

New York University Tandon School of Engineering  
Computer Science and Engineering

CS-UY 4563: Written Homework 4.

Due Monday, April 20th, 2020, 11:59pm.

*Collaboration is allowed on this problem set, but solutions must be written-up individually. Please list the names of any collaborators at the top of your solution set, or write “No Collaborators” if you worked alone.*

### Problem 1: 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:

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

- Let  $\vec{x}_i \in \{0,1\}^{25}$  denote the vectorized version of image  $I_i$ , obtained by concatenating the rows of the matrix representation of the image into a vector. Compute a  $4 \times 4$  kernel matrix  $\mathbf{K}$  for images  $I_1, \dots, I_4$  using the standard Gaussian kernel  $k_G(I_i, I_j) = e^{-\|\vec{x}_i - \vec{x}_j\|_2^2}$ .
- 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  $I_4$ ?
- Consider a “left-right shift” kernel, which is a similarity measure defined as follows:

For an image  $I_i$ , let  $I_i^{\text{right}}$  be the image with its far right column removed and let  $I_i^{\text{left}}$  be the image with its far left column removed. Intuitively,  $I_i^{\text{right}}$  corresponds to the image shifted one pixel to the right and  $I_i^{\text{left}}$  corresponds to the image shifted one pixel left. Define a new similarity metric  $k_{\text{shift}}$  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  $4 \times 4$  kernel matrix  $\mathbf{K}$  for images  $I_1, \dots, I_4$  using  $k_{\text{shift}}$ .

- 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  $I_4$ ?
- Prove that  $k_{\text{shift}}$  is a positive semi-definite kernel function. **Hint:** Use the fact that  $k_G$  is positive semi-definite.

### Problem 2: Thinking About Margin (10pts)

Consider the data set of four examples  $\vec{x}_1, \dots, \vec{x}_4$  with two features each:  $\vec{x}_i = (\vec{x}_i[1], \vec{x}_i[2])$ . Each data example has a binary class label  $y_i = \pm 1$ .

$x_i[1]$	0	1	1	2
$x_i[2]$	0	0.3	0.7	1
$y_i$	-1	-1	1	1

- (a) Find a linear classifier that separates the two classes. Your classifier  $f$  should be of the form:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } b + w_1\vec{x}[1] + w_2\vec{x}[2] > 0 \\ -1 & \text{if } b + w_1\vec{x}[1] + w_2\vec{x}[2] \leq 0 \end{cases}$$

State the intercept  $b$  and weights  $w_1$  and  $w_2$  for your classifier. Note there is no unique correct answer, as there are multiple linear classifiers that could separate the classes.

- (b) For the classifier you found in part (a), what is the maximum  $\gamma$  such that

$$y_i(b + w_1x_{i1} + w_2x_{i2}) \geq \gamma, \text{ for all } i?$$

- (c) Compute the margin of your classifier from part (a). The margin  $m$  is the minimum perpendicular distance from any point to the separating hyperplane defined by a linear classifier. Mathematically it can be computed as:

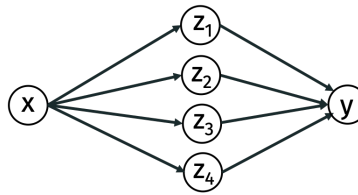
$$m = \frac{\gamma}{\|\vec{w}\|_2},$$

where  $\|\vec{w}\|_2 = \sqrt{w_1^2 + w_2^2}$ .

- (d) Which data examples are on the margin for your classifier? I.e. for which  $i$  does  $\vec{x}_i$  lie exactly distant  $m$  from your chosen separating hyperplane?
- (e) Is your classifier a maximum margin classifier, or does there exist a separating hyperplane with larger margin? Justify your answer briefly (in words or math). To answer this question, it might be helpful to plot the dataset and your classification rule.

### 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.

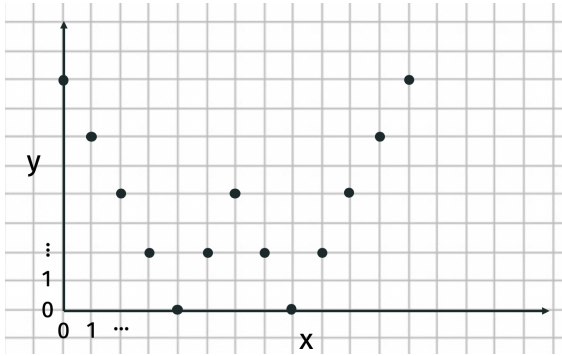
Specifically, for  $i = 1, \dots, 4$ ,  $z_i = \max(0, \bar{z}_i)$  where  $\bar{z}_i = W_{H,i}x + b_{H,i}$ . And

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

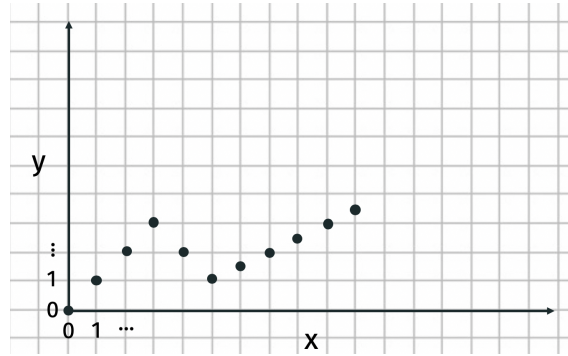
- (a) For each of the two datasets below, determine values for weights and biases which would allow this network to perfectly fit the data.
- (b) For input parameters  $\vec{\theta}$  let  $f(x, \vec{\theta})$  denote the output of the neural network for a given input  $x$ . We want to train the network under the squared loss. Specifically, given a training dataset  $(x_1, y_1), \dots, (x_n, y_n)$ , we want to choose  $\vec{\theta}$  to minimize the loss:

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

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



Dataset 1



Dataset 2

- (c) Suppose we randomly initialize the network with  $\pm 1$  random numbers:

$$\begin{aligned}
 W_{H,1} &= -1, W_{H,2} = 1, W_{H,3} = 1, W_{H,4} = -1 \\
 b_{H,1} &= 1, b_{H,2} = 1, b_{H,3} = -1, b_{H,4} = 1 \\
 W_{O,1} &= -1, W_{O,2} = -1, W_{O,3} = -1, W_{O,4} = 1 \\
 b_O &= 1
 \end{aligned}$$

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

- (d) Use back-propagation to compute  $\nabla f(x, \vec{\theta}_0)$  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.