PSYCO 452

Week 2: Nonlinearity, or Making Decisions

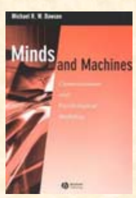
- Building Associations
  - Hebb Learning
  - Delta Learning
- **Making Decisions**
  - –Linear Activation Function
  - –Nonlinear Activation Functions
  - –Perceptrons, Pros and Cons

Course Trajectory

When	What
Weeks 1-3	Basics of three architectures (DAM, perceptron, MLP)
Weeks 4-6	Cognitive science of DAMs and perceptrons
Week 7	Connectionism and Cognitive Psychology
Weeks 8-10	Interpreting MLPs
Weeks 11-13	Case studies (interpretations, applications, architectures)

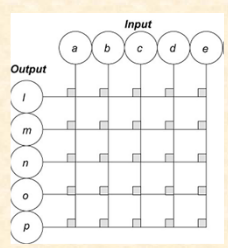
Chapter 9 Discussion

- Questions?
- Important Terms
  - Association
  - Associationism
  - Distributed associative memory
  - Processing unit
  - Modifiable connection
  - Net input function
  - Hebb learning
  - Delta rule
- General ideas are more important than the math, but the math can be useful



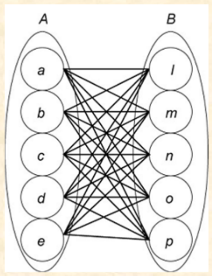
Distributed Associative Memory

- Modern views of neural association involve the strengthening of synapses (both excitatory and inhibitory) as well as the weakening of synapses
- These two processes have been combined to create many interesting models of distributed associative memory



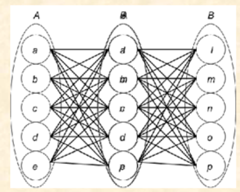
Problems With These Memories

- Hebb rule has many problems
  - Only learns orthogonal patterns
  - Produces error when overtraining
  - Unable to deal with linear dependence
- The delta rule overcomes many of these problems
  - Can deal with some correlated patterns
  - Only modifies weights when errors exist
  - Still cannot deal with linear dependence



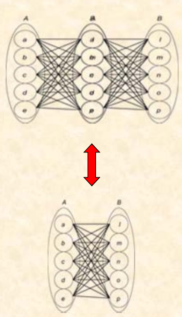
Distributed Associative Memory Sequences

- One possibility for overcoming these problems would be to build a more powerful network
- For example, perhaps a layer of hidden units would serve the purpose
- In this chain, the output of one DAM would be passed along as input to another, so that layers of connections would be exploited



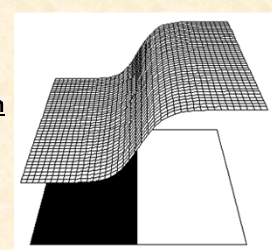
Hidden Unit #fail

- Linear algebra shows that these sequences can be reduced to a memory with one layer of connections
- In other words, the sequences don't add power to a linear system
- $r = W_1(W_2c) = (W_1W_2)c$
- $r = Xc$

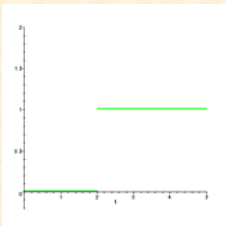


Why Won't Hidden Units Work?

- For layers to add something that can't be removed by linear algebra, a nonlinear transformation of net input must be provided
- In short, we need to use a <u>nonlinear activation function</u> in our processors
- Fortunately, many are available
- An each permits a unit to be interpreted as making a decision

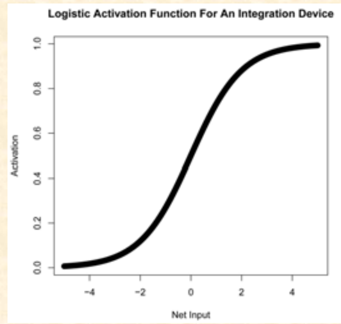


Threshold Device



- Monotonic
- Discontinuous
- Analogous to “all or none” law in neurons
- Used by Rosenblatt in the perceptron

