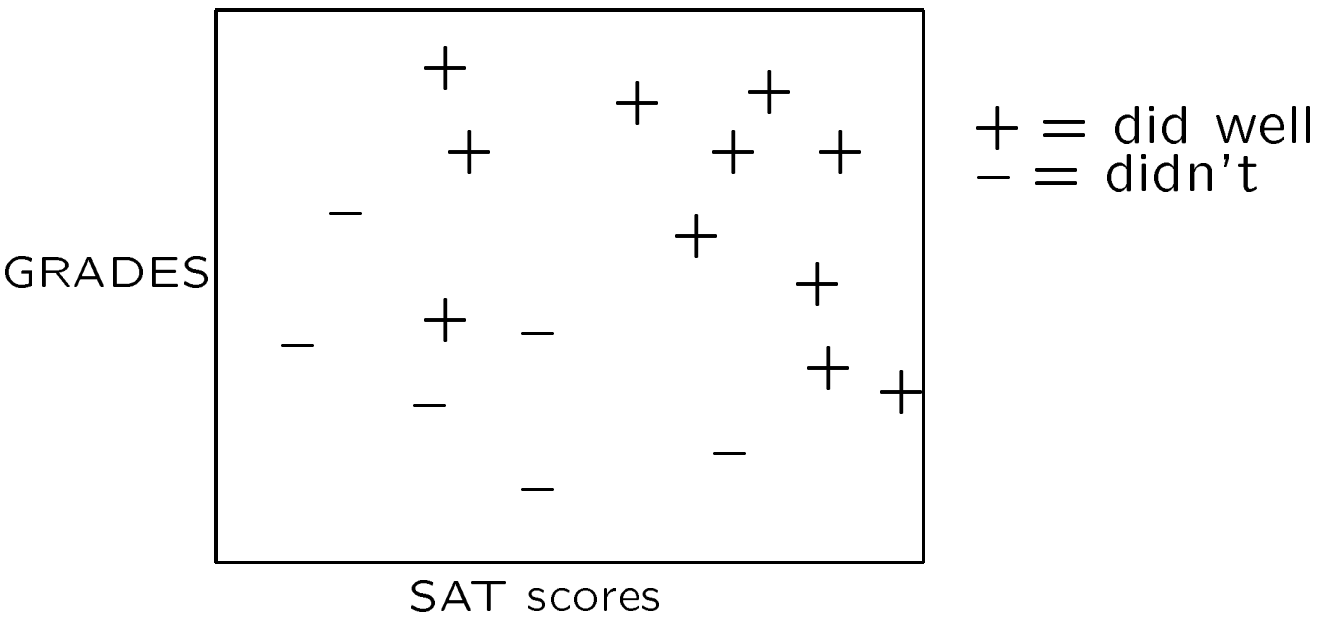
Linear threshold functions/Perceptrons

Linear threshold functions are natural hypothesis representation.

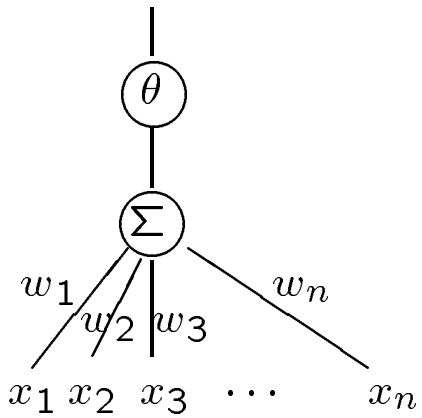


Linear threshold function: If

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \geq \theta$$

then output "positive", else output "negative".

View as "neuron" / "perceptron"



or geometrically as a halfspace (one side of a hyperplane).

Often simplify: Assume $\theta = 0$. (Plane through origin). Can do by adding an extra "dummy" feature.

Goal: given sequence of data to adjust weights to get best performance.

Perceptron Training Procedure

For simplicity, use threshold = 0, have "dummy" input x_0 always 1. So, positive iff $w_0x_0 + w_1x_1 + \ldots + w_nx_n \geq 0$.

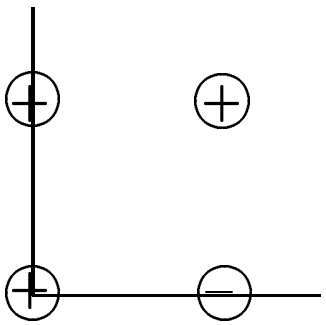
Start with some weights and cycle through data until converge or not improving:

- Mistake on positive: $w_i \leftarrow w_i + x_i$. $(\forall i)$
- Mistake on negative: $w_i \leftarrow w_i - x_i$. $(\forall i)$

If some examples have really large feature values, usually normalize by dividing by "length" of example:

$$\vec{w} \leftarrow \vec{w} \pm \vec{x}/|\vec{x}|$$

Example:



Guarantee: If there is a consistent LTF, this will converge.

