Matrix Factorization Techniques for Recommender Systems

Rwad Khatib, Michael Gudovsky, Erez Mahalu, Andrey Dodon
AFEKA – Tel Aviv Academic College of Engineering

1. Scope

This project will integrate the addition of external input to a known Collaborative Filtering recommender system (CF) in order to try to improve the system prediction accuracy. Some modifications were made in order to meet the limited timeline and computing power. Further, we will analyze the results and lay our conclusions and recommendation for further additions we think will have an impact on prediction accuracy.

2. Introduction

Collaborative filtering recommender systems (CF) methods produce user specific recommendations of items based on patterns of rating or usage without need for exogenous information about either items or users. Recommender systems rely on two types of inputs: *explicit* and *implicit.* Explicit feedback composed of direct data from users such as Netflix platform which collect stars ratings for movies, implicit feedback comprised of more abundant indirect data inferred from user's behavior such as purchase history, browsing history, etc., for example, user who watched many racing movies probably likes fast cars.
In order to establish recommendation CF system, need to relate two fundamentally different entities: items and users. There are two primary approaches to facilitate such a comparison, which constitute the two main techniques of CF: the neighborhood approach and latent factor models. Neighborhood models focus on relationships between items or, alternatively, between users. An item-item approach models the preference of a user to an item based on ratings of similar items by the same user.
Latent factor models, such as matrix factorization (aka, SVD), comprise an alternative approach by transforming both items and users to the same latent factor space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.
Latent factor models approach collaborative filtering with the holistic goal to uncover latent features that explain observed ratings.

CF suffers from cold start and data sparsity, when the CF starts working or there are missing data, the CF doesn't give a solution for those problems. Therefore, in such cases the recommendation errors might be extremely high.

Vader (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion.
it relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores.
The sentiment score of a text can be obtained by summing up the intensity of each word in the text.

SVD

A missing rating is predicted by the rule

$$\hat{r}_{ui}^{SVD} = \mu + b_i + b_u + q_i^T p_u$$

Where the base line predictor represented by (The average → $\boldsymbol{\mu}$ The overall average rating, **Item bias** → $\boldsymbol{b}_{\mathbf{i}}$ Observed Deviation of item rating form average μ, **User bias** → $\boldsymbol{b}_{\mathbf{u}}$ Observed Deviation of user rating form average μ). The inner product $\mathbf{q}_{\mathbf{i}}^{\mathbf{T}}\mathbf{p}_{\mathbf{u}}$ is the interaction between user u and item I where $\boldsymbol{q_i}$ and $\boldsymbol{p_u}$ measures the strength of the feature in the item and the strength of user's interest in those factors respectively. Thus, a rating is predicted by the rule

$$\hat{r}_{ui}^{SVD} = \mu + b_i + b_u + q_i^T p_u$$

In order to learn the model parameters ($\boldsymbol{b_i}$, $\boldsymbol{b_u}$, $\boldsymbol{q_i}$, $\mathbf{p_u}$) we minimize the regularized squared error:

$$\min_{b_*,q_*,p_*} \sum\nolimits_{(u,i)\in K} \left(r_{ui} - \mu - b_i - b_u - q_i^T p_u\right)^2 + \boldsymbol{\lambda}_1(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

The constant $\boldsymbol{\lambda}_1$, which controls the extent of regularization, is usually determined by cross-validation. Minimization is typically performed by either stochastic gradient decent (SGD) or alternating least squares (ALS). We use the SGD to update the parameters.
The algorithm loops through all ratings in the training data. For each given rating $\boldsymbol{r_{ui}}$, a prediction $\boldsymbol{\hat{r}_{ui}}$ is made, and the associated prediction error $\boldsymbol{e_{ui}} \overset{\text{def}}{=} \boldsymbol{r_{ui}} - \boldsymbol{\hat{r}_{ui}}$ is computed. For a given training case $\boldsymbol{r_{ui}}$, we modify the parameters by moving in the opposite direction of the gradient, yielding:

$$\mathbf{b_i} \leftarrow \mathbf{b_i} + \boldsymbol{\gamma}(\mathbf{e_{ui}} - \boldsymbol{\lambda}_1\mathbf{b_i}) \qquad \mathbf{b_u} \leftarrow \mathbf{b_u} + \boldsymbol{\gamma}(\mathbf{e_{ui}} - \boldsymbol{\lambda}_1\mathbf{b_u})$$
$$\mathbf{q_i} \leftarrow \mathbf{q_i} + \boldsymbol{\gamma}(\mathbf{e_{ui}}\mathbf{p_u} - \boldsymbol{\lambda}_1\mathbf{q_i}) \quad \mathbf{p_u} \leftarrow \mathbf{p_u} + \boldsymbol{\gamma}(\mathbf{e_{ui}}\mathbf{q_i} - \boldsymbol{\lambda}_1\mathbf{p_u})$$

When evaluating the Netflix data, we used the following hyper parameters, The learning rate ($\boldsymbol{\gamma} = \mathbf{0.005}$), the regularization ($\boldsymbol{\lambda_1} = \mathbf{0.02}$).

