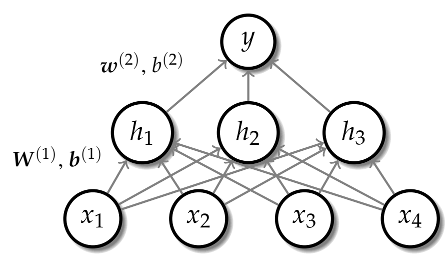
Instructor: Vishnu Boddeti
CSE 891-001: Deep Learning
Homework 1
September, 16

Instructions:

- **Filename:** Submit solutions in PDF format titled `written-assignment-1-{msunetid}.pdf`. Submissions in other formats or with other filenames will not be graded. You can produce your file however you like, LaTeX, Word or scan. Handwritten scans that are not legible will not be graded.
- **Submission:** Only homeworks uploaded to Google Classroom will be graded. Make sure to show all the steps of your derivations in order to receive full credit.
- **Integrity and Collaboration:** You are expected to work on the homeworks by yourself. You are not permitted to discuss them with anyone except the instructor. The homework that you hand in should be entirely your own work. You may be asked to demonstrate how you got any results that you report.
- **Clarifications:** If you have any question, please look at Google Classroom first. Other students may have encountered the same problem, and is solved already. If not, post your question there. We will respond as soon as possible.

1 Hard-Coding a Multilayer Perceptron (2pts):

In this problem you will find a set of weights and biases for a multilayer perceptron which determines if a list of four numbers are sorted in ascending order. More specifically, you receive 4 inputs x_1, x_2, x_3 and x_4 where $x_i \in \mathbb{R}$, and the network must output 1 if $x_1 < x_2 < x_3 < x_4$ and 0 otherwise. You will use the following Multilayer Perceptron (MLP) architecture consisting of one input layer with four nodes, one hidden layer with three nodes and one output layer with one node.



The activation function of the hidden units and the output unit can be assumed to be a hard threshold function.

$$\phi(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

Find a set of weights and biases for the network which correctly implements this function (including cases where some of the inputs are equal). Your answer should include:

1. A 3×4 weight matrix $\boldsymbol{W}^{(1)}$ for the hidden layer.
2. A 3-dimensional vector of biases $\boldsymbol{b}^{(1)}$ for the hidden layer.
3. A 3-dimensional vector of weights $\boldsymbol{w}^{(2)}$ for the output layer.
4. A scalar bias $b^{(2)}$ for the output layer.

1 Hand Coding a Multilayer Perceptron

Since the activation function is simply the hard threshold

$$\phi(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

the idea is to subtract the two adjacent inputs to check if the second is larger than the first. So h_1 will simply compare x_1 and x_2 and ignore x_3 and x_4 (zeros in $W^{(1)}$). h_2 compares x_2 and x_3 and h_3 compares x_3 and x_4. And then, y should only be 1 if h_1, h_2 and h_3 all output 1.

So, putting that idea into numbers, we get:

$$\boxed{W^{(1)} \overset{1.}{=} \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}, \quad \vec{b}^{(1)} \overset{2.}{=} \vec{0}}$$

This way we get $\vec{h} = \phi(W^{(1)}\vec{x} + \vec{b}^{(1)}) = \phi(W^{(1)}\vec{x})$

example: $\vec{x} = (0.1, 0.2, 0.3, 0.4)$ $\quad \vec{h} = \phi((-0.1+0.2, -0.2+0.3, -0.3+0.4)^T)$

$$= \phi((0.1, 0.1, 0.1)^T)$$

$$= (1, 1, 1)^T$$

This gives what we want: $h_i = \begin{cases} 1, & x_i < x_{i+1} \\ 0, & x_i \geq x_{i+1} \end{cases}, \quad i \in \{1, 2, 3\}$

Now to get from $\vec{h}$ to y we only need to make sure that all entries of $\vec{h}$ are 1 by summing them up and subtracting the scalar bias 2 from it (2, not 3 because of the sharp inequality in ϕ):

$$\boxed{W^{(2)} \overset{3.}{=} (1, 1, 1), \quad b^{(2)} \overset{4.}{=} -2}$$

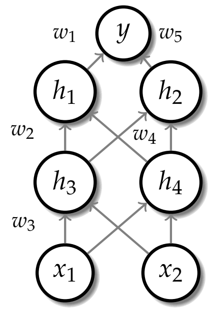
(Note that $W^{(2)}$ is a row vector!)

