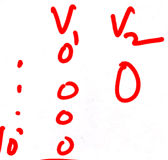
Expressive power of Neural Networks

$(+1, -1)$

A boolean function involving n variables maps from a domain $\{\pm 1\}^n$ to $\{\pm 1\}$ can be implemented by a neural network $\mathcal{H}_{V,E,\sigma}$

architecture

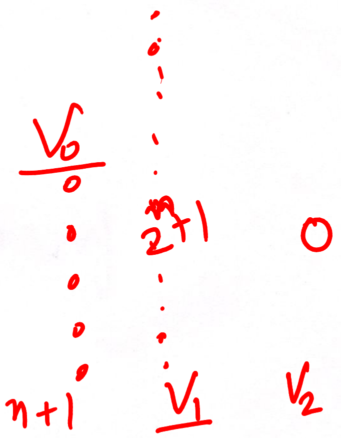


2 layers: # nodes $|V|$??

Claim: For every n, there exists a graph (V, E) of depth 2 such that $\mathcal{H}_{V,E,sign}$ contains all functions from $\{\pm 1\}^n$ to $\{\pm 1\}$.

Domain

σ: sign(threshold)



Proof: We construct a graph with $|V_0| = n+1$, $|V_1| = 2^n+1$, $|V_2| = 1$

Let $f : \{\pm 1\}^n \longrightarrow \{\pm 1\}$ be some boolean function.

We can adjust the weights so that the network will implement f.

Let $u_1 \,...\, u_k$ be all vectors in $\{\pm 1\}^n$ on which f outputs 1.

	1 2 ... n	f()	
u_1	1 1 ... 1	+1	← 1st
2^n rows	1 1 ... −1	−1	
u_2	−1 1 ... 1	+1	← 2nd

Consider a vector $x \in \{\pm 1\}^n$ (bool)

