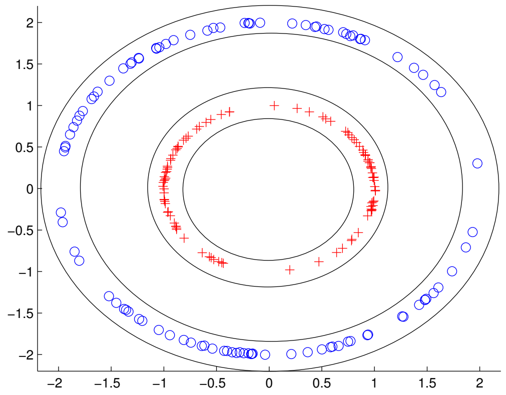
CS 391L **Machine Learning WB** **Spring 2023**

Homework 3

Lecturer: Inderjit Dhillon Date Due: Mar 29, 2023

Keywords: *Netflix, Regression, PyTorch*

1. Recall the 2-dimensional data in HW2 having two classes as shown in the figure below. The 'o' class lies in the tube $\{\boldsymbol{x} | 3.9 \leq ||\boldsymbol{x}||_2^2 \leq 4.1\}$ while the '+' class lies in the tube $\{\boldsymbol{x} | 0.9 \leq ||\boldsymbol{x}||_2^2 \leq 1.1\}$. Outline a solution



using support vector machines that would yield good classification accuracy on this data set.

2. **Neural Network based solution for Netflix prize dataset**
Recall the Netflix dataset from HW1, where one needs to predict missing (movie, user) ratings. For this assignment we will again work on the smaller subset of 2000 users, 1821 movies and 220,845 ratings. You can download this subset dataset from Canvas → Files → HW1 → dataset.zip.

In this assignment, we will be using a different approach than Alternating Least Squares (ALS) method that we used in HW1. More specifically, we will use a neural network based approach. On a high level the idea is to use a neural network that takes the user ID and movie ID as input and outputs the predicted rating. Your implementation should consist of the following steps:

- Load the training and validation data and create a PyTorch dataset class and dataloader class for the training and validation data.
- Create a neural network class with the following 4 parameters: 1) an embedding matrix for users (look `torch.nn.Embedding`) 2) an embedding matrix for movies 3) a hidden layer with `num_hidden` neurons and tanh activation 4) a linear output layer with 1 neuron and ReLU activation. The output of the network should be the predicted rating.
- In the forward pass of the network, you need to first convert the user and movie indices to their respective embeddings using the embedding matrices. Then concatenate the two embeddings and pass it through the hidden layer. Finally, pass the output of the hidden layer through the linear layer to get the predicted rating.

- Use the SGD optimizer for training and the mean squared error as the loss function
- Train your network for 20 epochs, with a mini-batch size of 2048 and report the RMSE on the training and validation set.
- Perform hyperparameter tuning to find the optimal learning rate, user/movie embedding dimension and number of hidden neurons. Report the optimal RMSE on validation set along with the optimal hyperparameters.

3. **Improving your neutral network solution**
   Now we'll improve our neutral network solution by changing different components of our training pipeline:

   - Change the optimizer, use AdamW (`torch.optim.AdamW`) instead of SGD (AdamW is an adaptive version of SGD, we'll learn more about AdamW later in the class). Again, tune your learning rate and report optimal validation RMSE.
   - In neural network training, typically L2 regularization is applied in form of weight decay parameter in the optimizer. Try passing non-zero values to the `weight_decay` argument to your AdamW optimizer and tune this parameter. Report optimal validation RMSE. Read more on weight decay here.
   - Normalize your train rating matrix such that all ratings lie between 0 and 1 (i.e. divide the ratings by maximum rating which in this case is 5). This allows us to treat this problem as a logistic regression problem as now we can pretend that the ratings are probability values (here the normalized rating values represents the probability of the movie being relevant to the user). Use binary cross entropy loss (`torch.nn.BinaryCrossEntropy`) to train the network (you'll need to use sigmoid activation after the last layer and during testing you need to renormalize your predicted scores such that they are between 0 to 5 again).