Machine Learning

Neural Networks

Fabio Vandin

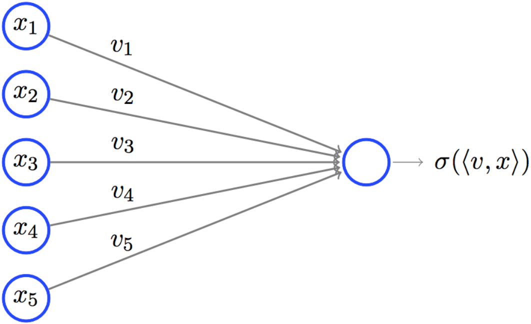
December 6^{th}, 2022

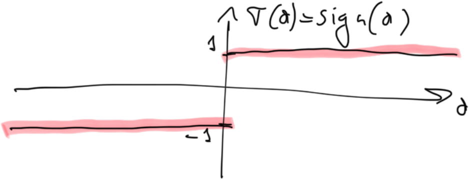
Neuron

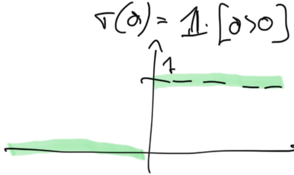
Neuron: function $\mathbf{x} \to \sigma(\langle \mathbf{v}, \mathbf{x} \rangle)$, with $\mathbf{x} \in \mathbb{R}^d$

$\sigma : \mathbb{R} \to \mathbb{R}$ is the *activation function*

Example: $\mathbb{R}^5$

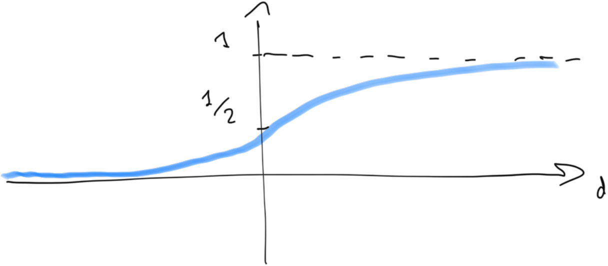






We will consider σ to be one among:

- sign function: $\sigma(a) = \text{sign}(a)$
- threshold function: $\sigma(a) = \mathbb{1}[a > 0]$
- sigmoid function: $\sigma(a) = \frac{1}{1+e^{-a}}$



Neural Network (NN)

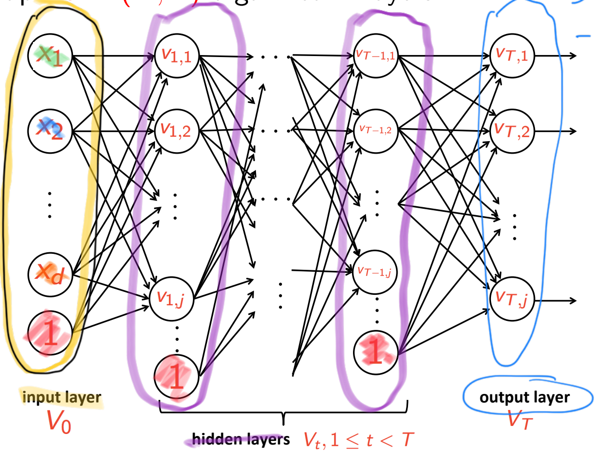
Obtained by connecting many neurons together.

We focus on **feedforward neural networks**, defined by a directed acyclic graph $G = (V, E)$ organized in layers

output of the nn
$-1; +1$

$\vec{x} \in \mathbb{R}^d$

$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

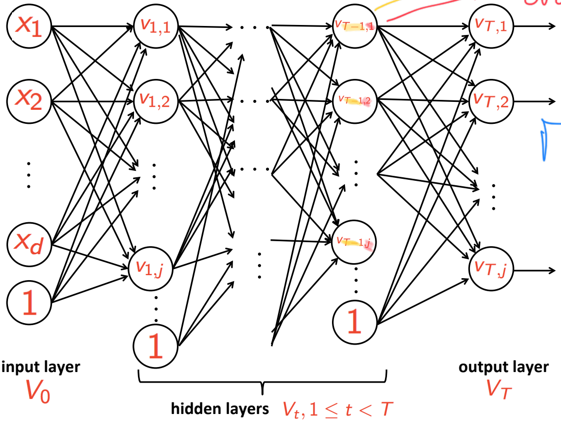


Neural Network (NN)

Obtained by connecting many neurons together.

