CIS 520, Machine Learning, Fall 2021
Homework 8
Due: Tuesday, December 9th, 11:59pm
Submit to Gradescope

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the **written homework** to understand the problem and reach a solution in groups. However, **it is recommended that each student also write down the solution independently and without referring to written notes from the joint session.** You must understand the solution well enough to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

Learning Objectives

After completing this assignment, you will be able to:

- Derive optimal behavior policies using Markov Decision Processes
- Understand and implement Q-learning
- Understand the relationship between Autoencoders and PCA

Deliverables

This homework can be completed **individually or in group of 2**. You need to make one submission per group. Make sure to add your team member's name on Gradescope when submitting the homeword's written and coding parts.

1. **A PDF compilation of `hw8_template.tex` with your solutions**
2. **A `hw8.ipynb` with the functions implemented**

1 Performance Measures for Face Detection in Images [20 points]

Consider a face detection problem, where the goal is to build a system that can automatically detect faces in images. Two research groups develop systems for this problem using slightly different approaches. In both cases, the central component of the system is a binary classifier which, when applied to a 24×24 image, decides whether or not it is a face. The two groups train their classifiers using different learning algorithms. Moreover, when given a new image, they also apply their classifiers in slightly different ways: group A tests each region of 24×24 pixels of the image taking strides of size 2 (so, for example, for a 100×100 image, $(39)^2$ regions would be tested); group B tests each region of 24×24 pixels of the image taking strides of size 5 (so here, for a 100×100 image, only $(16)^2$ regions would be tested).[1] On a standard benchmark suite of test images that contains 300 40*40 images of faces altogether, the two groups have the following performances (assume the regions tested by both systems include all the 300 true face regions, while a region of a centered face translated for ± 2 pixels is also considered as a region of face, but is not considered as a face if translated for ± 5 pixels, so there will be 2700 face regions for method A, and 300 true face regions for method B):

Research group	**Number of regions tested**	**Number of faces detected correctly**	**Number of non-face regions detected as faces**
A	24,300	2457	243
B	4800	276	72

1. **[10 points]** Based on the above numbers, calculate the TPR (recall), TNR, and precision of each group's system as tested. Also calculate the resulting geometric mean (GM) and F_1 measures for each system. If you were to select a system based on the GM measure, which system would you choose? Would your choice change if you were to select a system based on the F_1 measure? (Note, the geometric mean (GM) is defined as $\sqrt{TPR \times TNR}$.)

2. **[4 points]** Which performance measure would be more suitable for this problem – the GM measure or the F_1 measure? Why?

3. **[2 points]** Another way to determine which method to choose would be to look at the ROC curve. Because you are given instances of different algorithms, not the algorithm itself, each method corresponds to a *point* on the TPR vs. FPR graph, not a curve.

   What is the Euclidean distance from each instance to the 0-error (perfect classification) point $(0, 1)$? Based on this metric, which method would you choose?

4. **[4 points]** Now assume that there is a third group C which trains their classifier using a newly discovered learning algorithm. Some of the resulting statistical measures are described below:

Research group	**TPR**	**TNR**	**FPR**	**FNR**
C	0.95	0.990	0.01	0.05

   (a) Suppose you worked for a social media platform, where you have a large number of pictures, and want to avoid wrongly tagging some objects as faces. Would you prefer the algorithm created by group C over the algorithms created by groups A and B?

   (b) Now suppose you worked for law enforcement, where every face detected will be checked against a criminal database.You'd like maximize specificity TPR in this case to maximize the probability of finding the criminal. Would you prefer the algorithm created by group C over the algorithms created by groups A and B?

[1] In practice, the 24×24 classifier would also be applied to multiple scaled versions of the input image; we ignore this issue here for simplicity.

2 Autoencoders [38 points]

Recall from Homework 6 that we examined reconstruction using PCA. We will now look at the same problem of reconstruction from a neural networks perspective. In this section, you will be working with the Fashion MNIST dataset [2] and look at fitting a standard autoencoder to it, and look at the reconstruction properties. See the notebook template for a primer on autoencoders.

You will be using PyTorch for this on Google Colab. Template code is provided along with the notebook.

2.1 Part 1: Constructing the Autoencoder

1. **[10 points]** You will first be constructing the autoencoder class and training it with a fixed architecture. Let n be the size of the bottleneck layer. The autoencoder architecture is as follows:

Layers	Size
Fully Connected Layer 1	Input size: 28*28*1, Output Size: 256
ReLU 1	—
Fully Connected Layer 2	Input size: 256, Output size: 64
ReLU 2	—
Fully Connected Layer 3	Input size: 64, Output size: n
ReLU 3	—
Fully Connected Layer 4	Input size: n, Output size: 64
ReLU 4	—
Fully Connected Layer 5	Input size: 64, Output size: 256
ReLU 5	—
Fully Connected Layer 6	Input size: 256, Output size: 28*28
Tanh 1	—

Table 1: Architecture for a vanilla autoencoder

Note that the first three layers and ReLUs are part of the "encoder," while the last three layers plus the final Tanh output are part of the "decoder."

For 3.1 and 3.2, let the bottleneck layer $n = 2$ for this question. The Optimizer to use will be Adam, with a learning rate of $1e^{-3}$, and the loss is the Mean Squared Error. Use a batch size of 128, and train for 25 epochs. Normalize the input dataset with a mean and standard deviation of 0.5. **It will take up to 5 minutes to train a single model, so please plan accordingly.** Look for the code marked `#TODO` in the notebook and complete the code between the segments `#Start Your Code` and `#End Your Code`.

Please report the reconstructed images from Epoch 0, Epoch 10 and Epoch 20. These reconstructions are the output of your autoencoder for the last minibatch of the epoch. Record these outputs for 0, 10, 20 Epochs respectively. Also report the training curve of the autoencoder (mean epoch loss vs epochs).

2.2 Part 2: Latent Space Decomposition

1. **[5 points]** You will now plot the latent space of this autoencoder, which is essentially the encoding that is obtained for any given input. In this problem, the latent space is the output of the ReLU 3,

[2] https://github.com/zalandoresearch/fashion-mnist

after the Fully connected layer 3. Use your learned model from the previous section, and plot the images with the class label '5', '7', and '8' in the 2D latent space.

2. **[6 points]** Based on the previous image, demonstrate both the difference and similarity among the patterns of label '5', '7' and '8' in the latent space decomposition plot. How are they different and/or similar? And Why? Recall the dataset label are: T-shirt/top 0, Trouser 1, Pullover 2, Dress 3, Coat 4, Sandal 5, Shirt 6, Sneaker 7, Bag 8, and Ankle boot 9.

3. **[5 points]** From the latent space plot, explain what the encoding has done to the inputs. How is this effect related to what PCA does? Why is this useful?

2.3 Part 3: Reconstruction Error vs Bottleneck Layer

1. **[5 points]** In this section, you will extend your code in Part 1 to a variable size bottleneck layer. Use your existing code in Part 1 and run the same for the bottleneck layer sizes $[4, 8, 16, 32, 64]$. Record the training curves and the final reconstruction errors for the input images belonging to the class label 5 for each of the sizes. Report the combined training curves (mean epoch loss vs epochs) for all the configurations. Also report the reconstructed images for Epoch 20 for each of the configuration.

2. **[7 points]** Plot the mean reconstruction error of the 5 different trained models when the input images belong to the class label 5, with respect to the size of the bottleneck layer. What are you observing? How does what you see relate to PCA? What does this tell you about how you can potentially work with a high-dimensional data-space?