{w}^{(0)} = \underline{0} \\ e_k = y_{d,R(k)} - \underline{x}_{d,R(k)}^T \underline{w}^{(k)} \end{cases}$$

$R(k)$ is remainder if we divide k by p, $+1$
$R(k) \in \{1, \ldots, p\}$

case 2: $p > n$ $\Rightarrow$ more training points than number of weights.

assume rank $X_d^T = n$

$\min_{\underline{w}} \frac{1}{2} \| \underline{y}_d - X_d^T \underline{w} \|^2$ in general $X_d^T \underline{w} \neq \underline{y}_d$ unless $\underline{y}_d \in \mathcal{R}(X_d^T)$

least squares solution:

$$\boxed{\underline{w}^* = (X_d X_d^T)^{-1} X_d \underline{y}_d}$$

we can also use any unconstrained optimization algorithm on

$$\| \underline{y}_d - X_d^T \underline{w} \|^2 = (\underline{y}_d - X_d^T \underline{w})^T (\underline{y}_d - X_d^T \underline{w})$$

$$= \| \underline{y}_d \|^2 + \underline{w}^T X_d X_d^T \underline{w} - 2 \underline{w}^T X_d \underline{y}_d$$

$$F(\underline{w}) = \frac{1}{2} \| \underline{y}_d - X_d^T \underline{w} \|^2 = \frac{1}{2} \underline{w}^T X_d X_d^T \underline{w} - \underline{w}^T X_d \underline{y}_d + \| \underline{y}_d \|^2$$

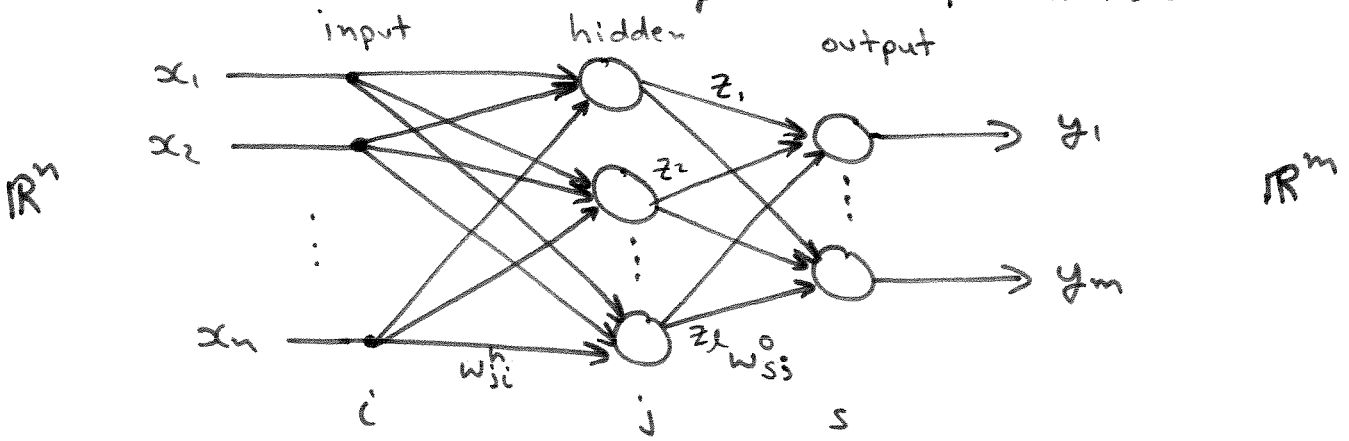
$$\nabla F(\underline{w}) = X_d X_d^T \underline{w} - X_d \underline{y}_d$$

descent direction $\underline{d} = -\nabla F = X_d \underline{y}_d - X_d X_d^T \underline{w}$

descent algorithm: $\underline{w}^{(k+1)} = \underline{w}^{(k)} + \alpha_k \underline{d}$

Backpropagation Algorithm

Consider the following three layer network:



n inputs x_i $i = 1(1)n$

m outputs y_s $s = 1(1)m$

l outputs from hidden layer z_j $j = 1(1)l$

l neurons in hidden layer.

