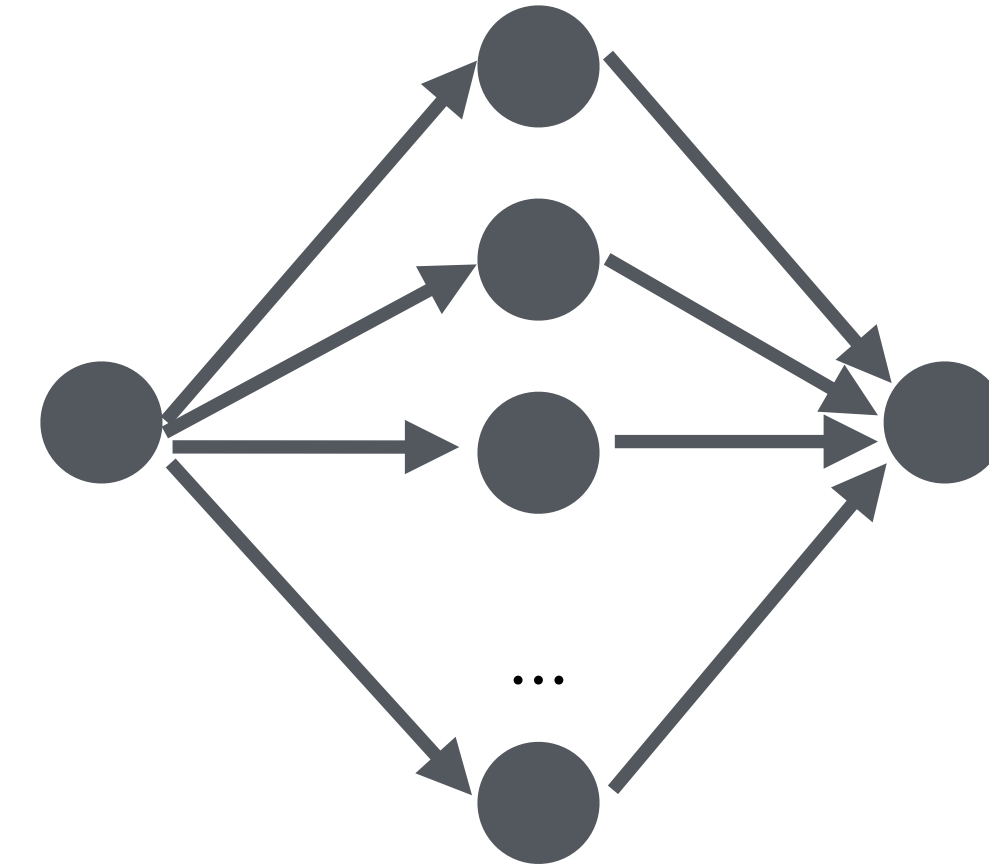


4. Approximation: 3-Layer ReLU net

Recap

- In the last class, we proved our first universal approximation result
 - Simple case, of **1D function** with **2-layer threshold nets**
- In particular, we proved that:



Proposition 2.1.

Suppose that $g : \mathbb{R} \rightarrow \mathbb{R}$ is ρ -Lipschitz. Then, for any $\varepsilon > 0$, there exists a 2-layer network with $\lceil \rho/\varepsilon \rceil$ threshold nodes, so that

$$\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$$

Distilling the Key Ideas

Proposition 2.1.

Suppose that $g : \mathbb{R} \rightarrow \mathbb{R}$ is ρ -Lipschitz. Then, for any $\varepsilon > 0$, there exists a 2-layer network with $\lceil \rho/\varepsilon \rceil$ threshold nodes, so that

$$\sup_{x \in [0,1]} |f(x) - g(x)| \leq \varepsilon$$

- Recall that the proof consists of two large steps:
 - Approximating $g(\cdot)$ with $h(\cdot)$ — a weighted sum of simple bases

$$h(x) = \sum_{i=1}^k \alpha_i \cdot b_i(x)$$

- Approximating $h(\cdot)$ with $f(\cdot)$ — a weighted sum of neurons

$$f(x) = \sum_{i=1}^m a_i \cdot n_i(x)$$

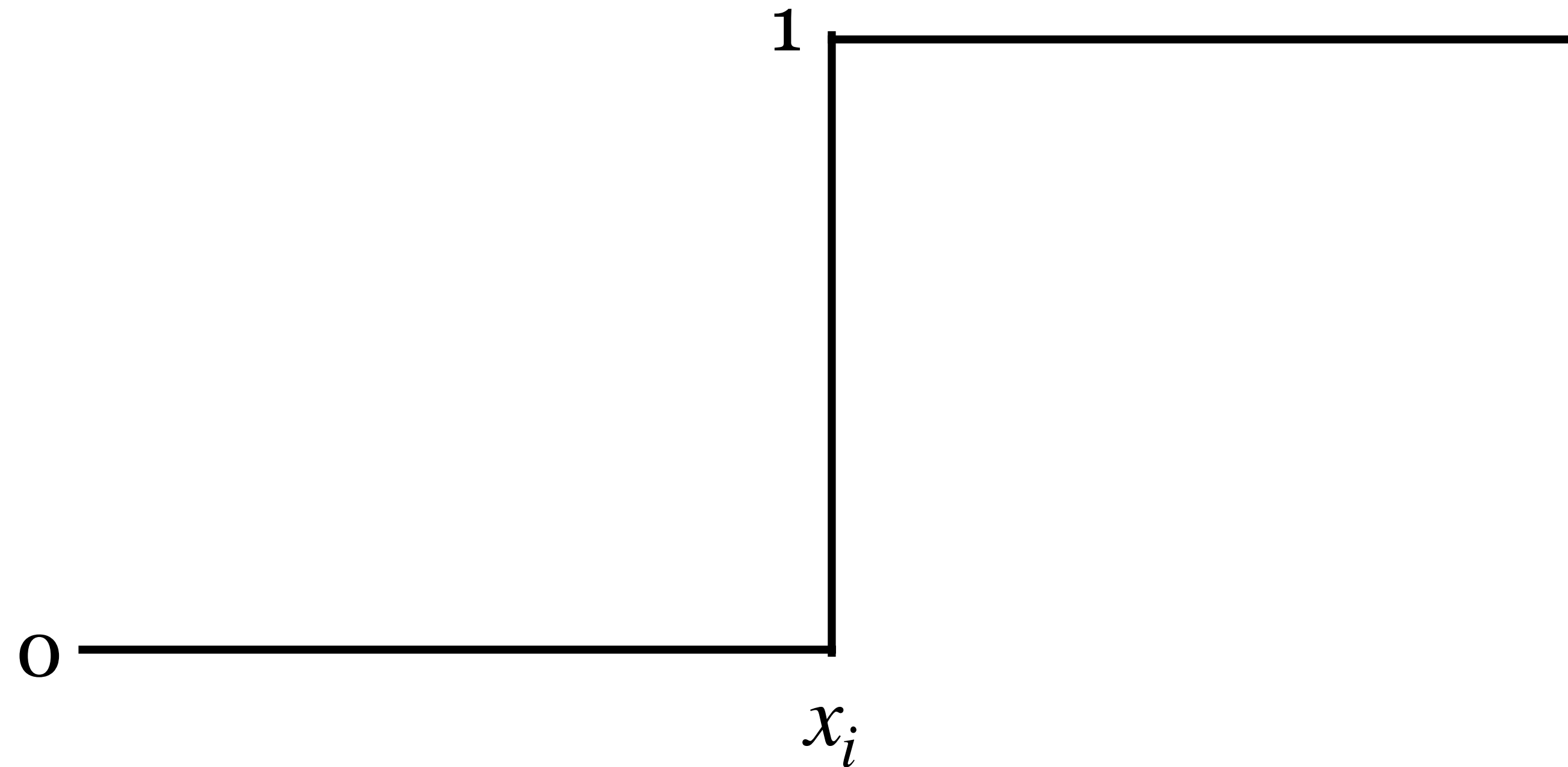
- Establish the relationship $g(\cdot) \approx h(\cdot) \approx f(\cdot)$

