CS/SE 4X03 — Assignment 4

20 March 2020

Due date: 31 March 1:20pm

- **If you need an extension, you can submit by 1:20pm on April 3rd.**
- **If you write your solutions by hand, please ensure your handwriting is legible and easy to read. Otherwise, please type your solutions. We will deduct marks for messy handwritten solutions.**
- **Submit all your Matlab code for problems 1 and 2 to Avenue.**
- **Submit your hardcopy as a PDF file to Avenue. Do not include code in the PDF.**

Problem 1 [14 points] Study sections 1 to 4 and 6 of https://arxiv.org/abs/1801.05894. I have covered 1–4 in class; see also http://www.cas.mcmaster.ca/~nedialk/COURSES/4X03/nn.pdf

Unzip the file NN.zip

[**4 points**] Modify the netbp.m code so you can pass parameters as follows

```
function  cost = netbp2(neurons, data, labels, niter, lr, file)
%Trains a neural network with 4 layers.
%Layer 1 has 2 neurons, layer 2 has neurons(1),
%layer 3 has neurons(2), and layer 4 has 2 neurons
%data is a matrix with two rows.  data(:,i) contains
%the (x,y) coordinates of point i.
%labels is a matrix with two rows. labels(:,i) is [1;0]
%if data(:,i) is in category A and [0;1] if it is in category B.
%lr is learning rate
%niter is maximum number of iterations
%file is a filename where the file is created by
%save(file, 'W2','W3','W4','b2','b3','b4');
%cost is a vector storing the value of the cost function
%after each iteration; i.e. cost(j) is the cost at iteration j
```

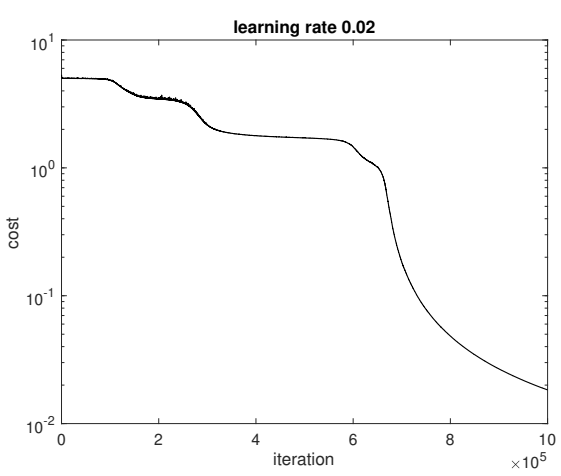
[**2 points**] Implement the Matlab function

```
function category = classifypoints(file, points)
%Reads parameters from file and classifies points into two
%categories, A or B. This file is created by netbp2.m.
%points is a matrix with two rows, where points(:,i) contains the
%(x,y) coordinates of point i.
%Returns vector category, where category(i) is 1 if
%points(1,i) >= points(2,i) and 0 otherwise.
```

[**8 points**] Chose values for X, Y and Z in the script main_nn.m such that in the plot classified.eps the grey region contains only A training points, and the white region contains only B training points. For example, this script should produce plots like in Figure 1

Submit

- hard copy: a figure like Figure 1
- Avenue: all Matlab code



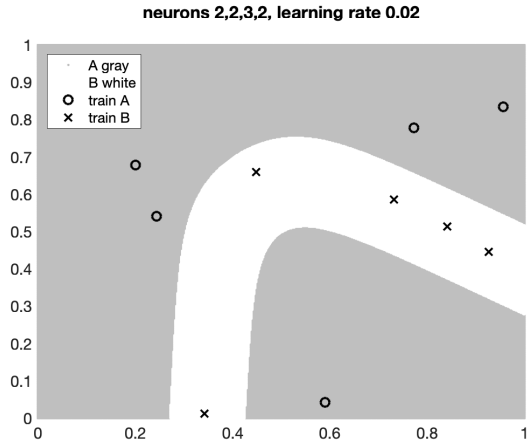


Figure 1: Learning rate vs iteration number and classified points

Problem 2 [5 points] Implement in Matlab the bisection method for finding a root of a scalar equation. Consider the polynomial

$$\begin{aligned} f(x) &= (x-2)^9 \\ &= x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512. \end{aligned}$$

(a) (3 points) Write a Matlab script to evaluate this function at 161 equidistant points in the interval $[1.92, 2.08]$ using two methods:

- evaluate $(x-2)^9$ directly
- evaluate $x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$ using Horner's method

Plot the results in two different plots. Explain the differences between the two plots.