input layer $\Rightarrow$ identity map for activation function

hidden layer $\Rightarrow$ activation function f^h_j $j = 1(1)l$

output layer $\Rightarrow$ activation function f^o_s $s = 1(1)m$

$f: \mathbb{R} \to \mathbb{R}$

weights for inputs into hidden layer: w^h_{ji} $i = 1(1)n$, $j = 1(1)l$

weights for inputs into output layer: w^o_{sj} $s = 1(1)m$

input to a hidden layer neuron: $\underline{x} \Rightarrow (\underline{w}^h_j) \to v_j$

$$v_j = \underline{w}^{h\,T}_j \underline{x} = \sum_{i=1}^{n} w^h_{ji} x_i$$

output from hidden layer:

$$z_j = f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)$$

output from s^{th} neuron of the output layer:

$$y_s = f_s^o\left(\sum_{j=1}^{l} w_{sj}^o z_j\right)$$

$$\therefore \quad y_s = f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h(v_j)\right)$$

$$= f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)\right)$$

$$= F_s(x_1, \ldots, x_n)$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} F_1(\underline{x}) \\ \vdots \\ F_m(\underline{x}) \end{bmatrix}$$

now how do we find the sets of weights w_{ji}^h and w_{sj}^o given a single pair of input/output vectors $(\underline{x}_d, \underline{y}_d)$ $\underline{x}_d \in \mathbb{R}^n$ $\underline{y}_d \in \mathbb{R}^m$ (desired output)

$$\min_{\underline{w}^o, \underline{w}^h} \frac{1}{2} \sum_{s=1}^{m} (y_{ds} - y_s)^2 \qquad y_s = F_s(\underline{x})$$

$$E(\underline{w}^o, \underline{w}^h) = \frac{1}{2} \sum_{s=1}^{m} (y_{ds} - y_s)^2$$

$$= \frac{1}{2} \sum_{s=1}^{m} \left(y_{ds} - f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_{di}\right)\right)\right)^2$$

we minimize $E(\underline{w}^o, \underline{w}^h)$ using a gradient algorithm with fixed step size, i.e. no line-minimization.

$\therefore$ need gradient of E with respect to $\underline{w}^o$, $\underline{w}^h$,

write: $$E(\underline{w}) = \frac{1}{2} \sum_{p=1}^{m} \left(y_{dp} - f_p^o \left(\sum_{q=1}^{l} w_{pq}^o z_q \right) \right)^2$$

where $$z_q = f_q^h \left(\sum_{i=1}^{n} w_{qi}^h x_{di} \right)$$

$$\frac{\partial E(\underline{w})}{\partial w_{sj}^o} = -(y_{ds} - y_s) f_s^{o'} \left(\sum_{q=1}^{l} w_{sq}^o z_q \right) z_j$$

$f_s^{o'} : \mathbb{R} \to \mathbb{R}$ is derivative of f_s^o.

now define: $$\delta_s = (y_{ds} - y_s) f_s^{o'} \left(\sum_{q=1}^{l} w_{sq}^o z_q \right)$$

error: $(y_{ds} - y_s)$ scaled by $f_s^{o'} \left(\sum_{q=1}^{l} w_{sq}^o z_q \right)$

$$\boxed{\frac{\partial E(w)}{\partial w_{sj}^o} = -\delta_s z_j}$$

in order to take $\partial E / \partial w_{ji}^h$ we re-write $E(\underline{w})$

$$E(\underline{w}) = \frac{1}{2} \sum_{p=1}^{m} \left(y_{dp} - f_p^o \left(\sum_{q=1}^{l} w_{pq}^o f_q^h \left(\sum_{r=1}^{n} w_{qr}^h x_{dr} \right) \right) \right)^2$$

$$\frac{\partial E}{\partial w_{ji}^{h}}(\underline{w}) = -\sum_{p=1}^{m} (y_{dp} - y_p)\, f_p^{o\prime}\left(\sum_{q=1}^{l} w_{pq}^{o} z_q\right) w_{pj}^{o} f_j^{h\prime}\left(\sum_{r=1}^{n} w_{jr}^{h} x_{dr}\right) \cdot x_{di}$$

again $f_j^{h\prime} : \mathbb{R} \to \mathbb{R}$ is derivative of f_j^{h}

we can write the above as:

$$\boxed{\frac{\partial E}{\partial w_{ji}^{h}}(\underline{w}) = -\left(\sum_{p=1}^{m} \delta_p w_{pj}^{o}\right) f_j^{h\prime}(v_j)\, x_{di}}$$

the fixed-step size gradient algorithm for each weight will have the form

$$w^{(n+1)} = w^{(n)} - \eta \frac{\partial E}{\partial w} \qquad \eta \text{ is the step-size.}$$