Distilling the Key Ideas

Step 1. Approximating $g(\cdot)$ with $h(\cdot)$ — a weighted sum of simple basis

- As the basis, we used **shifted step functions**

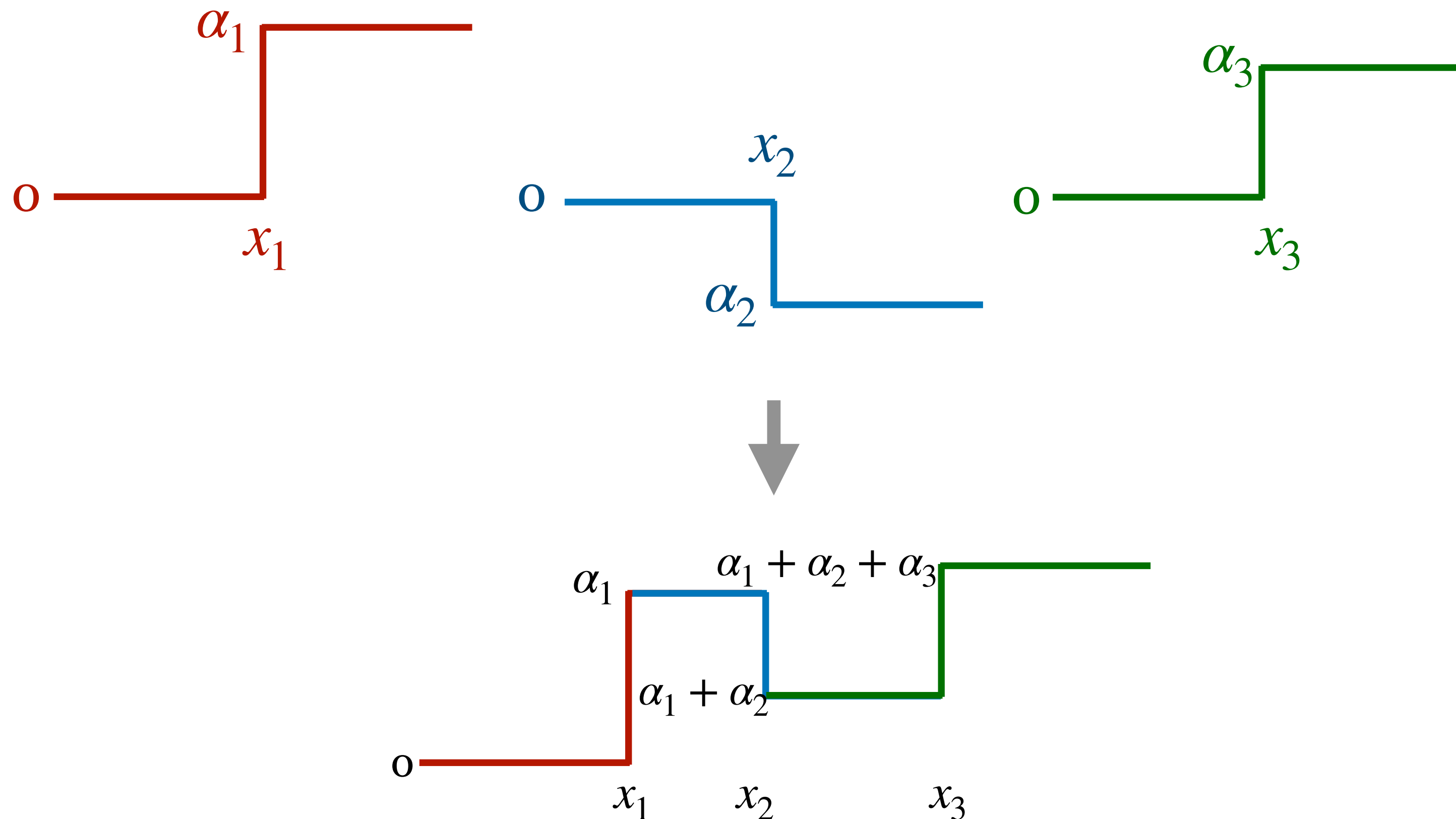
$$b_i(x) = \mathbf{1}\{x - x_i\}$$




Distilling the Key Ideas

- Weighted sum of bases gives **histogram-like function**

$$h(x) = \sum_{i=1}^k \alpha_i \cdot b_i(x)$$



Distilling the Key Ideas

- The histogram $h(\cdot)$ can approximate **any Lipschitz function** $g(\cdot)$ arbitrarily closely
 - Consider finer and finer granularity (i.e., increased k) 

Distilling the Key Ideas

Step 2. Approximating $h(\cdot)$ with $f(\cdot)$ — a weighted sum of neurons

- Two-layer threshold net can be written as a weighted sum of threshold neurons

$$f(x) = \sum_{i=1}^m a_i \cdot n_i(x)$$

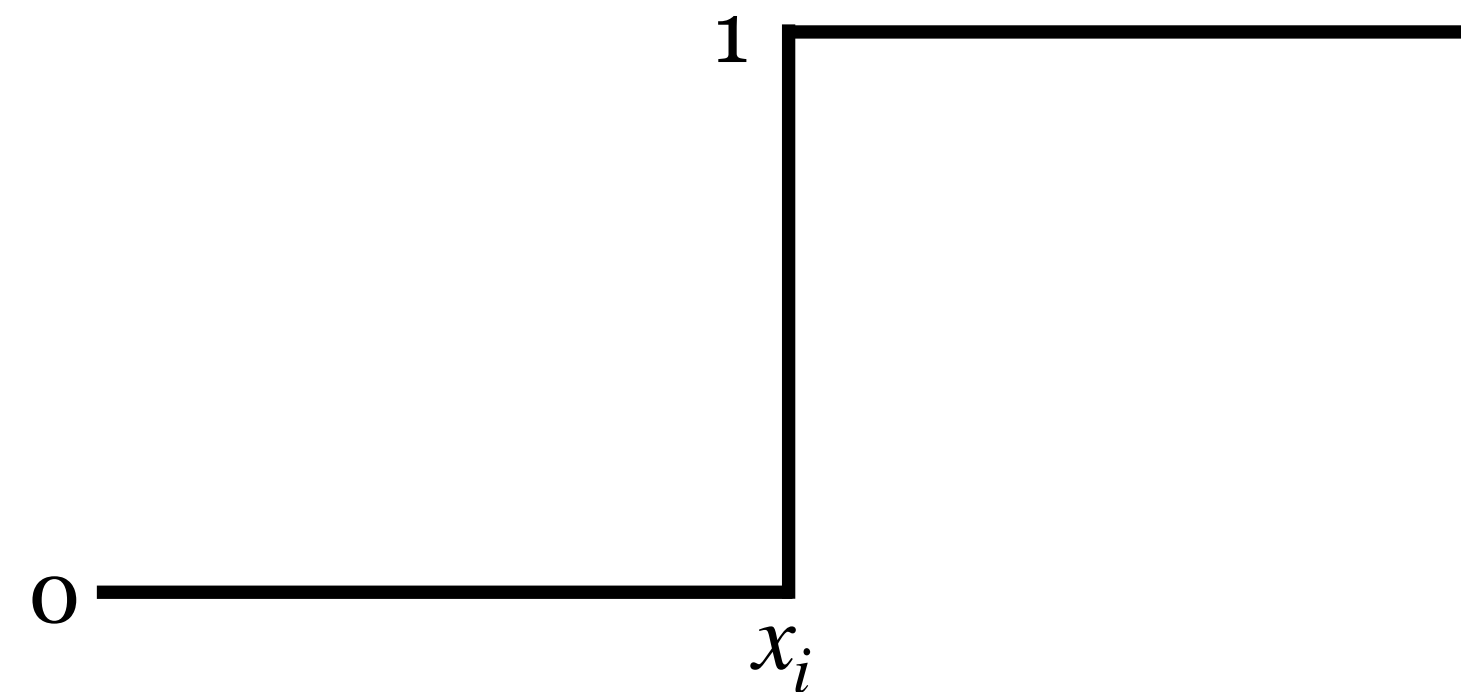
- Each threshold neuron can be written as:

$$n_i(\cdot) = \mathbf{1}\{w_i x + b_i \geq 0\}$$

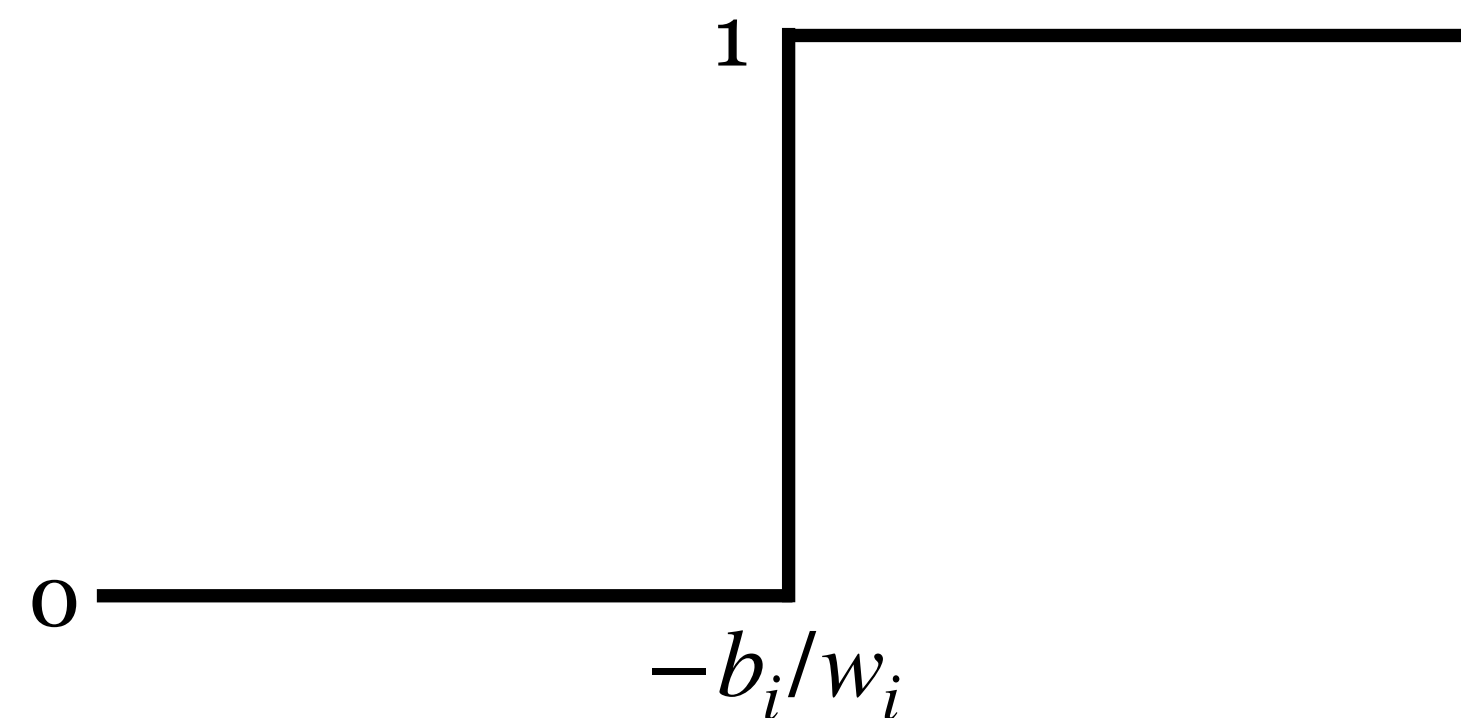
Distilling the Key Ideas

- Approximate each basis $b_i(\cdot)$ with neuron(s)!
 - Luckily, we can represent each basis using **one threshold neuron, without error**

- Basis $b_i(\cdot)$



- Neuron $n_i(\cdot)$



- Thus, k -basis $h(\cdot)$ can be approximated by k -neuron network $f(\cdot)$, with $\|h(\cdot) - f(\cdot)\|_\infty = 0$

Distilling the Key Ideas

- Thus we have:
 - $\|g(\cdot) - h(\cdot)\|_\infty < \varepsilon$, for some good k
 - $\|h(\cdot) - f(\cdot)\|_\infty = 0$

- By triangle inequality:

$$\begin{aligned}\|g - f\|_\infty &\leq \|g - h\|_\infty + \|h - f\|_\infty \\ &< \varepsilon + 0\end{aligned}$$

Today

- We depart from non-realistic assumptions
 - Use **ReLU**, not threshold
 - Use **d -dimensional input**, not 1-dimensional
- This is trickier, so we will first:
 - Use **three-layers**, not two
 - Upper-bound **L_1 norm**, not uniform

Main result

- In particular, we prove the following result

Theorem 2.1.