So we get $y = \phi(W^{(2)}\vec{h} + b^{(2)})$

$$y = \begin{cases} 1, & \vec{h} = (1,1,1)^T, \quad x_1 < x_2 < x_3 < x_4 \\ 0, & \text{otherwise} \end{cases}$$

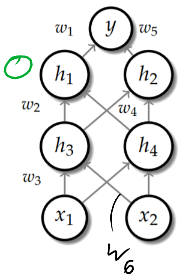
2 Sparsifying Activation Function (3pts):

An interesting property of the ReLU activation function is that it sparsifies the activations and the derivatives, i.e., sets a large fraction of the values to zeros for any given input vector. Consider the following network where the activation function of all the units is a ReLU function.



where each w_i refers to the weight of a single connection. Assume we are trying to minimize a loss function $\mathcal{L}$ which depends only on the activation of the output unit y. Suppose the unit h_1 receives an input -1 on a particular training case, so the ReLU evaluates to 0. Based only on this information, which of the weight derivatives $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, $\frac{\partial \mathcal{L}}{\partial w_3}$ are guaranteed to be 0 for this training case? Provide a YES or NO answer for each with your justification.

2 Sparsifying Activation Function



$$\mathcal{L}(y)$$

$$y = \text{Relu}(w_1 \overset{0}{h_1} + w_5 h_2)$$

$$\hookrightarrow h_1 = \text{Relu}(\overbrace{w_2 h_3 + w_4 h_4}^{-1})$$

$$\hookrightarrow h_3 = \text{Relu}(w_3 x_1 + w_6 x_2)$$

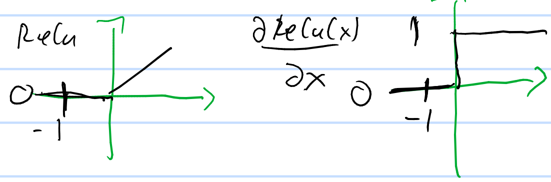
$$\Rightarrow \frac{\partial \mathcal{L}}{w_1} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y}\underset{}{\overset{=0}{h_1}} \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_1 h_1 + w_5 h_2}$$

$$\boxed{\text{so } \left.\frac{\partial \mathcal{L}}{w_1}\right|_{h_1 = 0} = 0 \quad \text{Yes! Garanteed to be 0 for } h_1 = 0}$$

$$\frac{\partial \mathcal{L}}{w_2} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial w_1 h_1}{\partial w_2} \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_1 h_1 + w_5 h_2}$$

$$\approx w_1 \frac{\partial h_1}{\partial w_2} = w_1 h_3 \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_2 h_3 + w_4 h_4}$$

$$= \frac{\partial \mathcal{L}}{\partial y} w_1 h_3 \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_2 h_3 + w_4 h_4 = -1} \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_1 h_1 + w_5 h_2}$$



So the derivative of the dead Relu is also 0, and therefore: $\boxed{\text{Yes! } \frac{\partial \mathcal{L}}{\partial w_2} \text{ is also guaranteed to vanish!}}$

$$\frac{\partial \mathcal{L}}{w_3} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial w_1 h_1}{\partial w_3} \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_1 h_1 + w_5 h_2}$$

$$= \frac{\partial \mathcal{L}}{\partial y} \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = w_1 h_1 + w_5 h_2} \cdot w_1 \frac{\partial h_1}{\partial w_3} \rightarrow w_2 \frac{\partial h_3}{\partial w_3} \cdot \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x = -1} = 0$$

$$\boxed{\text{Same for } \frac{\partial \mathcal{L}}{\partial w_3}\text{! Also garanteed to be 0 in this case, because } \left.\frac{\partial \text{Relu}(x)}{\partial x}\right|_{x=-1} = 0.}$$

And so would be any other earlier nodes.

