**Karan Vora (kv2154)**
**ECE-GY 7143 Introduction to Deep Learning, Assignment 1**

**Problem 1):**

**Solution a):**
The vector expression for $l_1$ – norm is

$$L(w)=\|Xw-Y\|_1$$

where, X is a $n\times d$ matrix, w is $d\times 1$ vector and Y is a $n\times 1$ vector.

**Solution b):**
Unlike, $l_2$ – norm, There's no closed form expression for $l_1$ – norm as it is non differentiable. We cannot solve for the gradient being zero.

**Solution c):**
Referred the provided notes for lecture 1, topic Warmup: Linear models. In three step recipe, in the 2nd step where we measure the quality of model, losses are an important metric. The lower the loss, the lower the overall penalty for an incorrect prediction.

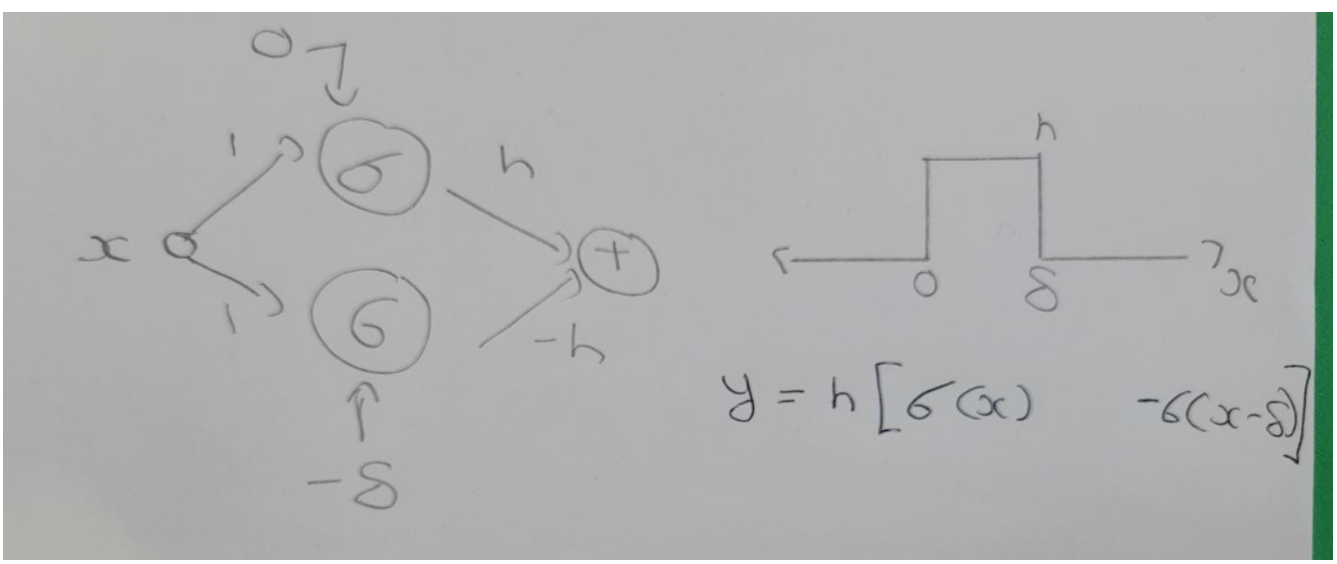
==========================================================================

