Do Not Share this document

CS/SE 4X03 — Assignment 4

Ned Nedialkov

18 March 2019

Due date: 3 April, 1:30pm, in class.

- Without MSAF, no assignments will be accepted after 1:35pm on the 3th.
- With MSAF, no assignments will be accepted after 1:35pm on the 8th.

The SVN submissions must be under subdirectory with name `A4` of your main directory.

Problem 1 [13 points] This is an exercise to obtain some minimal experience in machine learning.

Study sections 1 to 4 and 6 of `https://arxiv.org/abs/1801.05894`. I have covered them in class. Surely you would benefit if you learn as much as you can by reading the rest.

[(a) 5 points, `netbp.m`] Take the Matlab code from p. 17 of this article and modify it so you can pass parameters as follows

```
function netbp(points, labels, neurons, learning_rate, niter, file)
%Trains a neural network with 4 layers.
%points is a matrix with two rows.  points(:,i) contains the (x,y)
%coordinates of point i.
%labels is a matrix with two rows. Colum i is [1;0] if point i is in
%category A and [0;1] if it is in category B.
%neurons is a vector of size 3. neurons(1) is the number of neurons in
%layer 2, neurons(2) is the number of neurons in layer 3, and neurons(3)
%is the number of neurons in layer 4.
%learning_rate is the learning rate
%niter is the maximum number of iterations
%file is a filename where the file is created by
%save(file, 'W2','W3','W4','b2','b3','b4','savecost','learning_rate');
```

Then run the script `train.m`

```
clear all;
%obtain data for training
[points,labels] = gettrainingdata;
%we try 3 learning rates
learning_rates = [0.1 0.05 0.01];
files = {'w1', 'w2', 'w3'};
niter = 1e6;
neurons = [2 3 2];
for i=1:numel(files)
    netbp(points, labels,neurons, learning_rates(i), niter, files{i});
end
```

Do Not Share this document

where gettrainingdata.m is

```
function [points, labels] = gettrainingdata
points(1,:) = [0.1,0.3,0.1,0.6,0.4,0.6,0.5,0.9,0.4,0.7,0.2,.5,.8,0.1];
points(2,:) = [0.1,0.4,0.5,0.9,0.2,0.3,0.6,0.2,0.4,0.6,0.3,.5,.4,0.3];
labels = [ones(1,6) zeros(1,6) 1 0; zeros(1,6) ones(1,6) 0 1];
```

After executing this script, you should have the files w1.mat, w2.mat, w3.mat.

[(b) 3 points, classifypoints.m] 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 netbp.
%points is a matrix with two rows, where point(:,i) contains the
%(x,y) coordinates of point i.
%Returns vector category, where category(i) is 1, A,  if
%points(1,i) >= points(2,i) and 0, B, otherwise.
```

Then run the script visualize.m

```
close all; clear all;
[trainpoints,labels] = gettrainingdata;
%create points to classify
h = 0.01;
[x, y] = meshgrid(0:h:1,0:h:1);
points = [x(:), y(:)]';
files = { 'w1.mat', 'w2.mat', 'w3.mat' };
for i = 1:numel(files)
    figure(i)
    categories = classifypoints(files{i},points);
    plotpoints(trainpoints,labels,points,categories);
    load(files{i});
    title(sprintf('learning rate %g', learning_rate));
    set(gca, 'FontSize', 18);
    print('-depsc2', sprintf('w%d.eps', i));
end
figure(4)
%plot cost
for i = 1:numel(files)
    load(files{i});
    n  = 1:1000:numel(savecost);
    h(i) = semilogy(n, savecost(n));
    hold on;
    s{i} = sprintf('learning rate %g\n', learning_rate);
end
legend(h, s);
xlabel('iteration')
ylabel('cost')
set(gca,'FontSize', 18);
print('-depsc2', 'cost.eps');
```

where plotpoints.m is

Do Not Share this document