$\therefore$ we have what is called the "<u>backpropagation algorithm</u>"

$$\begin{cases} w_{sj}^{o(k+1)} = w_{sj}^{o(k)} + \eta\, \delta_s^{(k)} z_j^{(k)} \\ w_{ji}^{h(k+1)} = w_{ji}^{h(k)} + \eta \left(\sum_{p=1}^{m} \delta_p^{(k)} w_{pj}^{o(k)}\right) f_j^{h\prime}\left(v_j^{(k)}\right) x_{di} \end{cases}$$

$$\begin{cases} v_j^{(k)} = \sum_{i=1}^{n} w_{ji}^{h(k)} x_{di} \\ z_j^{(k)} = f_j^{h}\left(v_j^{(k)}\right) \\ y_s^{(k)} = f_s^{o}\left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)}\right) \\ \delta_s^{(k)} = \left(y_{ds} - y_s^{(k)}\right) f_s^{o\prime}\left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)}\right) \end{cases}$$

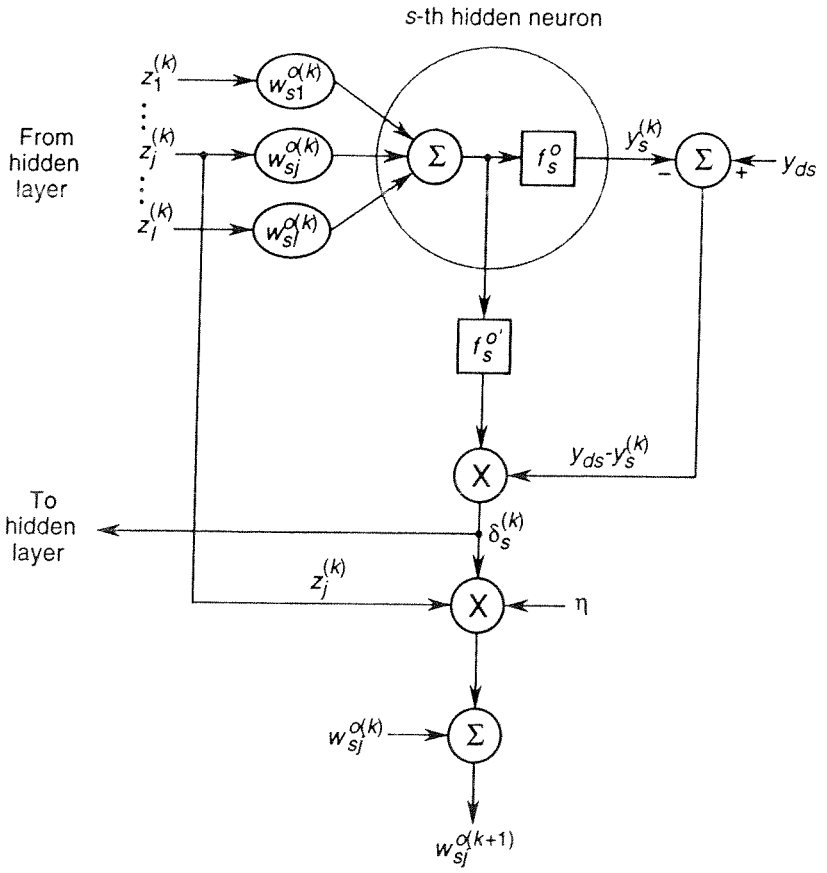


Figure 13.7. Illustration of the update equation for the output layer weights

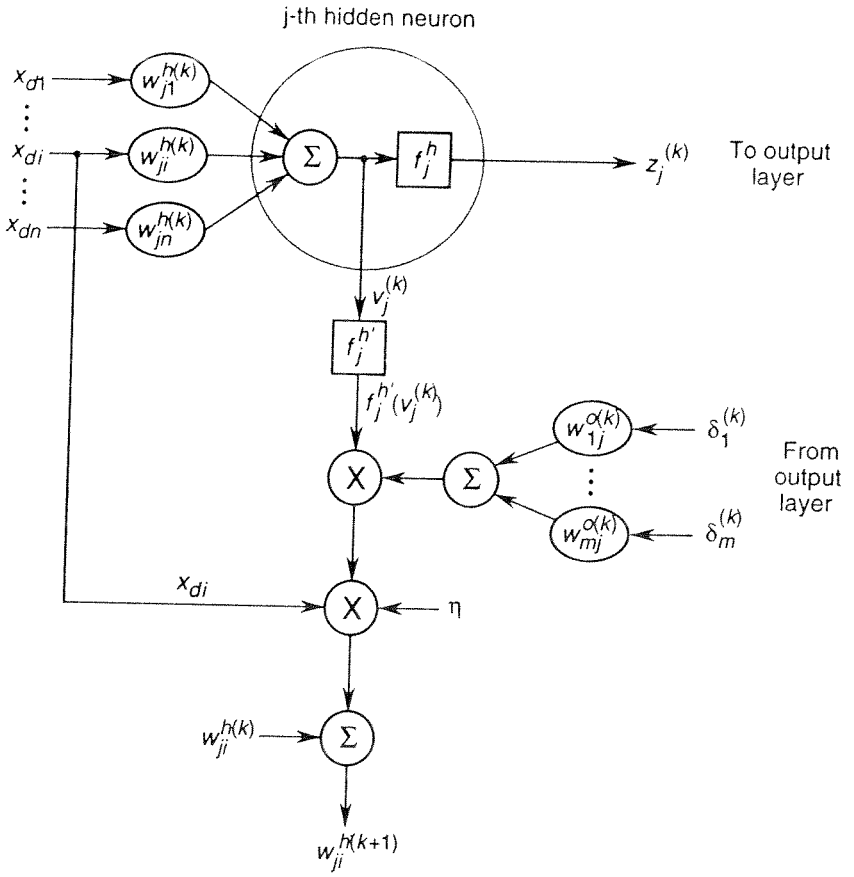
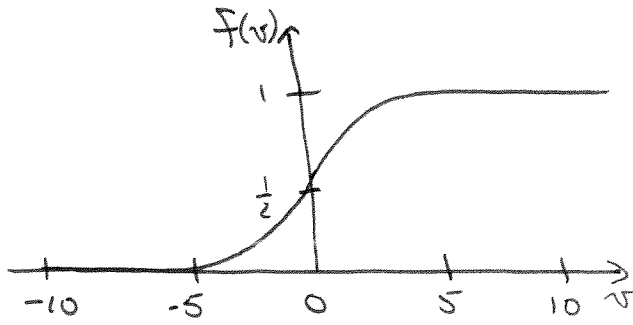


Figure 13.8. Illustration of the update equation for the hidden layer weights

Sigmoid Function

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

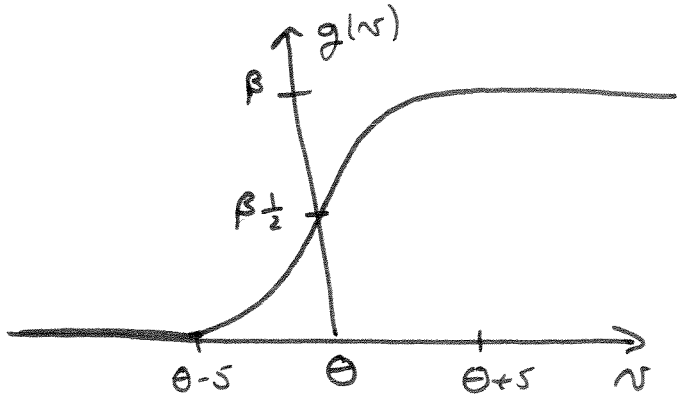


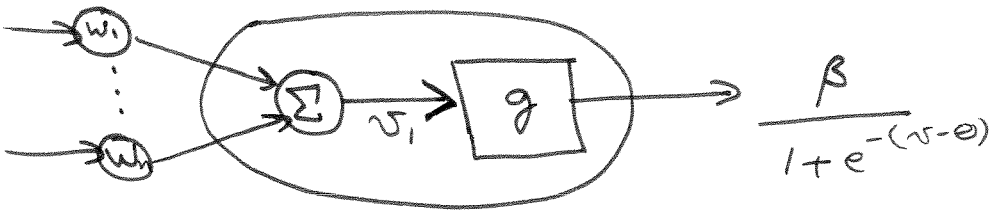
$$g(v) = \frac{\beta}{1+e^{-(v-\theta)}}$$

