Deep AutoEncoder Recommendation Engine using novel Dense Representation

Anant Gupta
Anant.Gupta@morganstanley.com

Abstract- *Traditionally SVD (Singular Value Decomposition) has been used for collaborative filtering. However SVD has certain limitations. It can identify only linear relationships and if the data is very sparse, SVD is not able to approximate the matrix very well. The current methodology for deep collaborative filtering attempts to replicate SVD by using neural networks. This methodology is able to capture non-linear relationships between a user and an item. But even with this approach, the sparsity of data still exists and when we apply neural networks on sparse data, the chance of overfitting is very high.*

To prevent this, we alter the user item matrix form to a simple tabular representation, where each row denotes the interaction between a single user and a single item. Those items which have no interaction with any user need not to be fed into the neural network. This provides a novel way to feed only useful information into the network. This data representation helps us effectively train the neural network with better accuracy.

Additionally this gives us a way to club similar users (and also similar items) into groups. This is because we are not using the autoencoder directly to generate the recommendation but instead using it to have separate latent representation of users and items using user item interactions. This latent representation can also be used to find similar users and similar item

Keywords: Deep Learning, Artificial Intelligence, Market Making, Portfolio Construction, Investment Banking, Recurrent Neural Networks, Auto Encoders, Modern Portfolio Theory

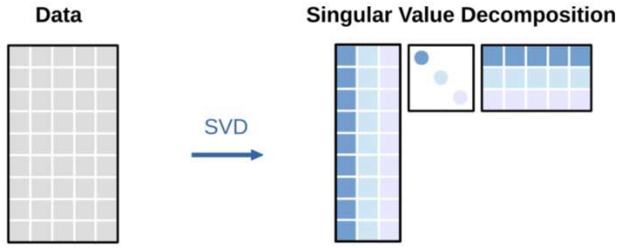
SECTION 1: INTRODUCTION

The current market is seeing a massive increase in the number of available products to be sold. Coupled with this, due to digitization, there is an increase in the number of tech-savvy consumers. This has increased the volume of both users and products. In order to stay relevant, retail companies are employing multiple methodologies to ensure that the customer receives correct recommendations. The ability to send customized recommendations and customer profiling is directly linked to revenues of a retail firm. The current methodologies employed for recommendations are Collaborative Filtering, SVD for arriving at recommendations. Most of these methods either fail at handling sparsity or non-linear relationships between users and items.

In this work, we attempt to solve both the above mentioned issues. We take only the data points for which we have a concrete user item relationship. This takes care of the sparsity problem when we have a lot of products that cater only to a small section of the user dataset. We then use the autoencoder setup to train our model. Usages of autoencoders capture the non-linear relationships between users and items. The resultant embeddings also captures the condensed representation of users/items within themselves. The user embedding is not dependent on the intrinsic features of the users but is dependent on the buying patterns of different items. This representation becomes a powerful tool to arrive at several kind of recommendation in the retail business

SECTION 2: RELATED WORK

A lot of research has been done on leveraging cutting edge technologies to arrive at targeted product recommendations for a user. The earliest and most commonly used methodology is Collaborative Filtering [1]. One of the most common methodologies in Collaborative Filtering is using **Matrix Factorization** (SVD Singular Value Decomposition). This process converts the user and item data into the same latent space. This representation is used to come up with recommendations for items for a user present in the same neighborhood. The drawback of this system is the high sparsity of data causes issues with the matrix factorization step. Feeding a highly sparse data into SVD fails sometimes because one is not able to calculate the eigen vectors



There are **Knowledge Based** recommender systems[2], in which the logic is translated into a function that scores each items against the user. This system is highly dependent on the function which can be created out of historical user activity. The biggest drawback for such a system is the inability to understand non-linear relationships between different users and items

There are also **Community Based** recommender systems [3] which leverage the social connections (friends/family etch) to determine the recommended products for a user. This system is highly dependent on the social connect and many retail firms do not have access or the means to access such relationships

The limitations faced by current recommender systems are

1) **Data Sparsity**

The input into the systems is not just the relevant data, but there is a lot of sparsity introduced because of the formation of user-item matrix. The actual events are extrapolated to the entire user x item space. This causes an error or weak recommendations in most methodologies [4]

Some ways in which this problem is overcome is by clubbing similar users based on user intrinsic features [4] This causes issues with recommendations because the user clubbing is done on intrinsic user similarity and not on user item relationships