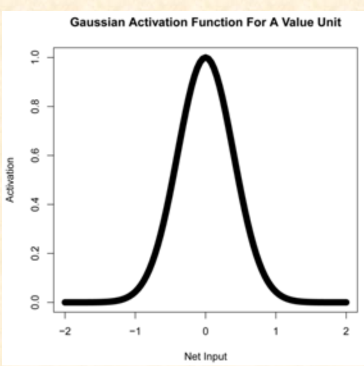
Integration Device



- Continuous
- Monotonic
- Approximates the step function
- Permits calculus to be used to derive step function

$$a_{pi} = \frac{1}{1 + e^{(-net+\theta)}}$$

Value Unit

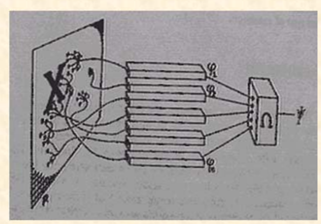


- Continuous
- Nonmonotonic
- Behaves as if it has two thresholds
- Permits calculus to be used to derive step function
- Lots of nice properties as we will see in the course

$$a_{pi} = e^{-\pi(net-\mu)^2}$$

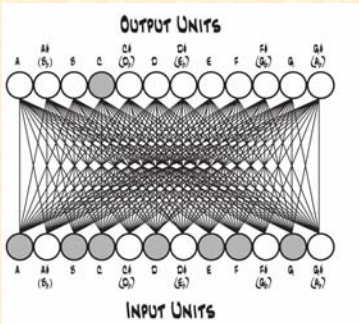
The Perceptron

- A perceptron can be viewed as a distributed memory whose output units use nonlinear activation functions
- It is used to associate an input pattern with a category name
- A perceptron was a trainable pattern classifier!



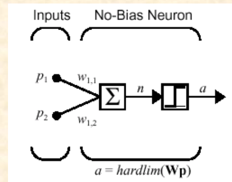
An Example Perceptron

- An example perceptron takes as input the seven notes that are found in a particular major or minor musical scale
- The perceptron then categorizes the scale by identifying the note that serves as the scale's root



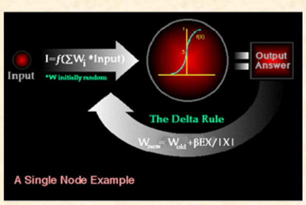
Perceptrons Learn

- Perceptron weights are set via a learning rule
- Assume that the perceptron uses a threshold activation function in its output. Rosenblatt used this rule
- $W_{ij(new)} = W_{ij(old)} + \eta(t_j - o_j)a_i$
- Compare this learning rule to the delta rule for DAM. Why does this rule make sense?
- $\Delta_{t+1} = \eta\,((t - o) \bullet c^T)$



Gradient Descent Rule

- Assume that the perceptron uses a sigmoid activation function
- Calculus can be used to determine a gradient descent rule that moves the network downhill in error space as fast as possible
- The calculus is only possible because the sigmoid is a continuous approximation of the threshold function



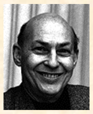
Deriving A Gradient Descent Rule

$$E = \sum E_p = \sum \sum \left(t_{pi} - o_{pi}\right)^2$$

- Define a "least squares" error term
- Use calculus to determine how this error term is changed by a weight change
- Use this information to define the fastest decrease in error possible
- For f(net) = 1/1+exp(-net):
- $W_{ij(new)} = W_{ij(old)} + \eta(t_j - o_j)f'(net)a_i$
- $W_{ij(new)} = W_{ij(old)} + \eta(t_j - o_j)(a_j)(1 - a_j)a_i$

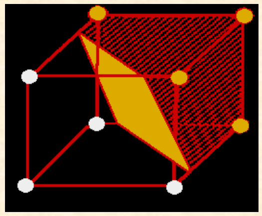
Perceptron Limitations

- In their book *Perceptrons*, Minsky and Papert used mathematics to investigate what perceptrons could and could not learn to do
- They discovered some interesting, and serious, limitations to the capabilities of perceptrons
- The result was an extreme decline in neural network research



Pattern Recognition

- Networks are frequently used to classify patterns
- They carve a pattern space into decision regions
- Patterns are classified according to these decision regions