Latent factor models approach collaborative filtering with the holistic goal to uncover latent features that explain observed ratings.

CF suffers from cold start and data sparsity, when the CF starts working or there are missing data, the CF doesn't give a solution for those problems. Therefore, in such cases the recommendation errors might be extremely high.

3. Research Question

How does external explicit data affect Matrix Factorization model prediction accuracy?

4. Study Case

Due to the CF disadvantages, we would like to try improving the accuracy and answer the research question.
We choose to use IMDB reviews of 1150 movies, for each movie the DB include an average of thousands review by different users. Each review is an English free text. The first step is to perform adequate sentiment analysis, the IMDB DB is available in csv (comma separated vectors) file, after loading the DB into Date Frame using the python pandas library, NLTK library VADER model was used in order to get sentiment score which later we integrate in the SVD model that used by the article authors [1].
The code that was written searches and finds all the IMDB and Netflix prize common movies, then we apply VADER model on the list of common movies in order to get VADER Score for each IMDB Movie.
The Change in the original SVD model, we introduce list of the average VADER score for each movie and DELTA: hyper parameter that controls the percentage that the VADER Score affect the item bias in the initialization process, and ultimately it will be set using cross validation while analyzing the RMSE for each combination on the test set of the common movies.

5. Data Review

- IMDb Movie Reviews Dataset | IEEE: https://ieee-dataport.org/open-access/imdb-movie-reviews-dataset / dataset.zip
  The data from IMDB we obtained for our project contained a folder with 1150 csv files, each file is included thousands of reviews for each movie. The file name is in format: *Movie name_date.csv*

- The Netflix dataset was taken from Kaggle Netflix Prize data | Kaggle
  The data was provided with one csv file including movie index movie names and date, four txt files, divided by corresponding movie id, each containing customer id, ratings and the date when the movie was rated

6. Methodology

6.1 Flow chart, expectations, algorithm, motives

In order to evaluate the research question (that external explicit data can improve accuracy), first we perform necessary preprocessing procedure which described in Sect. ***6.2*** on the IMDB dataset in order to make it possible for sentiment analysis analyzing, then for each movie VADER sentiments score was calculated using VADER model in NLTK library. We elaborate on VADER in Sect.***6.4*** At our research of the MF (SVD) latent models we did not encounter models that introduced exogenous implicit data to the bias calculations, also the use of implicit data that was derived from Natural Language reviews may reflect the reviewer opinion more accurately, this is an innovative approach in latent models that worth to evaluate.
Vader . Next step was to calculate mean VADER Score for each movie, as result of these steps we got list of all the common movies with VADER sentiment score. The previous treatment of the IMDB data was to prepare the input of the SVD model Matrix Factorization code.

Driven by the assumption that external explicit input induces more variance and complement sparse data specially when tries to recommend items for new users, we looking to achieve slight improvement (we anticipate a small improvement because of small DS of common movies at one hand, and the bias modification was aimed to help with cold start).

The Algorithm begin with perform the above preprocessing and acquiring the Vader Score per the common movies, after we run the first iteration of the algorithm that will run the original Matrix Factorization code (property of the article writers and we elaborate on it in Sect. ***7.1***) which achieved by integrating Vader score with zero weight into our modified MF code and calculating RMSE (Root Mean Square Error). The next iterations will repeat run the modified MF code each time with different Vader score weight that produces modified $\mathbf{b_i}$ (the initial Item bias predictor), at the end of each iteration we will check if the RMSE is improved as a result of the new modified $\mathbf{b_i}$. If not then we execute another iteration with different Vader score weight, and we perform next iteration until we get an improvement in the RMSE, the Algorithm end when we either run all the predefined Vader score weights of when we get an accuracy improvement, the Algorithm is depicted in the following diagram:

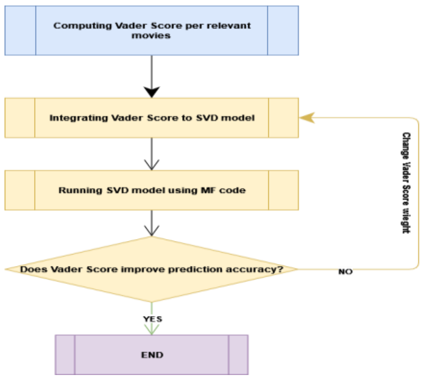

Computing Vader Score per relevant movies
Integrating Vader Score to SVD model
Running SVD model using MF code
Does Vader Score improve prediction accuracy?
NO
Change Vader Score wieght
YES
END

Figure 1: Flow chart

If we run the algorithm on predefined Vader score weights and did not achieve accuracy improvement, then we modify the hyper parameters of the MF model and run the Algorithm again.

6.2 Preprocessing

At first, we randomly choose one movie, we extracted from the files just the review column for analyzing the semantic of the text and acquiring the mean Vader compound score over all the reviews. Next step was finding the common movies we have from the Netflix dataset and IMDB dataset, we noticed that many of IMDB file names format included underscore between words while in the Netflix dataset the words are separated by space, we replaced all the underscore of all movies with spaces and removed the '.csv ' suffixes. In the Netflix data file consisting of all the movie names we concatenated the movie name with movie date so now the movie names correspond on both datasets. Next step is filtering all the movies existing in both datasets. For the Netflix data we have four text files containing the Netflix ratings for each movie, we concatenated those four files into one file containing all the rating data for all movies common in both datasets. Now for each movie we will work on we have the movie name, the Netflix rating and the IMDB review.

6.3 MF description + MF code

The Matrix Factorization model map both users and items to a joint latent factor space of dimensionality. Such that the user-item interactions are modeled as inner products in that space, high correspondence between item and user factors leads to a recommendation. The model is used in order to find two matrixes Q and P with smaller dimensions than the original matrix M. It characterizes both items and users by vectors of factors inferred from item rating patterns, which are taken from matrix M. The motivations are to reduce the calculation time, reduce the memory size that is needed and to get the best recommendation in case of data sparsity. This method is popular due to the good scalability and flexibility of the method.
The goal is to find latent factors Q &P and compose prediction matrix $\widehat{M} = Q * P$ that achieves Error ($E \overset{\text{def}}{=} M - \widehat{M}$) small as possible.
The approximation of $user_u$ to $item_i$ is calculated by the dot product, and denoted by $\hat{r}_{ui}$:

$$\hat{r}_{ui} = q_i^T p_u$$

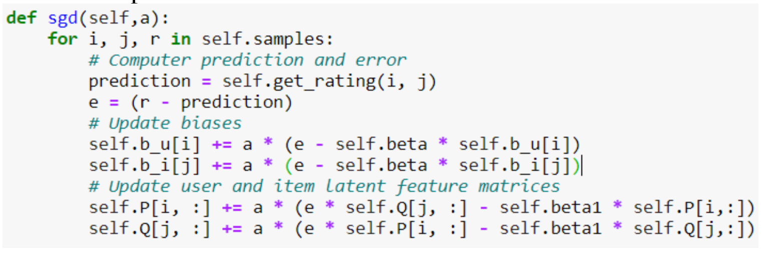
The model training based on SVD, starts when there is no data, therefore the values of Q and P are taken randomly. The model is trained during multiple iterations and improved in each iteration by using stochastic gradient descent, as described in the SVD section. In each iteration, the prediction error is calculated and used to update the Q and P.

$$Prediction\ error: e_{ui} = r_{ui} - \hat{r}_{ui}$$
$$q_i \leftarrow q_i + \gamma(e_{ui} p_u - \lambda q_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u)$$

$p_u - user\ u\ latent\ features$
$q_i - item\ i\ latent\ features$
$\gamma, \lambda - hiper\ parammeters$

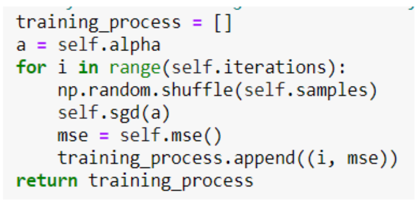
All the ratings in the original matrix are independent. In order to overcome the variance between different users/ items, biases are also calculated, and updated during the training process, as described is the SVD section.