Another method has been to use Truncated version of SVD which converts a large sparse matrix into a smaller matrix (low rank factorization). But this method still does not do away with the limitations of linear relationships

2) Linear Relationships

The usage of matrix factorization/linear methods (e.g SVD)[5] finds out the linear relationships between the users and items, but in reality there is a lot of non-linearity in these relationships which needs to be taken into consideration while providing recommendations

3) Similar Users

A retail business would definitely like to profile their users into groups for better targeting. The current methodologies of user similarities are based on intrinsic user features. The issue with this approach is that the user similarity should be based on transactions performed by the users

4) Similar Items

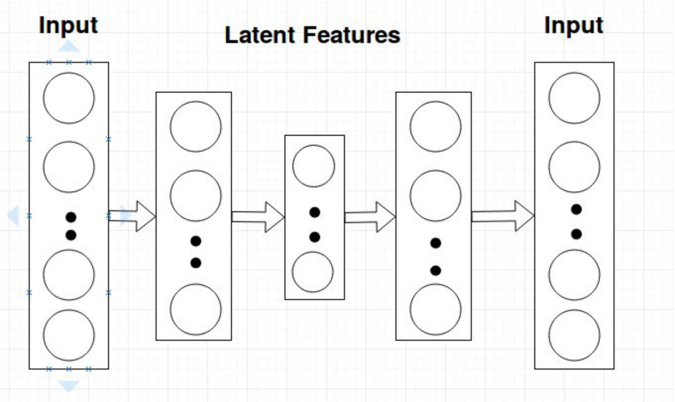
Similar to users, a retail organization would like to profile their items into groups as well. This helps the organization to introduce a new item into the recommendation without retraining the model

SECTION 3: RECOMMENDATION ENGINE USING DEEP LEARNING

AutoEncoders

Autoencoders are a powerful concept in the realm of deep learning. They are particularly strong in deriving latent features from an input dataset. The latent features derived capture all the intricate relationships between the input feature vectors. This is a significant win over the linear methodologies applied for feature generation like PCA etc. It is particularly helpful for compression techniques and feature generation.

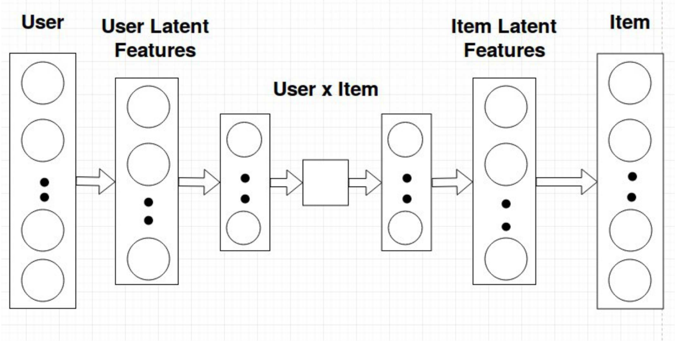
However, the use of it in recommender system is unheard of because it requires the same input and output. We can either pass the user data ot the item data. What we would like to have is an autoencoder that will generate the latent features for user and item data based on the input relationships between the user and item



Deep Auto Encoders are used quite extensively in feature learning in Deep Learning Domain. The encoder part of this Network takes a vector of Length N and an output vector of M where M<N. In short, it tries to condense all the information present in vector N into vector M. Then the Decoder part of this network takes vector M as an input and tries to reconstruct vector N. , This entire system is trained using backpropagation algorithm where the objective is to reduce the reconstruction error. In this process, the encoder part of the system learns the different linear and nonlinear distributions present in the data, because it has to reduce the size, but still needs to capture all the required information which will be used later by decoder to reconstruct the input. Once an Auto Encoder is trained, it is capable of representing all the information present in a vector in terms of linear and nonlinear distributions.

