Anchor paper

AutoRec: Autoencoders meet collaborative filtering (2015)
The main objective of the paper is to present an **innovative approach for solving Collaborative Filtering**. On Collaborative Filtering we **exploit information about users' preferences** for items (e.g., 1-5 star rating or consumed / non-consumed) to **provide personalized recommendations**.

The approach presented is based on *neural network* models, specifically, using the *Autoencoder* paradigm. At the time, the paradigm proved successful for vision and speech tasks.

For each training example, the network **receives a user** and the **items they rated/interacted with**, each rated item is an input node. The input is **projected to a k-dimensional space** by connecting the input nodes to a k-dimensional hidden layer, producing the user embeddings.

The user embedding is then connected to the output layer, trying to **reconstruct the ratings for all items**, including the ones not rated by the user. On training, the goal is to **minimize MSE over the known ratings**.

The authors suggested two approaches, *item-based AutoRec*, where the input is a list of all the users who rated item i, and *user-based AutoRec*, where the input is a list of all the items user u has rated. Item-based AutoRec produced **better results** on the benchmark datasets because the average number of ratings per item is much more than those per user.

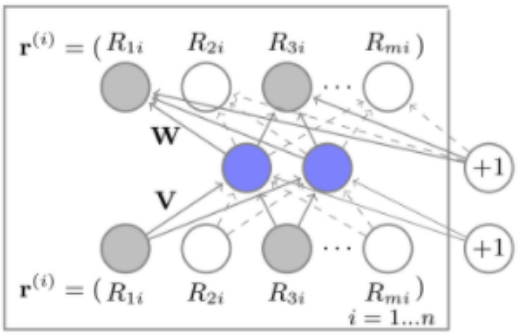


Figure 1: Item-based AutoRec model
Input is the list of users who rated item i
Output is the reconstructed list of users holding item i ratings for all users

The authors discuss **variations** of the model: identity/sigmoid/ReLU activation functions for V and W, different number of hidden units, and adding deeper hidden layers. All variations are tested on popular movie recommendations datasets - ML-1M, ML-10M, and Netflix datasets.

Suggested improvement

Problem

We tackle a profound problem in recommender systems - the **cold-start problem**. When recommending content for new users, with none or several rated items, strict **collaborative filtering algorithms under perform** w.r.t context aware approaches. On context aware approaches we **utilize auxiliary information** about the user. Information that usually exists while recommending content to the user, such as age, gender, occupation, and more.

Approach

Our approach extends the *user-based AutoRec* (U-AutoRec) to *context-aware user-based AutoRec* (CAU-AutoRec), by concatenating user context features to the input vector and to the hidden layer.

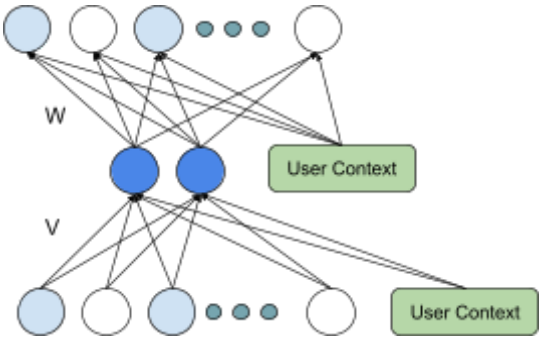


Figure 2: Context-aware user-based AutoRec model
User context features are concatenated to the input and hidden layer vectors

Features

First, we transform all user context features to a single vector by applying 1-hot transformation for categorical features, and concatenating all the 1-hot results. The concatenated result is concatenated once again to the input vector and to the hidden layer. An example for user context features is found on the MovieLens dataset, the user context contains:

Feature	# of categories	Examples
Age	7	Under 18, 35-55, 56+
Occupation	21	artist, farmer, programmer, unemployed
Gender	2	male, female

The concatenated vector is of size 30.