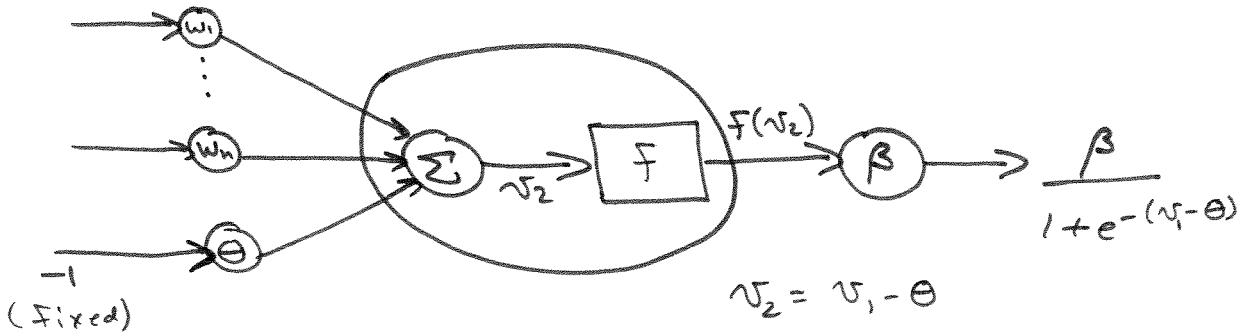
β, θ are parameters

β, scale parameter

θ, shift parameter







Note: $f'(v) = \frac{-1(-e^{-v})}{(1+e^{-v})^2} = \frac{e^{-v}}{(1+e^{-v})^2} = \frac{1+e^{-v}-1}{(1+e^{-v})^2}$

$$= \frac{1}{(1+e^{-v})}\left(\frac{1+e^{-v}}{1+e^{-v}} - \frac{1}{(1+e^{-v})}\right) = f(v)\big(1-f(v)\big)$$

with the Sigmoid Function as the activation Function,

$$y_s = \beta_s^o \, f_s^o \left(\sum_{j=1}^{l} w_{sj}^o \, \beta_j^h \, f_j^h \left(\sum_{i=1}^{n} w_{ji}^h \, x_i - \theta_j^h \right) - \theta_s^o \right)$$

$$E(\underline{w}, \underline{\theta}, \underline{\beta}) = \frac{1}{2} \sum_{s=1}^{m} (y_{ds} - y_s)^2$$

$$= \frac{1}{2} \sum_{s=1}^{m} \left(y_{ds} - \beta_s^o f_s^o \left(\sum_{j=1}^{l} w_{sj}^o \, \beta_j^h \, f_j^h \left(\sum_{i=1}^{n} w_{ji}^h \, x_{di} - \theta_j^h \right) - \theta_s^o \right) \right)^2$$

input to j^{th} hidden neuron

output from j^{th} hidden neuron

output from s^{th} output neuron

again letting $z_q = f_q^h \left(\sum_{i=1}^{n} w_{qi}^h \, x_{di} - \theta_q^h \right)$

output from hidden neuron before multiplication with β_q^h

$$E = \frac{1}{2} \sum_{p=1}^{m} \left(y_{dp} - \beta_p^o \, f_p^o \left(\sum_{q=1}^{l} w_{pq}^o \, \beta_q^h \, z_q - \theta_p^o \right) \right)^2$$

$$\frac{\partial E}{\partial w_{sj}^o} = -(y_{ds} - y_s) \, \beta_s^o \, f_s^{o\prime} \left(\sum_{q=1}^{l} w_{sq}^o \, \beta_q^h \, z_q - \theta_s^o \right) \beta_j^h \, z_j$$