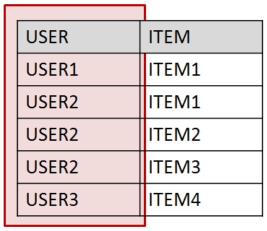
New Methodology

For achieving, this we will be using the principle of autoencoders in a slightly different and unique way



The above image has 2 extra layers in between. The traversal through each of the layers holds different significance.
The important layers of this neural network

Layer 1: User Input



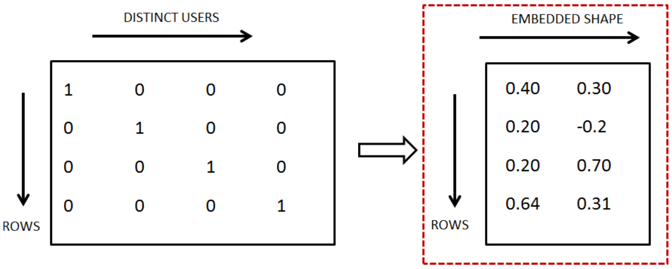
USER	ITEM
USER1	ITEM1
USER2	ITEM1
USER2	ITEM2
USER2	ITEM3
USER3	ITEM4

The above image is the tabular data for user and item columns. We take the user data out of it as a vector $U_{n;uo}$

Layer 2: User Embedding Vector

This is the first hidden layer: $UEmbedding_{n,ue}$
ue denotes the condensed user vector space

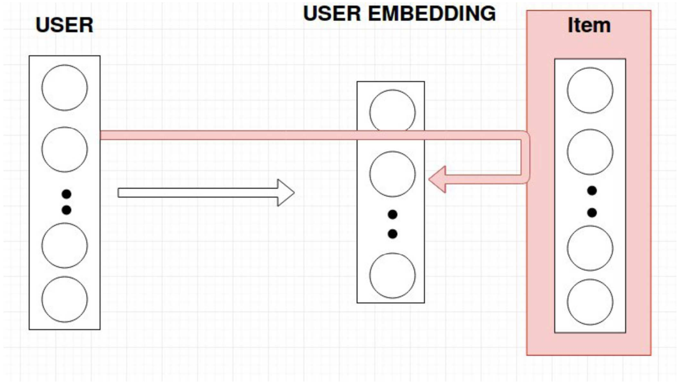
This layer will capture the input user data in a condensed vector space. The dimensions of this layer can be decided based on the number of users (< 1% of distinct users)



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

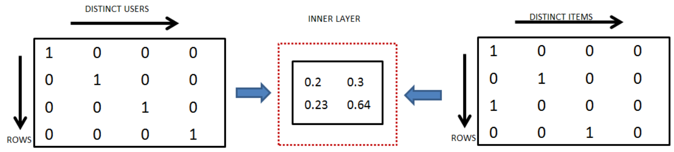
0.40	0.30
0.20	-0.2
0.20	0.70
0.64	0.31

We can see in the below figure that for the translation of User to User Embedding, the feedback mechanism takes into consideration the corresponding Item data as well. Due to this, the embedding generated captures the relationships between the users as well as the items



Layer 4: Inner most Layer

This layer represents the users and items in the same vector space similar to the user item matrix



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0.2	0.3
0.23	0.64

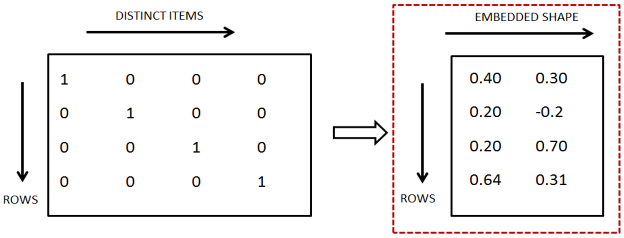
1	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0

This layer captures the interactions between the input user vector and the corresponding target item vector. An analogy can be drawn between the collaborative filtering matrixes where we have the user item matrix.

However, the major difference is that the features are compressed and we have no sparsity which is a big advantage.

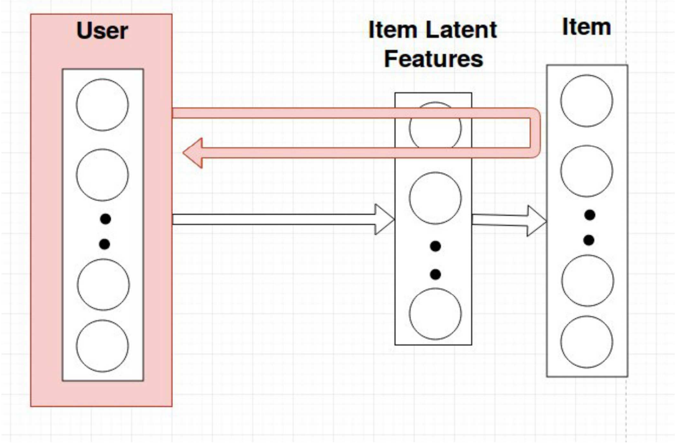
Layer 6: Item Embedding Vector

Similar to users, this layer represents the Embeddings of the Items. The number of rows remains the same



