Hadi Askari
ECS 271 Assignment 2

Part 1: Regular Clustering with Matrix Factorization

My Approach and Reasoning:

In clustering, multidimensional data can be laid out in forms of a graph and the differentiating parameters for these nodes dictate their similarity or 'distance' from other nodes in the multidimensional space. Clustering allows us to classify data into clusters or groups such that nodes belonging to the same cluster are similar to each other and nodes which are unlike each other lie in different clusters. Spectral Clustering is particularly effective in this domain.

Detailed:

First I loaded the training data and created a pivot table between the users and the movies. This table was sparse since not every user had watched every movie. To get around this, I used SVD's expectation maximization (EM) formulation to impute the missing values and get a better representation of the graph. After that I used sklearn's "kneighbours_graph" as a metric to compute the distances between the the users and kept the number of nearest neighbours as 10, while using "minowski" as a metric. I also tried several permutations of cosine_similarity and kneighbours_graph with both the entire and the user/movie matrix after SVD but the aforementioned combination yielded the best results. Following that I used scikit_learns SpectralClustering library to perform the clustering. My parameters were: n_clusters=3, n_components (the top eigenvalues to select)=10, n_neighbours=10, affinity='precomputed_nearest_neighbours" and assign labels as kmeans. Again, I experimented with various combinations here and eventually settled on these numbers as they were performing the best in terms of MSE. In the background the Spectral Clustering algorithm performs these steps:

1. Calculating the Laplacian matrix
2. Compute the Eigen values and the Eigen vectors
3. Sorting and considering the non-zero Eigen values and corresponding Eigen vectors
4. And finally assigning labels to the data points from the training data where users with similar ratings for similar movies are clustered together.

Finally, after assigning each user to similar clusters, I predict the movie ratings by taking the mean of all ratings from all users belonging to the same cluster as the user whose rating is being predicted. This led to an MSE of 1.14443. In my 90-10 split validation set. I use this model to predict the ratings on the test set.

Flow Chart:

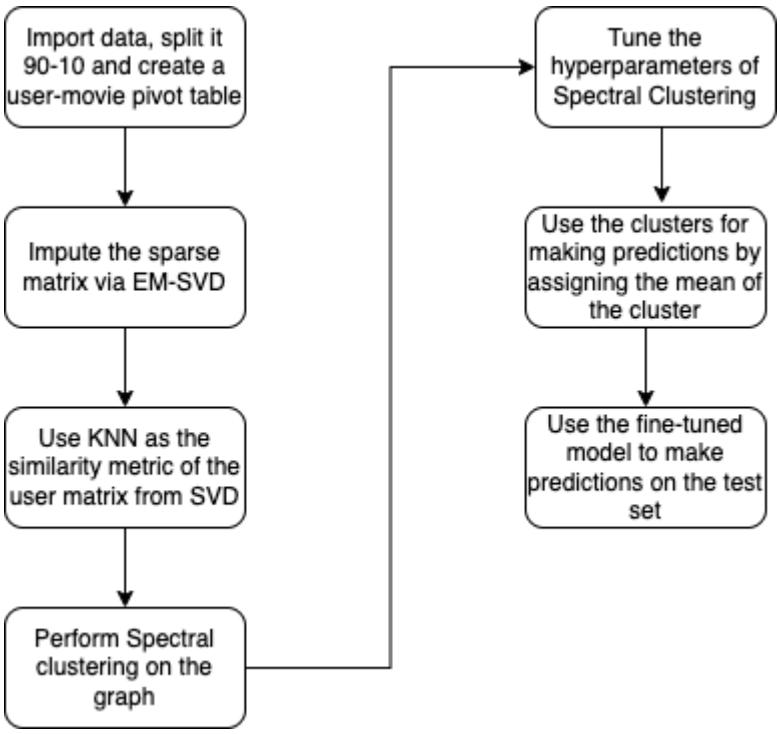


Figure1: The above flow chart shows the steps involved in Spectral Clustering used for the Netflix Recommender.

Part 2: Deep Clustering with IDEC and SDAE

My Approach and Reasoning:

Deep clustering algorithms can be broken down into three essential components: deep neural network, network loss, and clustering loss. There are many benefits of using deep clustering.

Autoencoders have found extensive use in unsupervised representation learning tasks ranging from denoising to neural machine translation. The simple yet very powerful framework is also used extensively in deep clustering algorithms. Most of the AE based deep clustering approaches use a pre-training scheme in which the encoder and decoder network parameters are initialized with the reconstruction loss before clustering loss is introduced. Thus we use a combination of **SDAE** and **IDEC**. The Stacked Denoising Autoencoder (SDAE) is an extension of the stacked autoencoder. This goes through a pretraining and a fine-tuning phase. IDEC proposes to incorporate an autoencoder into the DEC framework. In this way, the proposed framework can jointly perform clustering and learn representative features with local structure preservation.

Then the pipeline can be used as before as IDEC will provide similar clusters to Spectral clustering in part 1 and then we can generate our predictions.

