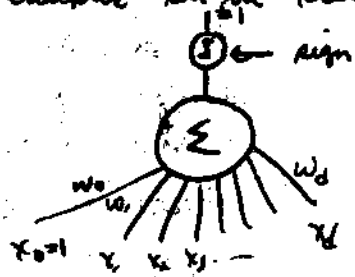
9/17/90

Outline
- linear threshold fns.
- training using lin. prog.
- " " Perceptron alg.
- Perceptron Conv. alg.
- Lower bounds
- Discussion

- instance ⟷ $(x_1, x_2 \ldots x_d) \in \mathbb{R}^d$
- w.l.o.g. ~~$x_0 = 1$~~ $x_0 = 1$ (could be an extra value that is added, etc.)
- learning from examples (positive & negative - i.e. labelled examples)

prior knowledge: examples can be classified by a linear threshold function



LTF:

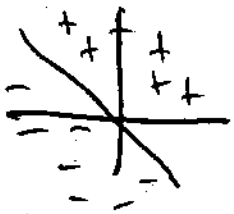
$w = (w_0 \ldots w_d)$

$x = (x_0 \ldots x_d)$

$w \cdot x = 0$ - hyperplane in d dimen

↑ graphically

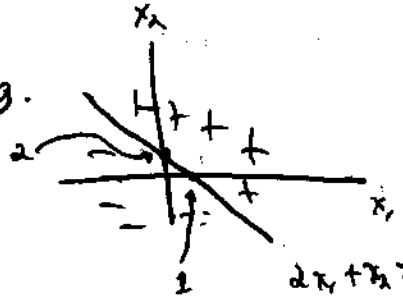
↑ algebraically



$w \cdot x > 0$ positive example

$w \cdot x < 0$ neg. exam.

E.g.



$2x_1 + x_2 > 2 = 2 \cdot x_0$

$x_0 = 1$

or $(2, 1, -2) \cdot \vec{x} > 0$

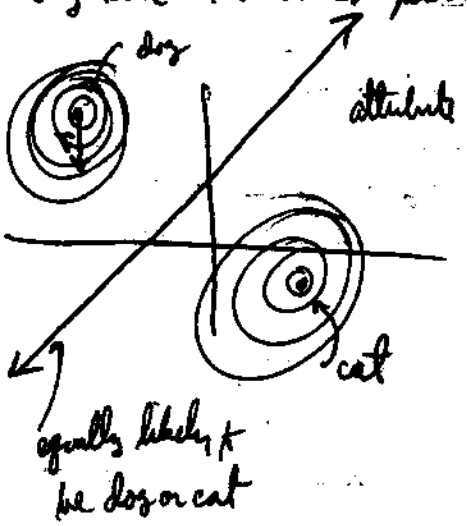
$(-2, 2, 1) \cdot \vec{x} > 0$

X_+ - positive training examples
X_- - negative training examples

Def (X_+, X_-) are *linearly separable* if $\exists \omega$: $\omega \cdot x > 0 \quad \forall x \in X_+$
$\omega \cdot x < 0 \quad \forall x \in X_-$

Why is this stuff interesting?

1) neuron is something like this ... (abstract model of a neuron)
2) ??
3) they are sometimes provably optimal

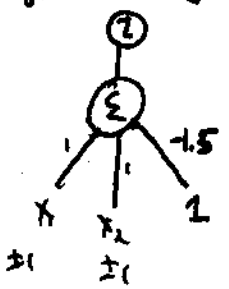


- examples are given by multivariate distribution from prototypical dog or cat - i.e. likelihood of dog (or cat) is $e^{-r^2/2}$ where r is distance from prototypical dog (or cat)

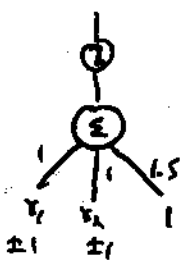
↳ this linear separator does as well as you can do.

4) can represent great many things: AND, NOT, OR gates, etc.

AND



OR

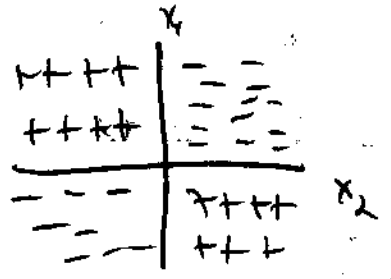


NOT

Note: can't do XOR or parity (can easily do majority)

XOR

x_1 \ x_2	+1	−1
+1	−1	+1
−1	+1	−1



↑ can draw linear separator

How many boolean fns. on d variables (inputs) are threshold fns?

