New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Written Homework 3.
Due Tuesday, April 12th, 2022, 11:59pm.

Collaboration is allowed on this problem set, but solutions must be written-up individually.

Problem 1: Complexity of Hypothesis Classes (20pts)

(a) In the lecture on learning theory we saw how to bound the number of training examples required to PAC-learn certain functions (aka models, aka hypothesis classes) in the realizable setting. For the following functions, give as tight an upper bound as you can on how many samples are required for PAC-learning with accuracy ϵ and success probability $(1-\delta)$.

(i) A *decision list* is a boolean function mapping $\{x_1, \ldots, x_d\} \in \{0,1\}^d \to \{0,1\}$ of the following form:

If(y_1) return z_1
Else if(y_2) return z_2
$\vdots$
Else if(y_k) return z_k,
Else return z_{k+1}.

Above $y_i \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_d, \bar{x}_d\}$ for each i and $z_i \in \{0,1\}$ for each i. Here $\bar{x}_i$ denotes the logical "not" of variable x_i. k is the number of terms in the decision list and is not bounded apriori. Try to give a bound that does not scale with k.

(ii) A linear *threshold function* is a function mapping $\{x_1, \ldots, x_d\} \in \{-1,1\}^d \to \{0,1\}$ of the following form:

$$\mathbb{1}[w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \geq \lambda]$$

where $w_i \in \{-d, \ldots, -1, 0, 1, \ldots, d\}$ is an integer weight for variable x_i and $\lambda \in \mathbb{R}$ is an arbitrary threshold. **Hint:** Show that we only need to consider a finite number of choices for λ.

(b) Suppose I want to perform model selection. I have q different hypothesis classes $\mathcal{H}_1, \ldots, \mathcal{H}_q$ and I know that for some i, there is a function $h^* \in \mathcal{H}_i$ that perfectly fits my data. I.e. with $R_{pop}(h^*) = 0$. My goal is to find some h such that $R_{pop}(h) \leq \epsilon$. Give as tight an upper bound as you can on how many samples are required to solve this problem with success probability $(1-\delta)$.

Problem 2: Kernels for Shifted Images (20pts)

In class we discussed why the Gaussian kernel is a better similarity metric for MNIST digits than the inner product. Here we consider an additional modification to the Gaussian kernel.

For illustration purposes we consider 5x5 black and white images: a pixel has value 1 if it is white and value 0 if it is black. For example, consider the following images of two 0s and two 1s:

$$I_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad I_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad I_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad I_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) Let $\mathbf{x}_i \in \{0,1\}^{25}$ denote the vectorized version of image I_i, obtained by concatenating the rows of the matrix representation of the image into a vector. Compute the 4×4 kernel matrix $\mathbf{K}$ for images $I_1, \ldots, I_4$ using the standard Gaussian kernel $k_G(I_i, I_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}$ (this is just a simple calculation).

(b) Suppose I_1 and I_2 are in our training data and I_3 and I_4 are in our test data. Which training image is most similar to each of our test images according to Gaussian kernel similarity? Do you expect a kernel classifier (k-NN, kernel logistic regression, etc.) to correctly or incorrectly classify I_3 and I_4?

(c) Consider a "left-right shift" kernel, which is a similarity measure defined as follows:

For an image I_i, let I_i^{right} be the image with its far right column removed and let I_i^{left} be the image with its far left column removed. Intuitively, I_i^{right} corresponds to the image shifted one pixel to the right and I_i^{left} corresponds to the image shifted one pixel left. Define a new similarity metric k_{shift} as follows:

$$k_{\text{shift}}(I_i, I_j) = k_G(I_i^{\text{right}}, I_j^{\text{right}}) + k_G(I_i^{\text{left}}, I_j^{\text{left}}) + k_G(I_i^{\text{right}}, I_j^{\text{left}}) + k_G(I_i^{\text{left}}, I_j^{\text{right}})$$

Intuitively this kernel captures similarity between images which are similar *after a shift*, something the standard Gaussian kernel does not account for.

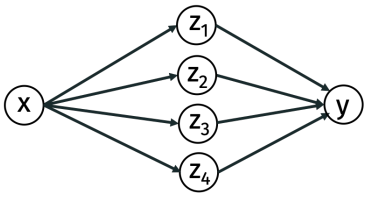
Recompute the a 4×4 kernel matrix $\mathbf{K}$ for images $I_1, \ldots, I_4$ using k_{shift}.