- SGD implementation:



```
def sgd(self,a):
    for i, j, r in self.samples:
        # Computer prediction and error
        prediction = self.get_rating(i, j)
        e = (r - prediction)
        # Update biases
        self.b_u[i] += a * (e - self.beta * self.b_u[i])
        self.b_i[j] += a * (e - self.beta * self.b_i[j])
        # Update user and item latent feature matrices
        self.P[i, :] += a * (e * self.Q[j, :] - self.beta1 * self.P[i,:])
        self.Q[j, :] += a * (e * self.P[i, :] - self.beta1 * self.Q[j,:])
```

- MF training process implementation:



```
training_process = []
a = self.alpha
for i in range(self.iterations):
    np.random.shuffle(self.samples)
    self.sgd(a)
    mse = self.mse()
    training_process.append((i, mse))
return training_process
```

6.4 Innovation + Vader

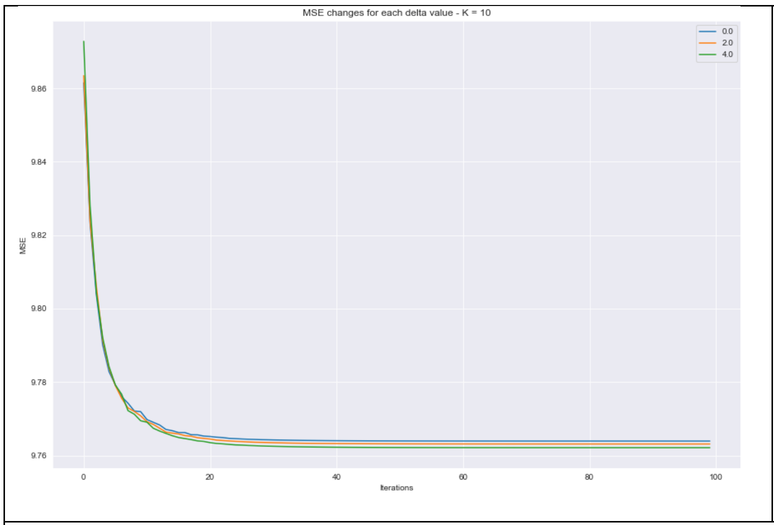
At our research of the MF (SVD) latent models we did not encounter models that introduced exogenous implicit data to the bias calculations, also the use of implicit data that was derived from Natural Language reviews may reflect the reviewer opinion more accurately, this is an innovative approach in latent models that worth to evaluate.
Vader (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. it relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.
For example- Words like 'love', 'enjoy', 'happy', 'like' all convey a positive sentiment. Also, VADER is intelligent enough to understand the basic context of these words, such as "did not love" as a negative statement. It also understands the emphasis of capitalization and punctuation, such as "ENJOY". The model also aggregates all the sentences in the document (or review in our case) to achieve an overall opinion, it's not trying to guess how many stars the reviewer awarded the item instead to which extent the review was positive or negative.
The compound score is calculated from $x = \frac{x}{\sqrt{x^2+\alpha}}$, where X sum of valence scores of constituent words and α is normalization constant (default value is 15)