→ $2^{\Theta(d^2)}$

How many total? → 2^{2^d}

note $\frac{2^{\Theta(d^2)}}{2^{2^d}} \to 0 \quad d \to \infty$ (so not as many asymptotically)

Learning Algorithm

① Batch Alg. - X_+, X_- lin. sep.

- find w s.t. $w \cdot x = 0$ separates X_+ from X_+

now, w.l.o.g. assume $X_- = \emptyset$ (add $-x$ to X_+ $\forall x \in X_-$)

if $\vec{w} \cdot \vec{x} < 0$ $\forall \vec{x} \in X_-$ then $\vec{w} \cdot (-\vec{x}) > 0$ $\forall \vec{x} \in X_-$

⇒ add $-x$ to X_+ $\forall x \in X_-$

now, find $w \cdot x > 0$ $\forall x \in X_+$

- this is just linear programming

∴ can solve using favorite LP algorithm (3 alg which can be done in poly time)

② Incremental (on-line) alg.

Perceptron alg. (Rosenblatt) (assume again that all examples are in X_+)

$w = 0$
repeat
  for each training example
    if $w \cdot x > 0$ do nothing
    if $w \cdot x \leq 0$
      $w \leftarrow w + x$
until $w \cdot x > 0 \ \forall x \in X_+$

(a batch alg.)

Why does this work?

$w' \leftarrow w + x$

so, $w' \cdot x = w \cdot x + \underbrace{x \cdot x}_{>0}$

↳ will be closer to 0 or positive

Convergence proof

Thm: Assume $\exists v$ s.t. $v \cdot x > \delta > 0 \ \forall x \in X_+$ (δ = margin of hyperplane)
$\exists M$ s.t. $|x|^2 \leq M \ \forall x \in X_+$

then Perceptron Alg converges after finite # of updates.

Pf: ① After k updates, $w = x^{(1)} + \cdots + x^{(k)}$ ($x^{(i)}$ = i^{th} mistake example)

$v \cdot w = v \cdot x^{(1)} + \cdots + v \cdot x^{(k)}$
$\geq k \cdot \delta$

$(\sum a_i b_i)^2 \leq (\sum a_i^2)(\sum b_i^2)$
$\Rightarrow (v \cdot w)^2 \leq |v|^2 |w|^2$

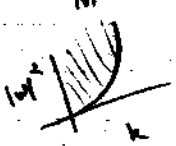
Cauchy Swartz $\Rightarrow |w|^2 \geq \frac{(v \cdot w)^2}{|v|^2} \geq \frac{(k\delta)^2}{|v|^2} = k^2 \frac{\delta^2}{|v|^2}$ — growing at least quadratically w/ updates

$\frac{v \cdot w}{|v||w|} = \cos\theta$
$\frac{(v \cdot w)^2}{|v|^2 |w|^2} = \cos^2\theta \leq 1$

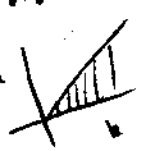
② $w' \cdot w' = |w'|^2 = |w|^2 + \underbrace{2w \cdot x}_{<0} + |x|^2$ (each update)
$= |w + x|^2 =$

$\Rightarrow |w'|^2 - |w|^2 \leq |x|^2$

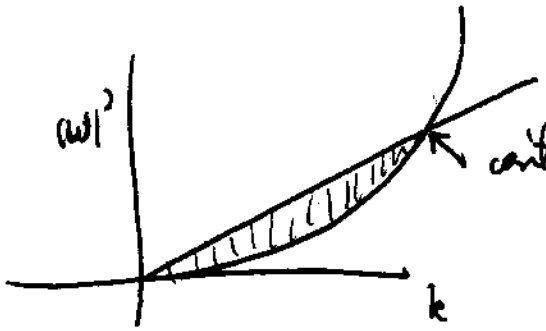
$\Rightarrow$ after k updates, $|w|^2 \leq k \cdot M$ — growing at most linearly



when does $k^2 \frac{\delta^2}{|v|^2} > kM$? $\Rightarrow k > \frac{M|v|^2}{\delta^2}$ ✗ so, $k < \frac{M|v|^2}{\delta^2}$