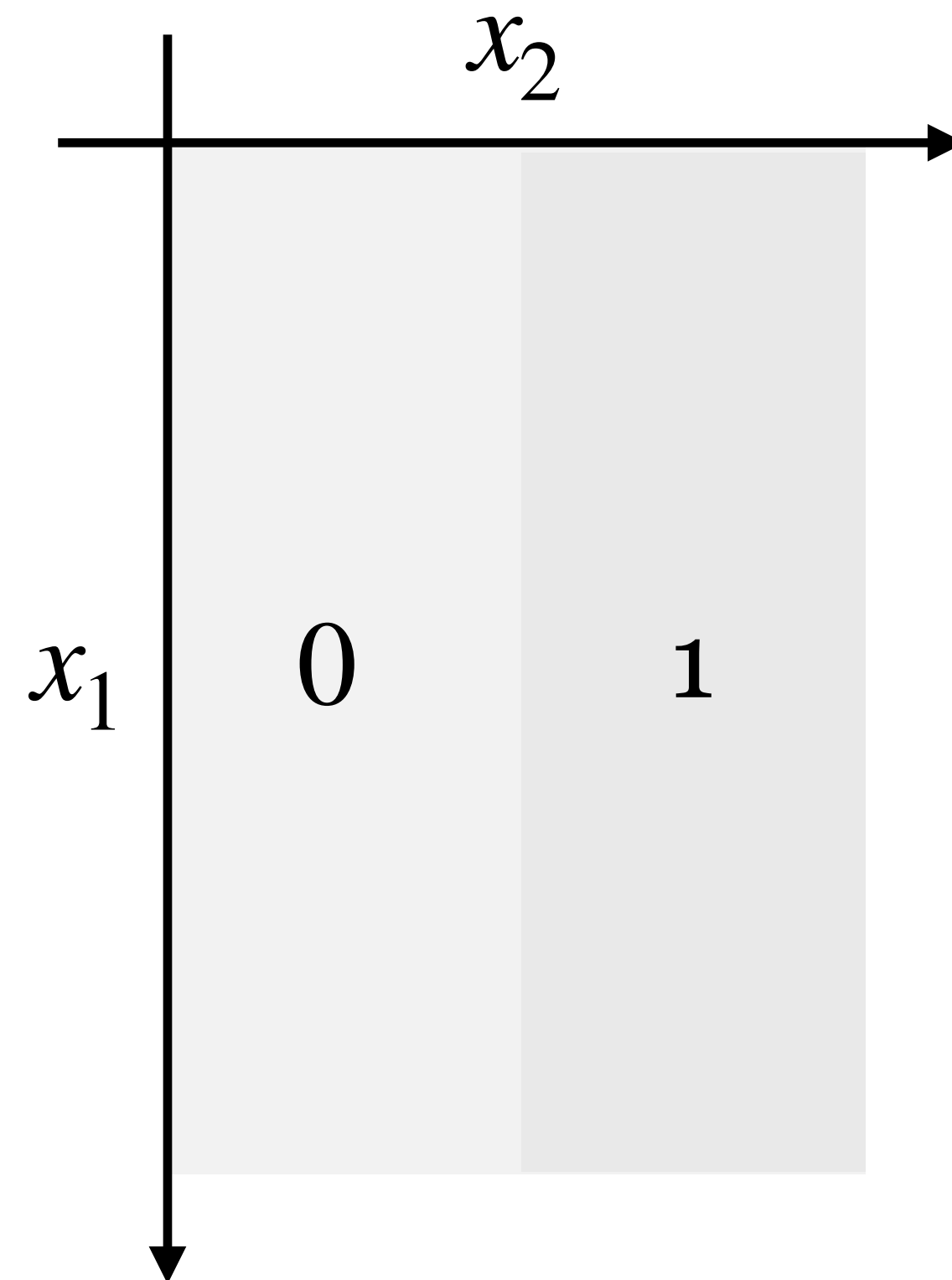
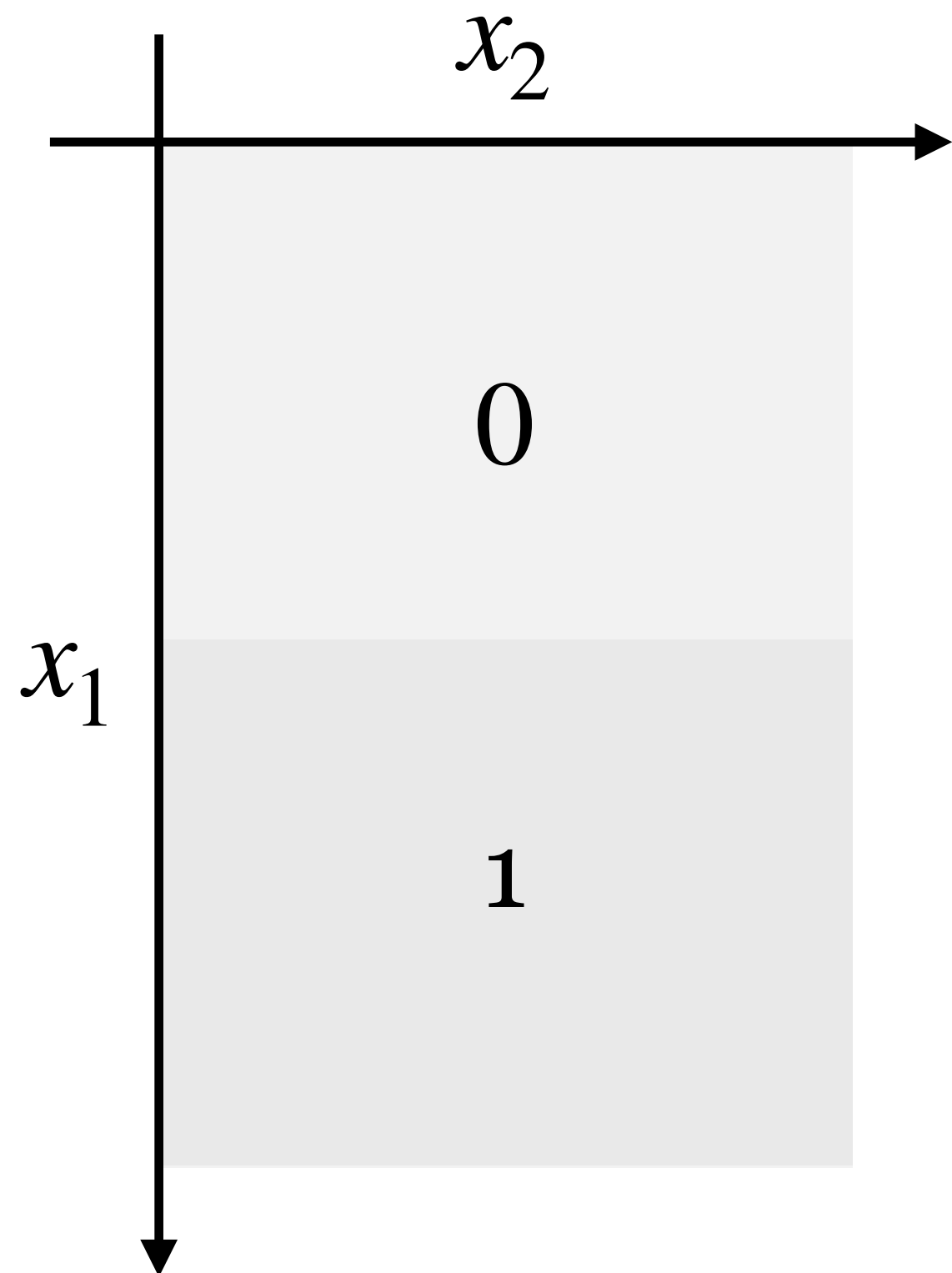
Let $g : [0,1]^d \rightarrow \mathbb{R}$ be a ρ -Lipschitz function, and let $\varepsilon > 0$. Then, there exists a three-layer ReLU network with $O(d \cdot \lceil \rho/\varepsilon \rceil^d)$ neurons such that

$$\int_{[0,1]^d} |f(\mathbf{x}) - g(\mathbf{x})| \, dx \leq 2\varepsilon$$

- Here, the Lipschitz constant is w.r.t. $\|\cdot\|_\infty$ at the input and $|\cdot|$ at the output
- Our strategy is basically the same:
 - Build a nice $h(\cdot)$, but with **d -dimensional bases**
 - Approximate each basis with neurons, but with **ReLU neurons**
- Let's think about these questions first, then move onto the final proof

Basis

- In 1D, we used step functions...
 - Can we also use **axis-parallel 1D step functions**?



Basis

- **Problem.** Not exactly sure whether we can decouple everything...
 - Brainteaser: Can anybody prove that we can/cannot do this?

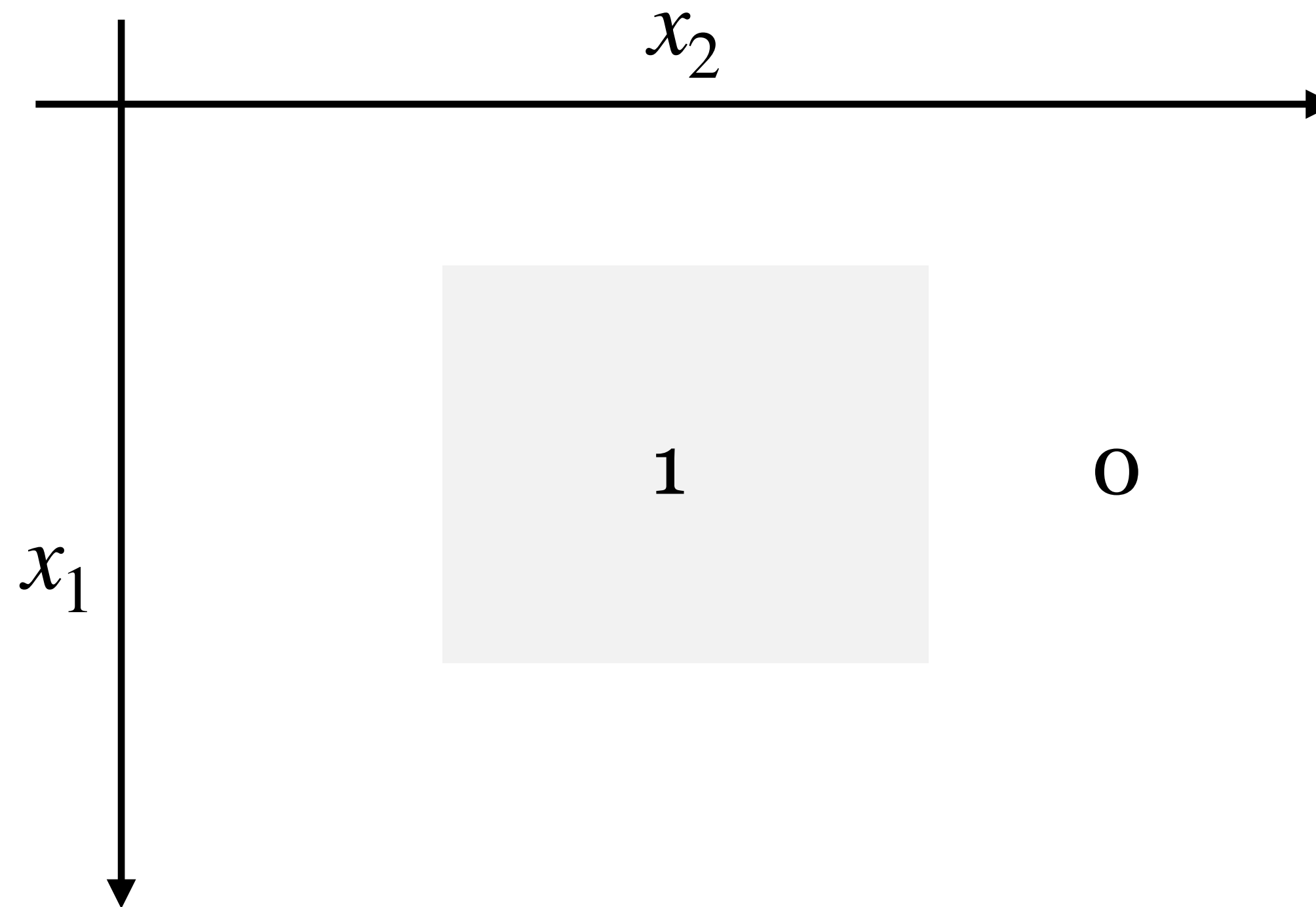
	x_2		
	0	α_2	$\alpha_2 + \alpha_4$
x_1	α_1	$\alpha_1 + \alpha_2$	$\alpha_1 + \alpha_2 + \alpha_4$
	$\alpha_1 + \alpha_3$	$\alpha_1 + \alpha_2 + \alpha_3$	$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$

