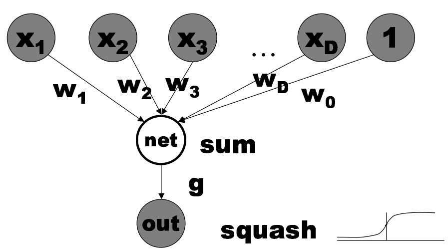
Backpropagation

Introduction to
Artificial Intelligence
COS302
Michael L. Littman
Fall 2001

Administration

Questions, concerns?

Classification Percept.



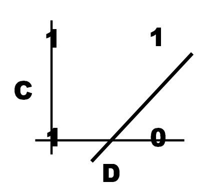
Perceptrons

- Recall that the squashing function makes the output look more like bits: 0 or 1 decisions.
- What if we give it inputs that are also bits?

A Boolean Function

A	B	C	D	E	F	G	out
1	0	1	0	1	0	1	0
0	1	1	0	0	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1
0	0	1	1	0	0	0	1
1	1	1	0	1	0	1	0
0	1	0	1	0	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	0

Think Graphically



Can perceptron learn this?

Ands and Ors

$out(\underline{x}) = g(sum_k\ w_k\ x_k)$

How can we set the weights to represent $(v_1)(v_2)(\sim v_7)$? AND

w_i=0, except

w_1=10, w_2=10, w_7=-10, w_0=-15 (5-max)

How about $\sim v_3 + v_4 + \sim v_8$? OR

w_i=0, except

w_1=-10, w_2=10, w_7=-10, w_0=15 (-5-min)

Majority

Are at least half the bits on?

Set all weights to 1, w_0 to –n/2.

A	B	C	D	E	F	G	out
1	0	1	0	1	0	1	1
0	1	1	0	0	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	0
1	1	1	0	1	0	1	1
0	1	0	1	0	0	1	0
1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1

Representation size using decision tree?

Sweet Sixteen?

ab	(~a)+(~b)
a(~b)	(~a)+b
(~a)b	a+(~b)
(~a)(~b)	a+b
a	~a
b	~b
1	0
a = b	a exclusive-or b (a ≠ b)

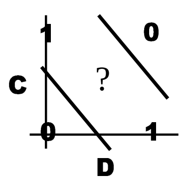
XOR Constraints

A	B	out	
0	0	0	$g(w_0) < 1/2$
0	1	1	$g(w_B+w_0) > 1/2$
1	0	1	$g(w_A+w_0) > 1/2$
1	1	0	$g(w_A+w_B+w_0) < 1/2$

$w_0 < 0$, $w_A+w_0>0$, $w_B+w_0>0$,

$w_A+w_B+2\ w_0>0$, $0 < w_A+w_B+w_0 < 0$

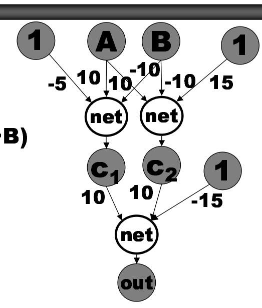
Linearly Separable



XOR problematic

How Represent XOR?

A xor B
= (A+B)(~A+~B)



Requiem for a Perceptron

Rosenblatt proved that a perceptron will learn any linearly separable function.

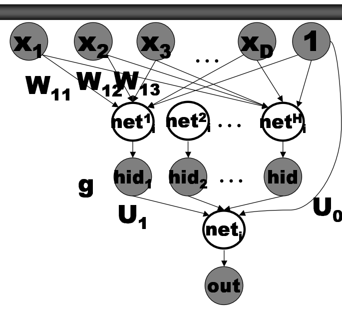
Minsky and Papert (1969) in *Perceptrons*: "there is no reason to suppose that any of the virtues carry over to the many-layered version."

Backpropagation

Bryson and Ho (1969, same year) described a training procedure for multilayer networks. Went unnoticed.

Multiply rediscovered in the 1980s.

Multilayer Net



Multiple Outputs

Makes no difference for the perceptron.

Add more outputs off the hidden layer in the multilayer case.

Output Function

$out_i(\underline{x}) = g(sum_j\ U_{ji}\ g(sum_k\ W_{kj}\ x_k))$

H: number of "hidden" nodes

Also:

- Use more than one hidden layer
- Use direct input-output weights

How Train?

Find a set of weights U, W

that minimize

$sum_{(\underline{x},y)}\ sum_i\ (y_i - out_i(\underline{x}))^2$

using gradient descent.

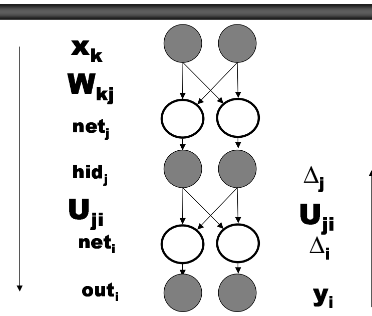
Incremental version (vs. batch):

Move weights a small amount for each training example

Updating Weights

1. Feed-forward to hidden:
   $net_j = sum_k\ W_{kj}\ x_k;\ hid_j = g(net_j)$
2. Feed-forward to output:
   $net_i = sum_j\ U_{ji}\ hid_j;\ out_i = g(net_i)$
3. Update output weights:
   $\Delta_i = g'(net_i)\ (y_i - out_i);\ U_{ji} \mathrel{+}= \eta\ hid_j\ \Delta_i$
4. Update hidden weights:
   $\Delta_j = g'(net_j)\ sum_i\ U_{jj}\ \Delta_i;\ W_{kj} \mathrel{+}= \eta\ x_k\ \Delta_j$

Multilayer Net (schema)



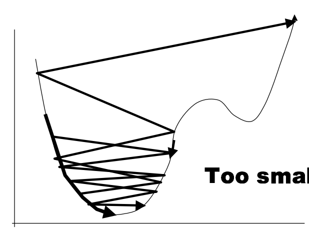
Does it Work?

Sort of: Lots of practical applications, lots of people play with it. Fun.

However, can fall prey to the standard problems with local search…

NP-hard to train a 3-node net.

Step Size Issues



Too small? Too big?

Representation Issues

Any continuous function can be represented by a one hidden layer net with sufficient hidden nodes.

Any function at all can be represented by a two hidden layer net with a sufficient number of hidden nodes.

What's the downside for learning?

Generalization Issues

Pruning weights:
"optimal brain damage"

Cross validation

Much, much more to this. Take a class on machine learning.

What to Learn

- Representing logical functions using sigmoid units
- Majority (net vs. decision tree)
- XOR is not linearly separable
- Adding layers adds expressibility
- Backprop is gradient descent

Homework 10 (due 12/12)

1. Describe a procedure for converting a Boolean formula in CNF (n variables, m clauses) into an equivalent network? How many hidden units does it have?
2. More soon