Autoencoders for Collaborative Filtering Recommender Systems

YanPeng Gao
101090653

April 14, 2020

Abstract

The task of this project was to investigate autoencoders for collaborative filtering recommendation systems on the MovieLens 1 Million dataset. A baseline matrix factorization approach was used to compare performances. Several autoencoder architectures such as shallow, deep, batch normalized, denosing and variational autoencoders were implemented and experimented.

1 Introduction

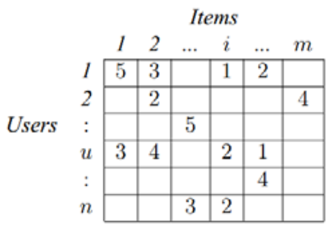
The most popular type of collaborative filtering approach for recommender systems are a family of matrix factorization techniques. In the past few years however, it has been proposed that autoencoders may be better suited for collaborative filtering. This project aims to implement and demonstrate the abilities of various types of autoencoder architectures to suggest the field of collaborative filtering should not be dominated only by matrix factorization. A background of the techniques will be discussed followed by the related work, how our experimentation was performed and the results of our experimentation.

2 Background

A recommender system is a system that uses a user's past activities to optimally suggest the best future activity for that particular user. Such activity could be the enjoyment of a movie, buying a product or new social media friend. Recommender systems are typically classified into two varieties: content based systems and collaborative filtering systems.

Content based system utilizes data on the properties of items and users. For example, if a user is listening to a song which is labeled a rock song, the system may recommend other rock songs for our user. Meanwhile, collaborative filtering examines the relationship between all the users and all the items to generalize a certain users preference of all items.

A popular approach used by the winning team in the Netflix challenge [3] is a form of matrix factorization known as Singular Value Decomposition.



		Items					
		1	*2*	...	*i*	...	*m*
	1	5	3		1	2	
	2		2				4
Users	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		

Figure 1: Example of User-Item Matrix

Given a sparse user item matrix M, the empty ratings are first imputed and then M is factorized as $M = U\Sigma V^T$. Matrix U has orthogonal columns that span the row space of M while Matrix V has columns that span the column space of M. Σ is a scaling matrix. In other words, the rows of U represent users in latent space while the rows of V represent items in hidden space. Predictions are made as $\hat{M}(i,j) = U(i) \times \Sigma \times V(j)$. SVD's will not be explored much further but will be used as a baseline to compare our autoencoder model's performance.

Autoencoders are a type of unsupervised neural network model where the goal is to compress and then reconstruct the input for as little loss as possible. For an encoder g_ϕ and a decoder f_θ, our reconstructed $\hat{x} = f_\theta(g_\phi(x))$. We train our model using the mean square error loss of x and $\hat{x}$. It can be shown that autoencoders perform a similar task as SVD for finding latent values of the item-user matrix. Recall the dimensionality reduction technique PCA where we are trying to find matrices U and P to minimize the $\|X - XPU^T\|$. P is a matrix that spans the columns of X, and is found by the eigenvectors of X^TX. This is therefore the same V in the SVD of $X = U\Sigma V^T$. We also note that for a linear autoencoder: $g_\phi(x) = Wx$, $f_\theta(z) = Vz$, we are optimizing for $\|X - VWX\|$, which yields the same subspace as PCA, hence showing autoencoders in general also look for latent variables.

3 Related Work

The first autoencoder collaborative filtering approach was introduced in 2015 known as AutoRec[6]. Work of AutoRec primarily centered around 1 layer encoding/decoding autoencoders with L2 Regularization in every layer using data from MoveieLens 1 million, MovieLens 10 Million and the Netflix challenge. Activiations were either sigmoid or identity(linear) and the optimizer used was resilient propagation (RProp). In 2017, the work of Kuchaiev and Ginsburg [4] demonstrated that with SELU activiation functions and dropout after the latent layer, deeper autoencoders can be trained. Dense refeeding was also introduced as a way to increase accuracy. In 2018, variational autoencoders [5], a generative architecture was used to achieve state of the art results. In this project, all 3 designs will be implemented as well as modified.

4 Methodology

4.1 Dataset and Baseline

Movielens 1-Million [1] was used as the dataset. Preliminary work had shown the 100k dataset to be too trivial for evaluation purposes while the 10 million dataset was too time and computationally intensive for this project. The dataset contained 1,000,209 reviews from 6040 users on 3706 movies resulting in an overall sparsity of 4.47%. The data was then cleaned by removing movies where there were less than 10 reviews resulting in 1670 fewer total reviews. The user-item matrix was created with users as rows and movies as columns. Five train/test splits of 80:20 ratio were created. The test sets are not disjoint but rather sampled randomly from the full dataset. Tuning and experimenting of the model was done using only one split while results are presented using all five splits.

