# Linear Units, Perceptrons, and the LMS Algorithm

15-496/782: Artificial Neural Networks
David S. Touretzky

Spring 2004

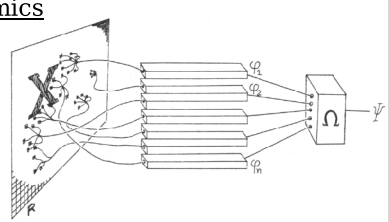


# Linear Units and Perceptrons

Perceptrons were the original "neural nets".

Rosenblatt (1962)
Principles of Neural Dynamics

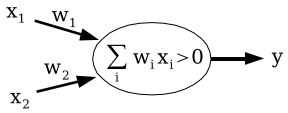
Minsky & Papert (1969)
Perceptrons



Bernie Widrow:
weather prediction
adaptive equalization in modems

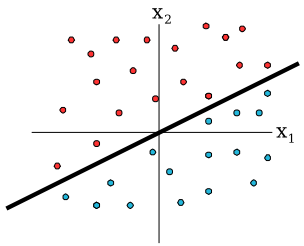


# Peceptrons Are Linear Classifiers




$$net = \sum_i w_i x_i$$

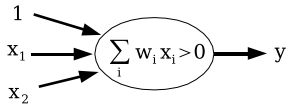
$$y = \begin{cases} 1 & \text{if } net > 0 \\ 0 & \text{otherwise} \end{cases}$$




$\vec{w} = [-1, 3]$

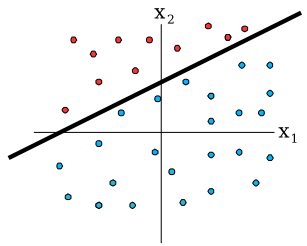


# Decision Boundary Off the Origin?




Add a bias term $w_0$:

$w_0 + w_1 x_1 + w_2 x_2 > 0$




$\vec{w} = [-6, -1, 3]$

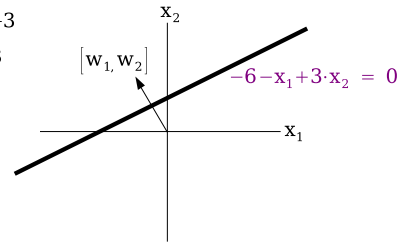
## The Decision Boundary is Always Perpendicular to the Weight Vector

$\vec{w} = [-6,\ -1,\ 3]$

slope of weight vector $= w_2/w_1 = -3$

slope of decision boundary $= 1/3$

(If a line has slope m, the perpendicular has slope -1/m.)




Scaling the weight vector has no effect on the decision boundary!



## Make the Weight Vector Touch the Decision Boundary

Let $\vec{w} = [-2, 3, 4]$

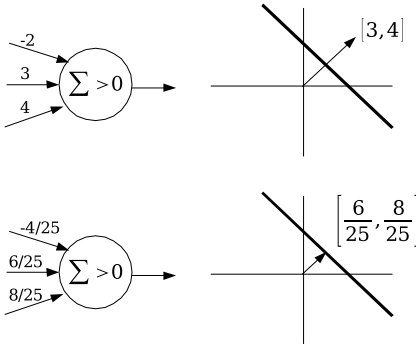
Let $\vec{v} = h\begin{bmatrix}3\\4\end{bmatrix}$

$w_0 + w_1 v_1 + w_2 v_2 = 0$

$w_0 + w_1^2 h + w_2^2 h = 0$

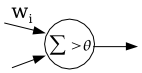
$h = \frac{-w_0}{w_1^2 + w_2^2} = \frac{2}{25}$

$\vec{v} = \left[\frac{6}{25}, \frac{8}{25}\right]$

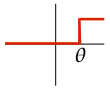





## Thresholds vs. Biases

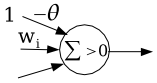



$$y = \begin{cases} 1 & \text{if } \sum w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$




Learning rules adjust both $\vec{w}$ and $\theta$

Simpler solution: $w_0 = -\theta$




A bias term of $-\theta$ is equivalent to a threshold of $\theta$



## Perceptron Learning Rule

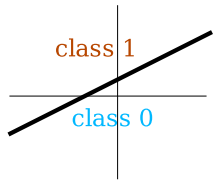
Initialize $\vec{w} \leftarrow 0$

For each $\vec{x}_i$ in training set:

$net = \vec{x}_i \cdot \vec{w}$

$$y = \begin{cases} 1 & \text{if } net > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\vec{w} \leftarrow \begin{cases} \vec{w} & \text{if } y = d_i \\ \vec{w} + \vec{x}_i & \text{if } y < d_i \\ \vec{w} - \vec{x}_i & \text{if } y > d_i \end{cases}$$



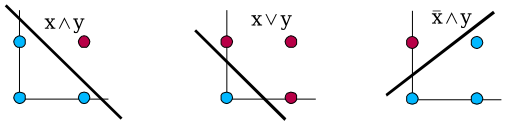

Repeat until all $\vec{x}_i$ classified correctly.

