DMML, 7 Mar 2019

Boosting a weak classifier

Start with a weak classifier h_0 over $T = \{w_1, \ldots, w_N\}$

makes mistakes on T,

accuracy $> 1/2$

Assign weight $1/N$ to each input

Add up weight of mistakes — boost weight of wrong inputs by a factor based on this

Build a classifier h_1 with new weights

Repeat

Sequence of weak classifiers $h_0, h_1, \ldots, h_m$

Error rate $e_0, e_1, \ldots, e_m$

Weighted sum of classifiers, using error rates

If classifiers vote as ± 1

$$h(x) = \text{sign}\left(\alpha_0 h_0(x) + \alpha_1 h_1(x) + \ldots + \alpha_m h_m(x)\right)$$

weight of the classifier

+1 or -1

Related setting

- A collection of weak classifiers ("experts")
- Combine expert advice

Topic classifier for articles

Expert i – checks for some words
and gives a topic,
else says "no guess"

"sleeping expert"

No sleeping - every expert classifies every input.

Experts : $k_1, k_2, \ldots$

Find a subset of $\leq M$ experts and assign weights α_i to this subset

Chosen set : $\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_M$ ← need not be distinct

Weights : $\alpha_1, \alpha_2, \ldots, \alpha_M$

$$h(x) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i \hat{k}_i(x) \right)$$

Weights for experts, weights for inputs

α_i expert k_i

$w_j = w_j^{(0)}$ input (x_j, y_j)

Initially $w_j = \frac{1}{N}$

Iteratively add $\hat{k}_1, \hat{k}_2, \dots, \hat{k}_M$ to our team of experts

Inductively assume we have chosen

$\hat{k}_1 \dots \hat{k}_m$ with weights $\alpha_1, \dots, \alpha_m$

Input weights $w_j^{(m)}$

Iteration $m+1$

Run each k_i on training data and get weighted sum of errors

$$E_i = \sum_{y_j \neq k_i(x_j)} w_j^{(m)}$$

Pick $\hat{k}_{m+1}$ as k_i with min E_i

$$e_{min} = \frac{E_i}{W}$$

sum of all weights

$$\alpha_m = \frac{1}{2} \ln\left(\frac{1 - e_{min}}{e_{min}}\right)$$

Update weights

$$w_j^{(m+1)} = w_j^{(m)} \cdot e^{\alpha_m} = w_j^{(m)} \sqrt{\frac{1-e_m}{e_m}}$$

if new $\hat{k}_{m+1}$ makes a mistake on x_j

$$= w_j^{(m)} e^{-\alpha_m} = w_j^{(m)} \sqrt{\frac{e_m}{1-e_m}}$$

if new $\hat{k}_{m+1}$ gets x_j correct

Sleeping Experts

No weights on inputs

Each expert k_i acts on a subset $T_i \subseteq T$

Assign weight $\alpha_i = 1$ to each k_i

Input $x_j \rightarrow K_j$ subset of experts active on x_j

$$W_{x_j} = \sum_{k_i \in K_j} \alpha_i$$ — weighted sum of experts who offer opinion on x_j

Given $k_i \in K_j$, either $k_i(x_j) = y_j$ or

$k_i(x_j) \neq y_j$

$m_i^{x_j} = 0$

$m_i^{x_j} = 1$

"miss value"

$$r_i^{x_j} = \frac{\left(\sum_{k_l \in K_j} P_l^{x_j} \cdot m_l^{x_j}\right)}{(1+\epsilon) - m_i^{x_j}}$$

$$P_l^{x_j} = \frac{\alpha_l}{W_{x_j}}$$

Update

$$\alpha_i = \alpha_i \cdot (1+\varepsilon)^{r_i^{x_j}}$$

How will this perform if the classification is skewed?

Minority case is "Yes" – most classifiers will have high accuracy overall?

Tune weight update to focus on minority case.

