Chapter 3

Feedforward Neural Networks

3.1 Model

A *feedforward neural network* (FFNN), or multilayer perceptron, is composed of alternating *linear layers* and nonlinear *activation functions.*

Definition 3.1. A *linear layer* is a function

$$\begin{aligned} \mathit{lin}\colon \mathbb{R}^{d} &\to \mathbb{R}^{d'} \\ \mathbf{x} &\mapsto \mathbf{W}\mathbf{x} + \mathbf{b} \end{aligned} \tag{3.1}$$

with attributes

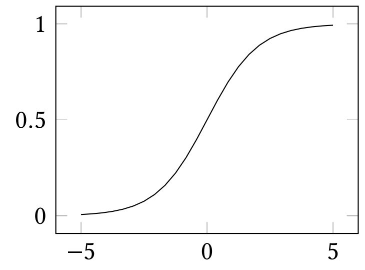
- $d \in \mathbb{N}$, called the *input size*
- $d' \in \mathbb{N}$, called the *output size*

and parameters

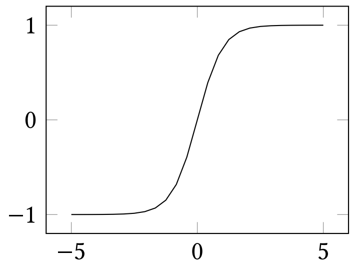
- $\mathbf{W} \in \mathbb{R}^{d' \times d}$, called the *weights*
- $\mathbf{b} \in \mathbb{R}^{d'}$, called the *bias.*

We'll use an idiosyncratic notation when defining and using neural networks (which should hopefully feel familiar, however, to anyone who has used PyTorch). If *lin* is a linear layer, we write *lin.d* for its input size, *lin.*$\mathbf{W}$ for its weights, and so on. In general, whenever we say that something is an "attribute" or "parameter" of a function, we use this dot notation to access it. A parameter of an attribute is also a parameter.

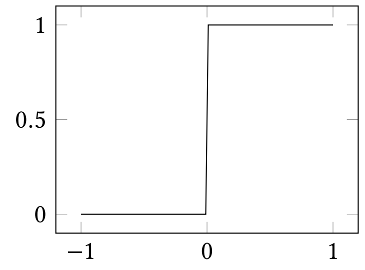
For the nonlinearities, several choices are common:



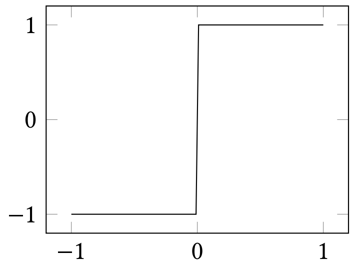
logistic sigmoid function
$\operatorname{sigmoid}(x) = \frac{1}{1+e^{-x}}$



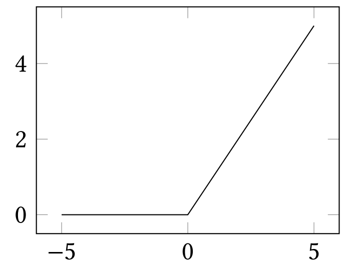
hyperbolic tangent
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



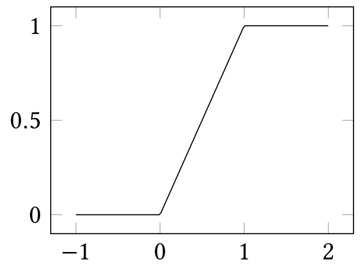
Heaviside step function
$$\operatorname{step}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$



sign function
$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ +1 & \text{if } x \geq 0 \end{cases}$$



rectified linear unit
$\operatorname{ReLU}(x) = \max\{0, x\}$



saturated linear unit
$\operatorname{SLU}(x) = \max\{0, \min\{1, x\}\}$

When applied to vectors, they are applied component-wise. That is, if σ is an activation function, then

$$\sigma\left(\begin{bmatrix} x_0 \\ \vdots \\ x_{d-1} \end{bmatrix}\right) = \begin{bmatrix} \sigma(x_0) \\ \vdots \\ \sigma(x_{d-1}) \end{bmatrix}. \tag{3.2}$$

Although sigmoid is a traditional activation function, our preference will be for ReLUs. They are often easier to work with in proofs, and they are also very widely used in practice.

