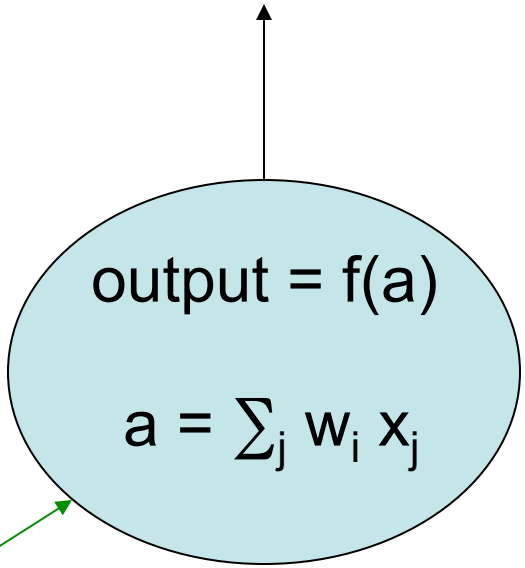
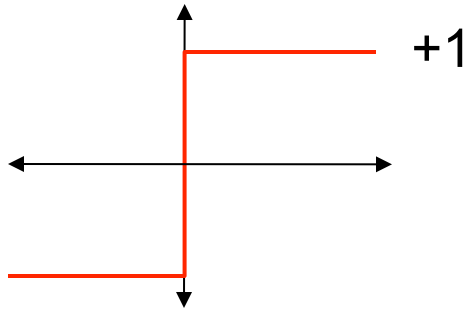
Perceptron

Linear Threshold Algorithm

General Perceptron





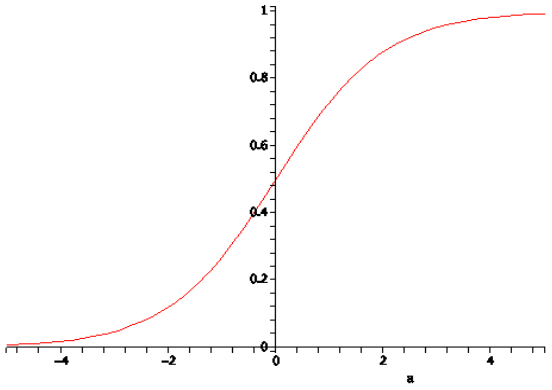
Linear threshold unit (LTU)

f(a) = a (identity function)

Can also have bias b (fixed to 1, add-a-dimen.)

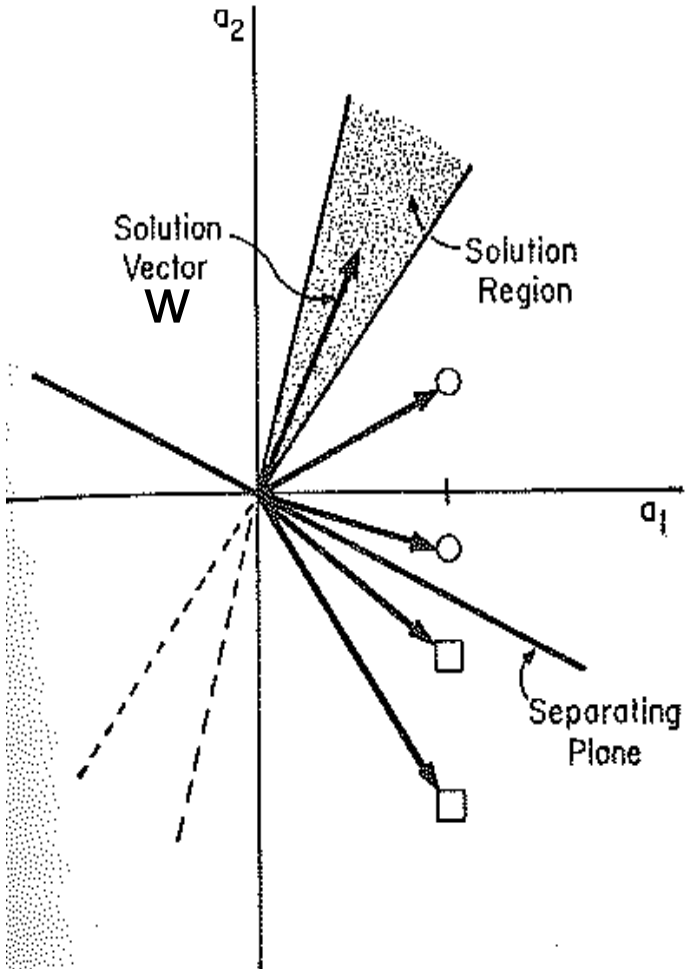
input values, x_i's

Each x_i weighted by a w_i



f(a) = 1/(1+exp(-a))

Classifying points – illustration of LTU "wiggle room"



From Duda and Hart, 1973

Perceptron Algorithm
(output sign($\boldsymbol{w} \bullet \boldsymbol{x}$), so *f(a)* step function)

- Keeps weights w_j, one per feature
- "Online algorithm", initially $\boldsymbol{w} = (0,\ldots,0)$
- Repeat (until consistent with data):

  get next training example *i*: $(\boldsymbol{x}_i, y_i)$

  if $(\boldsymbol{w} \bullet \boldsymbol{x}_i)\, y_i \leq 0$ then mistake:

  $\boldsymbol{w}$ gets $\boldsymbol{w} + \eta_i\, y_i\, \boldsymbol{x}_i$

η_i values are learning rates (step sizes)

Perceptron Class Exercise:

- Assume η_i always 1

x_1	x_2	y
1	3	+1
2	3	-1
-3	1	+1
1	-1	-1

(gap ≈ 2/15)

Perceptron as stochastic gradient descent

- *Perceptron criteria*: if $y_i \neq \text{sign}(\boldsymbol{w} \bullet \boldsymbol{x}_i)$ then minimize "badness" of mistake on example *i*:

$$-y_i \, (\boldsymbol{w} \bullet \boldsymbol{x}_i)$$

- Differentiate wrt w_j *gives gradient component:*

$$-y_i \, x_{i,j}$$

- Negative gradient, $y_i \, \boldsymbol{x}_i$, is direction of steepest descent, add $y_i \, x_{i,j}$ to w_j (for each j) or equivalently add vector $y_i \, \boldsymbol{x}_i$ to $\boldsymbol{w}$

Perceptron Convergence

- For arbitrary data it converges if

  η_i values go to 0 (as i goes to ∞)

  sum of η_i values goes to ∞

  sum of $(\eta_i)^2$ values finite

  (e.g. $\eta_i = 1 / i$; Robbins-Monro alg.):

- If data linearly separable with "gap" when instances normalized to length 1 then converges within $(1/\text{gap})^2$ mistakes

  (other slides)

Perceptron notes

- Can run in batch mode - delay updates until completed pass through data
- Voted perceptron idea
- Multiclass (1-vs-all): learn a $\boldsymbol{w}_y$ for each class, predict with y maximizing $\boldsymbol{w}_y \bullet \boldsymbol{x}$
- Learns discriminative classifier directly (no probability)