Neural Nets for Dummies

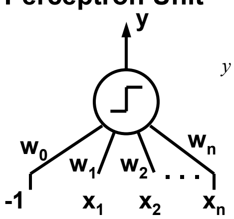
Training:
- Choose connection weights that minimize error

Prediction:
- Propagate input feature values through the network of "artificial neurons"

Advantages:
- Fast prediction
- Does feature weighting
- Very generally applicable

Disadvantages:
- Very slow training
- Overfitting is easy

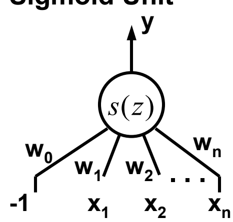
Perceptron Unit



$$y = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 \text{ otherwise} \end{cases}$$

Creates a decision plane (line) in feature space

Sigmoid Unit



$$z = \sum_{i}^{n} w_i x_i$$

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

Creates a "soft" decision plane (line) in feature space

Adopted from Tomas Lozano Perez's 6.034 Recitation Notes

The simplest two-layer sigmoid Neural Net

$$E = \frac{1}{2}(y^* - F(\vec{x}, \vec{w}))^2$$

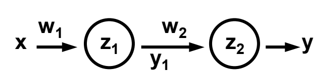
$y = F(\vec{x}, \vec{w})$ — $\vec{w}$: vector of weights; $\vec{x}$: vector of inputs

$$\frac{\partial E}{\partial w_j} = -(y^* - y)\frac{\partial y}{\partial w_j}$$

y^* = Desired output

$$y = F(\vec{x}, \vec{w}) = s(z_2) = s(w_2 y_1) = s(w_2 s(z_1)) = s(w_2 s(w_1 x))$$

Goal: find the weight vector that minimizes the error



Approach: Gradient Descent (How does the error change as we twiddle with the weights?)

$$\frac{\partial y}{\partial w_2} = \frac{\partial s(z_2)}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \frac{\partial s(z_2)}{\partial z_2} y_1 \quad \text{recall } z_2 = w_2 y_1 \text{ so, } \frac{\partial z_2}{\partial w_2} = y_1$$

$$\frac{\partial E}{\partial w_2} = \underbrace{\frac{\partial s(z_2)}{\partial z_2}(y - y^*)}_{\delta_2} y_1$$

$$\frac{\partial y}{\partial w_1} = \frac{\partial s(z_2)}{\partial z_2}\frac{\partial z_2}{\partial s(z_1)}\frac{\partial s(z_1)}{\partial z_1}\frac{\partial z_1}{\partial w_1} = \frac{\partial s(z_2)}{\partial z_2} w_2 \frac{\partial s(z_1)}{\partial z_1} x$$

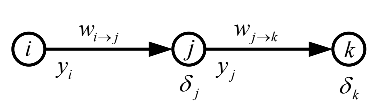
recall $z_2 = w_2 s(z_1)$ so, $\frac{\partial z_2}{\partial s(z_1)} = w_2$

recall $z_1 = w_1 x$ so, $\frac{\partial z_1}{\partial w_1} = x$

$$\frac{\partial E}{\partial w_1} = \underbrace{\frac{\partial s(z_1)}{\partial z_1}\delta_2 w_2}_{\delta_1} x$$

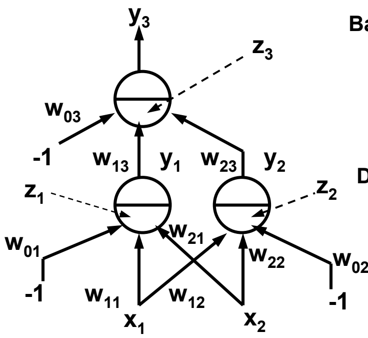
Descent rule: $w_{i\to j} = w_{i\to j} - r\delta_j y_i$

Backpropagation rule: $\delta_j = \frac{ds(z_j)}{dz_j}\sum_k \delta_k w_{j\to k}$



y_i is x_i for input layer

Example of Backpropagation



Backpropagation rule:

$\delta_3 =$

$\delta_2 =$

$\delta_1 =$

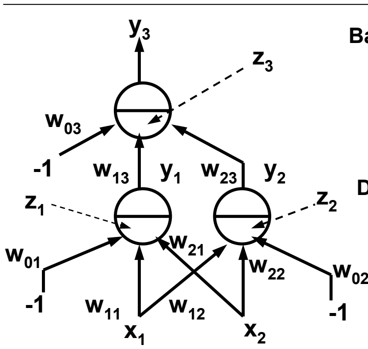
Descent rule:

$w_{03} =$	$w_{13} =$	$w_{23} =$
$w_{02} =$	$w_{12} =$	$w_{22} =$
$w_{01} =$	$w_{11} =$	$w_{21} =$

Initial Conditions: all weights are zero, learning rate is 8. Input: *$(x_1, x_2) = (0, 1)$*

$y^* =$	$\delta_3 =$	$w_{03} =$	$w_{13} =$
$z_1 =$	$\delta_2 =$	$w_{02} =$	$w_{12} =$
$z_2 =$	$\delta_1 =$	$w_{01} =$	$w_{11} =$
$y_1 =$			$w_{23} =$
$y_2 =$			$w_{22} =$
$z_3 =$			$w_{21} =$
$y_3 =$			

Example of Backpropagation



Backpropagation rule:

$\delta_3 = y_3(1-y_3)(y_3 - y_3^*)$

$\delta_2 = y_2(1-y_2)\delta_3 w_{23}$

$\delta_1 = y_1(1-y_1)\delta_3 w_{13}$

Descent rule:

$w_{03} = w_{03} - r\delta_3(-1)$	$w_{13} = w_{13} - r\delta_3 y_1$	$w_{23} = w_{23} - r\delta_3 y_2$
$w_{02} = w_{02} - r\delta_2(-1)$	$w_{12} = w_{12} - r\delta_2 x_1$	$w_{22} = w_{22} - r\delta_2 x_2$
$w_{01} = w_{01} - r\delta_1(-1)$	$w_{11} = w_{11} - r\delta_1 x_1$	$w_{21} = w_{21} - r\delta_1 x_2$

Initial Conditions: all weights are zero, learning rate is 8. Input: *$(x_1, x_2) = (0, 1)$*

$y^* = 1$	$\delta_3 = -1/8$	$w_{03} = -1$	$w_{13} = 1/2$
$z_1 = 0$	$\delta_2 = 0$	$w_{02} = 0$	$w_{12} = 0$
$z_2 = 0$	$\delta_1 = 0$	$w_{01} = 0$	$w_{11} = 0$
$y_1 = 1/2$			$w_{23} = 1/2$
$y_2 = 1/2$			$w_{22} = 0$
$z_3 = 0$			$w_{21} = 0$
$y_3 = 1/2$			