Name:

CMOR 518 · APPLICATIONS IN COMPUTATIONAL MATHEMATICS · FALL 2024

Homework 4

Posted Tuesday, October 8th 2024. Due Friday, October 18nd before 11:59PM (CST). Should be submitted via CANVAS

Please submit both .ipynb and published .pdf files. Your .iynb file should include the header stating your name, CMOR518, Homework number and the last date it was modified. If some manual computation needs to be done you can either type it to your .ipynb file or include the screenshots using:

A=plt.imread('name.png')

plt.figure

plt.imshow(A)

Reminder: Mathematically rigorous solutions are expected; strive for clarity and elegance. You may collaborate on the problems, but your write-up must be your own independent work unless stated otherwise. Transcribed solutions and copied Python code are both unacceptable, though, you may modify code provided in the course. *You may not consult solutions from previous sections of this class.*

This assignment is worth a total of 100 points.

1. [100 points] The training data is the set of 4 letters (r,i,c,e) defined on the plot below. For example, the letter r can be defineed as $r = [1,1,1,1,1,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0]$.

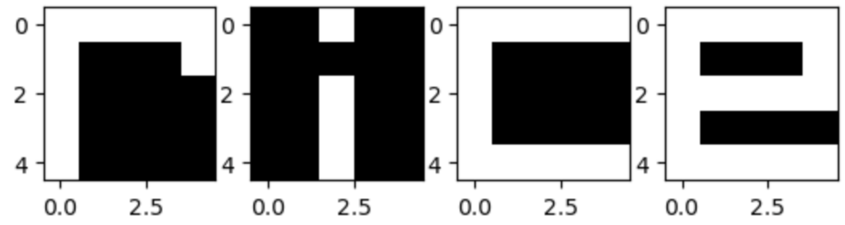

Figure 0.1: Train inputs

The truth (target) data is given as $[0,0],[0,1],[1,0],[1,1]$.

Write a function $neural(train, target, gamma, maxiter)$ that performs $maxiter$ iterations of the gradient descent. Let the user set the learning rate parameter (γ). Your code will compute the loss function, the partial derivatives of the loss function with respect to V and W. The matrices V, W should be initialized with the same random seed.

Test the code with $maxiter = 2000$ and $\gamma = 0.1$. Print the loss function at the end of all the gradient descent iterations.

Check the code is correct by writing a small code that produces a prediction from an input that belongs to the training dataset (this can be defined as a function so that we can reuse it). We should recover exact prediction if the training has been done correctly.

Finally, choose the letter r and perform its perturbations, using the function *change* which will flip one digit in the data. For a given perturbation, apply the neural network to obtain a prediction. Repeat the process 100 iterations. Count each time the prediction is good. Display the final number of good predictions (should be equal to 100: perfect score).