Definition 3.2. A *feed-forward neural network* (FFNN) with σ activations (where σ is any activation function) is a function

$$\begin{aligned} \mathit{ffn}\colon \mathbb{R}^d &\to \mathbb{R}^{d'} \\ \mathbf{x} &\mapsto \mathbf{h}^{(L)} \text{ where} \\ \mathbf{h}^{(0)} &= \sigma(\mathit{lin}_0(\mathbf{x})) \\ &\vdots \\ \mathbf{h}^{(L-2)} &= \sigma(\mathit{lin}_{L-1}(\mathbf{h}^{(L-3)})) \\ \mathbf{h}^{(L-1)} &= \mathit{lin}_L(\mathbf{h}^{(L-2)}) \end{aligned} \tag{3.3}$$

with attributes

- $d > 0$, $d' > 0$, called the *input* and *output size*
- $L > 0$, called the *depth*
- linear layers lin_ℓ for $\ell \in [L]$, such that

  $$\begin{aligned} \mathit{lin}_0.d &= d \\ \mathit{lin}_i.d &= \mathit{lin}_{\ell-1}.d' && i = 1, \dots, L-1 \\ \mathit{lin}_{L-1}.d' &= d'. \end{aligned}$$

  We call lin_{L-1} the *output layer* and all other layers *hidden layers*. (Thus a FFNN with depth L has $(L-1)$ hidden layers.)

The maximum input/output size of any layer of *ffn* is called the *width* of *ffn*.

Note that the output layer does not have a nonlinearity; I'm not sure how standard or nonstandard this is, but it's convenient for us.

We call the components of $\mathbf{h}^{(\ell)}$ *activation values*. We also call the whole vector $\mathbf{h}^{(\ell)}$ an "activation value," and sometimes an "activation vector" if we need to emphasize that we're talking about the whole vector. We sometimes call other values computed by the network "activation values" as well.

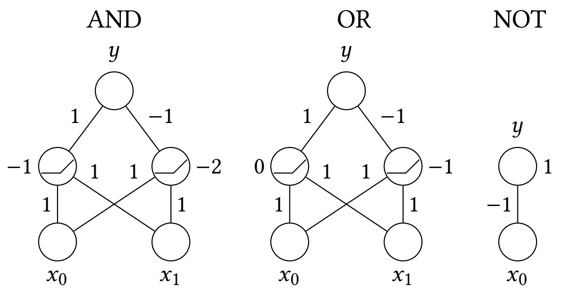
3.2 Expressivity

We'll review four expressivity results. First, to resolve the XOR problem from Theorem 2.4, we'll show that FFNNs can compute any Boolean function. Second, we'll exactly characterize all the functions that FFNNs with ReLU activations can compute. Third, for any Lipschitz-continuous function, we'll explicitly construct a depth-3 ReLU FFNN that approximates it. Finally, we'll prove the famous universal approximation theorem, that FFNNs can approximate any continuous function.

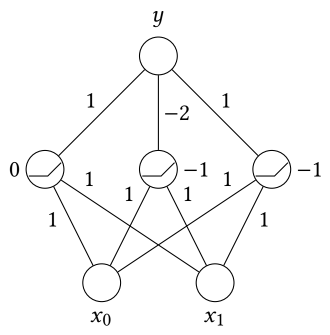
3.2.1 Exact: Boolean functions

In Section 2.2, we showed that perceptrons can compute AND, OR, and NOT. Unsurprisingly, FFNNs can as well, but we redo these examples using ReLUs and using 0 for false and 1 for true.

Example 3.3. The Boolean operations AND, OR, and NOT can be computed by FFNNs with ReLU activations. In the pictures below, nodes are units, edges are connections with their weights, and numbers written next to nodes are biases.



Example 3.4. Unlike perceptrons, FFNNs with ReLU activations can compute XOR:



Note that $\mathrm{XOR}(x_0, x_1) = \mathrm{OR}(x_0, x_1) - \mathrm{AND}(x_0, x_1)$.

Theorem 3.5. *Any Boolean function can be computed by a FFNN with* ReLU *activations.*

Proof. Just use the above constructions for AND, OR, and NOT. □

Definition (Full DNF).