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0.40	0.30
0.20	-0.2
0.20	0.70
0.64	0.31



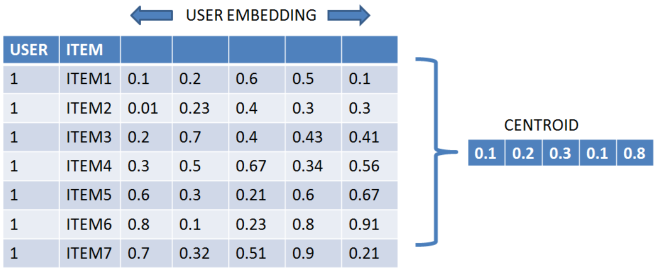
Layer 7 : Item Input

This is the input placeholder for the item vector

USER	ITEM
USER1	ITEM1
USER2	ITEM1
USER2	ITEM2
USER2	ITEM3
USER3	ITEM4

Final User Embedding

The final user embedding is calculated by taking the centroid of all the vectors for a particular user in the **User Embedding Layer** For a single user there can be multiple items in the dataset, and for each of the items, it will have a separate user vector in the user embedding space. We need to take the centroid to determine the singular vector for that user in the **User Embedding Space.** This vector will be the sole representation for the user

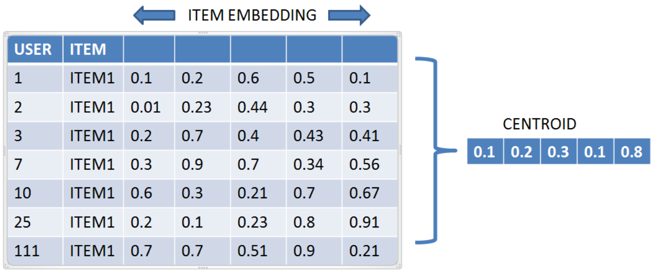

USER EMBEDDING
CENTROID

USER	ITEM					
1	ITEM1	0.1	0.2	0.6	0.5	0.1
1	ITEM2	0.01	0.23	0.4	0.3	0.3
1	ITEM3	0.2	0.7	0.4	0.43	0.41
1	ITEM4	0.3	0.5	0.67	0.34	0.56
1	ITEM5	0.6	0.3	0.21	0.6	0.67
1	ITEM6	0.8	0.1	0.23	0.8	0.91
1	ITEM7	0.7	0.32	0.51	0.9	0.21

0.1	0.2	0.3	0.1	0.8

Final Item Embedding

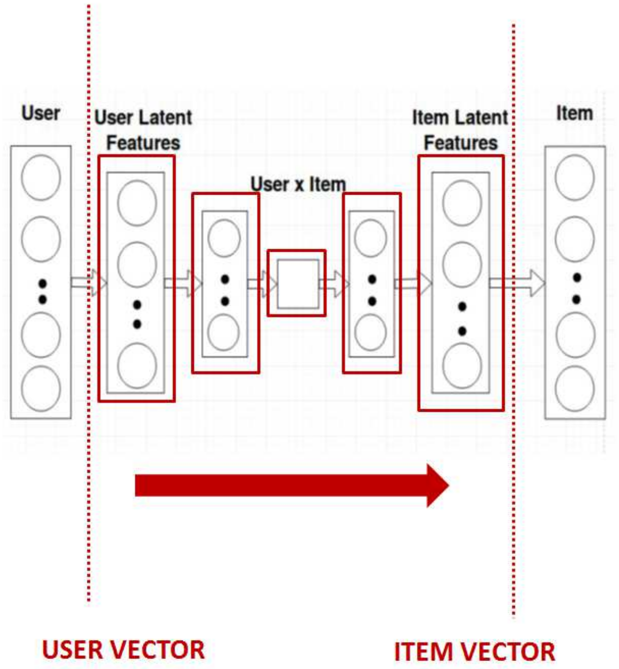
The final Item Embedding is calculated by taking the centroid of all the vectors for a particular item in the **Item Embedding Layer** Similar to users, there can be multiple user transactions on a single item, and for each of the users against this item, there will be a separate vector in the Item Embedding Layer. We need to take the centroid to determine the singular vector for that user in the **Item Embedding Space.** This vector will be the sole representation for the item



