Homework 2
CSCE 790 : Neural Networks and Their Applications
Fall 2024
Due Date - October 27

Instructions:

- Submit a single PDF with solutions. Use the following format to name the PDF file - "CSCE790-HW-2-Lastname."
- The solutions should be very clear and should follow all the instructions below.
- If the solutions are not readable, they will not be graded
- If you refer to any resource to get your solutions, add an acknowledgement and list the references (details of the source, e.g., book, website, etc.).
- Include codes to the problems (or link to GitHub or Notebook) and codes should be clearly commented
- Do not include colab printed page or .ipynb file in your solution as an additional document. You may add their links in the PDF and the codes should be accessible with the link without needing additional permissions.
- Add good figures; the figures should be generated as pdf, eps, or svg file and added to the solution.
- Figures should have detailed captions.

Part A: (Read and Summarize) (20 points)

- Read the following and submit a brief summary of each research article (no longer than a page, preferably as itemized points (your takeaways)).
  1. Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. Nature, 323(6088), pp.533-536.
     Link
  2. LeCun, Y.A., Bottou, L., Orr, G.B. and Müller, K.R., 2012. Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-48). Springer, Berlin, Heidelberg.
     Link

Part B: (Coding)

1. (15 points) It is desired to design a one-layer NN with two inputs and two outputs that classifies the following 10 points into 4 groups as shown:
   - Group 1: $(0.1, 1.2)$, $(0.7, 1.8)$, $(0.8, 1.6)$
   - Group 2: $(0.8, 0.6)$, $(1.0, 0.8)$
   - Group 3: $(0.3, 0.5)$, $(0.0, 0.2)$, $(-0.3, 0.8)$
   - Group 4: $(-0.5, -1.5)$, $(-1.5, -1.3)$

   To cast this problem in terms of a tractable NN design task, represent the four groups respectively using the binary codes $(1, 0)$, $(0, 0)$, $(1, 1)$, and $(0, 1)$. Define the input matrix as

$$X = \begin{bmatrix} 0.1 & 0.7 & 0.8 & 0.8 & 1.0 & 0.3 & 0.0 & -0.3 & -0.5 & -1.5 \\ 1.2 & 1.8 & 1.6 & 0.6 & 0.8 & 0.5 & 0.2 & 0.8 & -1.5 & -1.3 \end{bmatrix}$$

and the associated desired target as

$$Y = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Use sigmoid activation function, design an NN and train the network using backpropagation algorithm (do not use any pre-existing package - write your own backpropagation algorithm). Plot the following figures: (1) training error vs epoch number, (2) decision boundary and data points after 3 epochs, 10 epochs, and 100 epochs.

2. (15 points) Design and train a two layer neural network (again no packages for backpropagation - write your own backpropagation algorithm) to approximate a function $f : X \to Y$ within the domain $[-1, 1]$ whose sample data points are given by

$$X = \begin{bmatrix} -1 & -0.9 & -0.8 & -0.7 & -0.6 & -0.5 & -0.4 & -0.3 & -0.2 & -0.1 & 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \end{bmatrix}$$

and

$$Y = \begin{bmatrix} -0.96 & -0.577 & -0.073 & 0.377 & 0.641 & 0.66 \\ 0.461 & 0.134 & -0.201 & -0.434 & -0.5 & -0.393 & -0.165 & 0.099 & 0.307 & 0.396 & 0.345 & 0.182 & -0.031 & -0.219 & -0.321 \end{bmatrix}.$$

Plot the following figures: (1) training error vs epoch number, (2) $f(x)$ v.s. x and NN output v.s. x (actual and approximate functions) after 10, 100, 200, 400, 1000 epochs.

3. (15 points) Read the article in the link below. Use the codes in the article to reproduce the results given in it. Submit the results, your code, and a detailed step-by-step description of the algorithm. Link

4. (15 points) Read the article in the link below. Use the codes in the article to reproduce the results given in it for vanilla GAN and DCGAN. Submit the results, your code, and a detailed step-by-step description of the algorithm.Link

Part C: Derivation/Theory (20 points)

1. Derive the weight update rules of the backpropagation algorithm for a 2-layer feedforward NN with $\tanh$ activation function. You should derive all the gradients used in the update rules and give the explicit formula in matrix notation. Define the symbols along with the dimensions before doing the derivation.