Added definition 9/10; corrected 9/11

- A Boolean formula ϕ is a *literal* if it is of the form x or $\neg x$.
- A Boolean formula ϕ is a *monomial* if it is of the form

  $$\phi = \phi_0 \land \phi_1 \land \ldots \land \phi_{\ell-1}$$

  where each ϕ_i for $i \in [\ell]$ is a literal.
- A Boolean formula ϕ is in *disjunctive normal form* (DNF) if it is of the form

  $$\phi = \phi_0 \lor \phi_1 \lor \ldots \lor \phi_{m-1}$$

  where each ϕ_j for $j \in [m]$ is a monomial.
- A Boolean formula ϕ with free variables $x_0, \ldots, x_{n-1}$ is in *full DNF* if it is in DNF and, for any truth assignment to $x_0, \ldots, x_{n-1}$, at most one of the monomials is true.

Exercise 3.6. In fact, to prove Theorem 3.5, a two-layer network suffices. Any Boolean function f can be written as a Boolean formula in full DNF. Although it's not necessarily the most efficient, we can always do this by listing out all the inputs that make f true:

$$f(x_0, \ldots, x_{n-1}) = \bigvee_{\substack{\xi_0,\ldots,\xi_{n-1} \in \{0,1\} \\ f(\xi_0,\ldots\xi_{n-1})=1}} (x_0 \leftrightarrow \xi_0 \land \cdots \land x_{n-1} \leftrightarrow \xi_{n-1}).$$

For example, the XOR function looks like

$$\begin{aligned} \mathrm{XOR}(x_0, x_1) &= (x_0 \leftrightarrow 0 \land x_1 \leftrightarrow 1) \lor (x_0 \leftrightarrow 1 \land x_1 \leftrightarrow 0) \\ &= (\neg x_0 \land x_1) \lor (x_0 \land \neg x_1). \end{aligned}$$

Now please finish the proof!

3.2.2 Exact: continuous piecewise linear functions

Sticking with ReLU activations, we can give a simple characterization of the functions that FFNNs compute.

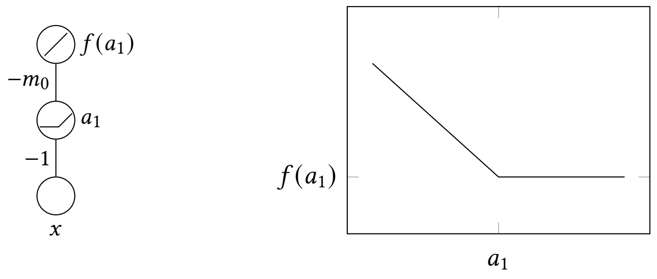
Theorem 3.7 (Arora et al., 2018)**.** *FFNNs with ReLU activations compute exactly the set of continuous piecewise linear functions with a finite number of pieces (or CPWL functions for short).*

Proof. We only prove the univariate ($d = 1$) case, which is easy and will come in handy later. Assume that f has the form:

$$f(x) = \begin{cases} m_0 x + b_0 & x < a_1 \\ m_1 x + b_1 & a_1 \le x < a_2 \\ \vdots & \\ m_{k-1} x + b_{k-1} & a_{k-1} \le x < a_k \\ m_k x + b_k & a_k \le x. \end{cases} \tag{3.4}$$

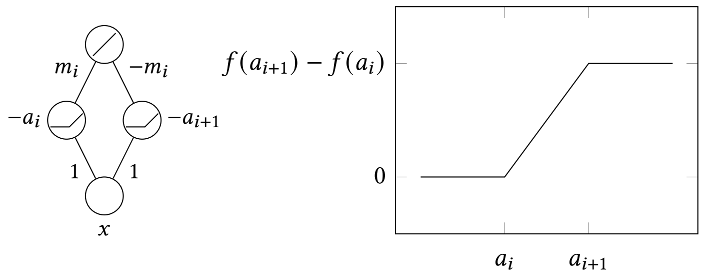
where $m_{i-1} a_i + b_{i-1} = m_i a_i + b_i$ for all i.

