1. Introduction

Recommendation systems are used in many online systems to help promote content to users according to users choices. In this exercise, we will create a matrix factorization neural network module using PyTorch.

The data set used in this exercise was from an IMDB database of anonymized users and their ratings for users. The data set contains roughly 1 million user-rating pairs with 72 000 users and 10 000 ratings.

Cross validation was used to determine the average loss and find the best model. Learning rate was also experimented with during the experiment.

2. Task 1

The first task involved creating the matrix factorization algorithm without biases. In this part, we trained on all 5 train/test splits generated by the shell script given in the dataset. There were 3 learning rates tested to find the best average model. These learning rates were 0.001, 0.01 and 0.1. Through experimentation with number of epochs run, it was found that the time taken for the 15 models to run for 5 epochs was a good compromise on time vs number epochs. Therefore, all data runs were done on 5 epochs.

The figure below shows a matrix of how each model performed for each learning rate when not accounting for bias, these numbers are the loss on the testing set for each of the cross-validation tiers.

	0.001	0.01	0.1
r1	1.99036	1.95970	1.69193
r2	1.98626	1.95878	1.69105
r3	2.00347	1.97396	1.70321
r4	1.98580	1.95741	1.68818
r5	2.00555	1.98354	1.76225

This next figures shows the average loss for each of the learning rates. This was used to identify the best learning rate for task 1. The best learning rate for task one (and for task 2 as seen later) was found to be lambda = 0.1. Thus, all models were run with a value lambda = 0.1.

```
0.001    1.994288
0.01     1.966678
0.1      1.707324
dtype: float64
```

3. Task 2

The second task was to perform the same task as Task 1 except taking into account specific user and movie biases. As above, the task was run by training on 5 epochs on all cross validation sets and on all 3 learning rates. The figure below shows the learning rates for all three learning rates for every cross validation set.

	0.001	0.01	0.1
r1	2.97840	2.94573	2.60989
r2	3.01181	2.97444	2.63317
r3	2.98273	2.94717	2.60980
r4	2.99956	2.96329	2.62450
r5	2.98710	2.96090	2.68367

The figure below shows the average loss across all cross-validation sets for each learning rate.

```
0.001    2.991920
0.01     2.958306
0.1      2.632206
dtype: float64
```

As seen in the above figure, the best cross-validated model for task 2 had a learning rate of 0.1. Which is the same as the learning rate of the best cross-validated model for task 1.

4. Task 3

The third and final task of this exercise was to create a prediction of the ratings for movies which were not seen by a specific user hadn't seen in the 5th cross-validation set. (r5.test) by training on r5.train with the best model found in the previous tasks. Recommendations were made using the forward method and then parsing for the top 5 argmaxes in the scores.