```
function plotpoints(trainpoints, labels, points, categories)
A = find(categories == 1);
B = find(categories == 0);
h1 = plot(points(1,A), points(2,A), 'y.');
hold on
h2 = plot( points(1,B), points(2,B), 'm.');
%plot the training points
category = labels(1,:) >= labels(2,:);
A = find(category == 1);
B = find(category == 0);
h3 = plot(trainpoints(1,A),  trainpoints(2,A), 'bo','LineWidth', 2);
h4 = plot(trainpoints(1,B),  trainpoints(2,B), 'bx','LineWidth', 2);
legend([h1,h2,h3,h4],'A yellow','B magenta','train A','train B', ...
    'Location', 'NorthWest')
```

After running the above `train` and `visualize`, I obtained the results in Figure 2. With a different

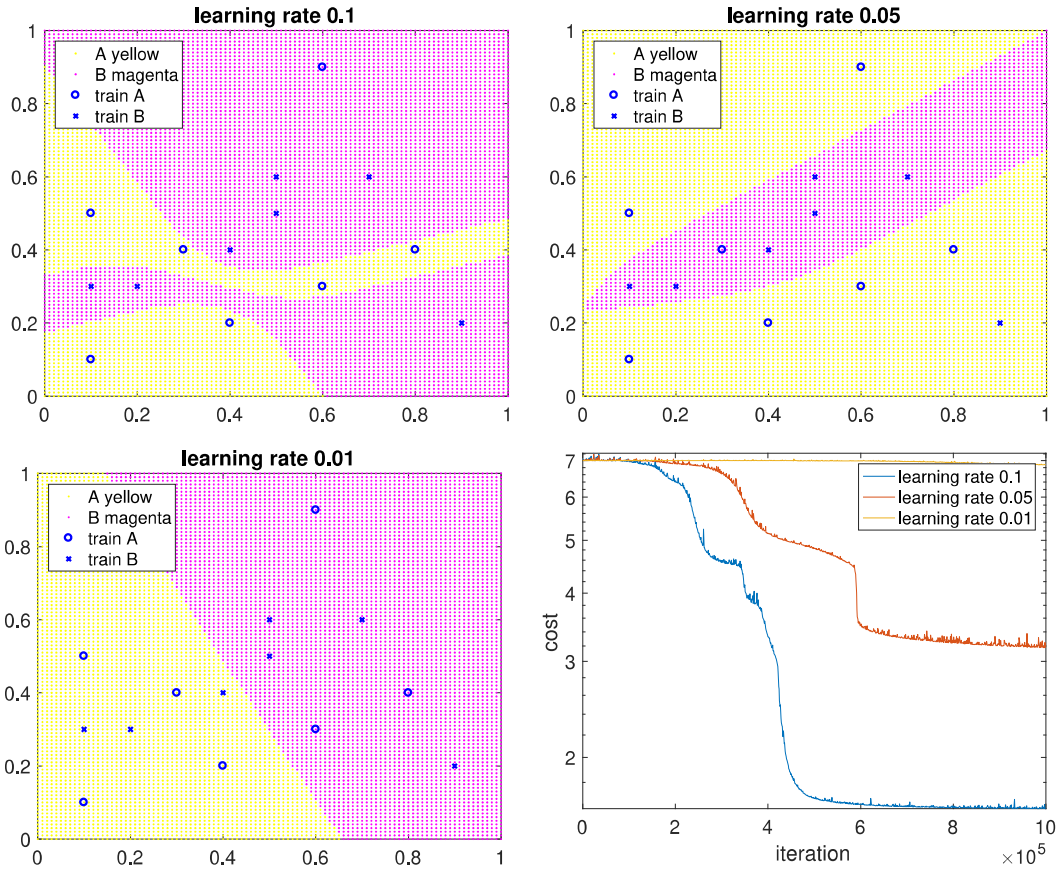

Figure 1: Layer 2 has 2 neurons, 3 has 3 and 4 has 2. "train" means point used in training.

choice of values for `neurons`, I obtained the results in Figure 2.

Experiment with different values for the learning rate and number of neurons. If you wish, you can add more layers to the network. The interface to `netbm` does not change. For example, if you add two more layers, you can set e.g. `neurons = [10 20 30 20 10]`.

Do Not Share this document

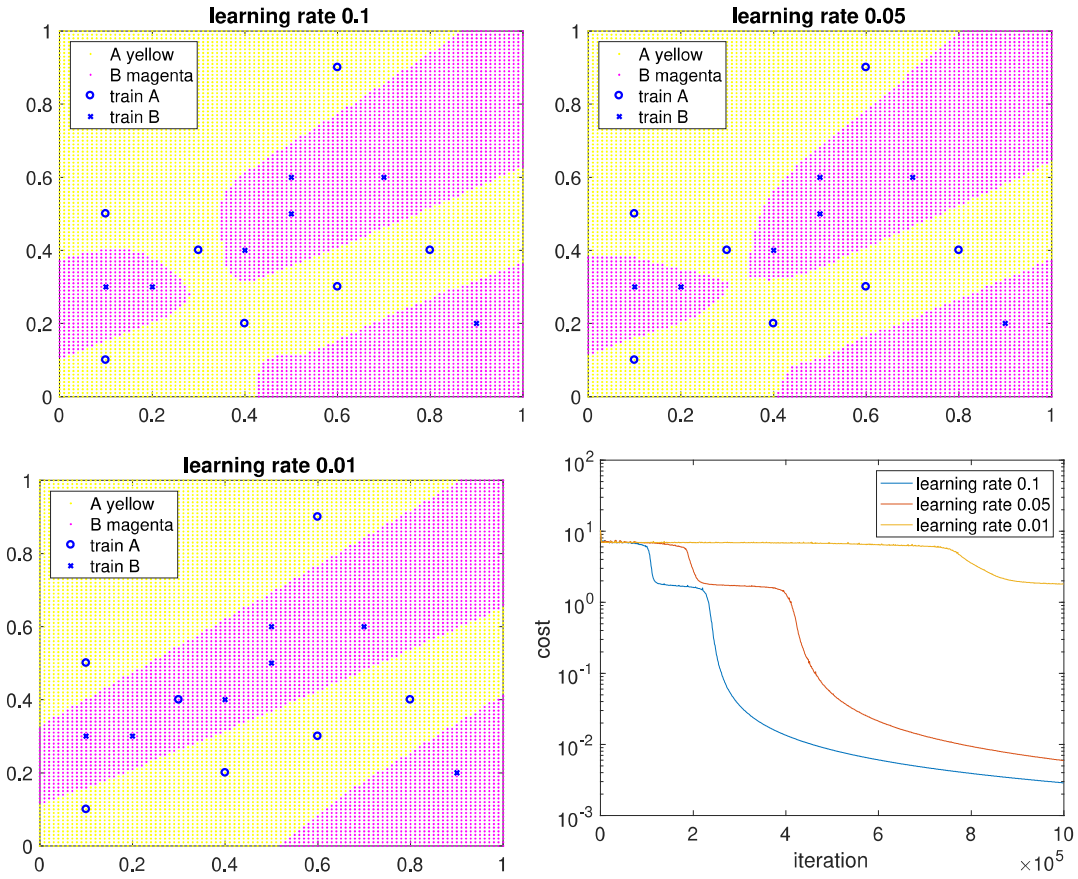

Figure 2:

[(c) 5 points] For what setup: learning rate, number of layers, neurons per layer you obtain the best result? (It is not required that you increase the number of layers.)

Submit

- hard copy: `netbm.m`, `classifypoints.m`, a figure like Figure 1 with your best choice for number of neurons and learning rates, and discussion for (c).
- SVN: `netbm.m`, `classifypoints.m`

Problem 2 [5 points] Implement in Matlab the bisection and Newton's method for finding roots of scalar equations.

Use your implementation of the bisection method to find a root of

1. $x - e^{2-\sqrt{x}} = 0$ in $[1, 3]$ and a root of
2. $x \sin(x^2) - 1 = 0$ in $[0, 4.5]$.

Use your implementation of Newton's method and Matlab's `fzero` to find a root of

1. $x^3 - 2x - 5 = 0$ with initial guess (for the root) $x_0 = 4.5$ and a root of
2. $x \sin(x^2) - 1 = 0$ with initial guess $x_0 = 4.5$.

Do Not Share this document

Submit

- hard copy: computed roots (use `format long e` before you output them).