## How to Run the Matlab Demos

- matlab
- cd /afs/cs/academic/class/15782-s04/matlab/perceptron
- ls
- perceptron



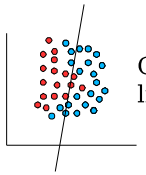
## Learning Boolean Functions



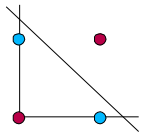

Are all Boolean functions learnable?



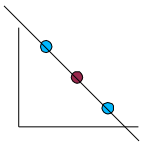
## Some Problems Aren't Linearly Separable



Convex classes aren't linearly separable



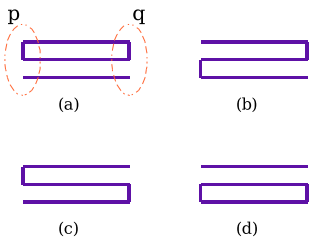
XOR



Not in "general position"

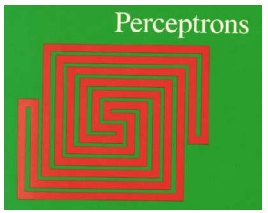


## Perceptrons Can't Compute XOR




| | p | q | pXORq |
|---|---|---|---|
| a | 0 | 0 | 0 |
| b | 1 | 0 | 1 |
| c | 0 | 1 | 1 |
| d | 1 | 1 | 0 |

Minsky & Papert:
Perceptrons can't compute "connectedness".

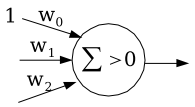


# Prove That Perceptrons Can't Compute XOR

| | $x_1$ | $x_2$ | d |
|---|---|---|---|
| 1. | 0 | 0 | 0 |
| 2. | 1 | 0 | 1 |
| 3. | 0 | 1 | 1 |
| 4. | 1 | 1 | 0 |

5. $w_0 \leq 0$ (by 1)
6. $-w_1 < w_0$ (by 3)
7. $-w_2 < w_0$ (by 2)
8. $w_1 + w_2 < -w_0$ (by 4)
9. $0 < w_0$ (add 6, 7
10. $w_0 > 0$ (by 9)

Lines 5 and 10 conflict




$w_0 + w_1 x_1 + w_2 x_2 >$

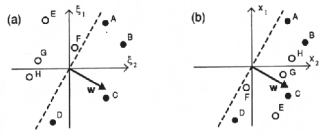


# Let's Use +1/-1 Outputs

$$y = \mathrm{sgn}(\mathrm{net}) = \begin{cases} +1 & \text{if net} > 0 \\ -1 & \text{otherwise} \end{cases}$$

Let $\phi^n$ = input pattern n
Let $t^n$ = class of pattern n (-1 or +1)

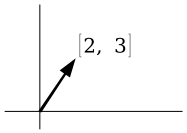
If a problem is linearly separable, all $\phi^n t^n$ lie on the same side of the decision boundary.




$\phi$ space $\quad$ $\phi \cdot t$ space



# Vectors



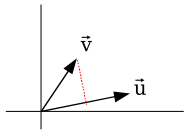

$\vec{v} = [2,\ 3]$

$\|\vec{v}\| = \sqrt{2^2+3^2} = \sqrt{13}$

$\frac{\vec{v}}{\|\vec{v}\|}$ = unit vector in same direction as $\vec{v}$



# Dot Product



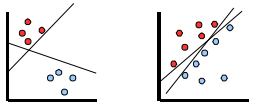

$\vec{u} \cdot \vec{v} = \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos\theta$

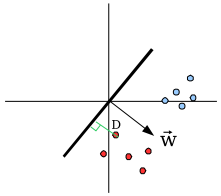
$\vec{u} = [u_1\ u_2\ u_3]$
$\vec{v} = [v_1\ v_2\ v_3]$

$\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 + u_3 v_3$

If $\vec{u}$ is a unit vector, then $\vec{u} \cdot \vec{v}$ is the length of the projection of $\vec{v}$ along $\vec{u}$.

# Easy vs. Hard Problems




$$D(\bar{w}) = \frac{1}{\|\vec{w}\|} \min_i (\vec{w} \cdot \phi^i t^i)$$

$$D_{max} = \max_{\vec{w}} D(\vec{w})$$

Large $D_{max}$ → easy problem.
$D_{max} < 0$ → not linearly separable.

For AND, $D_{max} = \frac{1}{\sqrt{17}}$. For XOR, $D_{max} = \frac{-1}{\sqrt{3}}$



# Perceptron ConvergenceTheorem
## Rosenblatt (1962)

This theorem is very famous.

The version that follows is from Bishop (1995), based on Hertz, Krogh, and Palmer (1991).

Theorem:

If a problem is linearly separable, then
a perceptron will learn it
in a finite number of steps.



# Proof of the Theorem (1)

Assume a vector $\vec{w}$ exists that correctly classifies all points.
Then $\vec{w} \cdot (\phi^n t^n) > 0$ for all n.

At each step of the algorithm:
$\bar{w}^{(\tau)}$ = weights at step $\tau$
$\phi^n$ is the misclassified vector at the current step
$\bar{w}^{(\tau+1)} \leftarrow \bar{w}^{(\tau)} + \phi^n t^n$

Suppose $\phi^n$ has been misclassified $\tau^n$ times so far.
Total misclassifications $\tau = \sum_n \tau^n$
Therefore $\bar{w}^{(\tau)} = \sum_n \tau^n \phi^n t^n$
(assuming $\bar{w}^{(0)} = 0$)



# Proof of the Theorem (2)

Find a lower bound on the growth rate of $\hat{w} \cdot \bar{w}^{(\tau)}$.

$$\hat{w} \cdot \bar{w}^{(\tau)} = \sum_n \tau^n \hat{w} \cdot \phi^n \cdot t^n$$
$$\geqslant \tau \min_n \left( \hat{w} \cdot \phi^n t^n \right)$$

So $\hat{w} \cdot \bar{w}^{(\tau)}$ is bounded from below by
a function that grows linearly in $\tau$.

If the algorithm runs forever, $\hat{w} \cdot \bar{w}^{(\tau)}$ diverges.

# Proof of the Theorem (3)

Find an upper bound on the growth rate of $\bar{w}^{(\tau)}$.

$$\bar{w}^{(\tau+1)} = \bar{w}^{(\tau)} + \phi^n \cdot t^n$$

$$\|\bar{w}^{(\tau+1)}\|^2 = \|\bar{w}^{(\tau)}\|^2 + \|\phi^n\|^2 (\tau^n)^2 + 2\,\bar{w}^{(\tau)} \cdot \phi^n t^n$$

$$\leqslant \|\bar{w}^{(\tau)}\|^2 + \|\phi^n\|^2 (\tau^n)^2$$

because $\bar{w}^{(\tau)} \cdot \phi^n t^n < 0$ since $\phi^n$ was misclassified.

Note: $(t^n)^2 = 1$ since $t^n = \pm 1$

Let $\|\phi\|_{max} = \max_n \|\phi^n\|$

Then $\|\bar{w}^{((\tau)+1)}\|^2 - \|\bar{w}^{(\tau)}\|^2 \leqslant \|\phi\|^2_{max}$

Since $\|\bar{w}^{(0)}\| = 0$, after $\tau$ weight updates we have:

$$\|\bar{w}^{(\tau)}\|^2 \leqslant \tau \|\phi\|^2_{max}$$



# Proof of the Theorem (4)

Show the bounds must cross.

$$\|\bar{w}^{(\tau)}\|^2 \leqslant \tau \|\phi\|^2_{max}$$

So $\|\bar{w}^{(\tau)}\|$ grows no faster than $\sqrt{\tau}$

But $\vec{w} \cdot \bar{w}^{(\tau)}$ has a lower bound that is linear in $\tau$:

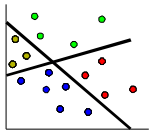
$$\vec{w} \cdot \bar{w}^{(\tau)} \geqslant \tau \min_n \left( \hat{w} \cdot \phi^n t^n \right)$$

The bounds would eventually cross if $\tau$ got large enough. Hence, $\tau$ must bounded, meaning we achieve correct classification of all points in a finite number of steps.

QED.

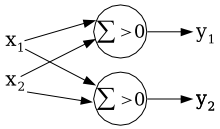


# More Than 2 Classes



Classes:

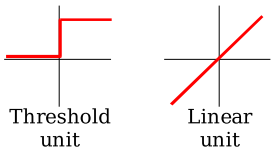
- 0 0
- 0 1
- 1 0
- 1 1

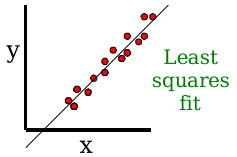


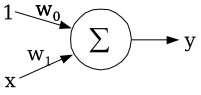

The two neurons learn independently.



# Linear Units: Function Approximators




$$y = \sum_i w_i x_i$$

# The LMS (Least Mean Squares) Learning Algorithm

Define total sum-squared error over the training set:

$$E = \frac{1}{2}\sum_j (d_j - y_j)^2$$

Do gradient descent in the error E:

$$\frac{\partial E}{\partial y} = y - d \qquad \frac{\partial y}{\partial w_i} = \frac{\partial}{\partial w_i}\sum_i w_i x_i = x_i$$

Chain rule:

$$\frac{\partial E}{\partial w_i} = (y - d)x_i$$

Gradient descent in E:

$$\Delta w_i = -\eta(y - d)x_i \qquad \eta \text{ is a learning rate constant}$$

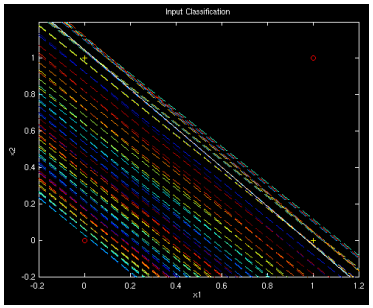


# LMS Convergence

If the learning rate $\eta$ is small enough, LMS will always converge.

When $|E(t+1) - E(t)| < 0.001$, stop.

What about XOR?

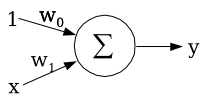




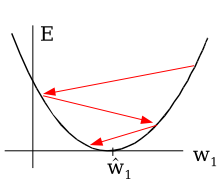

# Why LMS Can Blow Up

Error is quadratic in $\vec{w}$.

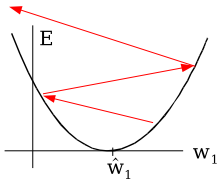
$$E = \frac{1}{2}\sum_i (d_i - y_i)^2$$




So the error surface forms a bowl.
The one-dimensional projection is a parabola.




Small $\eta$

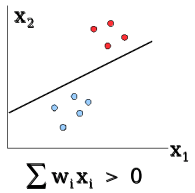



Large $\eta$

See bowl and parabolas demos.

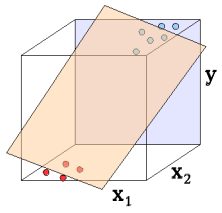


# Classification vs. Mapping




$$\sum w_i x_i > 0$$

Train with perceptron algorithm.



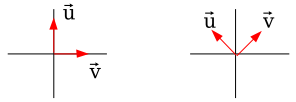

$$y = \sum w_i x_i$$

Train with LMS.

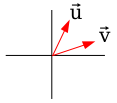
There are some pathological cases where LMS won't classify all points correctly, but the perceptron algorithm will.

## Orthogonality and Linear Independence

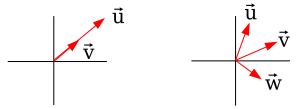
Orthogonal:




Linearly independent:



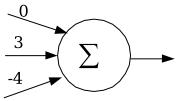

Not linearly independent:






## LMS Works Best with Orthogonal Input Patterns

$$\text{Patterns} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{Desired} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$




Even if not orthogonal, LMS will find a perfect solution as long as the patterns are linearly independent.

If not linearly independent, patterns interfere with each other and total sum-squared error cannot reach 0.



## The Rescorla-Wagner Model of Animal Learning

UCS = shock

UCR = jumps; tries to escape

$CS_1$ = light

$CS_2$ = tone

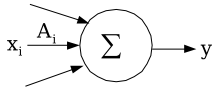
CR = fear response: freezing, shivering, inhibition of drinking



## Rescorla-Wagner is a Linear Model

$A_i$ = Associative strength between $CS_i$ and UCS.

$x_i$ = presence of $CS_i$: [0,1]




Conditioned Response = $y = \sum_i A_i x_i$

## Conditioning Experiments

Simple conditioning:

Train: light --> UCS
Tests: light --> CR
tone --> nothing

Conditioned inhibition:

Train: light --> UCS
light + tone --> no UCS
Tests: light --> CR
light + tone --> no CR
"summation test"
"retardation test"



## Rescorla-Wagner Learning Rule

The Rescorla-Wagner learning rule is the LMS rule, also called the Widrow-Hoff rule or the delta rule.

Problem: Rescorla-Wagner can't learn XOR.
But rats can.

Solution: Use a conjunctive unit as a third input.
(But this is a hack.)