Note that if $x \neq u_i$ then $\langle x, u_i \rangle \leq n-2$

if $x = u_i$ then $\langle x, u_i \rangle = n$

$+1 \; +1 \; -1 = 3$

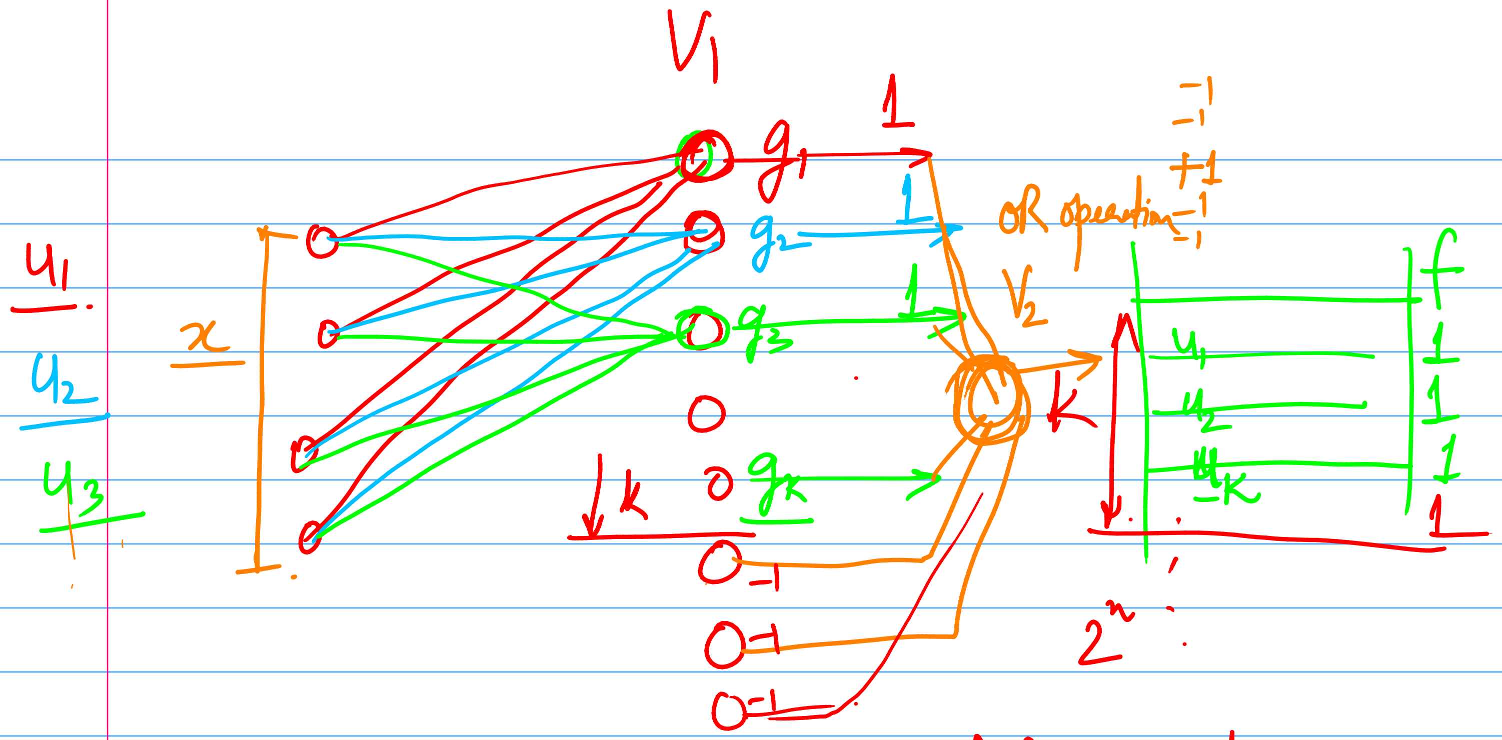
$+1 \; +1 \; -1$

$+1 \; -1 \; -1 = 1$

$\therefore$ a function $g_i(x) = \underbrace{sign}_{\sigma}(\underbrace{\langle x, u_i \rangle}_{\text{Affine transform}} \underbrace{- n + 1}_{\text{bias}})$ equals 1 if and only if $x = u_i$

$g_i(x)$ computational node

$x \neq u_i$ $\quad sign(n-2-n+1)$ $\quad sign(-1) = -1$ $\qquad n - n + 1$ $\qquad sign(1) = 1$



$u_1 \quad u_2 \quad u_3 \quad \ldots \quad u_K$ $\qquad f(\;) = 1$

XEnt

CE

loss

K: #classes.

output layer nodes = #classes

output	GT
o	y
o_1	y_1
o_2	y_2
o_3	y_3
⋮	
o_K	y_K

prob vector ← Xent → prob vector.

one-hot encoding.

0
1
0
0
0

100	0.2	3
101	0.21	94
98	0.19	6
97	0.19	5
74	0.16	3

K=2

confidence is low $\qquad$ high.

σ(Affine). vector of real numbers $\rightarrow$ $v_1, v_2, \ldots, v_K$ $\xrightarrow{\text{softmax}}$ $o_1, o_2, \ldots, o_K$ probability

We can adapt the weights between V_0 and V_1 so that for every $i \in [k]$, the i^{th} neuron is $g_i(x)$ i

The neuron in the output layer implements a disjunction i.e. OR of the functions $g_i(x)$.

$-k+k-1 \Rightarrow -1$

AND

Bias $-k+1$

$$f(x) = \operatorname{sign}\Big(\underbrace{\sum_{i=1}^{k} g_i(x)} + \underbrace{k-1}_{\text{bias}}\Big)$$

$-(k-2)+k-1 \Rightarrow 1$

$\operatorname{sign}(1)$

Even if we try to model functions of the form $\{0,1\}^n \rightarrow \{0,1\}$ the size of the network will be exponential in n.

A neural network can approximate 1-Lipschitz function $f: [-1,+1]^n \longrightarrow [-1,1]$ within a precision ϵ, but the size of the network will be exponential in n.

$[-1,+1]$

$f(u)$

$f(v)$

n_2 variables $[-1 \quad +1]$

$$\|f(u)-f(v)\| \leq \|u-v\|^2$$

Softmax converts k real valued predictions $v_1 \dots v_k$ into output probabilities $o_1 \dots o_k$ using the relation

$$\boxed{o_i = \frac{\exp(v_i)}{\sum_{j=1}^{k} \exp(v_j)}} \qquad \forall i \in \{1, \dots k\}$$

The softmax is mostly paired with the cross-entropy loss.

If the target probability distribution over the k-classes is given by the vector $y_1 \dots y_k$ then the cross-entropy loss is defined as $L = -\sum_{i=1}^{k} y_i \log(o_i)$