The first piece is made of a ReLU flipped left-to-right: Corrected 9/10



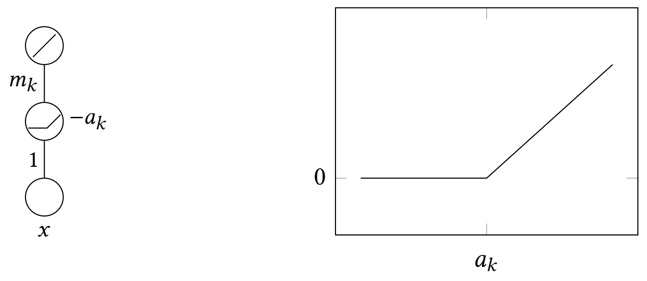
(3.5)

For each middle piece ($i = 1, \ldots, k-1$), we add in a "wedge" (also known as a *saturated linear unit*, or SLU) like this:



(3.6)

Finally, we add in one more ReLU for the last piece:



(3.7)

A brief sketch of the $d > 1$ case follows. Any CPWL function with k pieces can be written as the sum of several convex and concave CPWL functions (Wang and Sun, 2005). The convex ones can be written as the minimum of k affine functions, and the concave ones can be written as the maximum of k affine functions. All of this can be done by a ReLU FFNN, although the k-way minimum or maximum requires $O(\log k)$ depth. □

Exercise 3.8. Even for the $d = 1$ case, our proof of Theorem 3.7 is sketchy. Please write a more formal proof, writing out the definition of the FFNN in terms of Eq. (3.4).

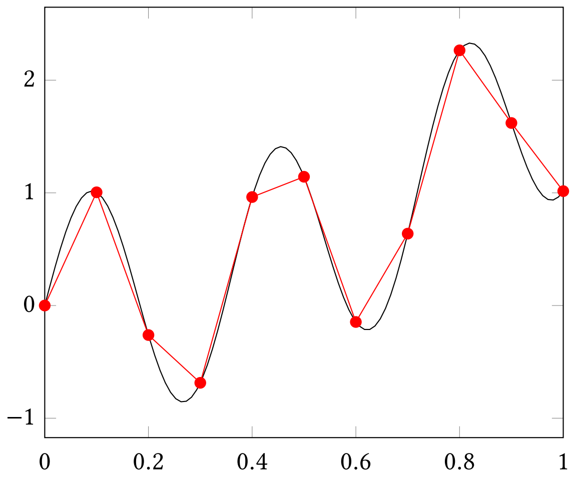
3.2.3 Approximate: some simpler cases

Before getting to the universal approximation theorem, which is very general but doesn't explicitly construct a network, we look at some simpler cases for which we can give an explicit construction.

Definition 3.9. Let $X \subseteq \mathbb{R}^d$. For any $\rho > 0$, a function $f : X \to \mathbb{R}$ is *ρ-Lipschitz-continuous* $|f(\mathbf{x}') - f(\mathbf{x})| \le \rho \|\mathbf{x}' - \mathbf{x}\|$ for all $\mathbf{x}, \mathbf{x}' \in X$.

Theorem 3.10. *For any ρ-Lipschitz-continuous function $f : [0,1] \to \mathbb{R}$ and any $\epsilon > 0$, there is a FFNN ffn with two layers and ReLU activations, such that for all $x \in [0,1]$, $|f(x) - \mathit{ffn}(x)| \le \epsilon$.*

Proof. This proof is adapted from Telgarsky (2021), but uses ReLU instead of sgn activations. The idea is to choose a δ depending on ϵ and construct a piecewise linear function that goes through all the points $(i\delta, f(i\delta))$ for all i such that $0 \le i\delta \le 1$.



Choose $\delta = \epsilon/(2\rho)$, and use Theorem 3.7 to construct the FFNN *ffn* to go through all points $(i\delta, f(i\delta))$ for all i such that $0 \leq i\delta \leq 1$; the first and last pieces can just be horizontal. If $\mathit{ffn}(i\delta) = f(i\delta)$, then for any $x \in [i\delta, (i+1)\delta]$, we have

$$|\mathit{ffn}(x) - f(x)| \leq |\mathit{ffn}(x) - f(i\delta)| + |f(i\delta) - f(x)|. \qquad \text{triangle inequality}$$

Working on the second term,

Explanation added 9/9

$$\begin{aligned} |f(i\delta) - f(x)| &\leq \rho|i\delta - x| && \text{Lipschitz continuity} \\ &\leq \rho\delta \\ &= \epsilon/2. \end{aligned}$$

Working on the first term,

$$\begin{aligned} |\mathit{ffn}(x) - f(i\delta)| &= |\mathit{ffn}(x) - \mathit{ffn}(i\delta)| \\ &\leq |\mathit{ffn}(i\delta) - \mathit{ffn}((i+1)\delta)| && \text{because } \mathit{ffn} \text{ is a straight line} \\ &= |f(i\delta) - f((i+1)\delta)| \\ &\leq \epsilon/2 && \text{by the same argument as before.} \end{aligned}$$

