

Intro. to machine learning (CSI 5325)

Lecture 7: neural networks

Greg Hamerly

Spring 2008

Some content from Tom Mitchell.

1 Connectionist models

2 Threshold units

3 Gradient descent

Connectionist Models

Consider humans:

- Neuron switching time $\approx .001$ second
 - Number of neurons $\approx 10^{10}$
 - Connections per neuron $\approx 10^{4-5}$
 - Scene recognition time $\approx .1$ second
 - 100 inference steps doesn't seem like enough
- much parallel computation

Properties of artificial neural nets (ANN's):

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

When to Consider Neural Networks

- Input is high-dimensional discrete or real-valued (e.g. raw sensor input)
- Output is discrete or real valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant

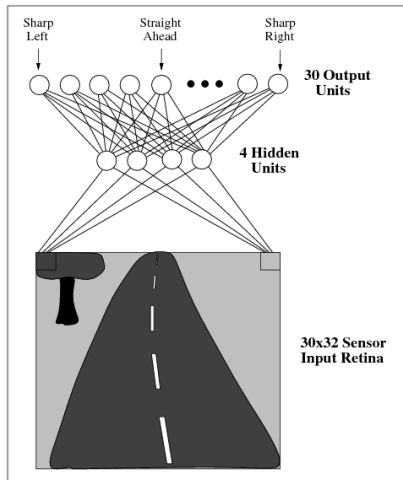
Examples:

- Speech phoneme recognition [Waibel]
- Image classification [Kanade, Baluja, Rowley]
- Emotion identification [Cottrell, Dailey]
- Detecting fraudulent credit card transactions [HNC, Fair Isaac]

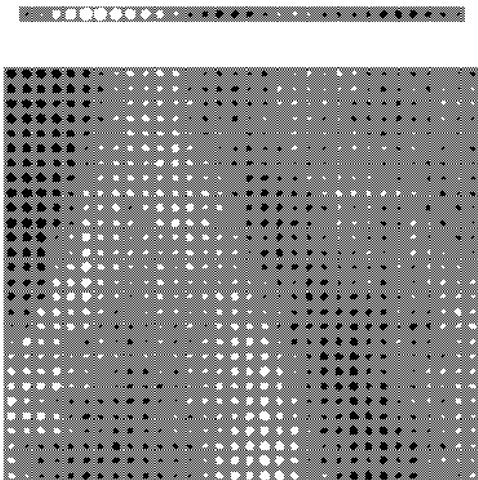
CMU's ALVINN drives 70 mph on highways



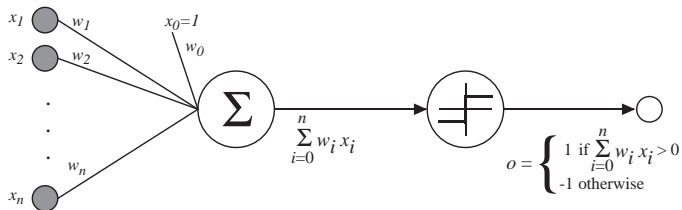
ALVINN's basic network configuration



ALVINN's input/output weights for a hidden unit



Perceptron

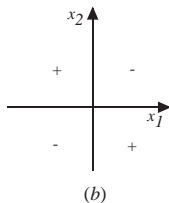
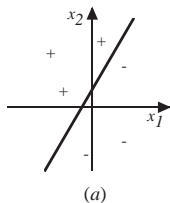


$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

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



Represents some useful functions

- What weights represent $g(x_1, x_2) = \text{AND}(x_1, x_2)$?

But some functions are not representable...

- e.g., not linearly separable
- Therefore, we'll want networks of these...

Visualizing the weight vector and decision surface

Consider the weight vector $\vec{w} = [w_1, w_2]$

What does it mean? What does it look like? (example on the board)

The vector \vec{w} is perpendicular (aka normal) to the decision surface.

\vec{w} points toward the positive half-space.

The bias term w_0 moves the decision surface away from the origin.

Perceptron training rule

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

where

$$\Delta w_i = \eta(t - o(\vec{x}))x_i$$

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

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

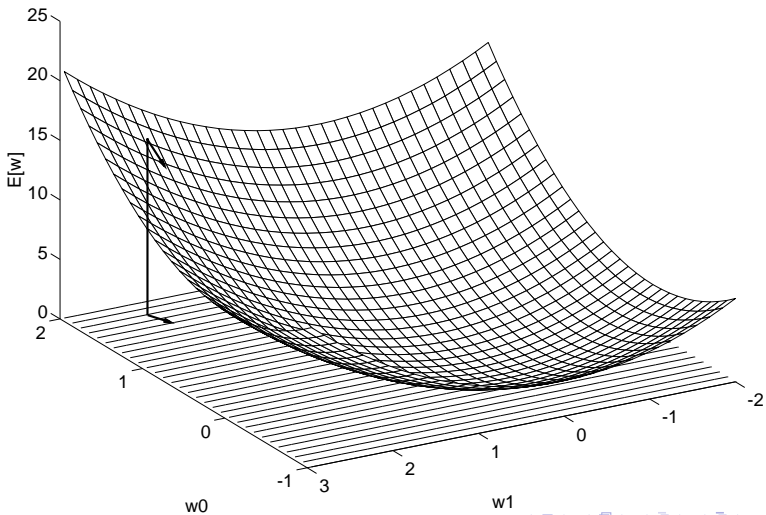
$$o(\vec{x}) = w_0 + w_1x_1 + \cdots + w_nx_n$$

Let's learn w_i 's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{\vec{x} \in D} (t(\vec{x}) - o(\vec{x}))^2$$

Where D is set of training examples.

Gradient Descent



Gradient Descent

Gradient

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

Training rule:

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

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient Descent

$$\begin{aligned}
 \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{\vec{x}} (t(\vec{x}) - o(\vec{x}))^2 \\
 &= \frac{1}{2} \sum_{\vec{x}} \frac{\partial}{\partial w_i} (t(\vec{x}) - o(\vec{x}))^2 \\
 &= \frac{1}{2} \sum_{\vec{x}} 2(t(\vec{x}) - o(\vec{x})) \frac{\partial}{\partial w_i} (t(\vec{x}) - o(\vec{x})) \\
 &= \sum_{\vec{x}} (t(\vec{x}) - o(\vec{x})) \frac{\partial}{\partial w_i} (t(\vec{x}) - \vec{w} \cdot \vec{x}) \\
 &= \sum_{\vec{x}} (t(\vec{x}) - o(\vec{x})) (-x_i)
 \end{aligned}$$