3 Universal Approximation Theorem (5pts):

In this problem you will build the intuition behind how the neural network function class can approximate a particular class of functions arbitrarily well.

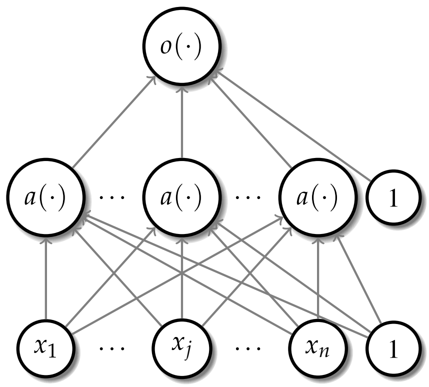
Suppose $f : I \to \mathbb{R}$, where $I = [a, b] \subset \mathbb{R}$ and $a \le b$ is a closed interval. Also, let $\hat{f}_\tau : I \to \mathbb{R}$ be some function approximator from our network where τ is a description of our network architecture and weights. Here, τ is a tuple of $(n, \boldsymbol{W}_0 \in \mathbb{R}^{n\times 1}, \boldsymbol{b}_0 \in \mathbb{R}^n, \boldsymbol{W}_1 \in \mathbb{R}^{\cancel{n\times 1}}_{1\times n}, \boldsymbol{b}_1 \in \mathbb{R}^{\cancel{n}})$, where n is the hidden layer size, $\boldsymbol{W}_0$ and $\boldsymbol{b}_0$ describe the input hidden parameters, and $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ describe the output hidden parameters.

The output is computed as $\hat{f}_\tau(\boldsymbol{x}) = \boldsymbol{W}_1 a(\boldsymbol{W}_0 \boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1$, where the activation $a(\cdot)$ is an indicator function, i.e., $a(y) = \mathbb{I}(y \ge 0)$, where $\mathbb{I}(s)$ is 1 when the boolean value s is true and 0 otherwise. For a vector, the activation function is applied to each element of the vector.

We want to show that there exist a series of neural networks $\{\tau_i\}_{i=1}^N$ such that:

$$\forall \epsilon > 0, \exists M : \forall m > M, \|f - \hat{f}_{\tau_m}\} < \epsilon$$

where $\|f - \hat{f}\| = \int_I |f(x) - \hat{f}(x)| dx$.

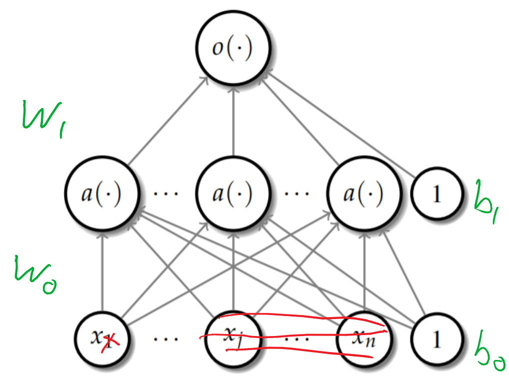


1. **(1.0 pt)** Consider a rectangular function $g(h, a, b, x) = h \cdot \mathbb{I}(a \le x \le b)$. Given some (h, a, b) show $\exists \tau : \hat{f}_\tau(x) = g(h, a, b, x)$. You answer should be a specific choice of n, $\boldsymbol{W}_0$, $\boldsymbol{b}_0$, $\boldsymbol{W}_1$, and $\boldsymbol{b}_1$, which will be functions of the selected (h, a, b), where $h \in \mathbb{R}$, $a \in \mathbb{R}$, and $b \in \mathbb{R}$.