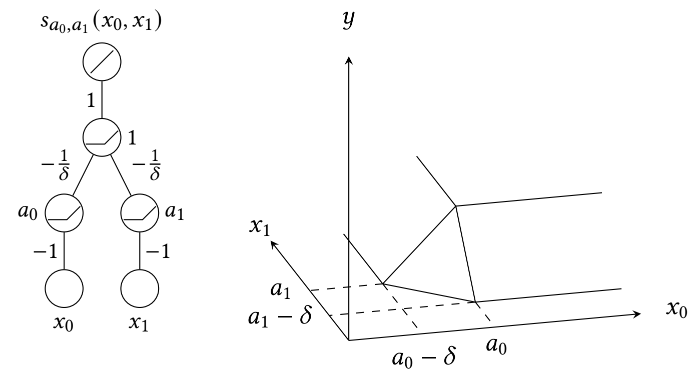
Putting them back together we have

$$|\mathit{ffn}(x) - f(x)| \leq \epsilon/2 + \epsilon/2 = \epsilon. \qquad \square$$

Theorem 3.11. *For any ρ-Lipschitz-continuous function $f\colon [0,1]^d \to \mathbb{R}$ and any $\epsilon > 0$, there is a FFNN ffn with three layers and ReLU activations, such that for all $\mathbf{x} \in [0,1]^d$, $|f(\mathbf{x}) - \mathit{ffn}(\mathbf{x})| \leq \epsilon$.*

Proof. This proof is similar to the previous one, but requires an additional hidden layer.

We just show how to construct ffn for $d = 2$. Define a two-dimensional wedge, s_{a_0,a_1}, with height 1:



The FFNN is going to be a linear combination of wedges:

$$\mathit{ffn}(x_0, x_1) = \sum_i \sum_j c_{ij}\, s_{i\delta,j\delta}(x_0, x_1). \tag{3.8}$$

We need to choose the coefficients c_{ij} such that for all i, j,

$$\sum_{i'} \sum_{j'} c_{i'j'} s_{i'\delta,j'\delta}(i\delta, j\delta) = f(i\delta, j\delta). \tag{3.9}$$

To see why Eq. (3.9) must have a solution, note that for each wedge,

$$s_{i'\delta,j'\delta}(i\delta, j\delta) = \begin{cases} 0 & \text{if } i < i' \text{ or } j < j' \\ 1 & \text{if } i \geq i' \text{ and } j \geq j' \end{cases} \tag{3.10}$$

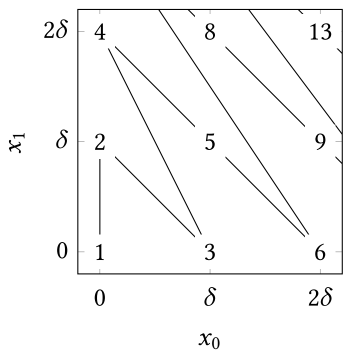
So Eq. (3.9) can be rewritten as:

$$\sum_{i'=0,\ldots,i} \sum_{j'=0,\ldots,j} c_{i'j'} = f(i\delta, j\delta). \tag{3.11}$$

Then solving for c_{ij}, we get:

$$c_{ij} = -\sum_{\substack{i'=0,\ldots,i \\ j'=0,\ldots,j \\ (i',j')\neq(i,j)}} c_{i'j'} + f(i\delta, j\delta). \tag{3.12}$$

To see that this must have a solution, imagine constructing the wedges in this order:



And so on. Each coefficient c_{ij} is defined only in terms of coefficients that came before it.

The last step is to show that the approximation error is at most ϵ. Choose $\delta = \epsilon/(2\rho\sqrt{2})$. Then, for $x_0 \in [i\delta, (i+1)\delta]$ and $x_1 \in [j\delta, (j+1)\delta]$,

Explanation added 9/9

$$|\mathit{ffn}(x_0, x_1) - f(x_0, x_1)| \le |\mathit{ffn}(x_0, x_1) - f(i\delta, j\delta)| + |f(i\delta, j\delta) - f(x_0, x_1)|.$$

Working on the second term,

$$\begin{aligned}
|f(i\delta, j\delta) - f(x_0, x_1)| &\le \rho\sqrt{(i\delta - x_0)^2 + (j\delta - x_1)^2} && \text{Lipschitz continuity} \\
&\le \rho\sqrt{\delta^2 + \delta^2} \\
&= \rho\delta\sqrt{2} \\
&= \epsilon/2.
\end{aligned}$$

The first term, by reasoning analogous to the proof of Theorem 3.10 and the above, is also at most $\epsilon/2$. So

$$|\mathit{ffn}(x_0, x_1) - f(x_0, x_1)| \le \epsilon.$$

□

Exercise 3.12. (a) Prove the case $d > 2$.

