[**https://github.com/RaptorMai/Deep-AutoEncoder-Recommendation/blob/master/Report.pdf**](https://github.com/RaptorMai/Deep-AutoEncoder-Recommendation/blob/master/Report.pdf)

**Introduction**

Reduces data dimensions by encoding for a set of data and training network to ignore the noise in the data.

Autoencoder used in CF for rec systems-classic problem is guessing the missing rating in a matrixR (MxN) where R(i,j) is the ratings given by the ith user to the jth item

**Preprocessing**

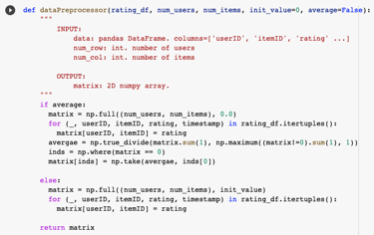
Train, Validation and Test Split

· 90%/10% train test split, 10% of training for validation

· Stratify with user\_id so then avoid bias issue,

o for example user A is only in training not in test, RMSE will be 0 for this user

o for example user B is only in test not in training then RMSE will be higher for this user



Transform Data Frame to Matrix

To apply Autorec dataset has to be transformed to matrixR (MxN) where R(i,j) is the ratings given by the ith user to the jth item

-init\_value is the default rating for unobserved ratings, when avg =True then unobserved rating will be set as average rating of the user

**AutoRec and Experiment**

AutoRec

User-based AutoRec-takes partially observed rating vector of a user, project it int a low dimensional latent space and then reconstruct back to the output space to redirect the missing rating

Loss Function

Minimize masked man squared error because it doesn’t make sense to redirect zero in the user rating vector

**Experiments**

Activations

Activations with non-zero negative part and unbounded positive part perform better for deep autoencoders for CF BUT results from Github show that they do not perform better than sigmoid+ linear baseline

Default Rating

Try different default ratings with missing values-best with avg or 0

For val RMSE-avg converges faster than 0 but much noiser

Unexpected Finding

ELU+ELU with default LD lambda = .001 test RMSE .877 (SELU+SELU =.878)

**Deep AutoRec and Experiments**

Deep Autoencoder Collaborative Filtering

Continue process with deeper learning

MMSE loss function

Tries activation function with non-zero negative art and unbounded positive parts works better

Uses dropout layers after the latent layer to prevent overfitting

Large dropout rate after the latent layer permits learning robust representations

****

Architecture

Smaller layer at beginning and end, largest layer in the middle (sbs-small big small)

Autoencoders tend to have BSB

[512,256,512] best 0.856

[256,512,256] second best 0.857

Adding more layers does not help, as we go deeper it is easier for overfitting and increasing reg parameters will bring test performance down->three hidden layers best option

Fewer parameters not only allow us t train mode with less data but also can mitigate overfitting

Gaussian Noise-doesn’t really help maybe because default has impact on performance thus by adding noise changes default rating?

Dropout Noise-dropout will mask out all elements randomly with a dropout (when dropout rate increases for the noise, RMSE started increasing)->cross validation needed for further verification

**Other experiments**

-every user is treated equally to update weights

-Assumption under this is that al the ratings from a user are generated from the same distribution but different people should have different distributions

-We cant have 1 autoencoder for every user but try one autoencoder for every group of users, assume users in each group rate movies similarly

-create user similarity matrix and cluster int different groups and train autoencoder for each group

**Conclusion & Future Work**

Conclusion

-Activations with non-zero negative part and unbounded positive part do not perform better than sigmoid+ linear baseline

-Avg vs 0 (default vs deep) patterns switched a bit convergence and noise

-ELU+ELU best model

- Gaussian Noise-doesn’t really help

**-Adding side info to user-based Autorec has limited effect**

Future Work

-dense re-feeding and dropout noise not fully implemented

-group by user (look at other experiments)

-Set up cross validation and more tuning