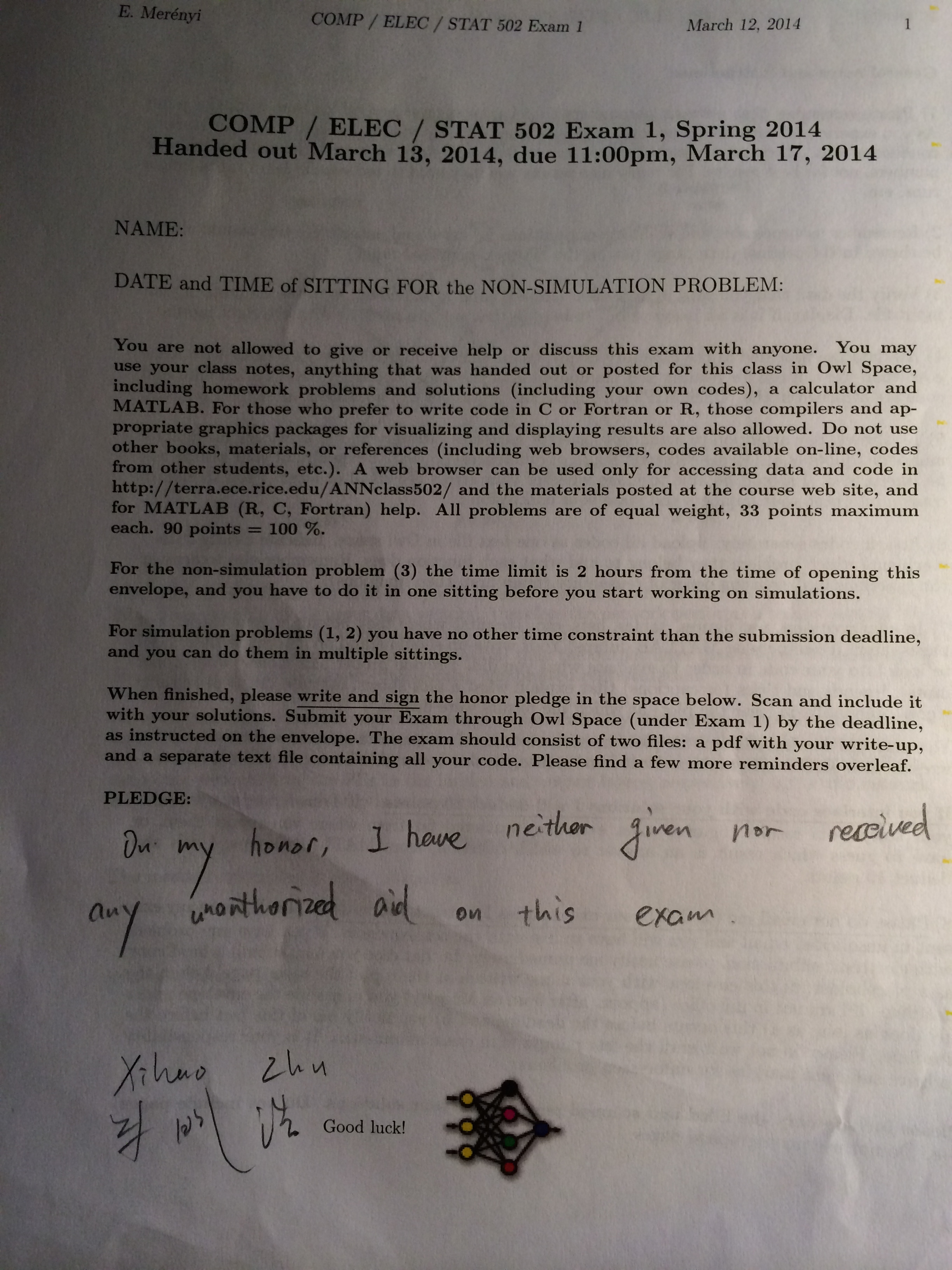
Xihao Zhu

Xz36

Comp502 midterm exam

Honor code:



P1.