USER	ITEM					
1	ITEM1	0.1	0.2	0.6	0.5	0.1
2	ITEM1	0.01	0.23	0.44	0.3	0.3
3	ITEM1	0.2	0.7	0.4	0.43	0.41
7	ITEM1	0.3	0.9	0.7	0.34	0.56
10	ITEM1	0.6	0.3	0.21	0.7	0.67
25	ITEM1	0.2	0.1	0.23	0.8	0.91
111	ITEM1	0.7	0.7	0.51	0.9	0.21

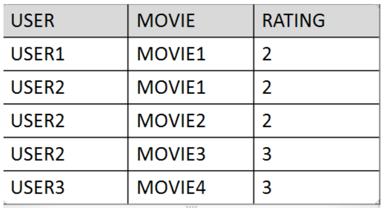
Recommendation

For coming up with recommendation, we need to pass the corresponding user vector into the neural network to get the corresponding vector in the Item Embedding space. We can then find the closest items within the neighborhood of this vector in the Item Embedding Space. We will take only the compressed Item Vectors



SECTION 4 : EXPERIMENTS AND RESULTS

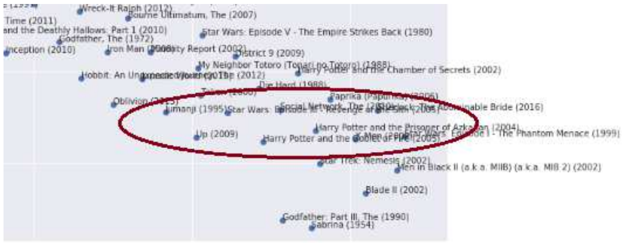
For testing, we will be using MovieLens dataset. We have ratings for the users against the movies and the data is in the following format



USER	MOVIE	RATING
USER1	MOVIE1	2
USER2	MOVIE1	2
USER2	MOVIE2	2
USER2	MOVIE3	3
USER3	MOVIE4	3

We will be considering the users with rating of 4 and 5 for experimentation and filtered on reviews made in the year of 2016. We took only 68 movies and sparse user reviews

For movie similarity based on viewership, we take the embedded vectors and perform t-SNE(7) and convert it to 2 dimensional graph. We then plot that graph



We can see that algorithm with very sparse data was able to distinguish movies that are childish or animated

SECTION 5: CONCLUSION AND FUTURE WORK

One of the major points of improvement will definitely be using a better algorithm to compress the multiple vectors in the embedding to a single vector. Currently, we are using centroid, but we can take a more neighborhoods approach rather than giving equal weights to every point. We can use Affinity Propagation methods to find the most representative vector for all the vectors [6]

One can also choose the embedding user/item shape to be 2 and that will force all users and items to be represented as (X,Y) format. This can then be visualized easily. One can also use T-SNE on the final user Embedding Vector to arrive at the (X,Y) format

One can also use Affinity Propagation methods to find the users that are most similar. The same can be done for items using the final embedding vector

References:

[1] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, pp. 291–324. Springer Berlin / Heidelberg (2007)

[2] Ricci, F., Cavada, D., Mirzadeh, N., Venturini, A.: Case-based travel recommendations. In: D.R. Fesenmaier, K. Woeber, H. Werthner (eds.) Destination Recommendation Systems: Behavioural Foundations and Applications, pp. 67–93. CABI (2006)

[3]Arazy, O., Kumar, N., Shapira, B.: Improving social recommender systems. IT Professional 11(4), 38–44 (2009)

[4] Gediminas Adomavicius, Alexander Tuzhilin : Towards the Next Generation of Recommender Systems
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.423.3802&rep=rep1&type=pdf

[5]Sonia Leach
Singular Value Decomposition – A Primer
http://people.csail.mit.edu/hasinoff/320/SingularValueDecomposition.pdf

[6] Inmar E. Givoni, Clement Chung, Brendan J. Frey
Hierarchical Affinity Propagation
https://arxiv.org/ftp/arxiv/papers/1202/1202.3722.pdf

[7] Laurens van der , Geoffrey Hinton
Visualizing Data using t-SNE
http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf