EEE 598 **HOMEWORK ASSIGNMENT 3** **Fall 2019**

Due on Monday, October 21, 2019 at 11:59 pm
Answer all problems following the instructions. Please include the written and programming portions in a zipped folder titled "Lastname_firstname_hw3.zip"

Problem 1. *(Perceptron)*

1. Let D be the following dataset consisting of points $x_i = (x_1, x_2) \in \mathbb{R}^2$ with associated label $y_i = \pm 1$, written in form $[(x_1, x_2), y]$. Let

   $$D = ([(1,1),+1],[(2,-2),-1],[(-1,-1.5),-1],[(-2,-1),-1],[(-2,1),+1],[(1.5,-0.5),+1])\,.$$

   Perform all steps of the perceptron algorithm by hand, going through each point in order (left to right, do not randomize the order). Show all calculations, and plot the hyperplane and all data points (with their labels) for every iteration. Note: you will have to come up with a way to take care of the bias term. Report the final converged solution, and plot it.

2. Now write a code to perform the perceptron algorithm given any set of data points. Show that you can reproduce the result of the dataset given in the step above.

3. Add the minimum number of additional points to make this particular dataset not linearly separable. Show that your code keeps looping and doesn't attain convergence (it suffices to measure the change in w from iteration to iteration, and plot this to show that it never converges to zero).

Problem 2. *(Support Vector Machines)*

1. Using the same dataset as the previous problem,

   $$D = ([(1,1),+1],[(2,-2),-1],[(-1,-1.5),-1],[(-2,-1),-1],[(-2,1),+1],[(1.5,-0.5),+1])\,,$$

   let's solve this problem using SVMs.

2. First download/import the Sci-kit Learn library, `https://scikit-learn.org/stable/`. The main SVM function is SVC in this libary. Write a code to find a SVM which linearly separates the above data. Plot this support vector machine and its (hard) margin. In particular, you may find the tutorial handy: `https://scikit-learn.org/stable/auto_examples/index.html#support-vector-machines`.

Problem 3. *(Image Classification on CIFAR-10)*

For this problem, you will need to use either Google Colab or Agave server. I highly recommend Agave if you are planning to use it for your project, as you can practice running code on the server.

1. This problem will get you used to PyTorch. First complete this tutorial to train a LeNet Style architecture on CIFAR-10. Make sure to train on the GPU, and report the best accuracy you can achieve (you can run for as many epochs as you like). **Tutorial link:** `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#sphx-glr-beginner-blitz-cifar` You also can use other PyTorch tutorials to get yourself familiar with the language (highly recommend the MNIST tutorial if its your first time with PyTorch).

2. **Dataset augmentation:** One technique to improve the accuracy of your classifier is to show more diverse data through dataset augmentation. Implement dataset augmentation with rotations, translations, flipping, and adding a small amount of noise to the images. Show some examples of the augmented training data, and then report the performance of this technique on yoru LeNet-style architecture. How much did performance improve?

3. **Resnet:** Implement Residual blocks from the "Resnet" architecture from a well-known paper: `https://arxiv.org/pdf/1512.03385.pdf`. Show that your new architecture with residual connections outperforms the previous network architecture. You can find a version of the Resnet pytorch model here: `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py`. Report your final performance.

Problem 4. *DC-GAN*

1. Follow the PyTorch tutorial for DCGANs: `https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html`. Try to train your DCGAN to get as best performance as possible. Show the resulting output images of your trained GAN.

2. **New Dataset of Colored Squares:** Create your own dataset of 64x64 images, with black backgrounds and colored squares. The dataset should have squares of different colors, sizes, and orientations. Create a dataset of at least 1000 images. Save these images when you generate them.

3. Train DC-GAN on your dataset of colored squares. How well does it perform? Does it learn to create squares? If not, keep increasing the size of the dataset until it does.

4. **Collect a Dataset of Favorite Animal:** Now pick a favorite animal of yours. Collect a small dataset of 500 images of this animal, resize all the images to the same size of 64 x 64. To make collecting images easier, you can try the following extension: `https://chrome.google.com/webstore/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm?hl=en`. Or any other resources that you can find on the web to download images in bulk.

5. Train DC-GAN on your new dataset. Try various tricks to improve performance (either add more data beyond the 500 images, dataset augmentation, different learning rates, network architecture/sizes). There's no right answer here, just try to get some performance or document any mode collapse you get.