So,



can't go past this crossover pt, or get contradict

$k M = |w|^2 = \frac{k^2 \delta^2}{|w|^2}$

$$kM = (k\delta)^2 / |w|^2$$

$$\Rightarrow \quad k \leq \frac{M \cdot |w|^2}{\delta^2}$$

In fact this alg. is exponential in d in worst case:

Thm (Minsky)

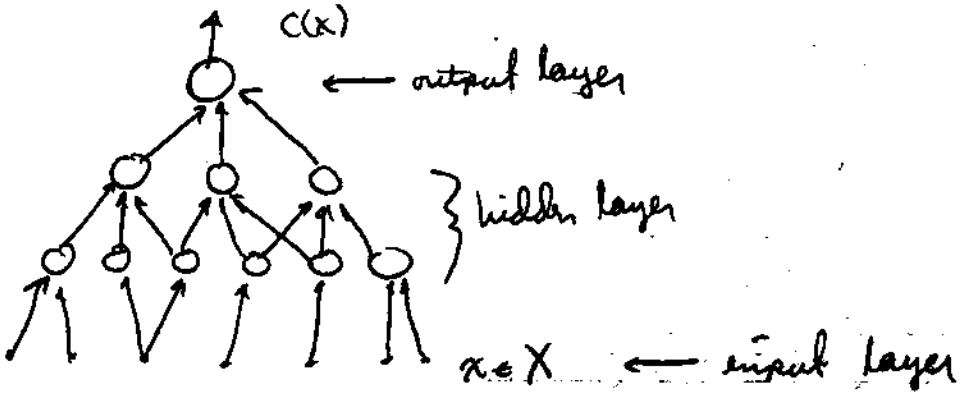
$(\ldots(((((x_1 \wedge x_2) \vee x_3) \wedge x_4) \vee x_5) \wedge \ldots \text{—} \ x_d)$

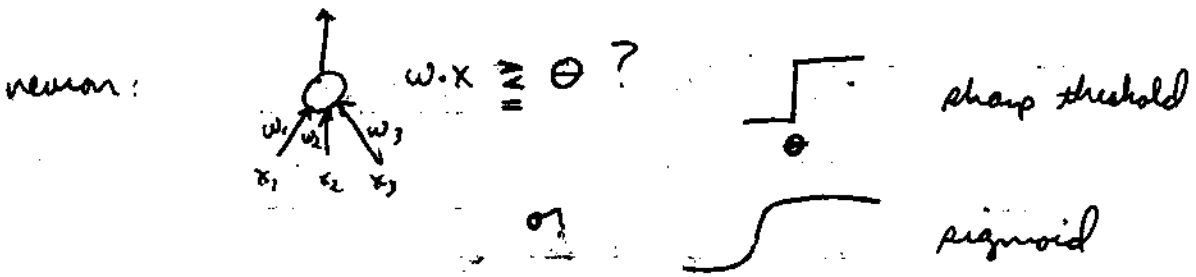
claim: ↑ is a threshold function, but weights grow exponentially in d.
(turn out to be Fibonacci #'s)

$\Rightarrow$ Perceptron alg. can take exp. time.

11/4/92

Neural nets





Network w/ sigmoid is a continuous function of inputs & weights.

Train using gradient descent... (back propagation)

Error we'd like to minimize: M.S.E. $E = \frac{1}{m} \sum_{x \in S} (f(x) - C(x))^2$

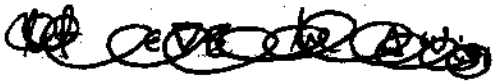
$\nabla E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_k} \right)$ ... (change in error as we tweek weights)

$w_0 \in \mathbb{R}^k$ initial weights (may be random)

$$\vec{w}_{i+1} = \vec{w}_i - \varepsilon \nabla E$$

(ε ← learning rate)

sometimes we <u>momentum</u> term...