A baseline using Singular Value Decomposition was first used. The RMSE achieved from this baseline is 0.8976, and so will be used to compare the utility of various autoencoders implemented below.

4.2 General Training

We define a loss function known as Masked Root Mean Square Error:

$$MRMSE = \sqrt{\frac{I[x](x - \hat{x})^2}{\sum I[x]}}$$

where for a given $x =< x_1, x_2, ..., x_j >$, $I[x]$ is an binary indicator vector that has $I[x]_k = 1$ when x_k is not equal to 0. During training, we are feeding the model our input x and getting a reconstructed $\hat{x}$. Since it is unsupervised, we do not require a label and are evaluating the error between our fed in vector and its reconstruction.

During testing however, we are still passing in the user-item training matrix but then are evaluating with the test matrix. The Masked RMSE becomes $\sqrt{\frac{I[x_{test}](x_{test} - \hat{x_{train}})^2}{\sum I[x_{test}]}}$ which allows us to only calculate the RMSE for our testing set and ignore the reconstruction of other values. This type of training was used in AutoRec[6] and by Kuchaiev and Ginsburg [4].

4.3 Single Hidden Layer Auto Encoder

We first implement the AutoRec design with one hidden layer which is connected by one set of weights that encode and another set of weights that decode. However, we used ELU and SELU activation functions instead of sigmoid and identity and trained with RMSProp. It was also found that while the weights were regularized with L2 in the orignal paper, we found that regularization at any magnitude hindered our training. Therefore, no regularization was applied. However, our findings of the latent dimension were similar to the findings from AutoRec. The accuracy increased until our latent vector had dimension 500 and was negligible after that (figure 3).

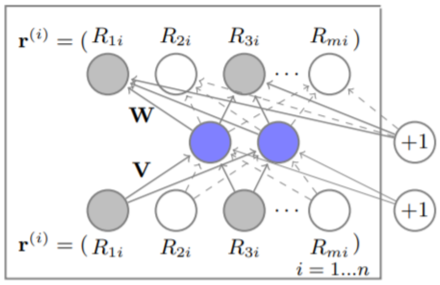

$\mathbf{r}^{(i)} = (R_{1i} \quad R_{2i} \quad R_{3i} \quad R_{mi})$
W
V
+1
+1
$\mathbf{r}^{(i)} = (R_{1i} \quad R_{2i} \quad R_{3i} \quad R_{mi})$
$i = 1...n$

Figure 2: 1 hidden layer autoencoder from AutoRec [6]

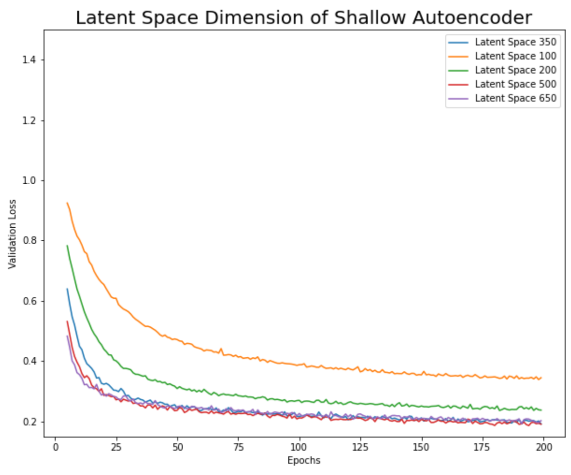


Figure 3: Dimensions of Latent Space for Single Hidden Layer Autoencoder

4.3.1 Denoising Autoencoder

It had been shown that for image based autoencoders, denoising of inputs demonstrated better reconstruction as well as increasing robustness of the feature weights [7]. Therefore, we implemented this for our collaborative filtering autoencoder. Each input is first passed through a dropout layer that removes a portion of the input. This was thought to increase generalizability as the loss function still includes the positions where dropout was applied. However, in experiments, it is shown that denoising autoencoders don't perform as well as regular autoencoders for this task. Therefore future experiments will not incorporate denoising.