Basis

- **Choice.** For now, we will use “high-D pulse” as our basis

$$b_i(\mathbf{x}) = \mathbf{1}\{\mathbf{x} \in R_i\} \quad (R_i \text{ is some hypercube})$$

- Much easier to approximate $g(\cdot)$
- Somewhat involved, to be approximated by ReLU neurons



ReLU for 1D pulse

- Now let's think about what a ReLU neuron can approximate:

$$\sigma(\mathbf{w}_i^\top \mathbf{x} + b_i)$$

- For 1D input, a single ReLU neuron looks like this:



- **Question.** Can this approximate 1D pulse, i.e., $\mathbf{1}\{x \in [a, b]\}$?

ReLU for 1D pulse

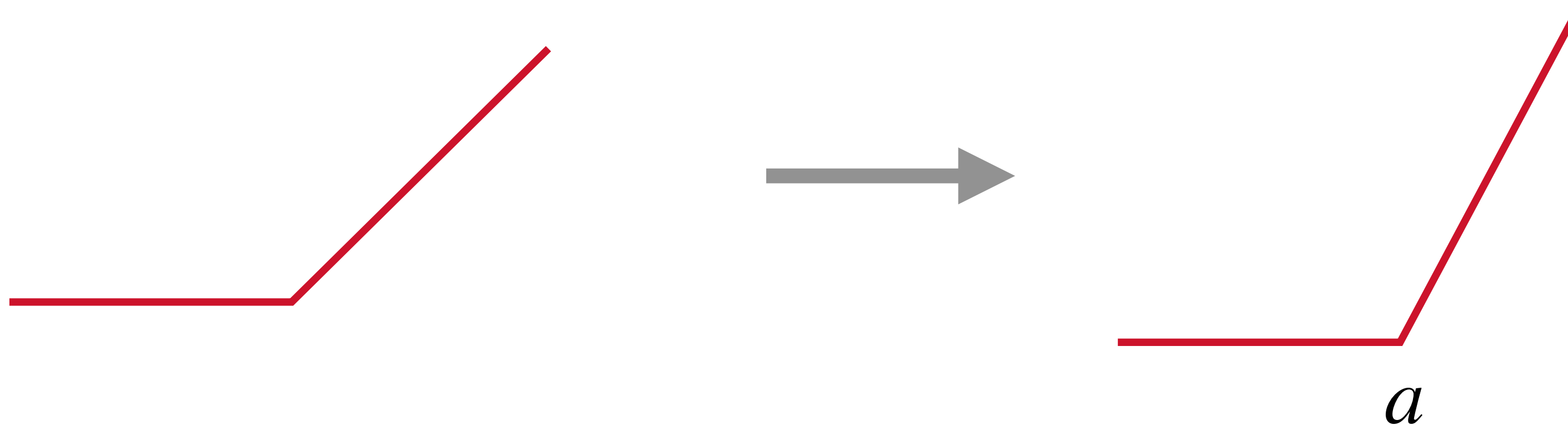
- We do it in three steps:

Step 1. Sharpen up ReLU

- Build ReLU like:

$$\sigma\left(\frac{x-a}{\gamma}\right)$$

using very small γ ($w = 1/\gamma, b = -a/\gamma$)



ReLU for 1D pulse

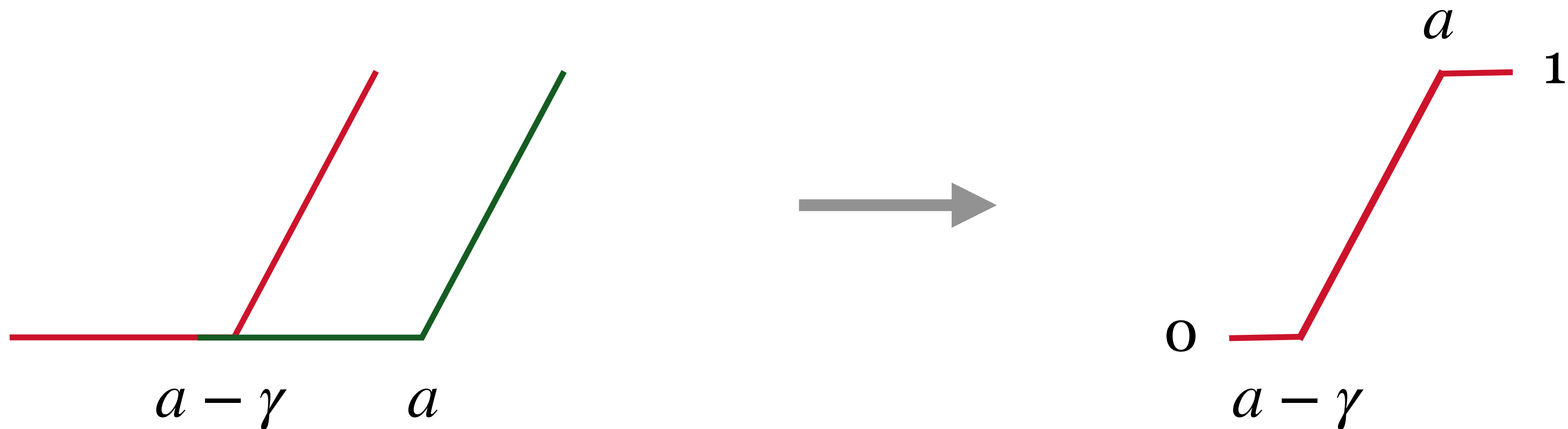
- We do it in three steps:

Step 2. Combine two ReLU to build a hard sigmoid

- Group two ReLU neurons

$$s_{a,\gamma}(x) = \boxed{\sigma\left(\frac{x - (a - \gamma)}{\gamma}\right)} - \boxed{\sigma\left(\frac{x - a}{\gamma}\right)}$$

- Brainteaser: What if we used sigmoid activations from the first place?