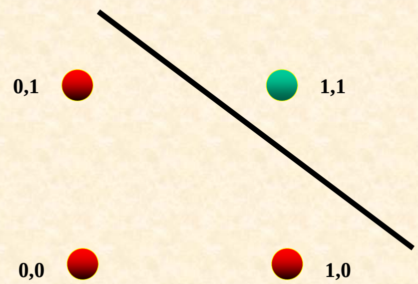


The AND Problem

INPUT 1	INPUT 2	OUTPUT
F	F	F
T	F	F
F	T	F
T	T	T

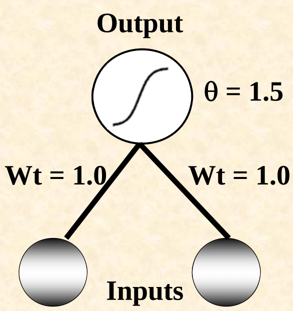
INPUT 1	INPUT 2	OUTPUT
0	0	0
1	0	0
0	1	0
1	1	1

The AND Pattern Space



An AND Network

- A single, straight cut through the pattern space solves the AND problem
- This means this problem is ***linearly separable***
- The networks of Old Connectionism could learn to solve such problems

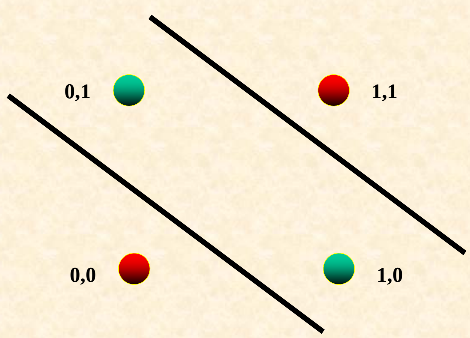


The XOR Problem

INPUT 1	INPUT 2	OUTPUT
0	0	0
1	0	1
0	1	1
1	1	0

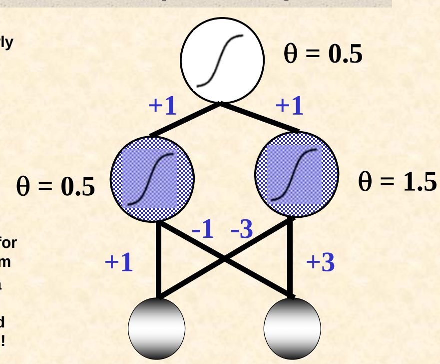
INPUT 1	INPUT 2	OUTPUT
F	F	F
T	F	T
F	T	T
T	T	F

The XOR Pattern Space

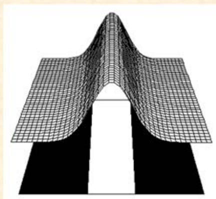


Linear Nonseparability

- XOR is not a linearly separable problem
- This is because more than 1 cut is required
- As a result, Old Connectionism could not train networks to deal with this problem
- XOR is a problem for New Connectionism
- Or, a problem for a perceptron with a more sophisticated activation function!



Value Unit



- Named after Ballard (1986)
- Gaussian activation function
- $G(net_{pj}) = \exp[-\pi(net_{pj} - \mu_j)^2]$

An Elaborated Error Term

- Standard error term in gradient descent rule

$$E = \sum E_p = \sum \sum (t_{pi} - o_{pi})^2$$

- Dawson & Schoplocher error term

$$E = \sum E_p = \sum \sum (t_{pi} - o_{pi})^2 + \sum \sum t_{pi} (net_{pi} - \mu_i)^2$$

- This second term keeps some of the patterns in the middle of the distribution!

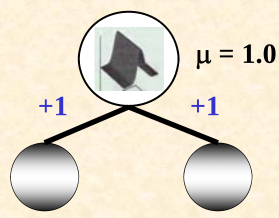
A New Learning Rule

- For $G(net_{pj}) = \exp[-\pi(net_{pj})^2]$
- $W_{ij(new)} = W_{ij(old)} + \eta(t_j - o_j)G'(net)a_i + \eta(t_j * net)G'(net)a_i$
- Using the Gaussian, and the Rumelhart Hinton & Williams chain rule procedure, one can derive a learning rule for value units:

$$\Delta w_{ij} = \eta(\delta_{pi} - \varepsilon_{pi})\, a_{pj}$$

- Essentially the same as the gradient descent rule, with the exception of an elaborated (two component) error term

Another XOR Network



Perceptron Performance

- Let's use a perceptron program to explore some of the issues raised this lecture
  - Ability to perform beyond DAM
  - Ability to deal with most of Boolean logic
  - Integration device vs. value unit power in terms of small, linearly nonseparable problems
- Limitations still exist – we will need to add layers of nonlinear processors to deal with them – and will talk about how to do this later in this course