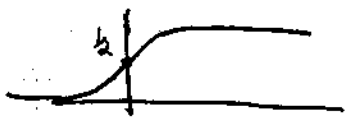
now $\Delta w_{i+1} = \left(\alpha \Delta w_i + (1-\alpha)\,\varepsilon(-\nabla E)\right)$

& $w_{i+1} = w_i + \Delta w_{i+1}$

---

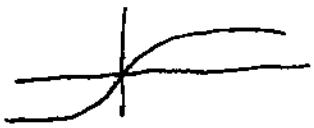
types of sigmoids:

1) logistic function $z = \dfrac{1}{1+e^{-x}}$



$$\frac{dz}{dx} = z(1-z) = \frac{-e^{-x}}{(1+e^{-x})^2}$$

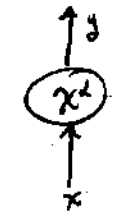
2) hyperbolic tangent $z = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$



$$\frac{dz}{dx} = \operatorname{sech}^2(x) = \frac{4}{(e^x+e^{-x})^2}$$

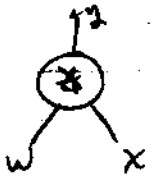
Chain rule (Back-propagation)

Let $\delta_z = \frac{\partial E}{\partial z}$ (notation)



e.g. $\delta_x = \delta_y \cdot 2x$ $\left(\text{e.g } \frac{\partial E}{\partial x} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x}\right)$ ⋮

$\delta_x = \delta_y \cdot y(1-y)$ (logistic)

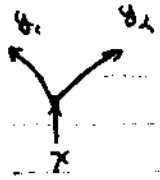


$\delta_w = \delta_y \cdot x$

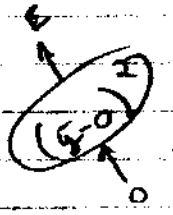
$\delta_x = \delta_y \cdot w$

$\delta_{x_1} = \delta_y$

$\delta_{x_2} = \delta_y$



$\delta_x = \delta_{y_1} + \delta_{y_2}$



$\frac{\partial E}{\partial 0} = 2(y=0)(-1)$
$= 2(0-y)$

y_1 y_2
| |
x x
$\delta_x = \delta_{y_1}$ $\delta_x = \delta_{y_2}$ — add to get cumulative effect

NetTalk (text ⟹ speech)

∴ actually, text ⟹ phonemes

— see paper

11/9/92

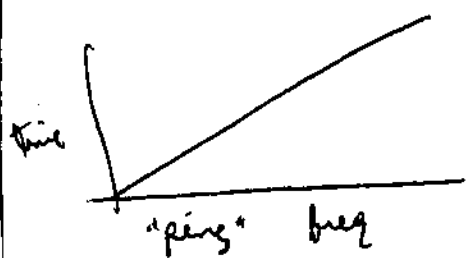
Net-Talk

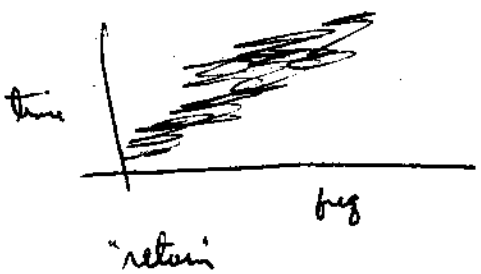
- Resistance to damage
  - added noise to weights
  - turns out that network performed well until fewer than 4 bits of weight left
  - ∴ ≈ 4 bits info/weight sufficient
- Vary # hidden units
  - 0 hidden units (perceptron) – 82% accuracy
  - 120 " " – 98% accuracy
- Vary # layers – similar performance w/two layers as w/one.
- Tea-leaf reading
  - can you see things (patterns, etc.) in the trained network?
  - generally, no. Info seems to be dist. over nodes.

(Neural Networks 1 (1988) 75-89 Gorman Sejnowski)

Sonar example

– try to tell rocks from mines from sonar "pings".





9/26/90

Well done with training a single neuron, what about a collection?
Need new techniques & criterion...

LMS criterion

target function: $t: \mathbb{R}^n \to \mathbb{R}$

$(x, t(x))$ typical training example

f = output of learner

before: prediction error on x = $\begin{cases} 0 & \text{if } f(x) = t(x) \\ 1 & \text{if } f(x) \neq t(x) \end{cases}$

now: square error on x = $(f(x) - t(x))^2$ (note: if f, t are 0,1 valued this is identical to the above).

mean squared error:

$$MSE = \frac{1}{M} \sum_{x \in S} (f(x) - t(x))^2$$

(try to minimize this).