2. **(1.5 pt)** Given $f(x) = -x^2 + 1$ where $I = [-1, 1]$ and some initial function $\hat{f}_0(x) = 0$ which is identically 0, construct a new function $\hat{f}_1(x) = \hat{f}_0(x) + g(h_1, a_1, b_1, x)$ such that $\|f - \hat{f}_1\| \le \|f - \hat{f}_0\|$, with the rectangle function in the previous question. Note that h_1, a_1, and b_1 are going to depend on your choice of f, $\hat{f}$ and I. Plot f and $\hat{f}_1$, write down h_1, a_1, and b_1, and justify why $\|f - \hat{f}_1\| < \|f - \hat{f}_0\|$.

3. **(2.5 pt)** Describe a procedure which starts with $\hat{f}_0(x) = 0$ and a fixed N, then construct a series $\{\hat{f}_i\}_{i=0}^N$ where $\hat{f}_{i+1}(x) = \hat{f}_i(x) + g(h_{i+1}, a_{i+1}, b_{i+1}, x)$, which satisfies $\|f - \hat{f}_{i+1}\| < \|f - \hat{f}_i\|$. Use the definition of g from above and the choice of f from the previous question. Plot f, $\hat{f}_1$,$\hat{f}_2$ and $\hat{f}_3$, write down how to generate h_{i+1},a_{i+1},b_{i+1}, and justify why $\|f - \hat{f}_{i+1}\| < \|f - \hat{f}_i\|$.

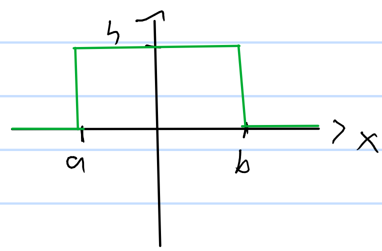
3 Universal Approximation Theorem



$$\hat{f}_\tau(\vec{x}) = W_1 a(W_0 \vec{x} + \vec{b}_0) + \vec{b}_1$$

$$a(y) = \mathbb{1}(y \geq 0)$$

1. $g(h, a, b, x) = h \, \mathbb{1}(a \leq x \leq b)$



I don't want to be too mathematically strict, but f and $\hat{f}_\tau$ are only defined on the interval $I = [a, b]$, so technically a correct answer would be:

$$\boxed{n = 1, \quad W_0 = 0, \quad b_0 = 0, \quad W_1 = 0, \quad b_1 = h}$$

which would always output h on the given interval on which f nd $\hat{f}_\tau$ are defined and on that interval we would have $g = f = \hat{f}_\tau = h$.

I assume though for this part you mean that $x \in \mathbb{R}$ not just $x \in I$. Or at least: $I = [\tilde{a}, \tilde{b}]$ and $g(h, a, b, x)$ with $\tilde{a} < a$ and $\tilde{b} > b$.
Then the idea is the same as in Problem 1: simply check if $a < x$ and if $x < b$:

$$\boxed{n = 2, \quad W_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \vec{b}_0 = \begin{pmatrix} -a \\ b \end{pmatrix}, \quad W_1 = (h, h), \quad b_1 = -h}$$

So this way we get $y = (h, h) \cdot \begin{pmatrix} a(x-a) \\ a(-x+b) \end{pmatrix} - h$

<u>Cases:</u>

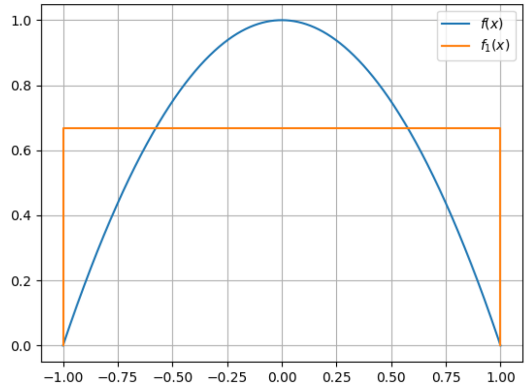
$a \leq x \leq b$: $y = (h,h) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} - h = 2h - h = h$ ✓