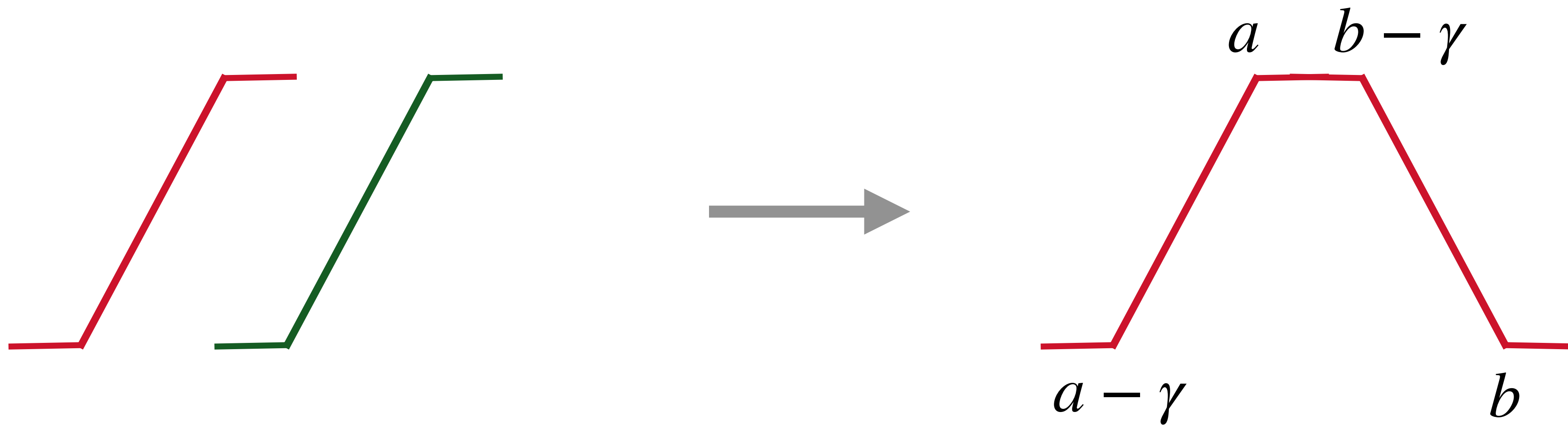
ReLU for 1D pulse

- We do it in three steps:

Step 3. Combine two hard sigmoids to build a pulse

- Group two hard sigmoids as:


$$u_{a,b,\gamma}(x) = \boxed{s_{a,\gamma}(x)} - \boxed{s_{b,\gamma}(x)}$$

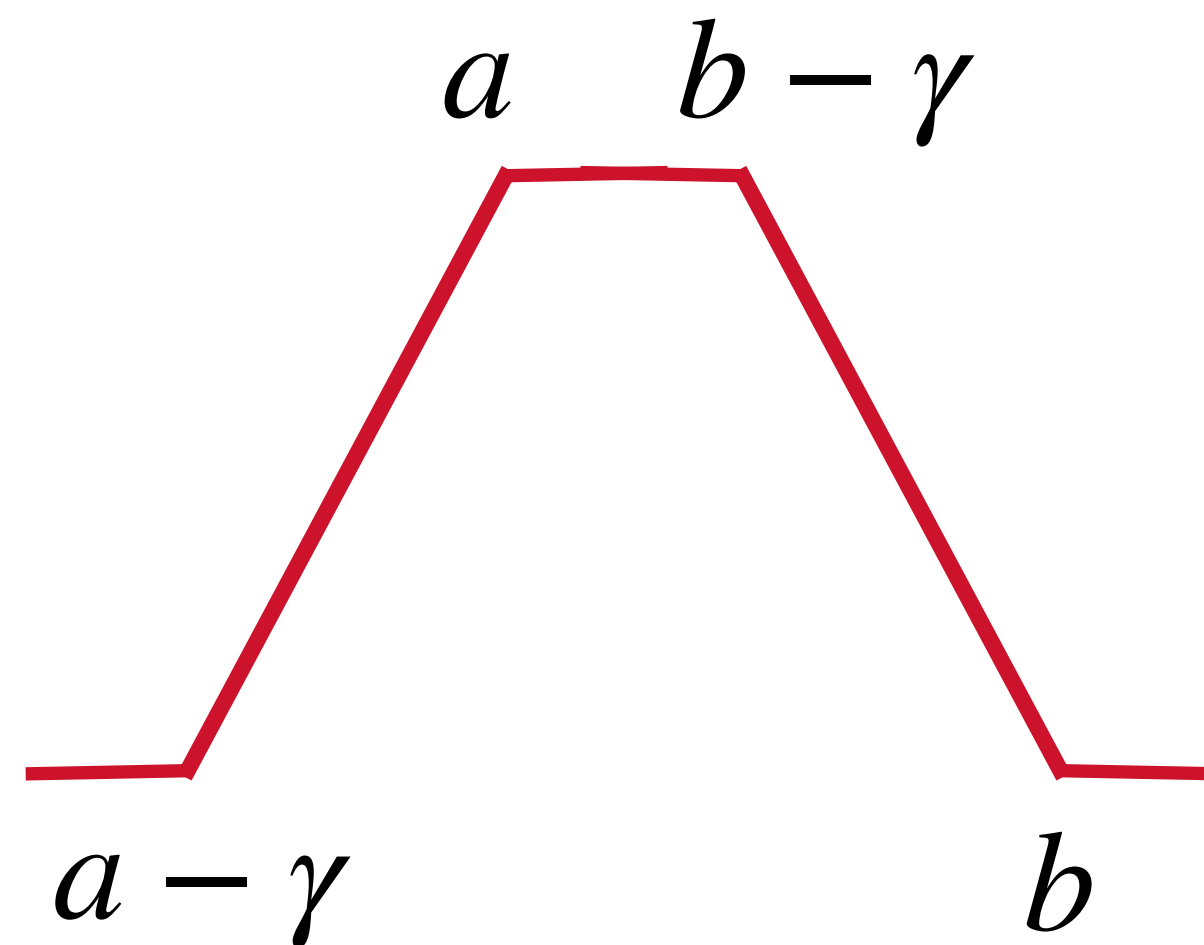


ReLU for 1D pulse

- Summing up, a pulse $\mathbf{1}\{x \in [a, b]\}$ can be approximated by:

$$u_{a,b,\gamma}(x) = \sigma\left(\frac{x - (a - \gamma)}{\gamma}\right) - \sigma\left(\frac{x - a}{\gamma}\right) - \sigma\left(\frac{x - (b - \gamma)}{\gamma}\right) + \sigma\left(\frac{x - b}{\gamma}\right)$$

- Question.** How good is this approximation?
 - Of course, the goodness depends on the sharpening factor γ
 - In fact, the L_1 error is exactly γ 



ReLU for multivariate pulse

- **Question.** How do we extend it to multivariate pulse?