Metrics

To measure the improvement on the cold-start problem, we define a **dedicated metrics-family**, *Uk-RMSE*, which is defined as follows:

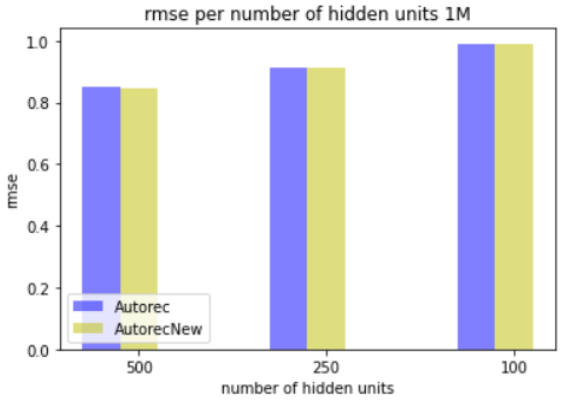
$$Uk = \sqrt{\Sigma_{r \in S} ||r - h(r;\theta)||_2^2} \; where \; S = \{r^{(i)} | \, |r^{(i)}| \leq k\}$$

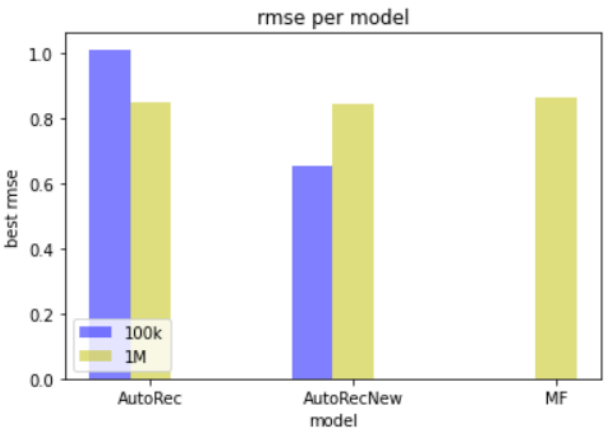
In other words, Uk-RMSE is RMSE computed for users with no more than k ratings. For example, interesting values of k for the MovieLens dataset are 10 and 20. Such values of ratings count are considered low w.r.t to the entire user-base.

Results

We compared our results to *AutoRec* and to the *matrix factorization* algorithm on the movieLens1M and movieLens100K datasets. We had to use datasets which provide some information on the users. Since movieLen100K is a subset of the 1M, this is probably not the best data set we can use. However, we had difficulties in finding data which is not too big, which provides good information on the users. In addition, movieLens100K has **on average**, less ratings per user compared to the movieLen100M, thus, suffers further from the **cold-start problem**.

We see that there is no significant difference between the different numbers of hidden units. We get the best results for 500 hidden units, but probably 100 are enough.