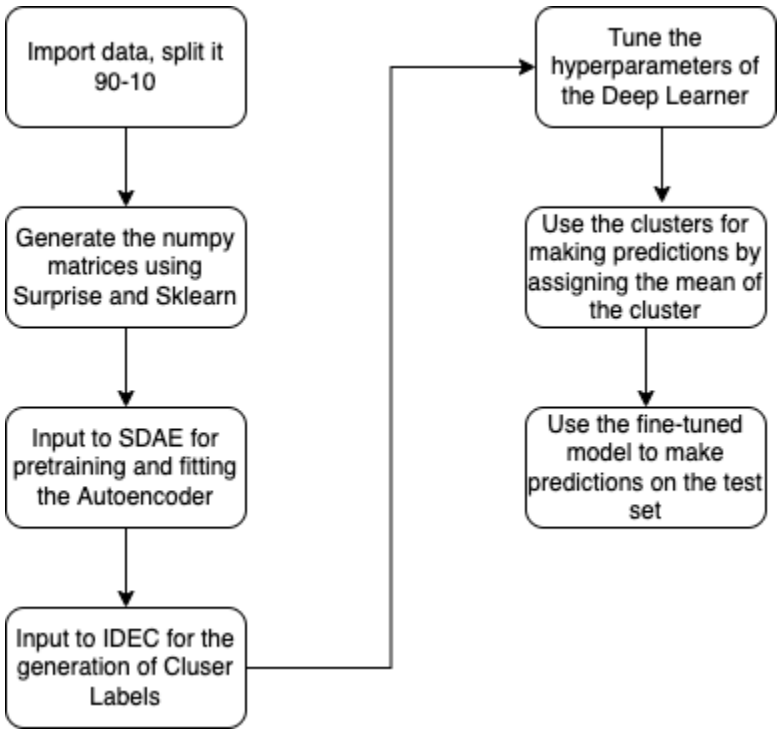
Detailed:

As stated before we don't need to perform any similarity distance metric to input to the deep learner so I just performed SVD and Kmeans on the original csv from the Surprise and sklearn library respectively and generated the npy files needed to run SDAE. The following was the architecture of SDAE:

```
StackedDAE(
  (encoder): Sequential(
    (0): Linear(in_features=15895, out_features=500, bias=True)
    (1): ReLU()
    (2): Linear(in_features=500, out_features=500, bias=True)
    (3): ReLU()
    (4): Linear(in_features=500, out_features=2000, bias=True)
    (5): ReLU()
  )
  (decoder): Sequential(
    (0): Linear(in_features=20, out_features=2000, bias=True)
    (1): ReLU()
    (2): Linear(in_features=2000, out_features=500, bias=True)
    (3): ReLU()
    (4): Linear(in_features=500, out_features=500, bias=True)
    (5): ReLU()
  )
  (_enc_mu): Linear(in_features=2000, out_features=20, bias=True)
  (_dec): Linear(in_features=500, out_features=15895, bias=True)
)
DenoisingAutoencoder(
  in_features=15895, out_features=500, bias=True
  (enc_act_func): ReLU()
  (dropout): Dropout(p=0.2, inplace=False)
)
```

I experimented with changing the parameters including the dropout, input_dimensions, changing the activations between relu and sigmoid, and z_dimensions. I experimented by just adding the user matrix after the SVD and the entire recomputed matrix after SVD as well. Since the Stacked Autoencoder performed well enough, I eventually went with the entire matrix. The final validation reconstruction loss was 0.1010 in SDAE. The architecture and parameters in IDEC were similar and I generated my cluster labels. Once again I saw that the minimum MSE was when I had 3 clusters and was 1.16. After that I re-used the methodology of assigning the average cluster labels from part 1 for generating the predictions on the test set.

Flow Chart:



Bonus:

Both of the methods had their pros and cons. The following are the benefits of Deep Clustering over normal Spectral Clustering which incorporate the differences in their methodologies as well.

- **High Dimensionality:** Many traditional clustering algorithms (like SC in part 1) suffer from the major drawback of the curse of dimensionality as they rely heavily on similarity measures based on distance functions. These measures work relatively well in low dimensional data, but as the dimensionality grows, they lose their discriminative power, severely affecting clustering quality. Deep clustering algorithms use deep neural networks to learn suitable low dimensional data representations which alleviates this problem to some extent. Even though classical algorithms like Spectral Clustering address this issue by incorporating dimensionality reduction in their design, neural

networks have been very successful in producing suitable representations from data for a large range of tasks when provided with appropriate objective functions.

- **End to End Framework:** Deep clustering frameworks combine feature extraction, dimensionality reduction and clustering into an end to end model, allowing the deep neural networks to learn suitable representations to adapt to the assumptions and criteria of the clustering module that is used in the mode

- **Scalability**: By incorporating deep neural networks, deep clustering algorithms can process large high dimensional datasets such as images and texts with a reasonable time complexity. The representation space learned are low dimensional spaces, allowing other clustering algorithms to efficiently cluster large real world datasets and infer cluster information in the real time after the initial training.

  However there are some drawbacks as well such as:

- **Hyper-parameters:** Deep clustering models have several hyper-parameters which are not trivial to set. The major drawback of deep clustering arises from the fact that in clustering, which is an unsupervised task, we do not have the luxury of validation of performance on real data.

- **Lack of interpretability:** Although interpretability is a big issue with neural networks in general, the lack of it is especially more significant in scenarios where validation is difficult. The representations learnt by deep neural networks are not easily interpretable and thus we have to place significant level of trust on results produced by the models

- **Lack of theoretical framework:** The majority of deep clustering algorithms we discussed above lack strong theoretical grounding. A model can be expressive and reliable without theoretical grounding, but it is often very difficult to predict their behavior and performance in out of sample situations, which can pose a serious challenge in unsupervised setup such as clustering.