We focus on **feedforward neural networks**, defined by a directed acyclic graph $G = (V, E)$ organized in layers

layer "order" within the layer



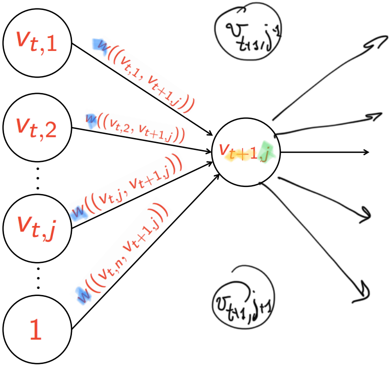
if $\sigma(a) = a$ is used for all nodes (but the output layer)? Linear model!

Each edge e has a *weight* $w(e)$ specified by $w : E \to \mathbb{R}$

Point of View of One Node

Consider node $v_{t+1,j}$, $0 \leq t < T$. Let

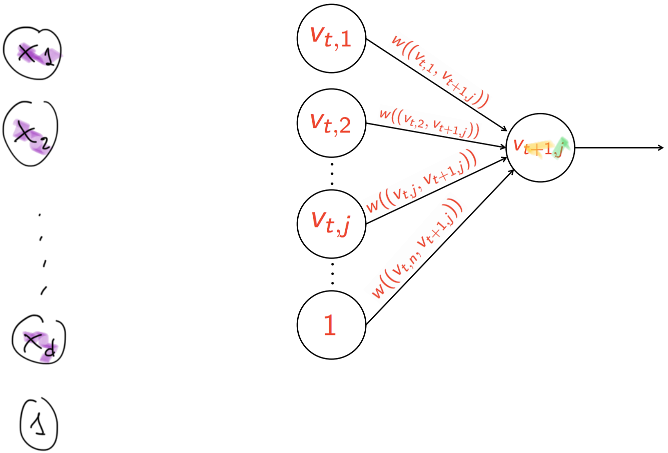
layer $t+1$, j-th node in the layer



Point of View of One Node

Consider node $v_{t+1,j}$, $0 \leq t < T$. Let

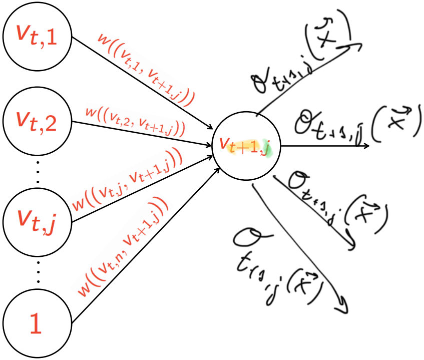
- $a_{t+1,j}(\mathbf{x})$: its *input* when $\mathbf{x}$ is fed to the NN before $\sigma(\,)$ is applied



Point of View of One Node

Consider node $v_{t+1,j}$, $0 \leq t < T$. Let

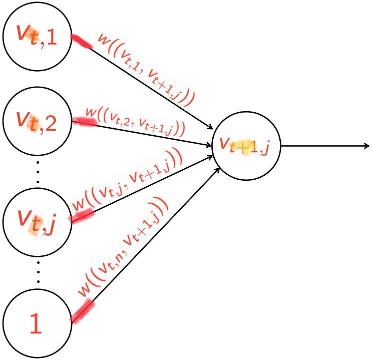
- $a_{t+1,j}(\mathbf{x})$: its *input* when $\mathbf{x}$ is fed to the NN
- $o_{t+1,j}(\mathbf{x})$: its *output* when $\mathbf{x}$ is fed to the NN



Point of View of One Node

Consider node $v_{t+1,j}$, $0 \le t < T$. Let

- $a_{t+1,j}(\mathbf{x})$: its *input* when $\mathbf{x}$ is fed to the NN
- $o_{t+1,j}(\mathbf{x})$: its *output* when $\mathbf{x}$ is fed to the NN