(b) What is the width of *ffn*, as a function of d, ρ, and ϵ?

3.2.4 The universal approximation theorem

Finally, the universal approximation theorem generalizes to any continuous function and requires only depth two. The proof depends on the following theorem, which we don't prove:

Theorem 3.13 (Stone–Weierstrass). *Let $\mathcal{F}$ be a family of functions $\mathbb{R}^d \to \mathbb{R}$ such that*

1. *Each $f \in \mathcal{F}$ is continuous.*
2. *For all $\mathbf{x} \in \mathbb{R}^d$, there is an $f \in \mathcal{F}$ such that $f(\mathbf{x}) \neq 0$.*
3. *For all $\mathbf{x} \neq \mathbf{x}' \in \mathbb{R}^d$, there is an $f \in \mathcal{F}$ such that $f(\mathbf{x}) \neq f(\mathbf{x}')$.*
4. *If $f, g \in \mathcal{F}$, then $f + g \in \mathcal{F}$, $fg \in \mathcal{F}$, and $cf \in \mathcal{F}$ for $c \in \mathbb{R}$.*

Then $\mathcal{F}$ is universal in the following sense: For any continuous function $g \colon [0,1]^d \to \mathbb{R}$ and $\epsilon > 0$, there is an $f \in \mathcal{F}$ such that for all $\mathbf{x} \in [0,1]^d$, $|f(\mathbf{x}) - g(\mathbf{x})| \leq \epsilon$.

Hornik et al. (1989) originally proved that FFNNs with cos activations are universal, but exp is easier (doesn't require you to remember your trigonometry identities) and also more plausible as an activation function.

Theorem 3.14 (Telgarsky, 2021). *FFNNs with* exp *activations are universal.*

Proof. Without loss of generality, we can ignore biases ($\mathbf{b}$). Let

$$\begin{aligned}
&\mathbf{x} \in \mathbb{R}^d \\
&\mathbf{U} \in \mathbb{R}^{m \times d} && \mathbf{u} \in \mathbb{R}^m \\
&\mathbf{V} \in \mathbb{R}^{n \times d} && \mathbf{v} \in \mathbb{R}^n \\
f(\mathbf{x}) &= \mathbf{u} \cdot \exp \mathbf{U}\mathbf{x} && (3.13) \\
g(\mathbf{x}) &= \mathbf{v} \cdot \exp \mathbf{V}\mathbf{x}. && (3.14)
\end{aligned}$$

Closure under scalar multiplication and addition are easy, and would work for any activation function:

$$\begin{aligned}
(cf)(\mathbf{x}) &= cf(\mathbf{x}) && (3.15) \\
&= c\mathbf{u} \cdot \exp \mathbf{U}\mathbf{x} && (3.16) \\
(f+g)(\mathbf{x}) &= f(\mathbf{x}) + g(\mathbf{x}) && (3.17) \\
&= \mathbf{u} \cdot \exp \mathbf{U}\mathbf{x} + \mathbf{v} \cdot \exp \mathbf{V}\mathbf{x} && (3.18) \\
&= \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \cdot \exp \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \mathbf{x}. && (3.19)
\end{aligned}$$

Closure under multiplication relies on the fact that $\exp a \exp b = \exp(a+b)$ for any a, b:

$$\begin{aligned}
(fg)(\mathbf{x}) &= f(\mathbf{x})\, g(\mathbf{x}) && (3.20)\\
&= (\mathbf{u} \cdot \exp \mathbf{U}\mathbf{x})\ (\mathbf{v} \cdot \exp \mathbf{V}\mathbf{x}) && (3.21)\\
&= \left(\sum_i \mathbf{u}[i] \exp(\mathbf{U}\mathbf{x})[i]\right)\left(\sum_i \mathbf{v}[i] \exp(\mathbf{V}\mathbf{x})[i]\right) && (3.22)\\
&= \sum_i \sum_j \mathbf{u}[i]\mathbf{v}[j] \exp(\mathbf{U}\mathbf{x})[i] \exp(\mathbf{V}\mathbf{x})[j] && (3.23)\\
&= \sum_i \sum_j \mathbf{u}[i]\mathbf{v}[j] \exp((\mathbf{U}\mathbf{x})[i] + (\mathbf{V}\mathbf{x})[j]) && (3.24)\\
&= \sum_i \sum_j \mathbf{u}[i]\mathbf{v}[j] \exp(\mathbf{U}[i,:] + \mathbf{V}[j,:])\mathbf{x} && (3.25)\\
&= \mathbf{w} \cdot \exp \mathbf{W}\mathbf{x} && (3.26)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{W} &\in \mathbb{R}^{mn \times d} & \mathbf{w} &\in \mathbb{R}^{mn} && (3.27)\\
\mathbf{W}[in+j,:] &= \mathbf{U}[i,:] + \mathbf{V}[j,:] & \mathbf{w}[in+j] &= \mathbf{u}[i]\mathbf{v}[j]. && (3.28)
\end{aligned}$$

□

Equation (3.26) added 9/9

Exercise 3.15. Prove that FFNNs with sinh activations are universal.

Theorem 3.16 (Hornik et al., 1989)**.** *Let σ be any continuous, nondecreasing function such that*

$$\lim_{z \to -\infty} \sigma(z) = 0 \qquad (3.29)$$

$$\lim_{z \to +\infty} \sigma(z) = 1. \qquad (3.30)$$

Then FFNNs with two layers and σ activations are universal.

Cybenko (1989) proved a similar theorem, but it requires some more advanced math.

Proof. The proof uses three nested approximations. First, by Theorem 3.14, any continuous function can be approximated by a linear combination of exp units. Second, by Theorem 3.10, exp can be approximated by a linear combination of wedges (Eq. (3.6)). Third, a wedge can be approximated by an appropriately transformed σ.

Let $f \colon \mathbb{R}^d \to \mathbb{R}$ be the function we are trying to approximate. For the first approximation, by Theorem 3.14, there is a FFNN *ffn* with 2 layers and exp activations, such that, for all $\mathbf{x} \in [0,1]^d$,

$$|\mathit{ffn}(\mathbf{x}) - f(\mathbf{x})| \le \frac{\epsilon}{3}. \tag{3.31}$$

The next step will be to approximate the exps, but we need two pieces of information from the first approximation. First, what interval will we need to approximate exp on? Let

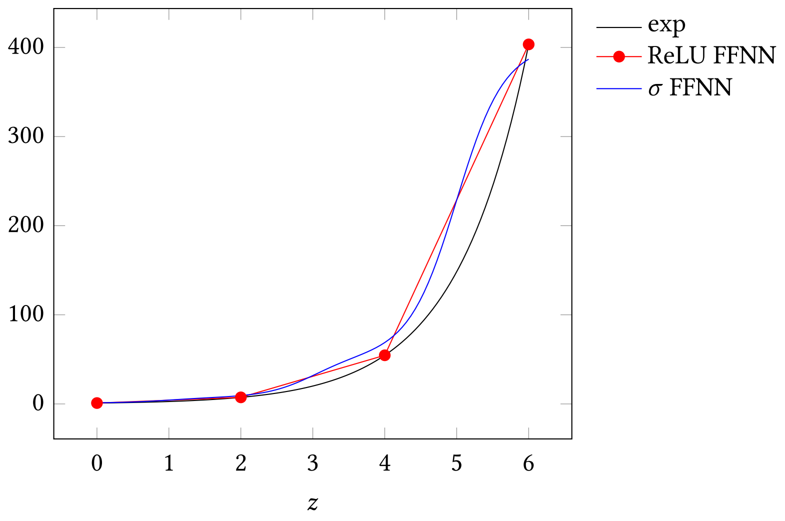
$$r = \max_i \left(\sum_j |\mathit{ffn}.\mathit{lin}_0.\mathbf{W}[i,j]| + |\mathit{ffn}.\mathit{lin}_0.\mathbf{b}[i]| \right) \tag{3.32}$$

so that the output of the first affine transformation (and therefore the input of the exp) is in $[-r, r]$. Second, how good does the approximation need to be? Let

$$m = \sum_i |\mathit{ffn}.\mathit{lin}_1.\mathbf{W}[1,i]| \tag{3.33}$$

which is the total of the scaling factors applied to the exps. We want the total error due to approximating the exps to be at most $\epsilon/3$, so we need exp to be approximated with error at most $\epsilon/(3m)$.