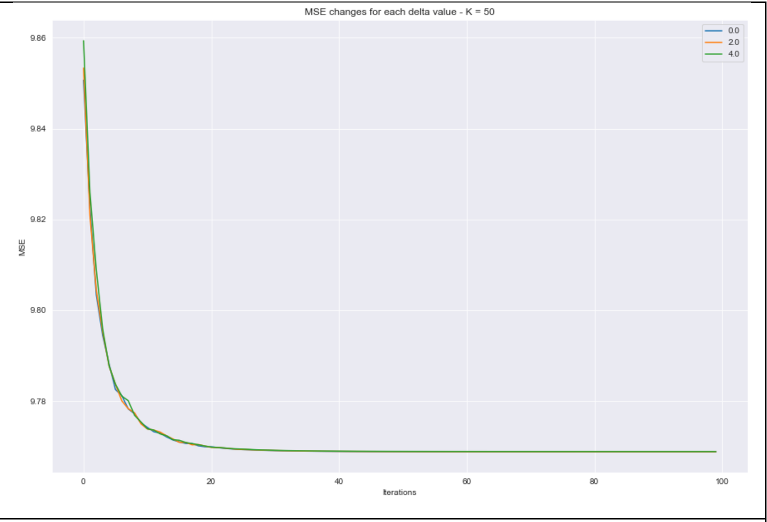
7. Comparison

7.1 Models

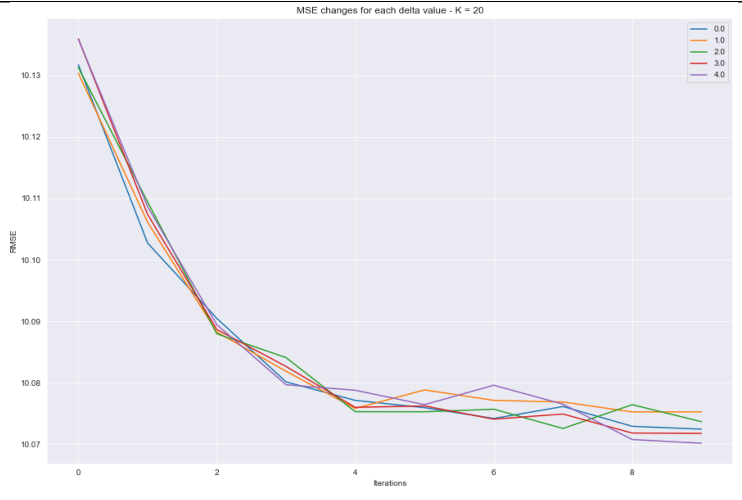
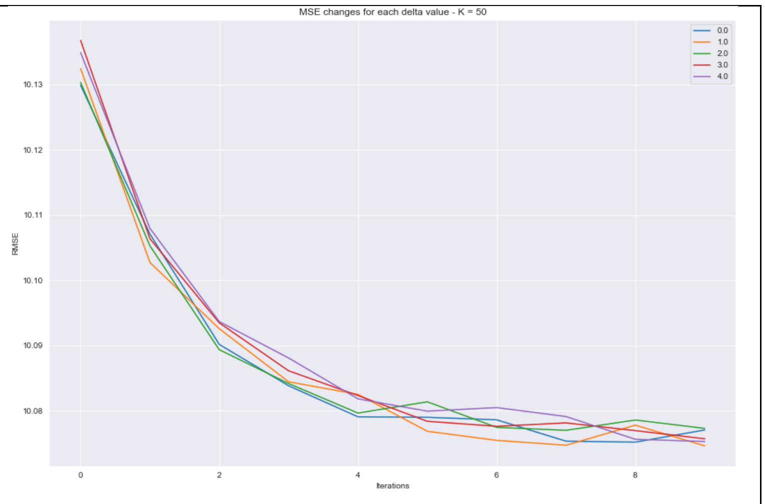
1. Link to Main Paper Suggested Model (SVD) [4]
2. Link to the improved Our addition to the model (github)

6.2 result comparison





When the number of latent features (K) is 10, we have much more effect on the RMSE than when we have K=50, our model doesn't affect at all and we see the same RMSE.
Both models ran over 100 iterations, and we can see improvement during all the iterations, and we see convergence after approximate 25 iterations.

	
When we ran 10 iterations, we can see the RMSE less smooth. In small number of iterations, we can't conclude that the Vader score addition improves the RMSE.	

Conclusions:

1. Due to high computation time, we were forced to cut a lot of data, we chose randomly from our data set and did all the computation on limited data of 50 movies with 200 reviews per movie.
2. We can see that indeed like was mentioned in the paper [1] SVD model cannot deal with sparse data, the data sparsity leads to high RMSE.
3. We see that our addition to the model affects the most when we have smaller number of features.
4. We assume that if we will use more data, movies, reviews and users, the Vader score addition will give a slight improvement.
5. As was mentioned in the papers [1],[2] to overcome data sparsity one can add explicit data of users. The temporal effects of user's behavior and item popularity can be considered to improve accuracy.

8. References

1. Main paper: Koren, Y., & Bell, R. (2015). Advances in collaborative filtering. In Recommender systems handbook (pp. 77-118). Springer, Boston, MA.
2. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
3. The Netflix dataset was taken from Kaggle Netflix Prize data | Kaggle
4. GitHub - harshraj11584/Paper-Implementation-Matrix-Factorization-Recommender-Systems-Netflix: [Python3.6] IEEE Paper "Matrix Factorization Techniques for Recommender Systems" by Koren,Bell,Volinsky/recommender.ipynb
5. Songhao Wu, Design your own Sentiment Score, toward data science web
6. .O.Isinkayea, Y.O.Folajimib, B.A.Ojokohc: Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal Volume 16, Issue 3, November 2015, Pages 261-273