$$\prod_i \mathbf{1}\{x_i \in [a_j, b_j]\}$$

ReLU for multivariate pulse

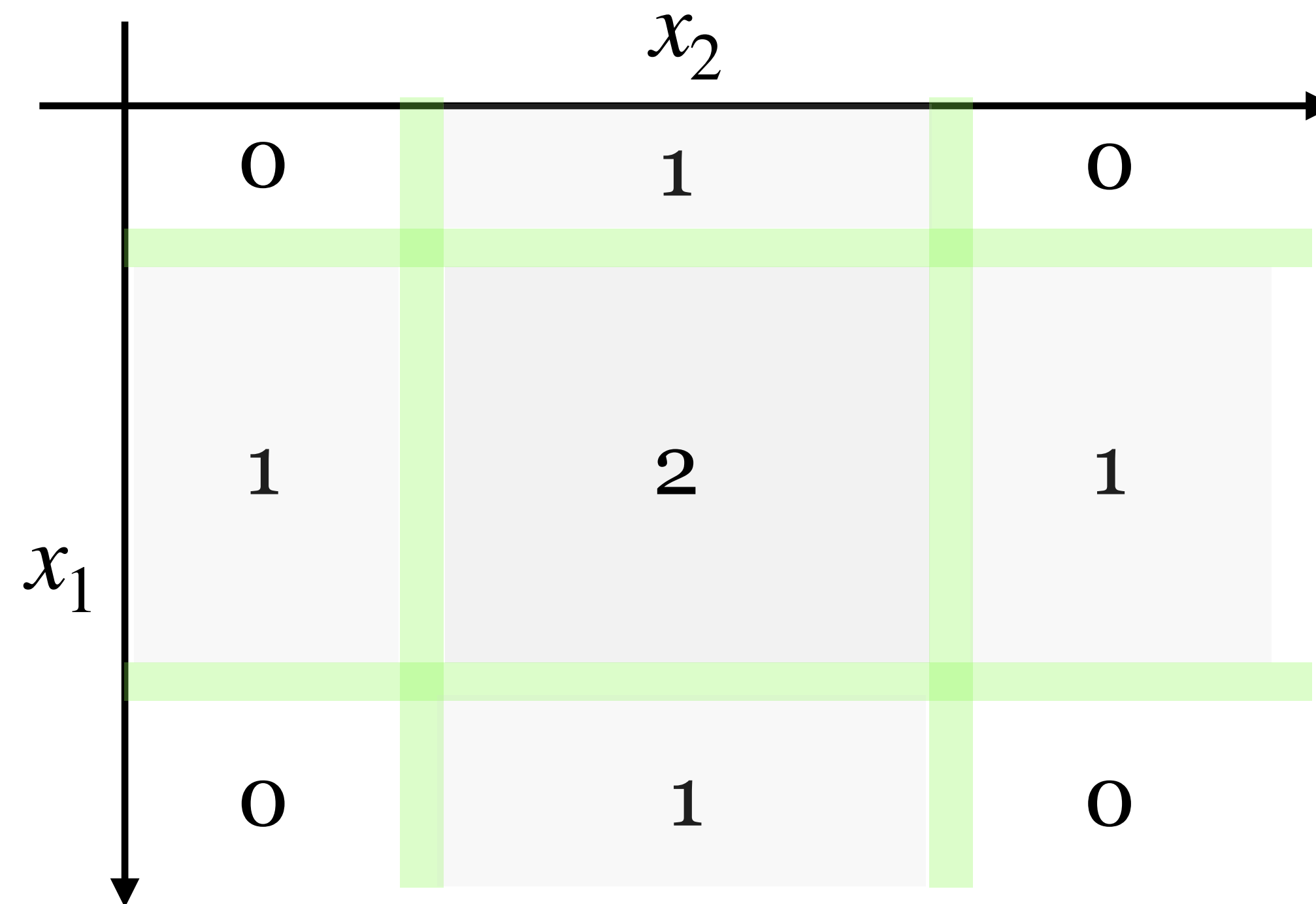
- Answer.
 - Very difficult if we use two layers (try it!)
 - Much easier if we use three layers

ReLU for multivariate pulse

- Again, we proceed in three steps:

Step 1. Sum univariate pulses in d axes

- We'll get something very ugly
- Total 4d ReLU neurons added, so far

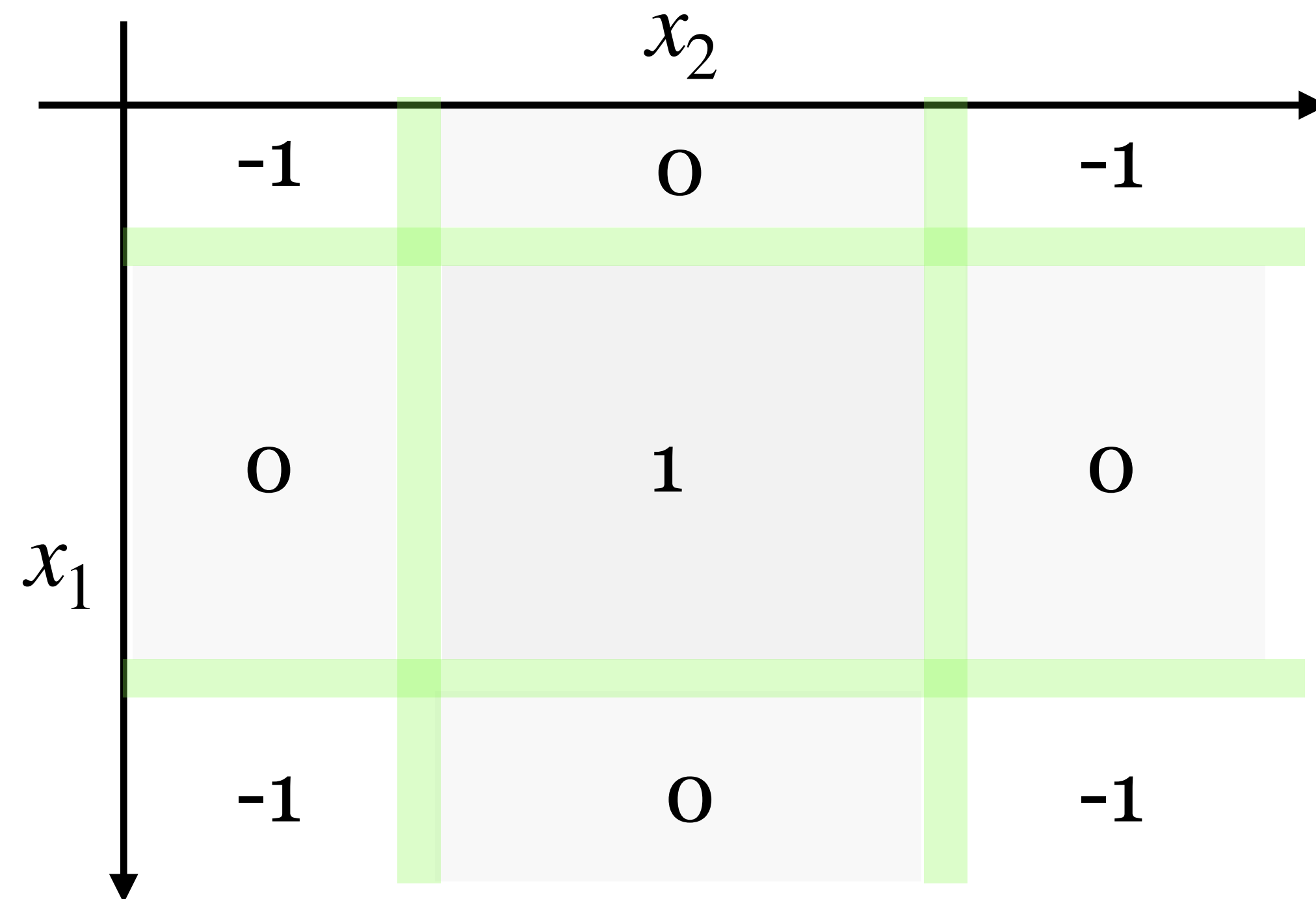


ReLU for multivariate pulse

- Again, we proceed in three steps:

Step 2. Subtract $d - 1$

- Think of this as adding a bias, while adding the second hidden layer

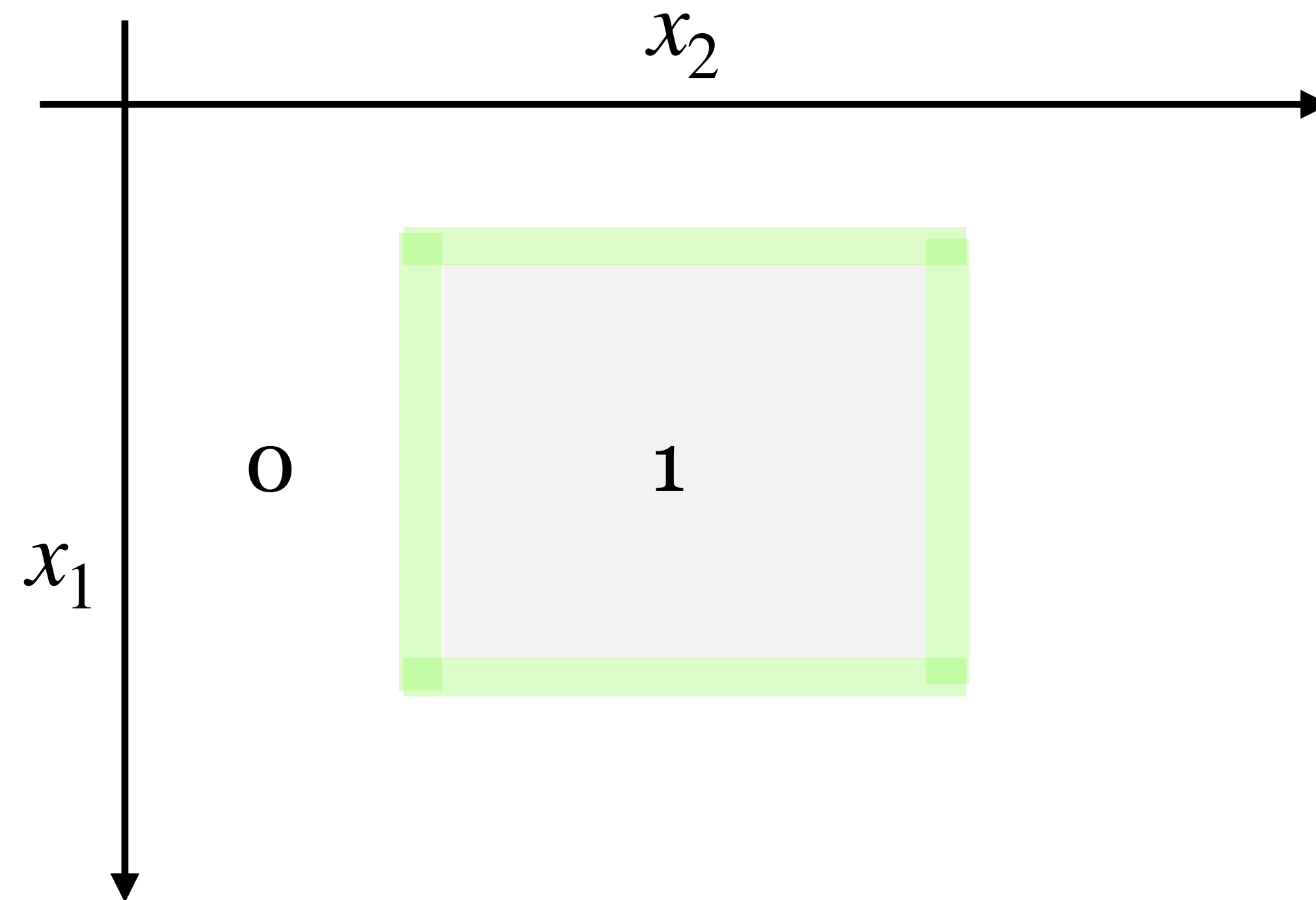


ReLU for multivariate pulse

- Again, we proceed in three steps:

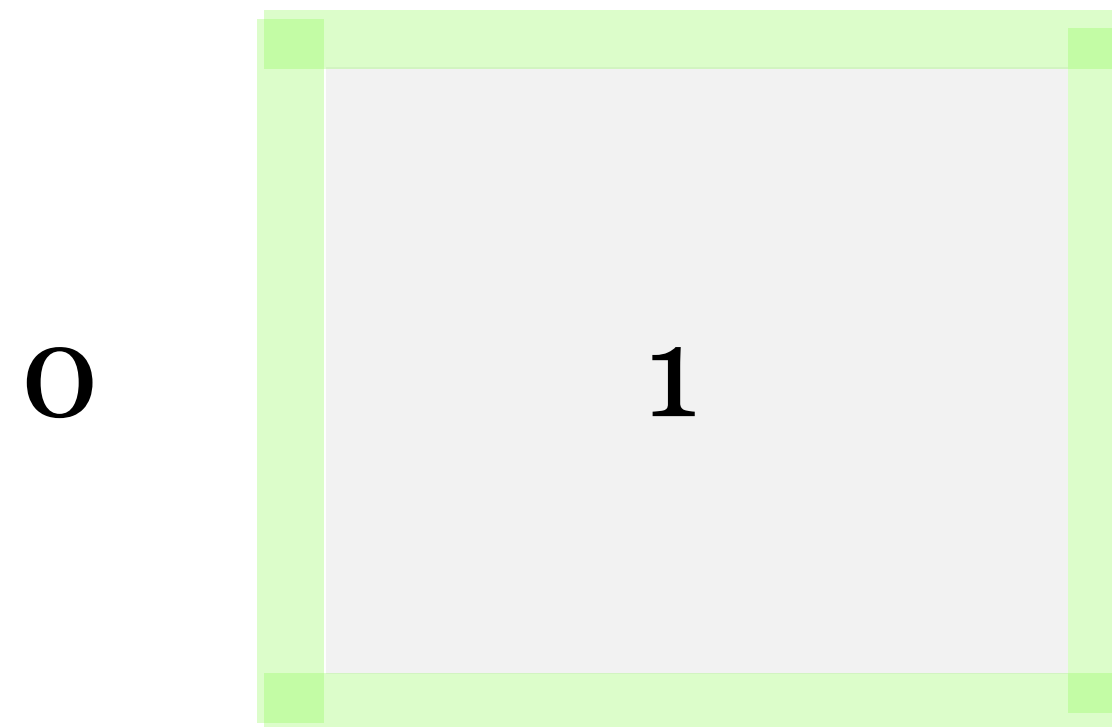
Step 3. Remove negative parts, with ReLU

- We have fully added the second hidden layer
 - Total number of neurons used: 4d in the 1st layer, 1 in the second layer



ReLU for multivariate pulse

- If we analyze the L1 error, it will be of order at most γ
 - Now we are ready for a formal proof!



Formalizing the claims

- We will prove three claims:
 - **Claim 1.** Existence of some $h(\cdot)$ with $\|g - h\|_{\infty} \leq \varepsilon$
 - **Claim 2.** Approximating bases with neurons
 - **Claim 3.** Connecting two claims

Formalizing the claims

Claim 1. There exists an $h(\cdot)$ such that

$$\|g(\mathbf{x}) - h(\mathbf{x})\|_{\infty} \leq \varepsilon, \quad \text{where} \quad h(\mathbf{x}) = \sum_{i=1}^N \alpha_i \cdot \mathbf{1}\{\mathbf{x} \in R_i\}$$

for $N = \lceil \rho/\varepsilon \rceil^d$.

Idea.

- Divide the domain $[0,1]^d$ into many sub-hypercubes, with sidelengths ε/ρ
- Select $\alpha_i = g(\mathbf{x}_{(i)})$, for some $\mathbf{x}_{(i)} \in R_i$

Formalizing the claims

Claim 2. For each R_i , there exists a three-layer neural net $f_i(\cdot)$ such that

$$\|f_i(\mathbf{x}) - \mathbf{1}\{\mathbf{x} \in R_i\}\|_1 \leq \frac{\varepsilon}{\sum_{i=1}^N |\alpha_i|}, \quad \forall i \in [N]$$

Idea.

- Already have the construction — all that remains is to choose the right γ

Formalizing the claims

Claim 3. We have $\|f - g\|_1 \leq 2\varepsilon$, for the three-layer neural network

$$f(\mathbf{x}) = \sum_{i=1}^N \alpha_i \cdot f_i(\mathbf{x})$$

Idea.

- Use the triangle inequality
- Congratulations — you have the claim!

Discussion

Theorem 2.1.

Let $g : [0,1]^d \rightarrow \mathbb{R}$ be a ρ -Lipschitz function, and let $\varepsilon > 0$. Then, there exists a three-layer ReLU network with $O(d \cdot \lceil \rho/\varepsilon \rceil^d)$ neurons such that

$$\int_{[0,1]^d} |f(\mathbf{x}) - g(\mathbf{x})| \, dx \leq 2\varepsilon$$

- We have used a lot of neurons — exponential dependence on d
 - Generally unavoidable, but gets better as we increase depth
- We can also use two layers
 - Requires sophisticated tools (next lecture)