(d) Again I_1 and I_2 were in our training data and I_3 and I_4 were in our test data. Now which training image is most similar to each of our test images according to the "left-right shift" kernel? Do you expect a typically kernel classifier to correctly or incorrectly classify I_3 and I_4?

(e) Prove that k_{shift} is a positive semi-definite kernel function. **Hint:** Use the fact that k_G is positive semi-definite.

Problem 3: Neural Networks for Curve Fitting (15pts)

Consider the following 2-layer, feed forward neural network for single variate regression:



Let $W_{H,1}, W_{H,2}, W_{H,3}, W_{H,4}$ and $b_{H,1}, b_{H,2}, b_{H,3}, b_{H,4}$ be weights and biases for the hidden layer. Let $W_{O,1}, W_{O,2}$, $W_{O,3}, W_{O,4}$ and b_O be weights and bias for the output layer. The hidden layer uses rectified linear unit (ReLU) non-linearities and the output layer uses no non-linearity.

Specifically, for $i = 1, \ldots, 4$, $z_i = \max(0, \bar{z}_i)$ where $\bar{z}_i = W_{H,i}x + b_{H,i}$. And

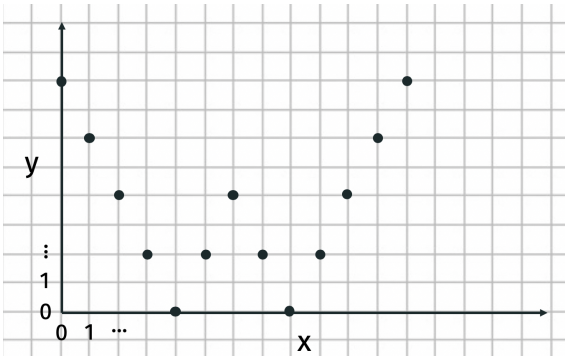
$$y = b_O + \sum_{i=1}^{4} W_{O,i} z_i.$$

(a) For each of the two datasets below, determine values for weights and biases which would allow this network to perfectly fit the data.

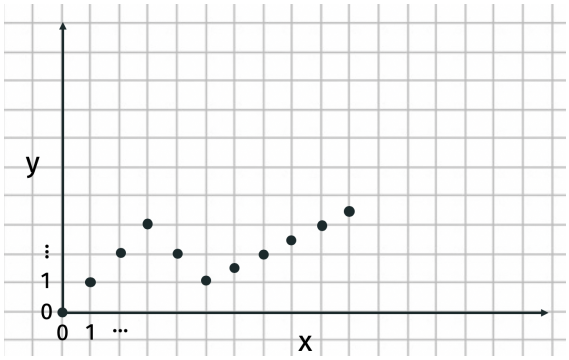
(b) For input parameters $\boldsymbol{\theta}$ let $f(x, \boldsymbol{\theta})$ denote the output of the neural network for a given input x. We want to train the network under the squared loss. Specifically, given a training dataset $(x_1, y_1), \ldots, (x_n, y_n)$, we want to choose $\boldsymbol{\theta}$ to minimize the loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - f(x_i, \boldsymbol{\theta}))^2.$$

Write down an expression for the gradient $\nabla \mathcal{L}(\boldsymbol{\theta})$ in terms of $\nabla f(x_1, \boldsymbol{\theta}), \ldots, \nabla f(x, \boldsymbol{\theta})$. **Hint:** Use chain rule.



Dataset 1



Dataset 2

(c) Suppose we randomly initialize the network with ± 1 random numbers:

$$W_{H,1} = -1, W_{H,2} = 1, W_{H,3} = 1, W_{H,4} = -1$$
$$b_{H,1} = 1, b_{H,2} = 1, b_{H,3} = -1, b_{H,4} = 1$$
$$W_{O,1} = -1, W_{O,2} = -1, W_{O,3} = -1, W_{O,4} = 1$$
$$b_O = 1$$

Call this initial set of parameter $\boldsymbol{\theta}_0$. Use forward-propagation to compute $f(x, \boldsymbol{\theta}_0)$ for $x = 2$.

(d) Use back-propagation to compute $\nabla f(x, \boldsymbol{\theta}_0)$ for $x = 2$. To do the computation you will need to use the derivative of the ReLU function, $\max(0, z)$. You can simply use:

$$\frac{\partial}{\partial z} \max(0, z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

This derivative is discontinuous, but it turns out that is fine for use in gradient descent.