Proof: (following Minsky & Papert)

First simplify problem:

- Use "$>$" instead of "$\geq$".
- Replace negative example at $(x_0, x_1, \ldots, x_n)$ by positive at $(-x_0, -x_1, \ldots, -x_n)$. So only pos's.
- Assume each example $\vec{x}$ has $|\vec{x}| = 1$.

Define vector of weights A^* to be a correct solution. So, for each example $\vec{x}$, $A^* \cdot \vec{x} > 0$.

Define $\delta = \min(A^* \cdot \vec{x})$ (min over the data set)

Will show: sufficient decrease in angle between current weight vector and A^* with each update.

Define A to be current weight vector.

Cosine of angle between A and A^* is: $\frac{A \cdot A^*}{|A|}$.

Add $\vec{x}$:

$$A \cdot A^* \rightarrow A^* \cdot (A + \vec{x}) = A^* \cdot A + A^* \cdot \vec{x}.$$

(increases by at least δ).

$$|A|^2 \rightarrow |A + \vec{x}|^2 = A \cdot A + x \cdot x + 2(A \cdot x).$$

(increases by at most 1)

If start with $A = \vec{0}$, in m steps: denominator $\leq \sqrt{m}$ and numerator $\geq m\delta$.

So, number of steps at most $1/\delta^2$.

Bad case for PTP:

Alternating decision list.

$(x_n \to T)$ else $(x_{n-1} \to F)$ else $(x_{n-2} \to T) \ldots$ else $(x_1 \to F)$ else T.

Requires exponential size coeffs.

$$-x_1 + x_2 - 2x_3 + 3x_4 - 5x_5 + 8x_6 - 13x_7 + \ldots \geq 0.$$

(magnitudes follow Fibonacci sequence)

Representation power of LTFs:

- Can represent conjunctions,disjunctions, DLs.
- Can't represent XOR, general decision trees.

XOR:

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

- Can if add extra feature "x_1x_2":

$$x_1 + x_2 - 2x_1x_2 > 0.$$

What's known about LTFs

n features:

- If exists a consistent LTF, several procedures will find one:
  - Linear programming
  - Perceptron Training Procedure
  - Winnow alg
- Linear programming is poly time. Others are usually fast but can take exponential time (in n) in extreme cases.
- If no consistent LTF, then PTP, winnow are more robust.
- Finding the LTF with fewest errors on a given data set is NP-complete.
  (but PTP, winnow do well in practice)

What's known, cont'd

If only a few features can minimize error by trying all possibilities.

- If m examples in 2-D, how many separations at most?

  In 3-D?

- Relation with Occam:

  (Intuitive) More degrees of freedom means more separations, so less trustworthy than hypotheses with fewer degrees of freedom.

  (NOT OBVIOUS!) Can roughly replace "# hypotheses" by "# separations" in the $\frac{1}{\epsilon}(\ln(N)+\ln(1/\delta))$ bound.

Similar problem: **Fitting best linear function**

E.g., Eval(board) $= w_1x_1 + w_2x_2 + \ldots + w_nx_n$.

Goal: minimize squared error:

$$\sum_{\text{examples } x} (desired(x) - actual(x))^2.$$

Iterative method:

- For example x,

$$\begin{aligned} \frac{\partial Error}{\partial w_i} &= -2(desired(x) - actual(x))\frac{\partial actual(x)}{\partial w_i} \\ &= -2(desired(x) - actual(x))x_i. \end{aligned}$$

- Go down gradient:

$$w_i \leftarrow w_i + c[desired(x) - actual(x)]x_i.$$

  where c is some constant, small enough to prevent oscillation.

Error is quadratic, so nice and "bowl shaped".
If steps small enough will converge.

Batch method

Given: examples $x^1, x^2, \ldots, x^m$.

- Set partials to 0:

$$\frac{\partial Error}{\partial w_i} = \sum_j -2(desired(x^j) - x^j \cdot w)x_i^j = 0$$

- Solve the linear equalities.

Equivalently, let A = matrix of examples, w = vector of weights, d = vector of desired outputs.

- Squared error: $||Aw - d||^2$.
- Solve: $2A^T(Aw - d) = 0$.

$$w = (A^T A)^{-1} A^T d.$$

Example from before (with dummy feature):

(Say positive means desired $= 1$, negative means desired $= -1$. So, desired outputs for 100, 101, 101, 111 are: $1, 1, -1, 1$ respectively.)

$$w = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

Solves to: $\begin{bmatrix} 1/2 \\ -1 \\ 1 \end{bmatrix}$, or: $1/2 - x_1 + x_2$.