$x \leq a \leq b$: $y = (h,h) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} - h = h - h = 0$ ✓

$a \leq b \leq x$: $y = (h,h) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} - h = h - h = 0$ ✓

Note that we have to assume $a \leq b$, otherwise the output $-h$ is also possible for the case $b \leq x \leq a$.

2.



There is an infinite amount of solutions for 2., but looking forward to 3., I choose:

$$\vec{f}_1(x) = \overset{=0}{\vec{f}_0} + g(h_1, a_1, b_1, x) = \boxed{g(h_1, -1, 1, x)}$$

and choose h_1 as the mean of f on the given interval:

$$h_1 = \frac{1}{2}\int_{-1}^{1} dx\, f(x) = \int_0^1 -x^2 + 1\, dx = 1 - 1/3 = \boxed{2/3 = h_1}$$

This is the best possible approximation using a single rectangle function.

Since $\vec{f}_0 = 0$ we need to show that $\|f - \vec{f}_1\| < \|f\|$:

$$\int_{-1}^{1} |f(x)|\, dx = 2\int_0^1 -x^2 + 1\, dx = \frac{4}{3}$$

$$\int_{-1}^{1} |f(x) - \vec{f}_1(x)|\, dx = 2\int_0^1 |-x^2 + 1/3|\, dx = \frac{4}{9\sqrt{3}} \approx 0.25660$$

And therefore $\frac{4}{9\sqrt{3}} = \|f - \vec{f}_1\| < \|f\| = \frac{4}{3}$ ✓

This is the best case for $N = 1$.

3. Generalizing this idea for any N, we simply devide I into N subintervals of equal size and iteratively add the mean of f in each subinterval as a rectangle function in our series:

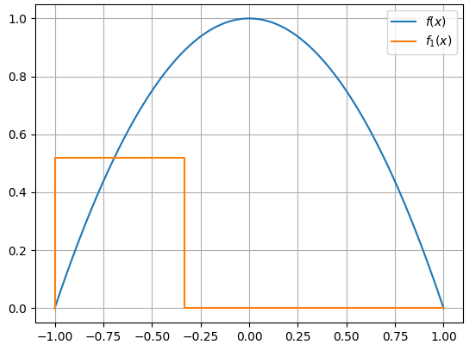
$$\vec{f}_0(x) = 0, \quad \vec{f}_{i+1}(x) = \vec{f}_i(x) + g(h_{i+1}, a_{i+1}, b_{i+1}, x)$$

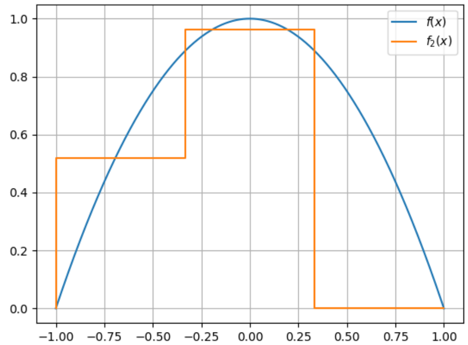
with $a_i = a + \frac{b-a}{N}(i-1)$, $b_i = a_i + \frac{b-a}{N} = a + \frac{b-a}{N} i$, here: $a = -1$, $b = 1$

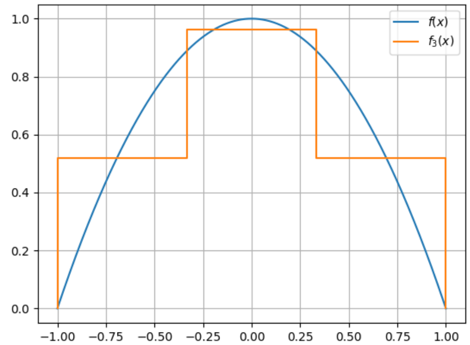
and $h_i = \frac{N}{b-a}\int_{a_i}^{b_i} f(x)\, dx$ The integral can be approximated by some numerical integration.