let $\delta_s = (y_{ds} - y_s) \, f_s^{o\prime} \left(\sum_{q=1}^{l} w_{sq}^o \, \beta_q^h \, z_q - \theta_s^o \right) \beta_s^o$

$$\therefore \frac{\partial E}{\partial w_{sj}^o} = -\delta_s \, \beta_j^h \, z_j$$

$$\frac{\partial E}{\partial \theta_s^o} = +(y_{ds} - y_s)\,\beta_s^o\, f_s^{o\prime}\left(\sum_{q=1}^{l} w_{sq}^o\, \beta_q^h\, z_q - \theta_s^o\right) = \delta_s$$

$$\frac{\partial E}{\partial \beta_s^o} = -(y_{ds} - y_s)\, f_s^o\left(\sum_{q=1}^{l} w_{sq}^o\, \beta_q^h\, z_q - \theta_s^o\right)$$

now we need $\dfrac{\partial E}{\partial w_{ji}^h}$, $\dfrac{\partial E}{\partial \theta_j^h}$, $\dfrac{\partial E}{\partial \beta_j^h}$

we re-write $E(\underline{w}, \underline{\theta}, \underline{\beta})$ as:

$$E = \frac{1}{2}\sum_{p=1}^{m}\left(y_{dp} - \beta_p^o\, f_p^o\left(\sum_{q=1}^{l} w_{pq}^o\, \beta_q^h\, f_q^h\left(\sum_{r=1}^{n} w_{qr}^h\, x_{dr} - \theta_q^h\right) - \theta_p^o\right)\right)^2$$

$$\frac{\partial E}{\partial w_{ji}^h} = -\sum_{p=1}^{m}(y_{dp} - y_p)\,\beta_p^o\, f_p^{o\prime}\left(\sum_{q=1}^{l} w_{pq}^o\, \beta_q^h\, z_q - \theta_p^o\right) w_{pj}^o\, \beta_j^h\, f_j^{h\prime}\left(\sum_{r=1}^{n} w_{jr}^h\, x_{dr} - \theta_j^h\right) \cdot x_{di}$$

$$= -\left(\sum_{p=1}^{m} \delta_p\, w_{pj}^o\right) \beta_j^h\, f_j^{h\prime}(\nu_j)\, x_{di}$$

where $\nu_j = \sum_{i=1}^{n} w_{ji}^h\, x_{di} - \theta_j^h$

$$\frac{\partial E}{\partial \theta_j^h} = +\sum_{p=1}^{m}(y_{dp} - y_p)\,\beta_p^o\, f_p^{o\prime}\left(\sum_{q=1}^{l} w_{pq}^o\, \beta_q^h\, z_q - \theta_p^o\right) w_{pj}^o\, \beta_j^h\, f_j^{h\prime}\left(\sum_{r=1}^{n} w_{jr}^h\, x_{dr} - \theta_j^h\right)$$

$$= \left(\sum_{p=1}^{m} \delta_p\, w_{pj}^o\right) \beta_j^h\, f_j^{h\prime}(\nu_j)$$

$$\frac{\partial E}{\partial \beta_j^h} = -\sum_{p=1}^{m} (y_{dp} - y_p)\, \beta_p^o\, f_p^{o\prime}\left(\sum_{q=1}^{l} w_{pq}^o\, \beta_q^h\, z_q - \theta_p^o\right) w_{pj}^o\, f_j^{h\prime}\left(\sum_{r=1}^{n} w_{jr}\, x_{dr} - \theta_j^h\right)$$

$$= -\left(\sum_{p=1}^{m} \delta_p\, w_{pj}^o\right) \quad f_j^{h\prime}(v_j) \Big\}$$