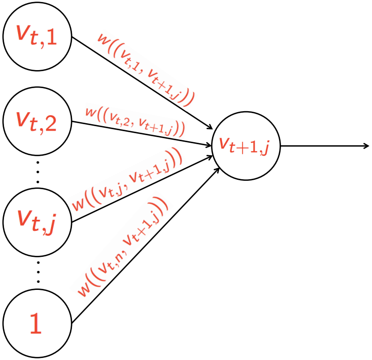


Then: $a_{t+1,j}(\mathbf{x}) = \sum_{r:(v_{t,r}, v_{t+1,j}) \in E} w((v_{t,r}, v_{t+1,j})) o_{t,r}(\mathbf{x})$

Point of View of One Node

Consider node $v_{t+1,j}$, $0 \le t < T$. Let

- $a_{t+1,j}(\mathbf{x})$: its *input* when $\mathbf{x}$ is fed to the NN
- $o_{t+1,j}(\mathbf{x})$: its *output* when $\mathbf{x}$ is fed to the NN



If $\sigma(a) = a$

$\Rightarrow$ neuron corresponds to a linear model

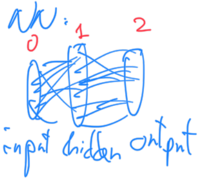
Then: $$a_{t+1,j}(\mathbf{x}) = \sum_{r:(v_{t,r}, v_{t+1,j}) \in E} w((v_{t,r}, v_{t+1,j}))\, o_{t,r}(\mathbf{x})$$

$$o_{t+1,j}(\mathbf{x}) = \sigma(a_{t+1,j}(\mathbf{x}))$$

Neural Network: Formalism

Neural network: described by directed acyclic graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}$

- $V = \cup_{t=0}^{T} V_t, V_i \cap V_j = \emptyset \ \ \forall i \neq j$
- $e \in E$ can only go from V_t to V_{t+1} for some t
- V_0 = *input layer*
- V_T = *output layer*
- $V_t, 0 < t < T$ = *hidden layers*
- T = *depth*
- $|V|$ = *size* of the network
- $\max_t |V_t|$ = *width* of the network



depth?

3? 4.25

2? 33.25

1? 12.25

0? 0.25

Neural Network: Formalism

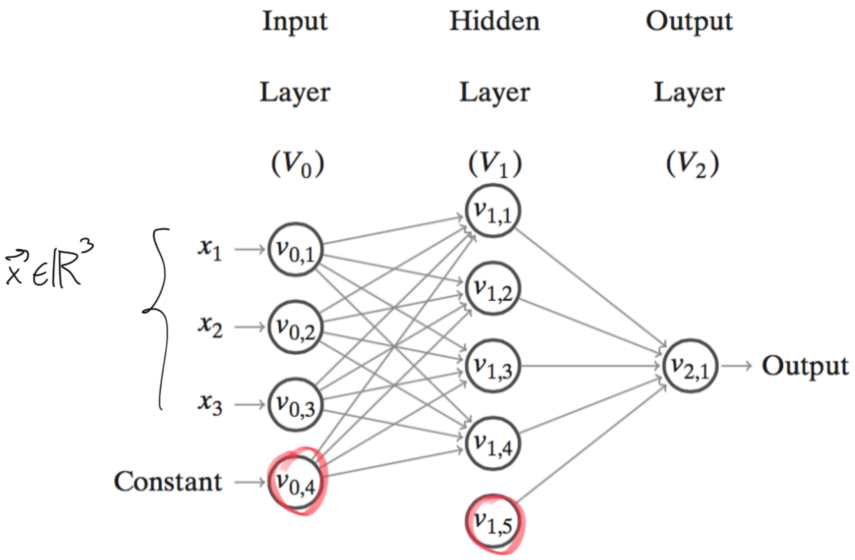
Neural network: described by directed acyclic graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}$

- $V = \cup_{t=0}^{T} V_t, V_i \cap V_j = \emptyset \;\; \forall i \neq j$
- $e \in E$ can only go from V_t to V_{t+1} for some t
- V_0 = *input layer*
- V_T = *output layer*
- $V_t, 0 < t < T$ = *hidden layers*
- T = *depth*
- $|V|$ = *size* of the network
- $\max_t |V_t|$ = *width* of the network

Notes:

- for binary classification and regression (1 variable): output layer has 1 node
- different layers could have different activation functions (e.g., output layer)

Example



depth $= 2$, size $= 10$, width $= 5$

Exercize

Assume that for each node the activation function $\sigma(z) : \mathbb{R} \to \mathbb{R}$ is defined as

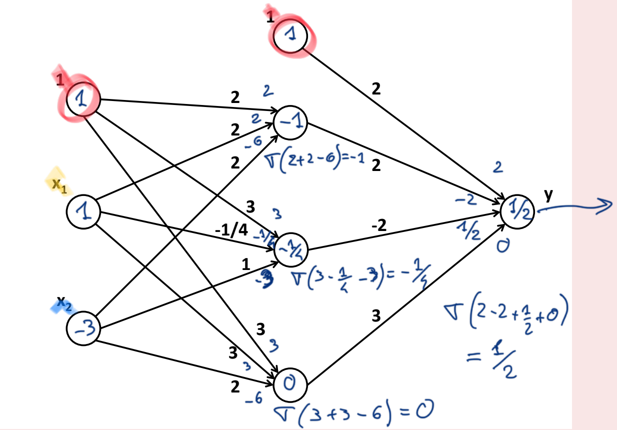
$$\sigma(z) = \begin{cases} 1 & z \geq 1 \\ z & -1 \leq z < 1 \\ -1 & z < -1 \end{cases}$$

and consider the neural network in the next slide, compute the value of the output y when the input $\mathbf{x} \in \mathbb{R}^2$ is

$$\mathbf{x} = [1 \quad -3]^\top$$

Exercise (continue)

$\vec{x} = \begin{bmatrix} 1 \\ -3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$



$\Rightarrow$ predicted value for $\vec{x} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$ is $1/2$.

Hypothesis Set of a NN

Architecture of a NN: (V, E, σ)

Once we specify the architecture and w, we obtain a function:

$$h_{\underbrace{V,E,\sigma}_{\text{architecture}},w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$$

weights (on edges)

Hypothesis Set of a NN

Architecture of a NN: (V, E, σ)

Once we specify the architecture and w, we obtain a function:

$$h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$$

The *hypothesis class* of a neural network is defined by *fixing* its architecture:

$$\mathcal{H}_{\underbrace{V,E,\sigma}_{\text{architecture}}} = \{h_{V,E,\sigma,w} : w \text{ is a mapping from } E \text{ to } \mathbb{R}\}$$

Hypothesis Set of a NN

Architecture of a NN: (V, E, σ)

Once we specify the architecture and w, we obtain a function:

$$h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$$

The *hypothesis class* of a neural network is defined by *fixing* its architecture:

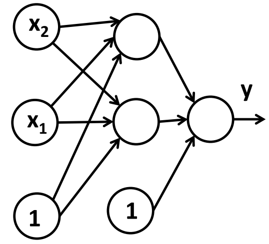
$$\mathcal{H}_{V,E,\sigma} = \{h_{V,E,\sigma,w} : w \text{ is a mapping from } E \text{ to } \mathbb{R}\}$$

Question: what type of functions can be implemented using a neural network?

Exercise (expressiveness of NNs)

Let $\mathbf{x} = [x_1, x_2] \in \{-1, 1\}^2$, and let the training data be represented by the following table:

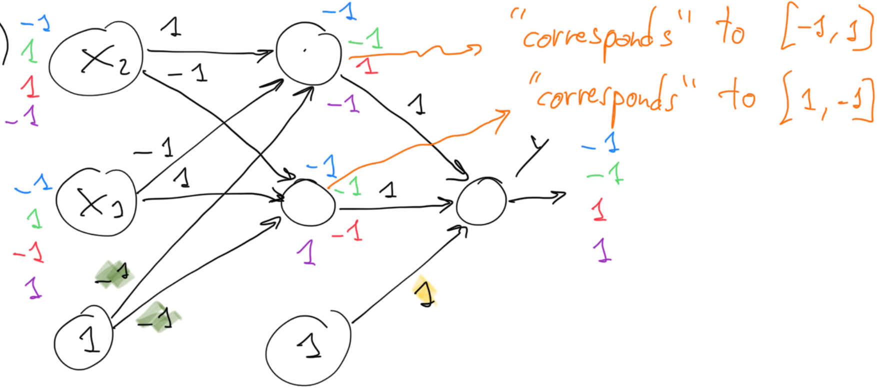
x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



