What does this do?

unit turns on when $\sum_j w_j x_j > 0$

$W_0 = -1.5$
$W_1 = 1$
$W_2 = 1$
What about this one?

unit turns on when \[ \sum_j w_j x_j > 0 \]
Neural Networks
Notation: a perceptron $j$ has

• a current output or *activation* level, $o_j$
• a target or desired output level, $t_j$
• links from input values of weight $w_{ij}$
• a threshold or bias $\theta$ at which it activates
  • treat as constant input of -1 connected by weight $\theta$
Input and Output

- each cell’s output $o_k$ is determined by:
  - $o_k = \text{step}(net_k)$
- where the net input $net_k$ to cell $k$ is summed:
  - $net_k = \sum_j w_j o_j$
Perceptrons:
Single layer feed-forward networks

- response given by: $o = \text{step}(\sum_j w_j x_j)$
Perceptron (or “Delta Rule”) Learning
[Widrow and Hoff; Rosenblatt 1960]

- recall
  \[ o = \sum_j w_j x_j \]

- LMS expresses error as sum of squared errors:
  \[ E = \frac{1}{2} Err^2 = \sum \frac{1}{2} (t - o)^2 \]

- calculate partial of Error with respect to each weight:
  \[ \frac{\partial E}{\partial w_j} = (\frac{\partial E}{\partial o})(\frac{\partial o}{\partial w_j}) = -(t - o)x_j \]

- now modify weights along negative of error gradient, for some learning rate, \(\alpha\):
  \[ \Delta w_j = \alpha (t - o)x_j \]
How do you build XOR?

- response given by: \( o = \sum_j w_j x_j \)
Linear non-separability problem
Minsky and Papert [1969]
Two-layer feedforward nets

- multilayer perceptron can solve XOR
Minsky and Papert’s Criticism

• no solution for how to *learn* weights for hidden units:
  • how to give credit (or blame) to the hidden layer units for their contribution to the error?
Backpropagation
[Werbos 1974; Rumelhart, Hinton & Williams 1986, etc.]

- recursive method for weight adjustment in multilayer feed-forward networks
- based on generalization of the delta rule for non-linear activation functions
- assigns error value to output in hidden layers
Input and Output

- each cell’s output $o_k$ is determined by:
  - $o_k = g(\text{net}_k)$
- where the net input $\text{net}_k$ to cell $k$ is summed:
  - $\text{net}_k = \sum_j w_{jk} o_j$
Activation Functions

- typically non-linear, monotonic (or discontinuous at one point)

- common options:
  - \( \text{tanh}() \)
  - \( \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \)
How does weight $w_{jk}$ to output $k$ affect error?

\[ E = \frac{1}{2} (t_k - o_k)^2 \]

\[ \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial w_{jk}} \]

\[ = -(t_k - o_k) \frac{\partial g(\text{net}_k)}{\partial w_{jk}} \]

\[ = -(t_k - o_k)g'(\text{net}_k) \frac{\partial \text{net}_k}{\partial w_{jk}} \]

\[ = -(t_k - o_k)g'(\text{net}_k) \frac{\partial}{\partial w_{jk}} \Sigma_j (w_{jk} o_j) \]

\[ = -(t_k - o_k)g'(\text{net}_k) o_j \]

**define modified error measure:**

\[ \delta_k = (t_k - o_k)g'(\text{net}_k) \]

\[ \frac{\partial E}{\partial w_{jk}} = -o_j \delta_k \]
Continuing...

recall:  \[ \delta_k = (t_k - o_k)g'(net_k) \]
now let’s consider \( g() \) as the sigmoid:  \[ g(x) = \frac{1}{1 + e^{-x}} \]
and so:  \[ g'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} \]
\[ = \frac{1}{1 + e^{-x}} \frac{e^{-x}}{1 + e^{-x}} \]
\[ = g(x)(1 - g(x)) \]

which gives:  \[ \delta_k = (t_k - o_k)g(net_k)(1 - g(net_k)) \]
\[ = (t_k - o_k)o_k(1 - o_k) \]
Network Architecture

output layer

\[ \text{i} \]

\[ w_{ij} \]

\[ w_{jk} \]

\[ j \]

\[ k \]

input layer

\[ \text{i} \]
How does weight $w_{ij}$ (to hidden unit $j$) affect error?

- consider impact of unit $j$ on error over all output-layer (L) units $k$ to which it is connected

$$\text{Error} = \sum_{k \in L} E_k$$
How does weight $w_{ij}$ (to hidden unit $j$) affect error?

- compute recursively via Chain Rule

\[
\frac{\partial E}{\partial w_{ij}} = \sum_k \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ij}}
\]

\[
= \sum_k -(t_k - o_k)o_k(1-o_k)w_{jk}o_j(1-o_j)o_i
\]

\[
= \sum_k -\delta_k w_{jk}o_j(1-o_j)o_i
\]

\[
= -o_i o_j (1-o_j) \sum_k \delta_k w_{jk}
\]
Knowing the Limits

• sigmoid is asymptotic at 0 and 1
  • \( g(\infty) = 1, \ g(-\infty) = 0 \) and recall:

\[
net_k = \sum_j w_{jk} o_j
\]

• since \( o_j \) is determined by output of other units, we can only vary \( w_{jk} \)

• adjusting \( w_{jk} \) risks leading to infinite weights
When should we use NN's?

• where complex mapping needs to be found for which a closed-form analytic solution is not readily available or computationally efficient
Backprop Applications
NOVICE

- multiple networks used for different activities (e.g. base steering, approaching, reaching)
- solves inverse perspective projection and inverse kinematic mappings
- adapts on-line to changes in configuration
LeNet: Numeric character recognition
LeNet: overlapping digits
Character Recognition

- **OCHRE**
  - Java demo
Mozer’s Adaptive House

... a home that essentially PROGRAMS ITSELF by observing the actions of the inhabitants.
Training
Testing the “anticipator” neural net
Multi-zone lighting selections
ALVINN - Driving a Car

- Gaussian output array PDF
  - overcomes problem of severe non-linearities
- training “on the fly”
  - management of training data to avoid overlearning
  - artificial synthesis of unseen training data
Speech Synthesis

1. create *if-then* rules to encode all regularities
2. maintain database of exceptions to rules
3. build a production system to resolve conflicts: very difficult

- e.g. ‘c’ can either be pronounced ‘s’ as in *center*, *icy*, and *city* or a ‘k’ as in *cat* and *crumb*
  - if a ‘c’ appears before an i, e, or y
  - then pronounce it like an ‘s’
  - else pronounce it like a ‘k’
- exception: *celtic*
NETTalk approach

- Rationale: non-intuitive pronunciations can be “discovered” by network whereas they might not be recognizable by human programmer
  
  - Trained on random pick of 16,000 of 20,000 words from Webster's
  - Testing yielded correct response to 90% of 4,000 remaining words
GloveTalk

- translates hand gestures to speech through an adaptive interface
- gestures are mapped continuously to ten control parameters of a parallel formant speech synthesizer
- allows the hand to act as an artificial vocal tract that produces speech in real time
- one subject trained to speak intelligibly with Glove-TalkII
Glove-Talk

S. Fels & G. Hinton

Sam I Am

© S. Fels, 1994