Assignment 9 (Reference lecture: 09/04)

For extra credit (5pts) : submit before 09/07 (8pm)

Final Due: 09/09 8:00 pm

Submission Instruction:

Step 1: A pdf file submitted on Gradescope. This file should contain the code and output. The easiest way to generate this file is to first do screenshots(Snipping Tool on Windows, Shift-Command-4 on Mac), then paste the screenshots on a word file and transform it into a pdf file. You can also directly transform an ipython notebook to pdf format: File -> Print Preview -> right click and Print -> change Print Destination to "Save as Pdf". You are welcome to share your proven method on piazza.

Step 2: Code file submission to either the email address(cse190na4ai@gmail.com) or your private repository. This file should match the pdf version in step 1 and will serve as a reference if we find some problems with the pdf file submitted.

Two subassignments

1. Implement backpropagation for two hidden layer Neural Networks

- $NN_2 1(n, m, m, n)$ (*two* hidden layer neural net with *linear* activations having n input/output neurons and m hidden layer neurons (this is a parameter in the code) no need to have bias b vector
  presented on slide 5 of week5
- $NN_2 2(n, m, m, n)$ (*two* hidden layer neural net with *sigmoidal* activation function), no need to have bias b vector
  presented on slide 7 of week5

Apply them for minimizing the same loss for the same dataset as in Assignment 8.
Perform **2000** steps of gradient descent using $\alpha = 0.01$ to train the nets (may need to decrease it).
Using three sizes of networks $m = 10, 25, 50$.

Output:

- Final loss value
- Plot of loss values along the gradient descent path
- Compare final loss with the one hidden layer with same m implemented in Assignment. Two hidden layer net with same m should have a smaller loss than its one hidden layer counterpart.

2. Let $n = 10,\ t = 50$.

Perform a classification task of the provided input points stored row-wise in data-matrix X into n different classes labeled using entries of vector Y by minimizing the SVM loss (**slide 14 of week5**).

Set the margin value using $\Delta = 1$ (*may need to be adjusted in the case of NN based SVM*).

As the function f in the SVM classifier, use

1. Linear SVM classifier $f(x_i, W) = W \cdot x_i$ (**slide 13 of week5**) ($W \in n * n$)

Let $X =$ np.load('assignment9_X.npy') ($t * n$ dimensional matrix) be the matrix storing training inputs arranged row-wise, let Y = np.load('assignment9_Y.npy') ($t * 1$ dimensional vector) be the vector storing desired labels (i-th entry of Y is the index of the correct class for the i-th training pair (in range 0-9)).

Perform 500 steps of gradient descent using $\alpha = 0.01$ or lower if necessary.

Output:

- Final loss value
- Plot of loss values along the gradient descent path

Partial credit rubric:

Part 1. (3pts)

NN1: 0.5pts for each correct implementation of m.

NN2: 0.5pts for each correct implementation of m.

Part 2. (6pts)

1.: 5pts for overall correct implementation, 1pt for obtaining relatively low loss.

Plots (1pt)

Clear and correctly formatted loss plots, even if the plotted values are incorrect.

Reminder on assignment grades:

ALL assignments are graded on a 10 pts scale, BUT they will be weighted according to their difficulties in calculating the total score for all assignments. In particular, later assignments (like this one) are more difficult than earlier ones, thus will be weighted more.