  Comment on the accuracy of your computed solutions compared to the solutions computed by `fzero`.

For the remaining problems, submit in the hard copy what they ask for.

Problem 3 [3 points] Consider Newton's method on

$$x^5 - x^3 - 4x = 0$$

(a) How do the computed approximations behave with $x_0 = 1$?

(b) Try your implementation with $x_0 = 1 + 10^{-10}$. Explain why this method behaves differently, when started with $x_0 = 1 + 10^{-10}$, compared to when it is started with $x_0 = 1$.

Problem 4 [5 points] Use your implementation of Newton's method to compute a root of

$$f(x) = e^{-x^2} - \cos(x) - 1$$

with $x_0 = 0$ and then with $x_0 = 1$. Try also `fsolve` with these initial values.

Explain the behavior of Newton when $x_0 = 0$. Do you obtain (nearly) the same results with your implementation and `fsolve` when $x_0 = 1$?

Problem 5 [5 points] Consider Newton's method for solving the scalar nonlinear equation $f(x) = 0$. Suppose we replace the derivative $f'(x_k)$ with a constant value d and use the iteration

$$x_{k+1} = x_k - f(x_k)/d.$$

(a) Under what condition for d will this iteration be locally convergent?

(b) What is the convergence rate in general?

(c) Is there a value for d that would lead to quadratic convergence?

Problem 6 [3 points] Suppose that you compute a root of $F(x, y) = 0$ given by

$$\begin{aligned} e^{-x} + e^{-y} - 1.0001 &= 0 \\ 10^6 xy - 1 &= 0 \end{aligned}$$

using Newton's method with an initial guess $(x_0, y_0) = (0, 1)$. Can this method encounter difficulties? Try `fsolve` by calling it as

```
[x,fval,exitflag,output] = fsolve(...)
```

Considering the output of `fsolve`, did it converge to a root? Plot the norm of the residual, i.e. $\|F(x_k, y_k)\|$ versus iteration number k. What conclusions can you make from this plot?

Do Not Share this document

Problem 7 [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

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

The initial values are

$$u_1(0) = 0.994, \qquad u_1'(0) = 0,$$
$$u_2(0) = 0, \qquad u_2'(0) = -2.00158510637908252242053786222 4.$$

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?

Problem 8 [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.