Outline for today

- Neural network learning
- Perceptrons/Linear threshold functions
- Gradient descent

Connectionist Models

Key idea much of "intelligence" must be in the connections

- Neuron switching time ~ .001 second
- Scene recognition time ~ .1 second
- 100 inference steps doesn't seem like enough

→ much parallel computation

Properties:

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically

Some Successful Applications

NETtalk [Sejnowski]

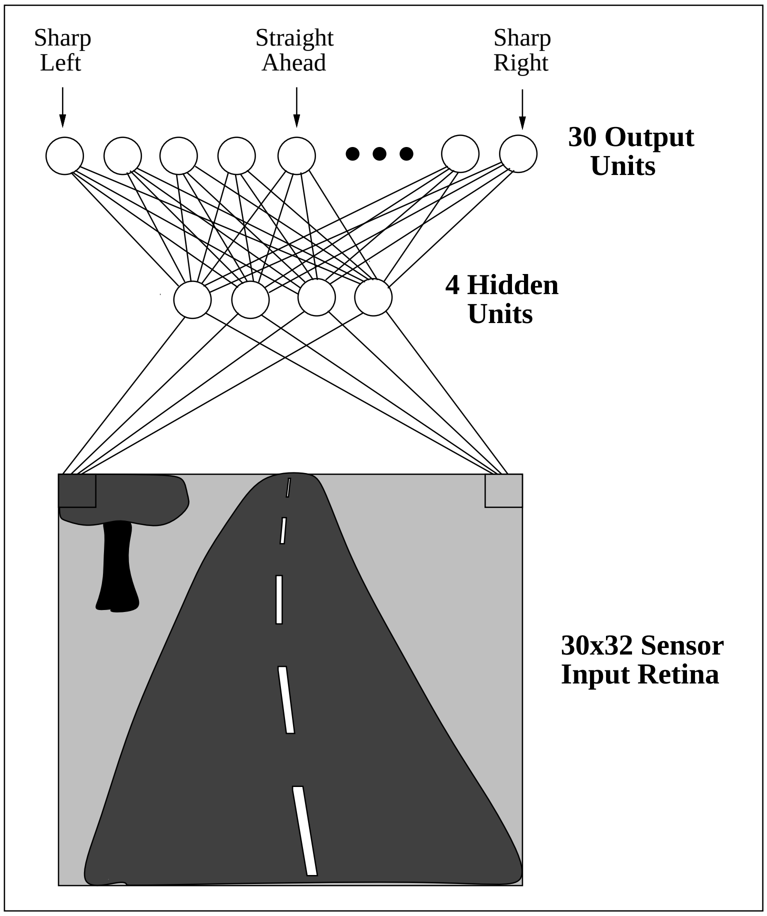
- Inputs: english text
- Output: spoken phonemes

Phoneme recognition [Waibel]

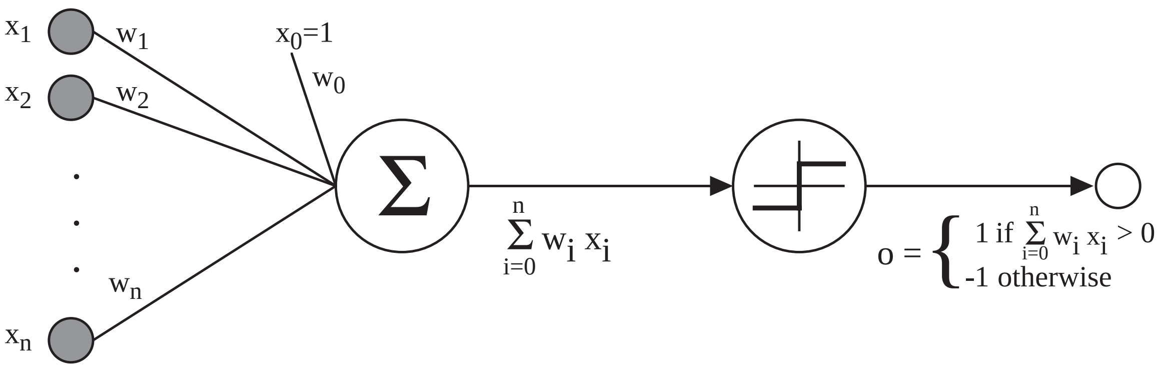
- Inputs: waveform features
- Outputs: b,c,d,...

Robot control [Pomerleau]

- Inputs: perceived features
- Outputs: steering control


Sharp
Left
Straight
Ahead
Sharp
Right
30 Output
Units
4 Hidden
Units
30x32 Sensor
Input Retina

Perceptron



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Decision Surface of a Perceptron

(a) (b)

Represents some useful functions

- What weights represent $g(x_1, x_2) = AND(x_1, x_2)$?

But some functions not representable

- e.g., not linearly separable
- we'll put together networks of these later

What's VC dimension of a perceptron?

Perceptron training rule

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i \tag{1}$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

Make sense?

- What if output o is too big?
  - and x_i positive, negative?

Perceptron training rule

Can prove it will converge

- If training data is linearly separable
- and η sufficiently small

Gradient Descent

To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Let's minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{p \in D} (t_p - o_p)^2$$

Where D is set of training examples

What does E versus $\vec{w}$ look like?

Gradient Descent

Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

Gradient Descent

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_p (t_p - o_p)^2 \\
&= \frac{1}{2}\sum_p \frac{\partial}{\partial w_i}(t_p - o_p)^2 \\
&= \frac{1}{2}\sum_p 2(t_p - o_p)\frac{\partial}{\partial w_i}(t_p - o_p) \\
&= \sum_p (t_p - o_p)\frac{\partial}{\partial w_i}(t_p - \vec{w}\cdot\vec{x_p}) \\
\frac{\partial E}{\partial w_i} &= \sum_p (t_p - o_p)(-x_{i,p}) \qquad (2)
\end{aligned}
$$