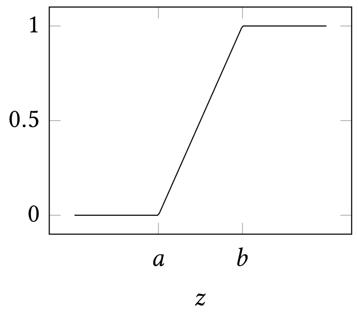
The remaining approximations are pictured below.



For the second approximation (red approximating black), by Theorem 3.10, there is a linear combination of k wedges that approximates $\exp(z)$ for $z \in [-r, r]$

with error $\epsilon/(3m)$. In preparation for the third approximation (blue approximating red), recall that in Theorem 3.10, we used wedges

$$s_{a,b}(z) = \begin{cases} 0 & \text{if } z \le a \\ \frac{z-a}{b-a} & \text{if } z \in (a,b) \\ 1 & \text{if } z \ge b \end{cases}$$



(sorry that this notation clashes with the notation we used in Theorem 3.11). But the only assumptions we made about the shape of a wedge $s_{a,b}$ were

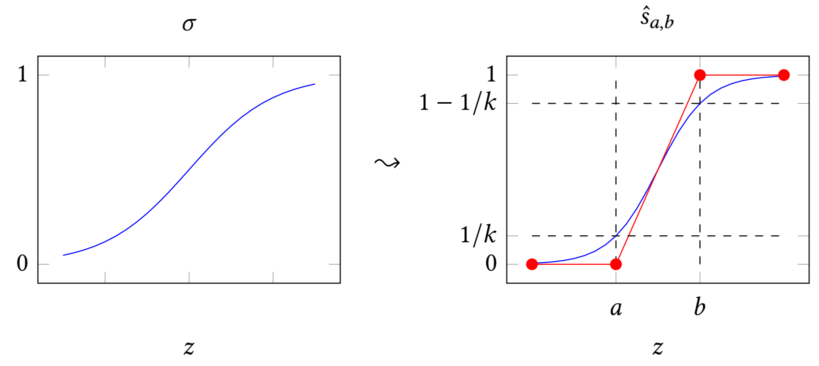
$$\begin{aligned} s_{a,b}(z) &= 0 && \text{if } z \le a \\ s_{a,b}(z) &\in [0,1] && \text{if } z \in (a,b) \\ s_{a,b}(z) &= 1 && \text{if } z \ge b. \end{aligned}$$

Since σ satisfies the middle condition, we will only have to worry about the error for $z \le a$ and $z \ge b$. Each wedge is scaled to have height at most $\epsilon/(3m)$, and we want to limit the remaining error to $\epsilon/(3m)$. Since there are k wedges, we want to approximate each unscaled wedge with error at most $1/k$.

For the third approximation (blue approximating red), by the definition of σ, there exists an A such that $\sigma(A) \le 1/k$ and B such that $\sigma(B) \ge 1 - 1/k$. Then for any a, b, let

$$\hat{s}_{a,b}(z) = \sigma\left(\frac{z-a}{b-a}(B-A) + A\right) \tag{3.34}$$

so that if $z \le a$ or $z \ge b$, then $|\hat{s}_{a,b}(z) - s_{a,b}(z)| \le 1/k$. In other words, σ can always be squashed (horizontally) so that it has small enough error outside the interval (a,b).



$\square$

3.2.5 Other topics

The universal approximation theorem says that an FFNN can approximate any function using only two layers, but says nothing about the width of the network. More recent work has studied the trade-off between depth and width.

Bibliography

Arora, Raman, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee (2018). Understanding deep neural networks with rectified linear units. In: *Proceedings of ICLR*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. In: *Math. Control Signal Systems* 2, pp. 303–314.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). Multilayer feedforward networks are universal approximators. In: *Neural Networks* 2.5, pp. 359–366.

Telgarsky, Matus (2021). Deep learning theory lecture notes. Unpublished lecture notes.

Wang, Shuning and Xusheng Sun (2005). Generalization of hinging hyperplanes. In: *IEEE Transactions on Information Theory* 51.12, pp. 4425–4431.