This fullfills the condition $\|f - \vec{f}_{i+1}\| < \|f - \vec{f}_i\|$, because $\vec{f}_{i+1}$ always adds the best possible approximation for the next subinterval, which is a constant 0 in $\vec{f}_i$ (see next page for visuallization).

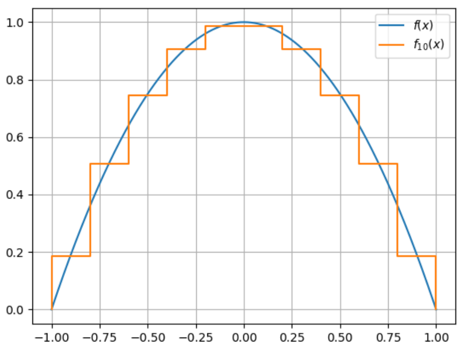
So for $N=3$, we get the series:

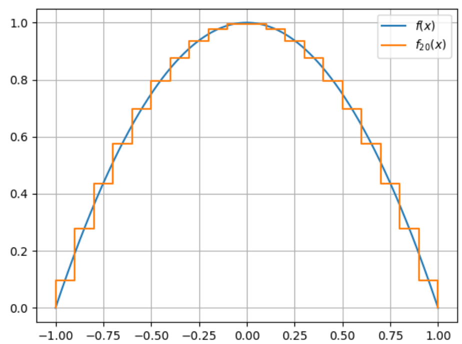


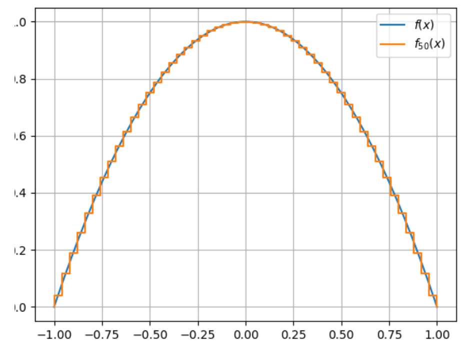


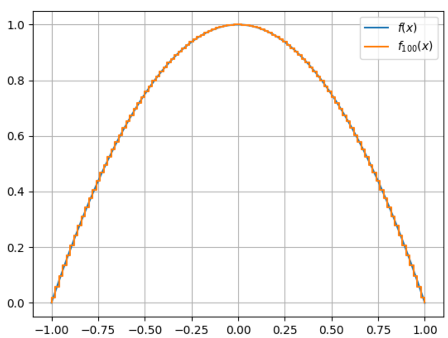


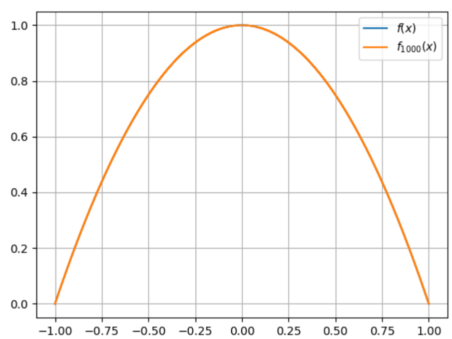
For $N \to \infty$: $\hat{f}_N \to f$ (without proof). E.g. let's look at $\hat{f}_N$ for $N \in \{10, 20, 50, 100, 1000\}$:











Alternatively to choosing the mean for h, one can also simply choose the function value at one of the boundaries or at the center of the subinterval, or $h_i = \frac{f(b_i) - f(a_i)}{2}$.

All of these alternative options are numerical approximations of the integral for the mean.

Choosing the mean is the best possible approximation of f!