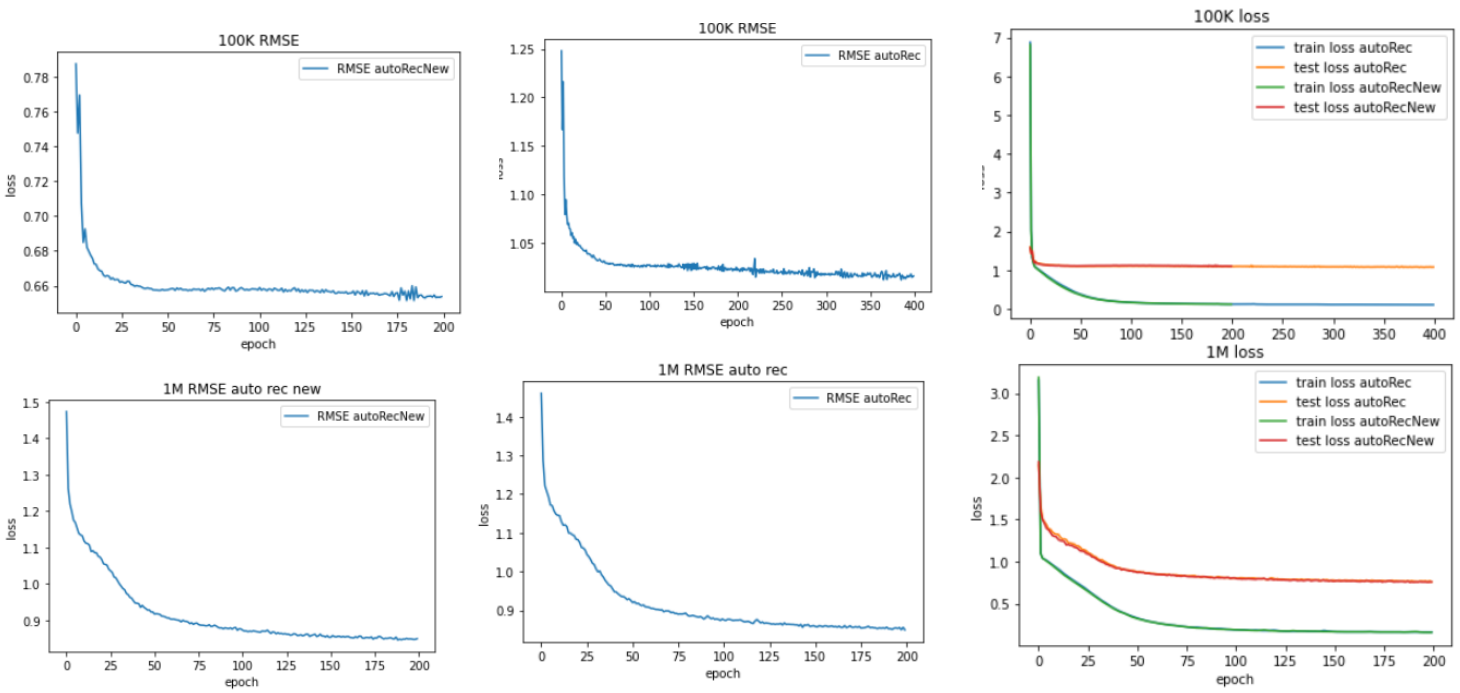




Auto rec rmse 100K	Auto rec new rmse 100K	Auto rec rmse 1M	Auto rec new rmse 1M	Number of hidden units
1.0118	**0.6514**	0.8496	**0.8457**	500
1.0268	0.6599	0.9138	0.9125	250
1.0268	0.6618	**0.9896**	0.9908	100

In order to find the best models we performed a hyper parameters search. In the graph above we can see the rmse value that we got for each model. On the **1M data set**, all models **give similar results**; our improvement gives the best results, however, it isn't significant. On the 100K data set, Our improvement gives much better results than in the original *AutoRec* model. As we hoped, it seems that the **improvement helps** when there are **less ratings for each user**. The fact that it performs better on the 100K dataset, is attributed to the user's features being more important when we have less ratings per user.

Auto rec new rmse 100K	Auto rec rmse 100K	MF rmse 1M	Auto rec rmse 1M	Auto rec new rmse 1M
1.0118	**0.6514**	0.8629	0.8496	0.8457



In the graphs above we can see the convergence of the models. The convergence seems similar but we can note that in the 100K dataset there is a steep start.
We can see that the improvement gave slightly better U30-RMSE and U20-RMSE. We hoped to see a larger difference. One of the reasons is probably the fact that we don't have much data on the users.

Auto rec	Auto rec new	
0.74771	**0.74560**	U30-RMSE1M
0.76075	**0.7564**	U20-RMSE1M

Conclusion

We tried to use some user's features to improve the auto rec algorithm. We thought that it would be hard to improve the results over the whole dataset, since it is very large and there are many items per user, while we don't have much and indicative information on the users. It seems that our improvement does have an impact on datasets with less ratings per user, such as movieLens100K. We didn't see significant improvement in the U20-RMSE and U30-RMSE. Probably the network can learn from all the other users, and predict good enough ratings for users with less ratings.

Further work

The fact that we saw an improvement in the results on movieLens100K, and the fact that we didn't see much improvement on the UK-RMSE metrics over movieLens1M worth further investigation.
Using our approach, we can try different methods. For example we can preprocess the user's features, and then concat the embeddings to the user's ratings.
We can try to combine this approach with deep auto encoder, and try different activations.