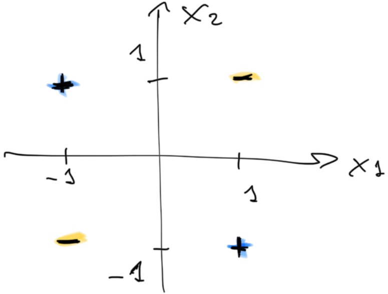
Consider the NN in the figure above, where the activation function for each hidden node and the output node is the *sign* function. Assume that the network's weights are constrained to be in $\{-1, 1\}$.

1. Find network's weights so that the training error is 0.
2. Use example above to motivate the fact that NNs are *richer* models than linear models.

1)



2) The training set represents the "XOR" function, that is not linear



$\vec{x} \in \{-1, 1\}^2$

Such dataset cannot be perfectly classified with a linear model, but NNs can perfectly classify the data $\Rightarrow$ NNs are richer models than linear models!

General construction: let's take an arbitrary function

$$f: \{-1, 1\}^d \to \{-1, 1\}.$$

Goal: build a NN that corresponds to f: if the input is $\vec{x}$, then the prediction of such NN is $f(\vec{x})$

i) consider $\vec{x}$ such that $f(\vec{x}) = 1$: for each such $\vec{x}$, there is a neuron in the (only) hidden layer that "corresponds" to $\vec{x}$. The neuron implements:

$$g_i(\vec{x}') = \text{sign}(\langle \vec{x}, \vec{x}' \rangle \underbrace{- d + 1}_{})$$

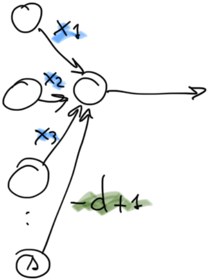
weights on the incoming edges

input to the neuron

neuron $g_i(\vec{x}')$

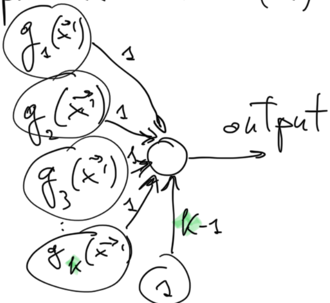
"corresponds" to

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

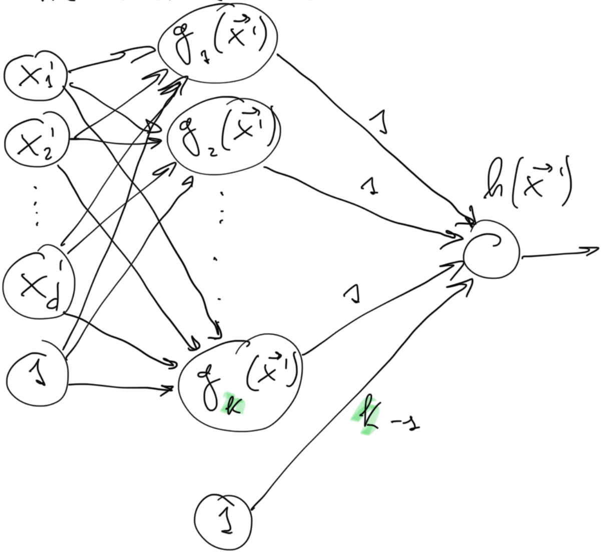


- output node: "implements" $h(\vec{x}') = \text{sign}\left(\sum_{i=1}^{k} g_i(\vec{x}') + k-1\right)$

where $k = \#$ of inputs $\vec{x}$ s.t. $f(\vec{x}) = 1$



The overall NN is :



Exercise: show that the NN "computes" $f()$.

Expressiveness of NN

set of hypotheses for the NN where the architecture is given by (V,E) and the sign activation function

Proposition

For every d, there exists a graph (V,E) of depth 2 such that $\mathcal{H}_{V,E,\text{sign}}$ contains all functions from $\{-1,1\}^d$ to $\{-1,1\}$

Expressiveness of NN

Proposition

For every d, there exists a graph (V, E) of depth 2 such that $\mathcal{H}_{V,E,\text{sign}}$ contains all functions from $\{-1, 1\}^d$ to $\{-1, 1\}$

NN can implement every boolean function!