Neural networks

Perceptron

$$w \cdot x \geq b$$

Instead

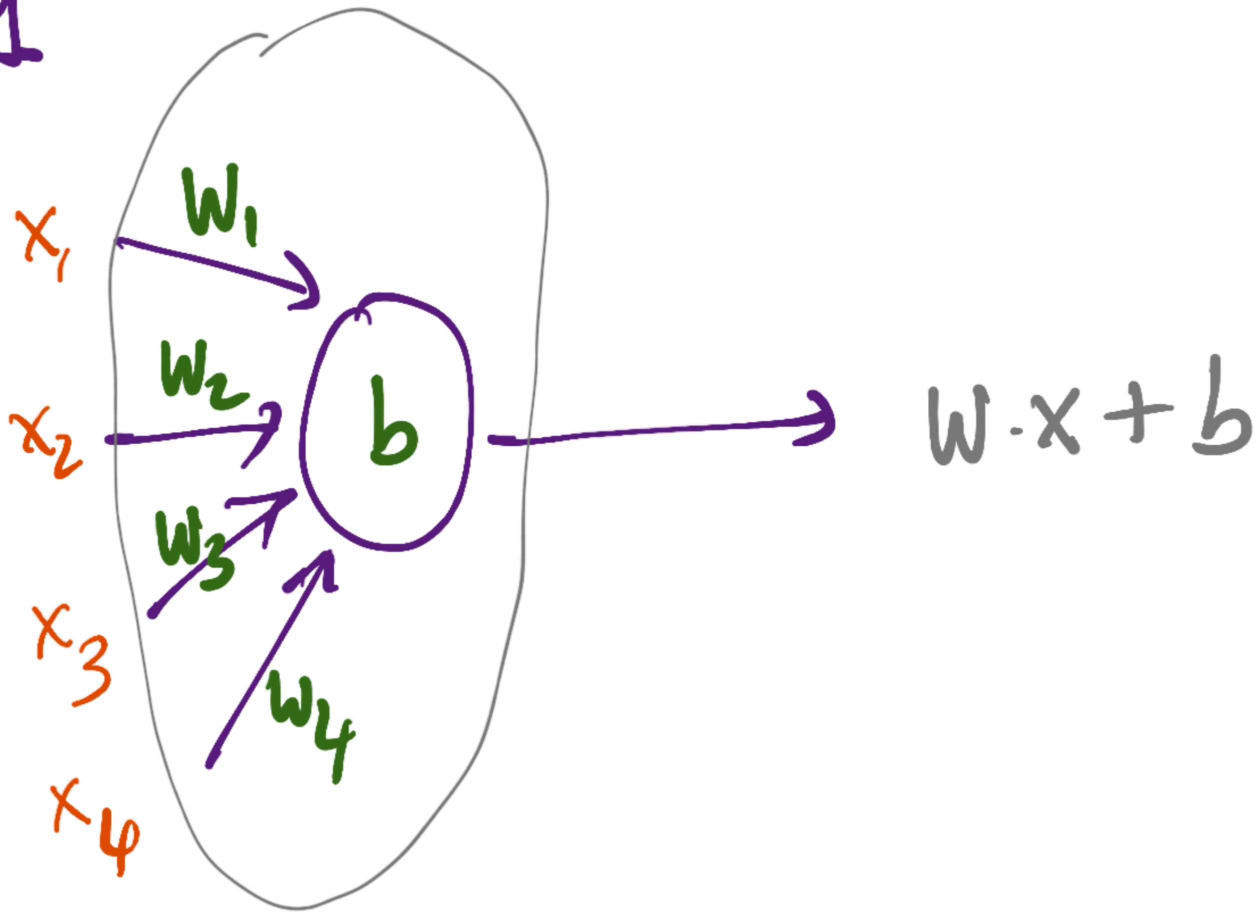
$$w \cdot x + b \geq 0$$

$$\leq 0$$

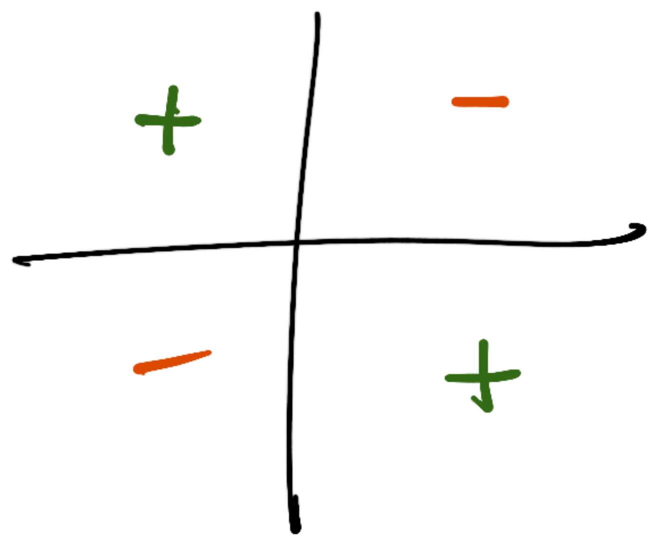
weight

bias

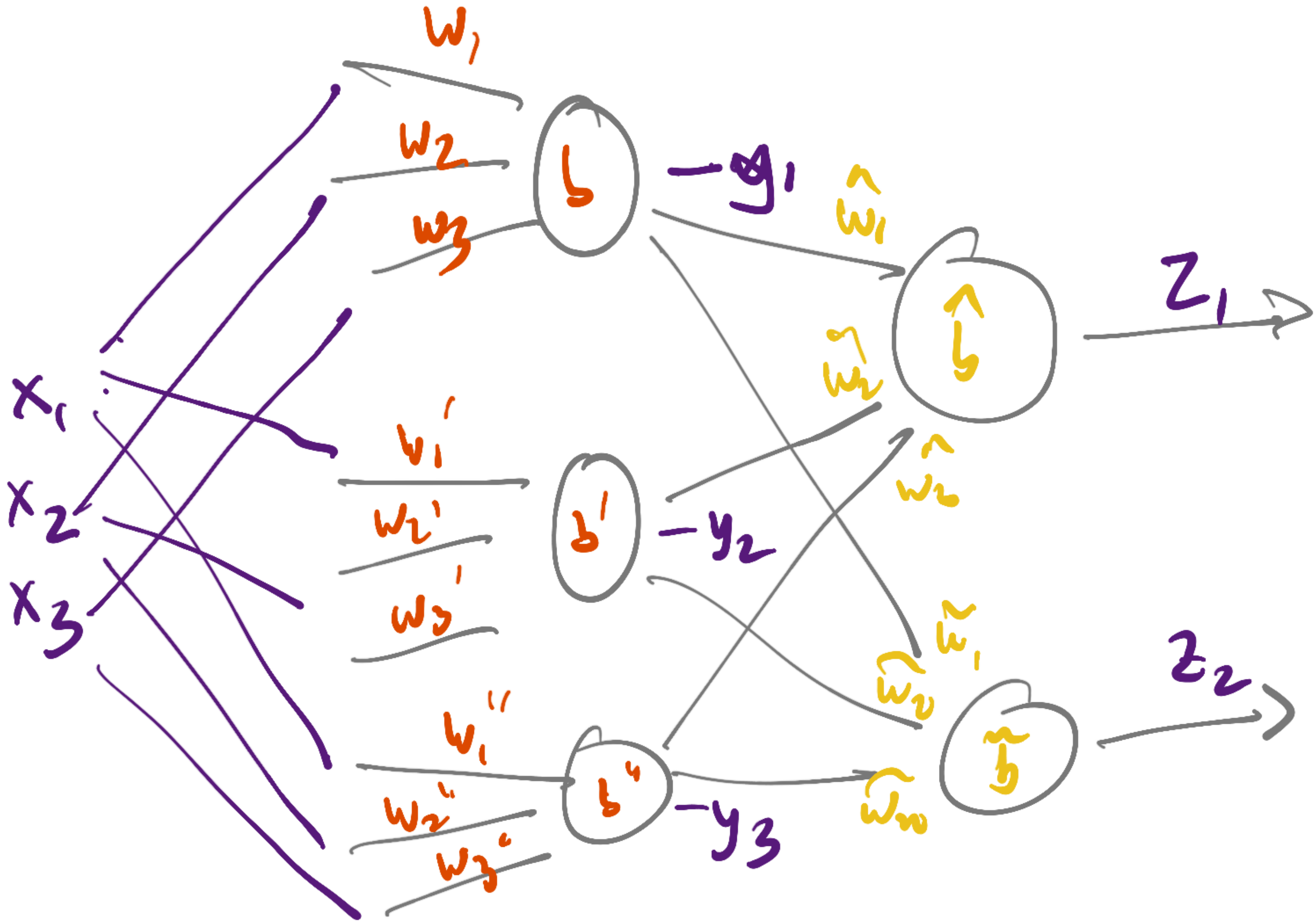
Pictorially

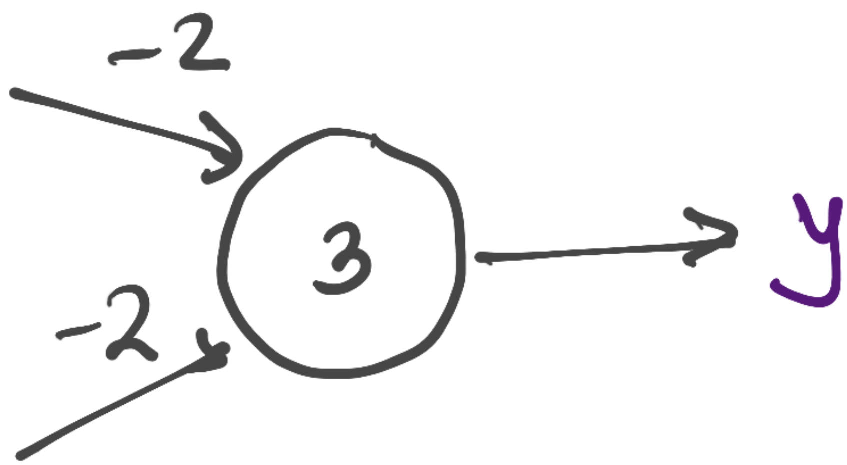


XOR function
cannot be modelled



Network of perceptrons





$x_1, x_2 = 0, 1$

NAND gate

x_1	x_2	y	
0	0	3	
0	1	1	$\rightarrow 1$
1	0	1	
1	1	-1	$\rightarrow 0$

Functional completeness

OR + NOT can express any boolean function over (x_1, x_2)

AND + NOT

NAND

Perceptron for NAND → networks of perceptrons are functionally complete