**Problem 2):**

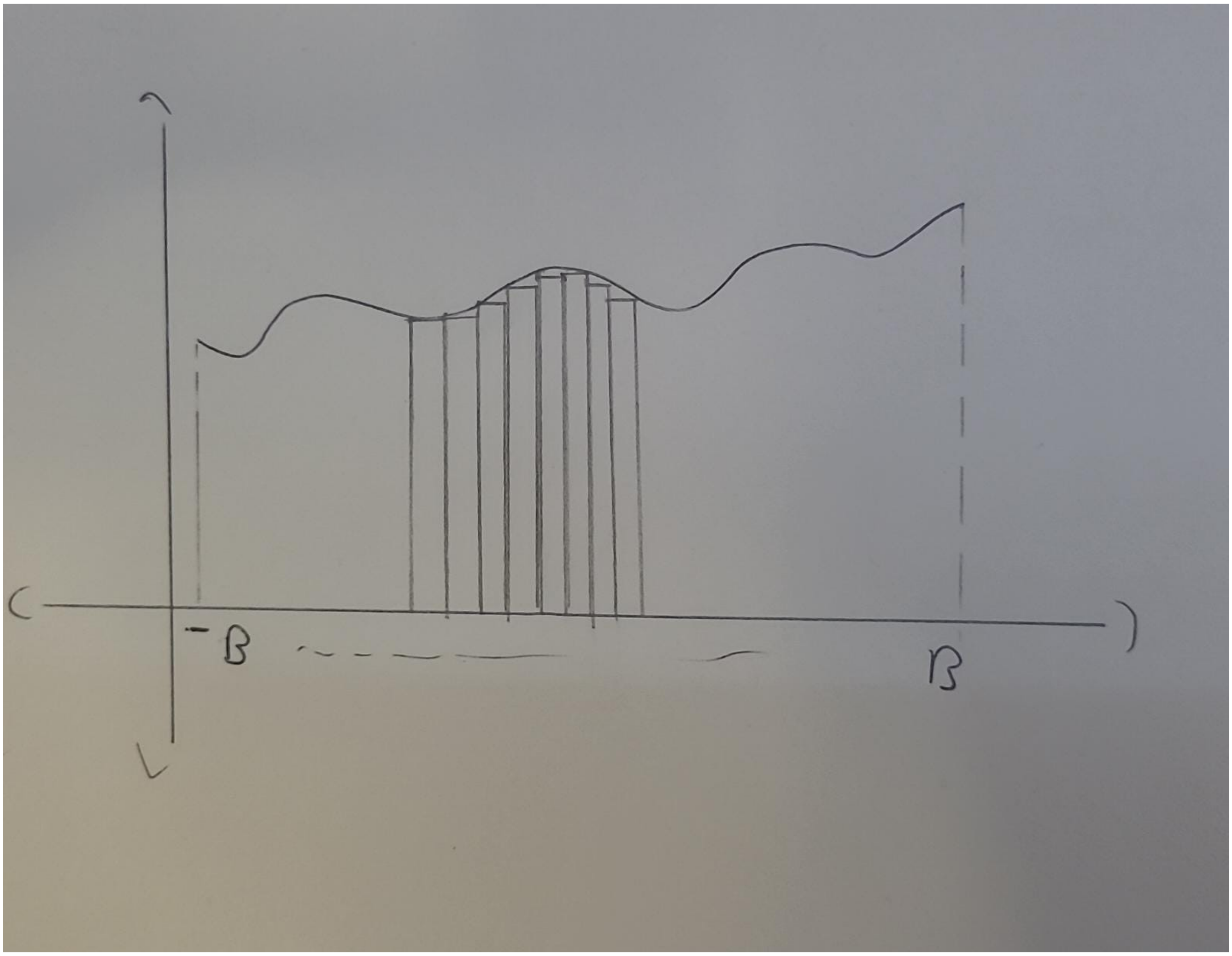
**Solution a):**
A box function can be realized by using the difference between two step function:

$$f(x)=h\,\sigma(x)-h\,\sigma(x-\delta)$$

This produces the following graph.



**Solution b):**
The approximate plot of smoothening a curve is given below. We can separate any arbitrary function distribution into an approximate superposition of box function. Each chunk can be plotted using 2 hidden neurons. To get the plot, we change the bias of first layer and value of second layer.




**Solution c):**
We can apply the same principle to a different dataset types aswell. As the dimensionality of the data increases so is the dimensions of the plotted boxes, thus we require more boxes to plot. Consider a domain,

$$D=[-B,B]$$

the approximate relation between dimensionality $d$ and number of boxes required is given by

$$N=\left(\frac{B}{\delta}\right)^d$$

With increase in $d$ the number of trainable parameters also explodes requiring a more complex network. The network architecture will always be a combination of depth and width to sufficiently approximate the said function.

**Solution d):**
From the above mentioned discussion, we can conclude that any sufficiently wide and deep neural network can approximate any arbitrary function with a great degree of accuracy considering the data provided as input is sufficiently good enough in terms of quality and the other parameters or an NN architecture are also optimal.

==========================================================================

**Problem 3):**

The softmax function is given by,

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

Taking log,

$$\log(y_i) = z_i - \log\left(\sum_{k=1}^{n} e^{z_k}\right)$$

Now taking partial derivative,

$$\frac{\partial \log(y_i)}{\partial z_j} = \frac{1}{y_i} \frac{\partial y_i}{\partial z_j}$$

When j = i,

$$\frac{\partial y_i}{\partial z_i} = 1 - \frac{\partial}{\partial z_i} \log\left(\sum_{k=1}^{n} e^{z_k}\right)$$

$$= 1 - \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

$$= 1 - y_i$$

and when j ≠ i,

$$\frac{\partial y_i}{\partial z_j} = -\frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}} = -y_j$$

Therefore, putting it together using Dirac Delta as indicator

$$J_{ij} = y_i(\delta_{ij} - y_j)$$

========================================================================================