Homework 4
600.482/682 Deep Learning
Spring 2019

March 11, 2019

Due Fri. 03/15 11:59pm.
Please submit a zip file including your runnable '.ipynb' jupyter notebook, name as 'First Name_Last Name_HW4.ipynb' to Gradescope "Homework4 - Zip Submission" with entry code MYRR74 and a separate PDF report to Homework 4 - Report.
You are supposed to program using PyTorch framework in Q2 and Q3.

1. In lecture 7, we have learned that a two-layer MLP can model polygonal decision boundaries. You are given data sampled from a "two-patagons" decision boundary (refer to lecture 7, slide 70). Points within this structure are "true" (output is 1) while points outside are considered "false" (output is 0). This decision boundary can be modeled with a total of $2 \times 5 + 2 + 1$ neurons with threshold activation function. There is no need to manually split the data, only consider results on this one single dataset for this toy example. The name of the data file is "HW4_data.npy". Please load this via np.load function. You will obtain a numpy array with the shape $(60000, 3)$. The first two colomns are x, y coordinates and the third one represent the labels. The vertices of the polygons are (500, 1000), (300, 800), (400, 600), (600, 600), (700, 800), (500, 600), (100, 400), (300, 200), (700, 200), (900, 400) Plase use the first 50000 points as your training set and the rest as your testing set. Please

   (a) **Manually** setup an MLP with threshold activation* by selecting weights that can model the above decision boundary and verify that the all samples are classified correctly. Threshold activation is $y_i = 1 \cdot (w_i^T x + b_i > 0)$

   (b) Now, we have found one network model that can perfectly classify this problem. The above manual setup used threshold activations that are non-trainable. Consequently, please replace the 'threshold' with sigmoid activation,

   $$y_i = \frac{1}{1 + e^{-(w_i^T x + b_i)}}$$

   and change every node/perceptron such that its parameters can be trained. Randomize the parameters and use gradient descent to optimize the parameters. (Hint: random initialization can be set using a uniform random between the largest and smallest parameter value in 1(a) you have set). In the last layer, consider an activation of > 0.5 as "true", i. e. output is 1. Please try multiple times (e.g. 5) and report your findings. Can you find a solution that performs as good as the above "true" solution?

   (c) Now, define an MLP with increased capacity (try playing with a minimum of two different setups trading off increased depth with increased width) and see whether you can achieve higher classification accuracy.

2. In this problem, we start to play with convolutional neural networks (CNN).

   (a) Please download Fashion MNIST official released dataset.(https://github.com/zalandoresearch/fashion-mnist). Design a small CNN (e.g. 3-4 hidden layers) using the tools we encountered in class (e.g. convolution and pooling layers, activation functions, regularization). **Please think hard for yourself, do not talk with your colleagues nor use code available online. Please design ahead with what you think makes sense. This is NOT about performance.**

(b) Train your network and report the results on the test data of Fashion MNIST.

(c) Compare your architecture and results to at least one (preferred two) colleagues in our class. **Briefly** state the differences between your and your colleagues' architectures and results, and reason about why one or the other may perform better.

(d) Now, design an improved architecture based on your discussions and try to improve on your (and your colleagues') performance: Implement at least one of dropout and batch norm, use augmentation, play with optimizers, and try to beat all previous scores. Report your best results on the test set of Fashion MNIST. Try to compare again to your colleagues' results.

3. Supervised v.s. unsupervised learning. Please again use Fashion MNIST dataset.

(a) An auto-encoder like structure is one of the most popular CNN architectures in computer vision. The encoding part reduces the original input dimension into a low-dimensional "code" (feature) while the decoding part expands this representation back into its original form (the image) as best as possible. The reason why this structure has gained popularity is because good filter kernels can be learned even though no labels (annotations) are available, because the target task is reconstruction of the input image albeit passing through a data-bottleneck. Learning without annotations is known as unsupervised learning. We have provided the implementation of such autoencoder together with this homework. Please refer to the instructions in Appendix. Train the provided autoencoder on the Fashion MNIST dataset and visualize the feature maps extracted in the first layer (closest to the input).

(b) These feature maps are likely noisy, and we hope to improve on these by training a denoising auto-encoder. The network structure of the denoising autoencoder remains the same as above, however, you will need to work with data augmentation of your inputs: In particular, please corrupt your input images by adding strong noise (Gaussian, Salt and Pepper, completely set pixels to zero) **but** keep the output (the target you are trying to reconstruct) unchanged. Please train your network again and visualize the feature maps. Do you see any difference?

(c) With the above trained model, please freeze your autoencoder layers and "cut off" the decoding portion of the autoencoder. Connect the last layer of the encoding branch to a fully connected layer, and train the parameters of this last fully connected layer to be able to perform classification. Compare the performance of this model to your result in Problem 2.

Appendix for problem 3

1. Please download the provided code, 'autoencoder_sample.py'. We have provided a sketch code of a sample autoencoder network structure. Please fill in your work and run the training. Detailed comments can be found in the file. Note: this is not an optimal structure for this problem, you are encouraged to do more testing and tweaking. Improvement of structure(for Q3(a)-(c)) with analysis will receive bonus points, but it is not required.
   * Make sure you have saved your model after your training (the same for Q3-(b)), because you will need to use this model in Q3-(c).

2. The sample code is using pytorch interface to download the Fashion MNIST dataset. You can either modify the input based on this method, or write your own data loader. Please state your method in your report.

3. Please refer to the comments in the sample code to load your trained model and freeze layers. You need to figure out how to connect the fully-connected layer, and modify the training section to make it work.