Unfortunately the graph (V, E) is very big...

Proposition

For every d, let $s(d)$ be the minimal integer such that there exists a graph (V, E) with $|V| = s(d)$ such that $\mathcal{H}_{V,E,\text{sign}}$ contains all functions from $\{-1, 1\}^d$ to $\{-1, 1\}$. Then $s(d)$ is an exponential function of d.

Note: similar result for $\sigma =$ sigmoid

Proposition

For every fixed $\varepsilon > 0$ and every Lipschitz function $f : [-1, 1]^d \to [-1, 1]$ it is possible to construct a neural network such that for every input $\mathbf{x} \in [-1, 1]^d$ the output of the neural network is in $[f(\mathbf{x}) - \varepsilon, f(\mathbf{x}) + \varepsilon]$.

Note: first result proved by Cybenko (1989) for sigmoid activation function, requires only 1 hidden layer!

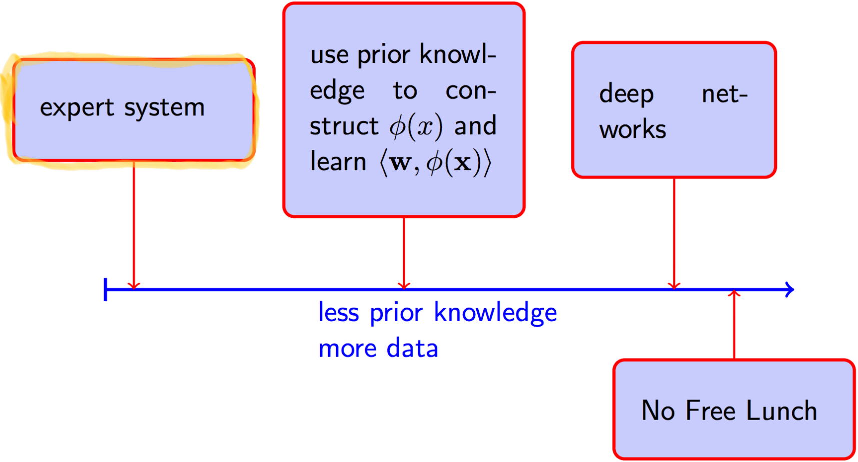
NNs are **universal approximators!**

But again...

Proposition

Fix some $\varepsilon \in (0, 1)$. For every d, let $s(d)$ be the minimal integer such that there exists a graph (V, E) with $|V| = s(d)$ such that $\mathcal{H}_{V,E,\sigma}$, with $\sigma =$ sigmoid, can approximate, with precision ε, every 1-Lipschitz function $f : [-1, 1]^d \to [-1, 1]$. Then $s(d)$ is exponential in d.

An Extremely Powerful Hypothesis Class...



Sample Complexity of NNs

How much data is needed to learn with NNs?

Proposition

The VC dimension of $\mathcal{H}_{V,E,\text{sign}} = O\left(|E| \log |E|\right)$

Different σ?

Proposition

Let σ be the sigmoid function. The VC dimension of $\mathcal{H}_{V,E,\sigma}$ is:

- $\Omega\left(|E|^2\right)$
- $O\left(|V|^2|E|^2\right)$

$\Rightarrow$ large NNs require a lot of data!

Question: assume we have a lot of data, can we find the best hypothesis?

Runtime of Learning NNs

Informally: applying the ERM rule with respect to $\mathcal{H}_{V,E,\mathrm{sign}}$ is *computationally difficult*, even for small NN...

Proposition

Let $k \geq 3$. For every d, let (V, E) be a layered graph with d input nodes, $k+1$ nodes at the (only) hidden layer, where one of them is the constant neuron, and a single output node. Then, it is NP-hard to implement the ERM rule with respect to $\mathcal{H}_{V,E,\mathrm{sign}}$.

Well maybe the above is only for very specific cases...

- instead of ERM rule, find h close to ERM? **Computationally infeasible!** (probably)
- other activation functions (e.g., sigmoid)? **Computationally infeasible!** (probably)
- smart embedding in larger network? **Computationally infeasible!** (probably)