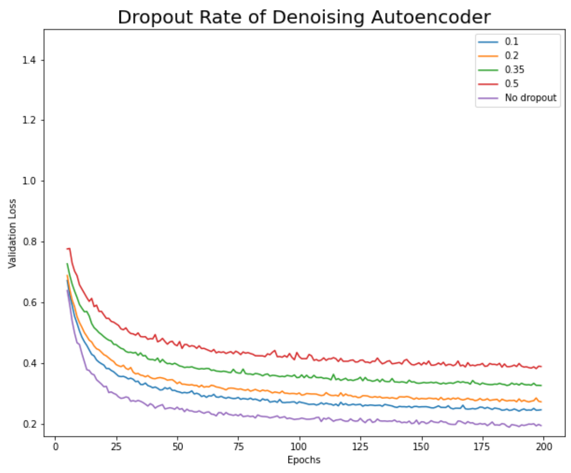


Figure 4: Denoising Level for Single Hidden Layer Autocncoder

4.4 Deeper Autoencoders

Instead of having one layer of encoding and one layer of decoding, multiple layers of each were explored. Dropout after the middle layer was performed for regularization as suggested by Kuchaiev and Ginsburg [4]. It was discovered that a deeper autoencoder with no modifications did not perform as well as a single layer autoencoder.

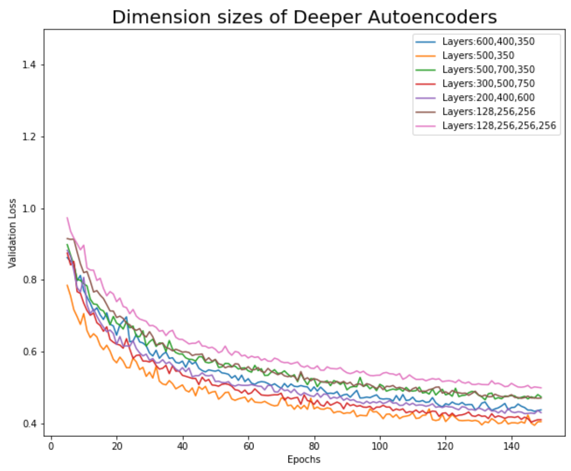


Figure 5: Performance of Deeper Autoencoders

4.4.1 Dense Refeeding

A proposed approach to increase the performance of autoencoders is dense refeeding. This technique refeeds the dense output of the reconstruction back as input. The goal is to provide the model with a non-sparse inputs each time to be able to train all the weights. However our results show that dense refeeding did not work for our project.

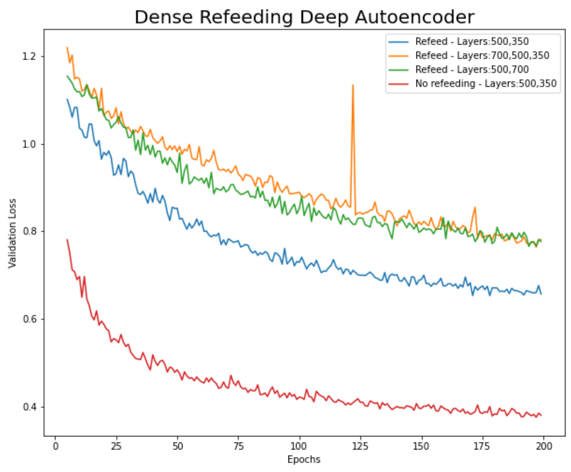


Figure 6: Performance of Deeper Autoencoders with Dense Refeeding

4.4.2 Batch Normalized Layers

Batch normalization is a recent technique applied to state of the art CNN's such a Residual Networks. No research has been done on applying batch normalization on autoencoders for collaborative filtering.

Batch normalization normalizes each layer of outputs with a linear transformation: $y_i = \gamma \hat{x}_i + \beta$ where $\hat{x}_i$ is the unit normalized feature and γ and β are learned features. Batch normalization helps solve the problem known as Internal Covariate Shift (Ioffe and Szegedy [2]) where the input distribution to layers deep in the network changes per mini-batch. With the elimination of the covariate shift, the model is able to stabilize the distribution regardless of the previous iteration's weight update, allowing for better training.

Indeed we found that an autoencoder with batch normalized layers performs better than the regular deep autoencoder.