(b) (2 points) Apply your bisection method to find a root starting with $[1.92, 2.08]$, tolerance 10^{-6}, and using the Horner's evaluation.

Obviously, $r = 2$ is a root of multiplicity 9. Can you determine a value for r such that $|r-2| < 10^{-6}$?

Discuss your observations.

Submit

- hard copy: your plots and discussion
- Avenue: all Matlab code. Name the main program `p2.m`. It should produce all required results.

Problem 3 [8 points] Implement Newton's method for systems of equations.

Each of the following systems of nonlinear equations may present some difficulty in computing a solution. Use Matlab's `fsolve` and your own implementation of Newton's method to solve each of the systems from the given starting point.

In some cases, the nonlinear solver may fail to converge or may converge to a point other than a solution. When this happens, try to explain the reason for the observed behavior.

Report (in your hardcopy) for `fsolve` and your implementation of Newton's method and each of the systems below, the number of iterations needed to achieve accuracy of 10^{-6} (if achieved). Do not submit Matlab code to Avenue.

(a)

$$\begin{aligned} x_1 + x_2(x_2(5 - x_2) - 2) &= 13 \\ x_1 + x_2(x_2(1 + x_2) - 14) &= 29 \end{aligned}$$

starting from $x_1 = 15$, $x_2 = -2$.

(b)

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 &= 5 \\ x_1 + x_2 &= 1 \\ x_1 + x_3 &= 3 \end{aligned}$$

starting from $x_1 = (1 + \sqrt{3})/2$, $x_2 = (1 - \sqrt{3})/2$, $x_3 = \sqrt{3}$.

(c)

$$\begin{aligned} x_1 + 10x_2 &= 0 \\ \sqrt{5}(x_3 - x_4) &= 0 \\ (x_2 - x_3)^2 &= 0 \\ \sqrt{10}(x_1 - x_4)^2 &= 0 \end{aligned}$$

starting from $x_1 = 1$, $x_2 = 2$, $x_3 = 1$, $x_4 = 1$.

(d)

$$\begin{aligned} x_1 &= 0 \\ 10x_1/(x_1 + 0.1) + 2x_2^2 &= 0 \end{aligned}$$

starting from $x_1 = 1.8$, $x_2 = 0$.

Problem 4 [5 points] Consider two bodies of masses $\mu = 0.012277471$ and $\hat{\mu} = 1 - \mu$ (Earth and Sun) in a planar motion, and a third body of negligible mass (moon) moving in the same plane. The motion is given by

$$\begin{aligned} u_1'' &= u_1 + 2u_2' - \hat{\mu}\frac{u_1 + \mu}{((u_1 + \mu)^2 + u_2^2)^{3/2}} - \mu\frac{(u_1 - \hat{\mu})}{\left((u_1 - \hat{\mu})^2 + u_2^2\right)^{3/2}} \\ u_2'' &= u_2 - 2u_1' - \hat{\mu}\frac{u_2}{((u_1 + \mu)^2 + u_2^2)^{3/2}} - \mu\frac{u_2}{\left((u_1 - \hat{\mu})^2 + u_2^2\right)^{3/2}}. \end{aligned}$$

The initial values are

$$u_1(0) = 0.994, \qquad u_1'(0) = 0,$$
$$u_2(0) = 0, \qquad u_2'(0) = -2.00158510637908252242053786 2224.$$

Implement the classical Runge-Kutta method of order 4 and integrate this problem on [0,17.1] with uniform stepsize using 100, 1000, 10,000, and 20,000 steps. Plot the orbits for each case. How many uniform steps are needed before the orbit appears to be qualitatively correct?

Submit

- hard copy: plots and discussion.

Problem 5 [5 points] The following system of ODEs, formulated by Lorenz, represents are crude model of atmospheric circulation:

$$y_1' = \sigma(y_2 - y_1)$$
$$y_2' = ry_1 - y_2 - y_1y_3$$
$$y_3' = y_1y_2 - by_3$$

Set $\sigma = 10$, $b = 8/3$, $r = 28$, take initial values $y_1(0) = 15$, $y_2(0) = 15$, and $y_3(0) = 36$, and integrate this ODE from $t = 0$ to $t = 100$ using Matlab's **ode45**. Plot each component of the solution as a function of t. Plot also (y_1, y_2), (y_1, y_3), and (y_2, y_3) (in separate plots).

Change the initial values by a tiny amount (e.g. 10^{-10}) and integrate again. Compare the difference in the computed solutions.

Submit

- hard copy: plots and discussion