Therefore the back propagation algorithm becomes

$$w_{sj}^{o(k+1)} = w_{sj}^{o(k)} + \eta\, \delta_s^{(k)}\, \beta_j^{h(k)}\, z_j^{(k)}$$

$$\theta_s^{o(k+1)} = \theta_s^{o(k)} - \eta\, \delta_s^{(k)}$$

$$\beta_s^{o(k+1)} = \beta_s^{o(k)} + \eta\,(y_{ds} - y_s^{(k)})\; y_s^{(k)} / \beta_s^{o(k)}$$

$$w_{ji}^{h(k+1)} = w_{ji}^{h(k)} + \eta\left(\sum_{p=1}^{m} \delta_p^{(k)}\, w_{pj}^{o(k)}\right) \beta_j^{h(k)}\, f_j^{h\prime}(v_j^{(k)})\, x_{di}$$

$$\theta_j^{h(k+1)} = \theta_j^{h(k)} - \eta\left(\sum_{p=1}^{m} \delta_p^{(k)}\, w_{pj}^{o(k)}\right) \beta_j^{h(k)}\, f_j^{h\prime}(v_j^{(k)})$$

$$\beta_j^{h(k+1)} = \beta_j^{h(k)} + \eta\left(\sum_{p=1}^{m} \delta_p^{(k)}\, w_{pj}^{o(k)}\right) z_j^{(k)}$$

$$v_j^{(k)} = \sum_{i=1}^{n} w_{ji}^{h(k)}\, x_{di} - \theta_j^{h(k)}$$

$$z_j^{(k)} = f_j^h\left(v_j^{(k)}\right)$$

$$y_s^{(k)} = \beta_s^{o(k)} f_s^o\left(\sum_{q=1}^{l} w_{sq}^{o(k)}\, \beta_q^{h(k)}\, z_q^{(k)} - \theta_s^{o(k)}\right)$$

$$\delta_s^{(k)} = (y_{ds} - y_s^{(k)})\, f_s^{o\prime}\left(\sum_{q=1}^{l} w_{sq}^{o(k)}\, \beta_q^{h(k)}\, z_q^{(k)} - \theta_s^{o(k)}\right) \Big/ \beta_s^{o(k)}$$

we can now use the fact that $F'(v) = F(v)(1-F(v))$ for the sigmoid function and the algorithm becomes:

$s = 1(1)m$ output neurons.
$j = 1(1)l$ hidden neurons
$i = 1(1)n$ input neurons.

$$\begin{cases} w_{sj}^{o(k+1)} = w_{sj}^{o(k)} + \eta\, \delta_s^{(k)}\, \beta_j^{h(k)}\, z_j^{(k)} \\ \theta_s^{o(k+1)} = \theta_s^{o(k)} - \eta\, \delta_s^{(k)} \\ \beta_s^{o(k+1)} = \beta_s^{o(k)} + \eta\,(y_{ds} - y_s^{(k)})\, \dfrac{y_s^{(k)}}{\beta_s^{o(k)}} \end{cases}$$

$$\begin{cases} w_{ji}^{h(k+1)} = w_{ji}^{h(k)} + \eta\, \Delta_j^{(k)}\, \beta_j^{h(k)}\, f_j^{h}(v_j^{(k)})\left(1 - f_j^{h}(v_j^{(k)})\right) x_{di} \\ \theta_j^{h(k+1)} = \theta_j^{h(k)} - \eta\, \Delta_j^{(k)}\, \beta_j^{h(k)}\, f_j^{h}(v_j^{(k)})\left(1 - f_j^{h}(v_j^{(k)})\right) \\ \beta_j^{h(k+1)} = \beta_j^{h(k)} + \eta\, \Delta_j^{(k)}\, z_j^{(k)} \end{cases}$$

$$\begin{cases} v_j^{(k)} = \sum_{i=1}^{n} w_{ji}^{h(k)}\, x_{di} - \theta_j^{h(k)} \\ z_j^{(k)} = f_j^{h}\left(v_j^{(k)}\right) \\ y_s^{(k)} = \beta_s^{o(k)}\, f_s^{o}\left(\underbrace{\sum_{q=1}^{l} w_{sq}^{o(k)}\, \beta_q^{h(k)}\, z_q^{(k)} - \theta_s^{o(k)}}_{u_s^{(k)}}\right) \\ \delta_s^{(k)} = (y_{ds} - y_s^{(k)})\, f_s^{o}\left(u_s^{(k)}\right)\left(1 - f_s^{o}(u_s^{(k)})\right) \beta_s^{o(k)} \\ \Delta_j^{(k)} = \sum_{p=1}^{m} \delta_p^{(k)}\, w_{pj}^{o(k)} \end{cases}$$