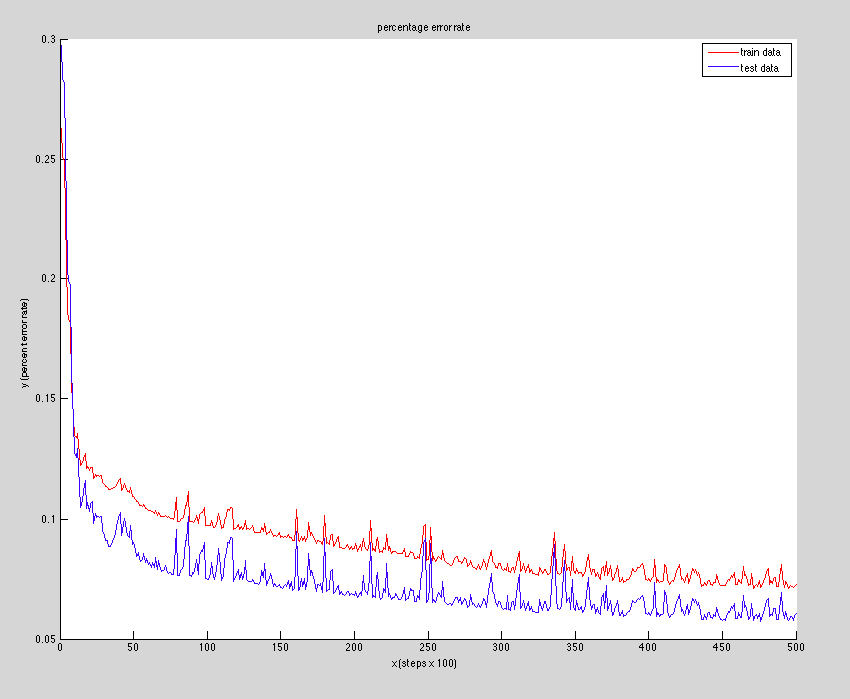
(1.1)

For my parameters, I use 50000 steps and set learning rate as 0.02. A momentum term constant is 0.3; stopping criteria is stop after 50000 steps, 15 hidden layers and 100 as epoch size.

|  |  |
| --- | --- |
|  | value |
| learning steps | 50000 |
| learning rate | 0.02 |
| momentum term constant | 0.3 |
| layer # | 15 |

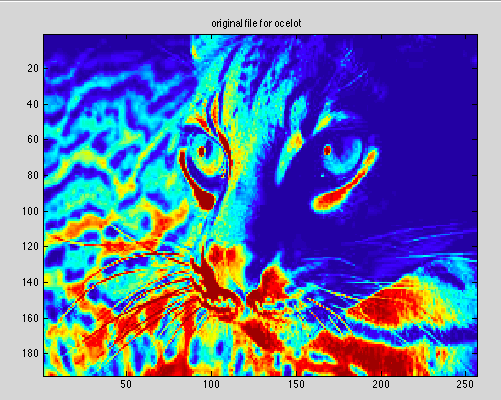
For error rate calculation, I use average percent error. The formula is to get sum of absolute value of every network’s output minus desired output. Then average the sum of error to be error per pixel. That is, once I got sum of errors(S) of my network output from every input, I calculate average error like this: E=S/(192\*256). (there are 192\*256 pixels as I understand)Then since data range is -1 to 1, so percent error Percent E=E/2.

Here is the plot(y is percent average error)

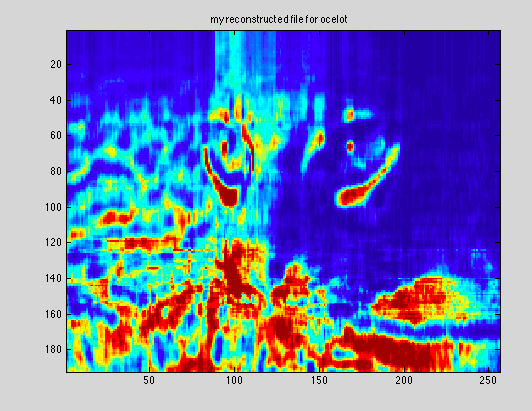


So finally I got average percent error rate for train data(ocelot) to be 8%, while that for test data(fruitfill) is 6%. I think this result is good.

For Ocelot picture, here is original file:

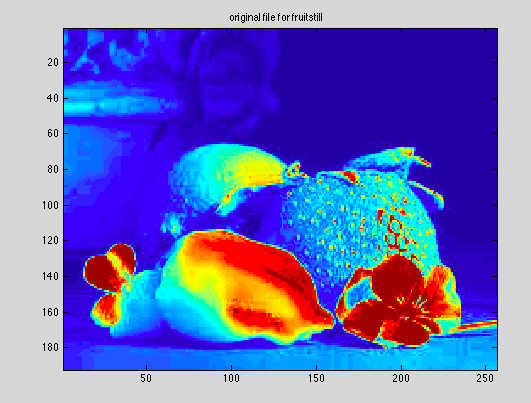


here is my reconstructed file:

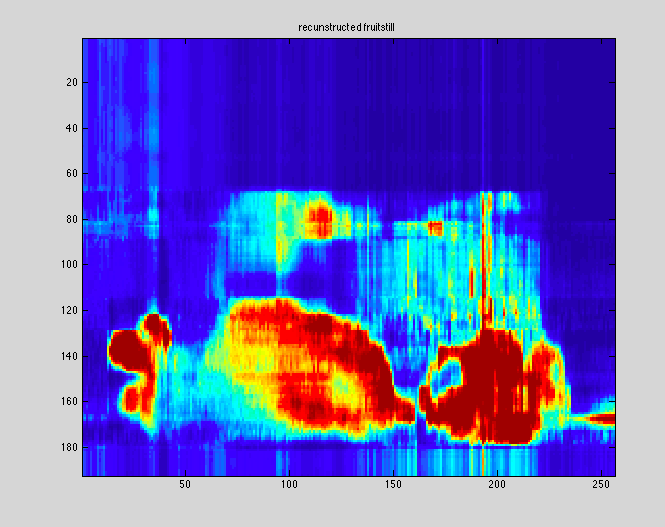


For fruitfill:

Here is original file



Here is reconstructed file:



P2.

(2.1)

So I use x2 to compute the eigenvectors for the left quarter of this image.

Eigenvector with x2 preprocessed to have 0 means in columns.

Eigenvectors =

[

0.0869 0.7317 0.6212 0.2669

0.5129 0.5450 -0.5633 -0.3500

0.6041 -0.3392 0.5270 -0.4912

0.6031 -0.2275 -0.1381 0.7510

]

(2.2)

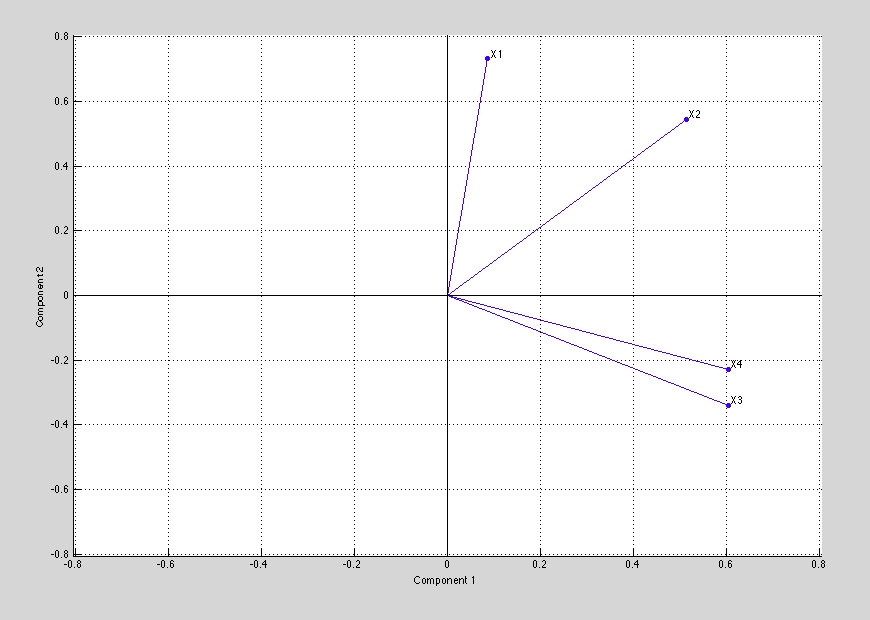
PCA analysis: 1st component:

0.0869 0.7317

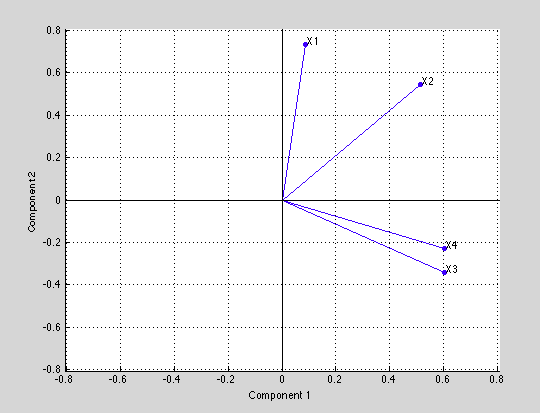
0.5129 0.5450

0.6041 -0.3397

0.6037 -0.2284



Matlab’s 1st PC



P3.