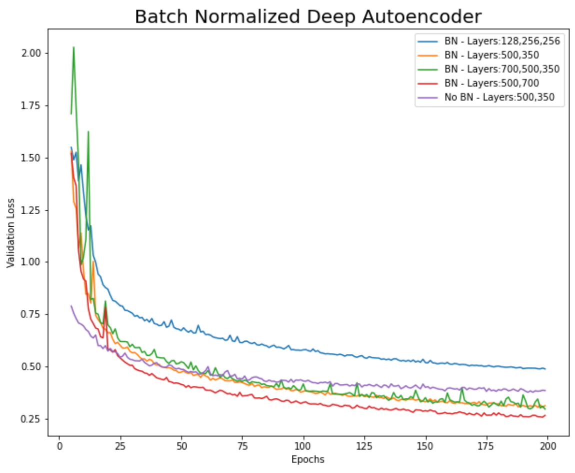


Figure 7: Performance of Deeper Autoencoders with Batch Normalization

4.5 Variational Autoencoders

Variational autoencoders are a type of generative neural network where instead of encoding a latent space representation, the encoder generates a mean parameter and a covariance parameter to form a distribution. We sample that distribution to form the latent representation which is then decoded.

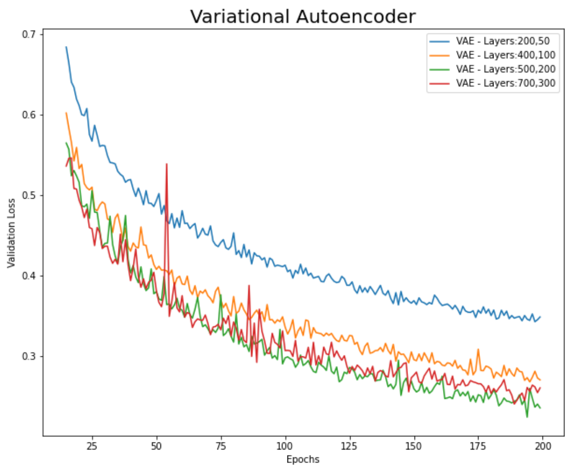


Figure 8: Performance of Variational Autoencoders

5 Results and Discusion

The best results for several autoencoder architectures are reported. Results were evaluated with all 5 folds and trained on 300 epochs. The loss is root mean square error at a 95% confidence interval.

Model	Test Loss	Training time (seconds)
Single Hidden Layer Autoencoder	0.1829±0.003	1557.20 ±11.42
Deep Autoencoder Layers:300,500,750	0.3634±0.008	1912.83±1912.83
Single Hidden Layer Denoising Autoencoder	0.2533±0.002	1676.45±9.03
Batch Normalized Deep Autoencoder Layers: 500,700	0.2263±0.004	2090.03±12.62
Variational Autoencoder Layers:500,200	0.2207±0.004	160.92±1.64
SVD (baseline)	0.8976±0.0004	136.12±0.69

In this project, it is demonstrated that all autoencoder architectures were able to perform better than our baseline SVD model. However it was very surprising that the simplest autoencoder design performed the best as related work indicated that both variational autoencoders and deep autoencoders with dense refeeding improved on the original one layer design. I believe this has to do with our dataset as our user-item matrix has a sparsity of 4.47%, while larger datasets may have a sparsity of around 0.1%. Furthermore, due to the relatively high sparsity, generative models such as VAE's were not necessary. Therefore we conclude that the choice of the autoencoder architecture greatly depends on the sparsity and size of the dataset; higher complexity may not always be better.

We also were able to suggest that batch normalization, a technique that has not been applied to collaborative filtering autoencoders before, is a useful method in training deeper autoencoders as supported by our results.

6 Conclusion and Future Work

In conclusion, we recommend that autoencoders be recognized as suitable alternatives for matrix factorization techniques on collaborative filtering. When training an autoencoder for this task, various models needs to be investigated as there is no one style of autoencoder that performs optimally on all datasets.

Future work should be done on more advanced matrix factorization techniques and not just singular value decomposition with mean item inputation. It was demonstrated that matrix factorization was significantly faster in training than most autoencoders so there could be a trade off decision between speed and accuracy when comparing advanced matrix factorization techniques and autoencoders.

References

[1] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL `https://doi.org/10.1145/2827872`.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 37:448–456, 07–09 Jul 2015. URL `http://proceedings.mlr.press/v37/ioffe15.html`.

[3] Yehuda Koren. The bellkor solution to the netflix grand prize. 09 2009.

[4] Oleksii Kuchaiev and Boris Ginsburg. Training deep autoencoders for collaborative filtering. 08 2017.

[5] Dawen Liang, Rahul Krishnan, Matthew Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. 02 2018.

[6] Suvash Sedhain, Aditya Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. pages 111–112, 05 2015. doi: 10.1145/2740908.2742726.

[7] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. page 1096–1103, 2008. doi: 10.1145/1390156.1390294. URL `https://doi.org/10.1145/1390156.1390294`.