Gradient Descent

$f(x) = f_\omega(x)$ $\qquad \omega \in \mathbb{R}^k$

(ω: weights, parameter, knobs, etc.)

finding good $f \equiv$ finding good $\vec{\omega}$

$$E(\omega) = MSE = \frac{1}{n} \sum_{x \in S} \left(f_\omega(x) - f(x)\right)^2$$

assume E, f are **differentiable** w.r.t. ω.

Idea: start somewhere, & run downhill until we get to a *local minimum*

$$\nabla E = \left(\frac{\partial E}{\partial \omega_1}, \frac{\partial E}{\partial \omega_2}, \ldots \frac{\partial E}{\partial \omega_n} \right)$$

if we go in direction of ∇E, go uphill...

so, go in direction of $-\nabla E$

Alg $\vec{\omega}_0 =$ initial pt. $= (0, 0, 0 \ldots 0)$ (perhaps) (or maybe random)

$$\vec{\omega}_{i+1} \leftarrow \vec{\omega}_i + \varepsilon \, \Delta\vec{\omega}_i \qquad \Delta\vec{\omega}_i = (-\nabla E)$$

(ε: learning rate)

Remarks

① $\nabla E(S) = \frac{1}{m} \sum_{x \in S} \nabla E(\{x\})$ [updates are usually done at a time]

↑ gradient of error over entire set

↑ sum of gradients of singletons

there, can either look at all points & then take a step (update) or, look at each point & then take step.

② momentum

want to avoid bouncing back & forth across val

↳ let our move this time be an average of our proposed move this time & our move last time (will avoid weird bouncing back & forth)

$$\Delta \vec{w}_i = \alpha \Delta \vec{w}_{i-1} + (1-\alpha)(-\nabla E)$$

Gradient Descent for Linear Functions

$f_w(x) = w \cdot \vec{x}$

$S = \{\vec{x}_i\}$ $\quad t(\vec{x}_i) = y_i$

$E(w) = \frac{1}{m} \sum_i (w \cdot \vec{x}_i - y_i)^2$

in this case, local min ⟹ global min

$\nabla E(w) = \frac{2}{m} \sum_i (w \cdot \vec{x}_i - y_i) \cdot x_i$

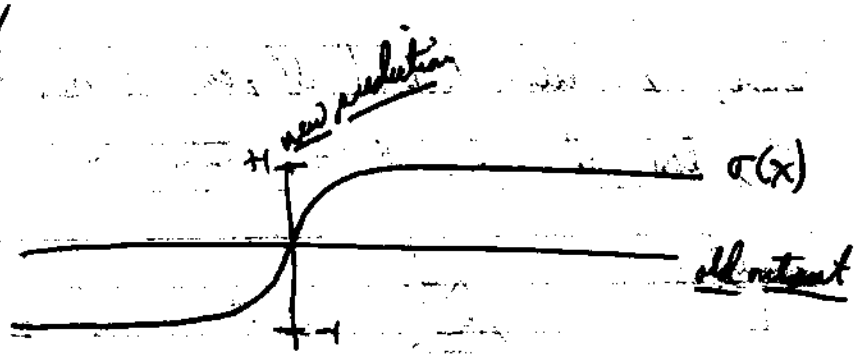
backpropagation / neural nets: Widrow/Hoff, Adaline (1960's)

Now, think about threshold functions.

- a data point which is positive (& thus we should output 1) may actually output 999 — MSE will be high. (A function will be dropped that way to compensate)

like to transform all positive outputs to +1 & all negative outputs to −1

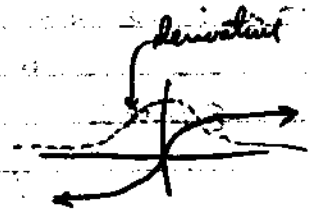
- Use sigmoid



↳ differentiable approximation to a threshold

① $z = \tanh(x) = \left[\frac{e^x - e^{-x}}{e^x + e^{-x}}\right]$

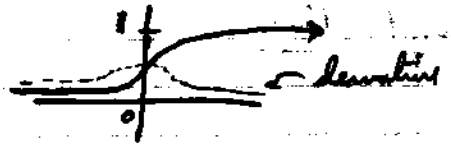
hyperbolic tangent



$\frac{dz}{dx} = \text{sech}^2(x) = \frac{4}{(e^x + e^{-x})^2}$

② logistic function

$z = \frac{1}{1+e^{-x}}$



$\frac{dz}{dx} = z(1-z)$

note: looking at threshold function, so $\theta \in \{\pm 1\}$

Now, let $f_\omega(x) = \sigma(\omega \cdot x)$

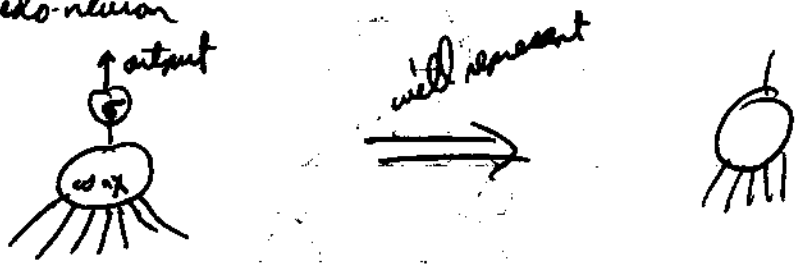
$$E(\omega)\{x_i\} = (\sigma(\omega \cdot \vec{x}_i) - y_i)^2$$

$$\nabla E(\omega)\{x_i\} = \underbrace{2(\sigma(\omega \cdot \vec{x}_i) - y_i)}_{\text{error}} \cdot \operatorname{sech}^2(\omega \cdot \vec{x}_i) \cdot \vec{x}_i$$

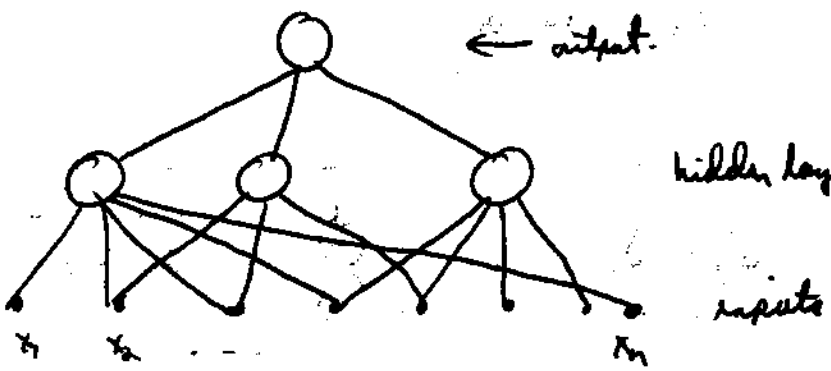
↑ weight → weight stuff closer to ±1 more heavily, don't weight outliers so much

---

neuroid, pseudo-neuron



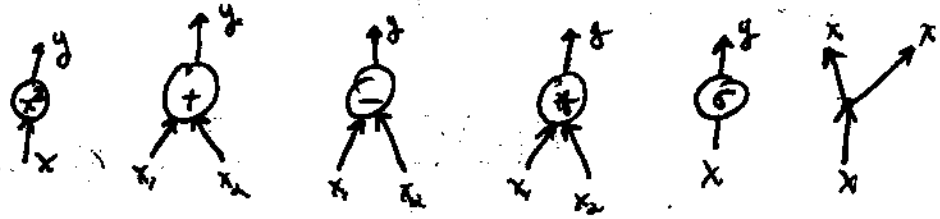
"Neural Networks"



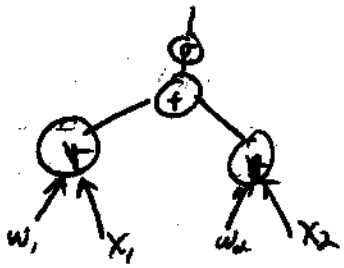
Now, gradient descent method still applies... back-prop

Computing ∇E

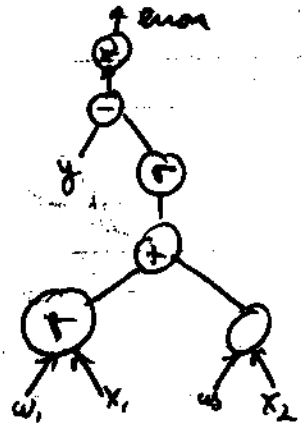
some primitive elements:



e.g. a unit neuron

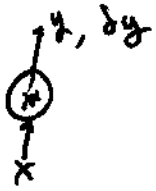


circuit for error:



idea: compute forward to compute y, δ_y & backward to get x, δ_x

e.g.



$$\delta_x = \delta_y \cdot \partial x$$

$$\delta_y = \delta x_1$$
$$= \delta x_2$$

$$\delta x_1 = \delta_y$$
$$\delta x_2 = \delta_y$$

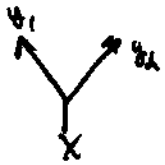
$\delta x_1 = \delta_y \cdot x_2$

$\delta x_2 = \delta_y \cdot x_1$

$\delta x = \delta_y \cdot (y)(1-y)$

or

$\delta x = \delta_y \cdot \text{sech}^2(x)$

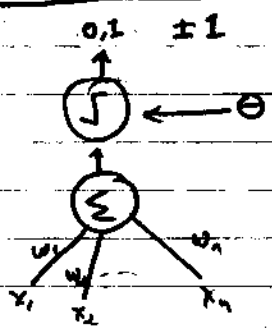


$\delta x = \delta_{y_1} + \delta_{y_2}$

Now, can back propagate & get $\delta w_i \ \forall i$

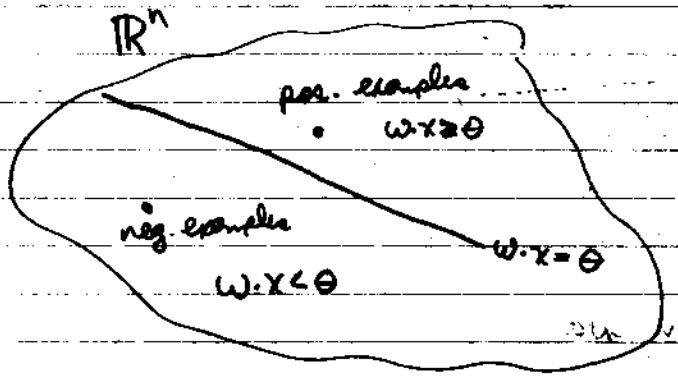
& this is gradient of error w/r.t weights.

Perceptrons

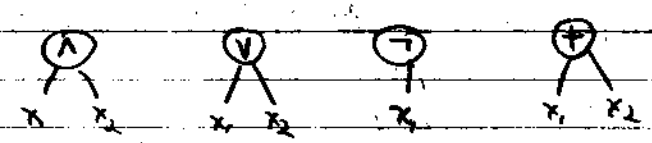


$\sum w_i x_i \geq \theta \implies$ output $= +1$

else $\implies$ output $= -1$



Boolean Functions



$x_1 + x_2 \geq 2$ $\quad x_1 + x_2 \geq 1$ $\quad -x_1 \geq 0$ $\quad$ XOR not linear

Thm: # Boolean threshold functions is $2^{\Theta(n^2)}$

(note: total # boolean functions is 2^{2^n})

Linear separators work well:

Consider two types of examples which are generated by multivariate gaussian distributions:

optimal classifier is a hyperplane
(perpendicular bisector of segment joining the centers)

Perceptron training algorithm

Assumptions:

- assume no examples on hyperplane, $w \cdot x \neq 0$ for "true" w & any x we see
- assume $\theta = 0$ (w.l.o.g.)

  pf: add new coordinate, x_0, to all examples with $x_0 = 1$

  $-w_0 = \theta$
- assume all examples are positive examples (w.l.o.g.)

  (negate x on all negative examples)

Perceptron Alg.

- Initially: $\vec{w} = (0, 0, \ldots, 0)$
- Given an infinite sequence $x^1, x^2, \ldots$ & examples:
  - predict using $\text{sign}(w \cdot x^i)$
  - get correct label for x^i

if $w \cdot x^i > 0$ do nothing

else update w by $w \leftarrow w + x^i$

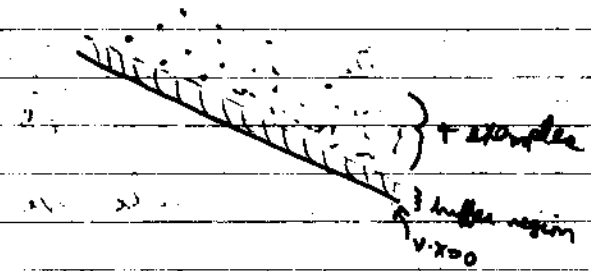
(note: $w' = w + x$

$w' \cdot x = wx + |x|^2$

$\hookrightarrow > 0$

AO more likely to correctly classify)

- Assume $|x^i| = 1 \quad (\forall i)$ (this can be weakened, but some upper bound is needed)
- Assume $(\exists \delta)(\exists v \in \mathbb{R}^n) \quad v \cdot x^i > \delta \quad (\forall i)$



Thm: Perceptron Convergence Theorem

Given assumptions, perceptron procedure only makes a finite # mistakes

pf: assume (w.l.o.g.) $|v|=1$ (else, scale v & change δ)

$$f(w) = \frac{v \cdot w}{|w|} \quad (= \cos\alpha \text{ where } \alpha \text{ is the angle between } v \text{ and } w)$$

$$\leq 1$$

How does $f(w)$ change when $w' \leftarrow w + x^i$

numerator: $v \cdot w' = v \cdot (w + x) = v \cdot w + v \cdot x \geq v \cdot w + \delta$

(increases by at least δ each update)

denominator: $|w'|^2 = w' \cdot w' = (w+x) \cdot (w+x)$

$$= |w|^2 + 2wx + |x|^2$$

$2wx < 0$ because we made a mistake; $|x|^2 = 1$

$$\leq |w|^2 + 1$$

after t updates: $f(w) \geq \frac{t \cdot \delta}{\sqrt{t}} = \delta\sqrt{t}$

note: yield contradiction if $t > 1/\delta^2$

$\Rightarrow$ **At most $1/\delta^2$ mistakes** ∎

Variations

$$w \leftarrow w - \lambda(w \cdot x)x$$

$$0 < \lambda < 2$$

(note: if $\lambda = 1$, moves "just enough" to correctly classify x; procedure converges for any $0 < \lambda < 2$)

Note: Boolean threshold functions can have $1/8 \times 2^{2^n}$

e.g. $x_1 \vee (x_2 \wedge (x_3 \vee (x_4 \wedge \ldots())))$

Note: $2^{\Theta(n^2)}$ threshold functions (boolean)

Hebbing alg. yield only $\Theta(n^2)$ mistakes (possibly much better than above).

- Can we bridge this gap?

VC-dimension

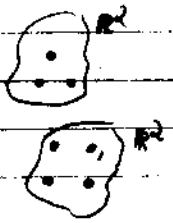
Let X be some domain

Let $\mathcal{C}$ be some concept class on X

Define The Vapnik-Chervonenkis dimension ($VC\text{-dim}(\mathcal{C})$) is the largest d s.t. $\exists$ d distinct pts $x^1, x^2, \ldots, x^d \in X$ s.t. for any $+/-$ labelling of these pts, $\exists$ a concept $c \in \mathcal{C}$ that realizes that labelling.

- If $|\mathcal{C}| < \infty$, then $VC\text{-dim}(\mathcal{C}) \leq \lg |\mathcal{C}|$
- $\mathcal{C}$ = all concepts, $|X| = \infty \Rightarrow VC\text{-dim} = \infty$
- $\mathcal{C}$ = half-spaces in $\mathbb{R}^2$

$VC\text{-dim} = 3$ — can do / can't do



- $\mathcal{C}$ = half-spaces in $\mathbb{R}^n$

$VC\text{-dim} = n+1$

Thm $\text{opt}(\mathcal{C}) \geq VC\text{-dim}(\mathcal{C})$

pf: take d points which yield $VC\text{-dim}(\mathcal{C})$, & force learning alg. to make d mistakes.

Suppose $x^1, x^2, \ldots$ are drawn independently according to some distribution P on X, and labelled according to some target concept c. Let $M_A(t)$ denote probability that learning alg. A makes a mistake on t^{th} trial.

Thm: $\exists A$ s.t. $(\forall t)(\forall P)(\forall c)$

$$M_A(t) \leq \frac{2 \cdot VC\text{-dim}(\mathcal{C})}{t}$$

(Haussler, Littlestone, Warmuth)

(tight to within constant factors)