The Netflix Challenge

Kaggle Team: Group 85

Celal Karakoç [redacted]

[redacted]

1 Introduction

Netflix, Spotify and other streaming services all use recommendation algorithms. These algorithms will tell you which movie or song you'd probably like. We developed a content based recommender system that uses ratings data to predict what rating a user would give to a movie. There are 2 main algorithms that we used in our system, namely Nearest Neighbour Collaborative Filtering and Latent Factors.

2 Methodology

2.1 Collaborative Filtering

Before the algorithm we load or calculate the similarity matrix. This calculation can be done with 2 different formulas.

$$Pearson(i,j) = \frac{\sum_{k \in I \cap J}(r_{ki} - \mu_i)(r_{kj} - \mu_j)}{\sqrt{\sum_u (r_{ui} - \mu_i)^2 \sum_u (r_{uj} - \mu_j)^2}} \quad (1) \qquad Cosine(x,y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||} \quad (2)$$

For global biases we make use of the global mean of every rating. ($\mu_{global} = 3.58$), the global mean of every rating by a male. ($\mu_{male} = 3.56$) and the global mean of every rating by a female. ($\mu_{female} = 3.61$). For local biases we use the bias for every user. ($\mu_{user} - \mu_{global}$), the bias for every movie. ($\mu_{movie} - \mu_{global}$) and the bias for every user, accounting the gender. ($\mu_{user} - \mu_{gender}$).

Cosine item-item	Cosine user-user	Pearson item-item	Pearson user-user
1.14565	1.08486	0.94104	1.01468

Table 1: RMSE of formulas

For our final ratings and subsequent results we have used the item-item Pearson similarity matrix, because this gave the best results (table 1).
In collaborative filtering we loop over all the predictions. For each prediction, we take the corresponding column of the similarity matrix. The array is then filtered. There are two filtering methods we could choose to decide which neighbours are the nearest.

1. **Top-N filtering**: A list of k nearest-neighbors is selected according to the similarity.
2. **Threshold filtering**: A list of all the neighbors is selected that has a similarity that is larger than a particular threshold.

Figures 1 and 2 in chapter 4 shows the optimization of both filtering methods. Through this we found that k=17 was the most optimal k in top-N filtering and the threshold 0.9 for threshold filtering. The high threshold of 0.9 is explainable, since you only want very similar users or movies. The k=17 for the top-N is also argumentative, because few nearest neighbour might result in overfitting and a lot might result in underfitting.

The final rating for our collaborative filtering model is calculated with:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}} \tag{3}$$

2.2 Latent Factors

For latent factors we first need a P and Q matrix, such that $R = P \cdot Q^T$.
We initialize both matrices, based on random values following a Gaussian distribution. Afterwards, we split our data in a training and validation set (80-20). Each iteration we calculated the error (4) with regularization (5).

$$err^2 = min_{qp} \sum_{x,y} (r_x i - (\mu + b_x + b_i + p \cdot q))^2 \tag{4}$$

$$+ \lambda \cdot (||q||^2 + ||p||^2 + b_x^2 + b_i^2) \tag{5}$$

Using the rules 6, 7, 8, 9, we update our biases and P/Q-Matrices respectively.

$$b_x = b_x + \alpha(err - \lambda\, b_x) \tag{6}$$

$$b_i = b_i + \alpha(err - \lambda\, b_i) \tag{7}$$

$$p_i = p_i + \alpha(err\, q_x - \lambda\, p_i) \tag{8}$$

$$q_x = q_x + \alpha(err\, p_i - \lambda\, q_x) \tag{9}$$

We also make use of our own modified adaptive learning rate (10, [1]), which ensures that our learning rate does not jump over the optimum, but rather becomes more and more precise through time. Furthermore, we noticed that a learning rate greater than 0.2 would cause an overflow. A single run of our adaptive learning rate over 50 iterations can be seen in figure 3. We found that the optimal set of factors and iterations, through trial-and-error to be both 50. Everything more would cost to much time to calculate without a significant increase in RMSE. The iterations and the learning rate also seem to have a correlation, which makes sense. A smaller α would need more iterations and vice versa. However, our adaptive learning rate also solves this problem. For the factors, it would depend on the size of your features of your dataset.

$$\alpha = \frac{1}{1 + decay \times iter_{max} \times momentum} \tag{10}$$

3 Conclusion

For our final rating we ensembled a top-N CF-model ($k = 17$), a threshold CF-model ($\theta = 0.9$), an LF-Model ($\alpha = 0.2$, $\lambda = 0.1$, $factors = 50$, $iter = 50$), an LF-Model ($\alpha = 0.05$, $\lambda = 0.01$, $factors = 500$, $iter = 10$) and an LF-Model ($\alpha = 0.05$, $\lambda = 0.05$, $factors = 100$, $iter = 25$) resulting in our score of 0.86857. The α was chosen based on the result of the adaptive learning rates, while the λ was through trial-and-error. The best-case of the top-N and threshold models were used. Furthermore, we noticed that, while factors and iterations mattered a lot at the start, eventually the difference between let's say 100 and 500 factors or 50 and 100 iterations would have very little effect on the RMSE (table 3).

4 Results

The Error we used to measure our results is the Root Mean Square Error.

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(i,x \in R)} (\hat{r}_{xi} - r_{xi})^2} \tag{11}$$

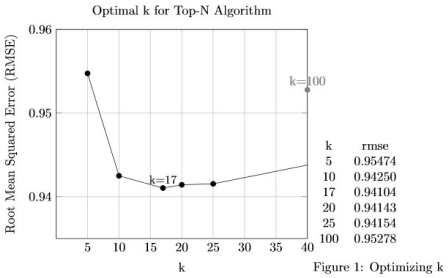

Figure 1: Optimizing k

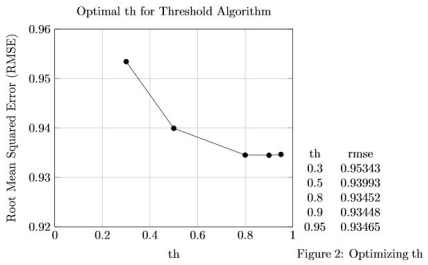

Figure 2: Optimizing th

Table 2: RMSE of different CF-models

Model	**RMSE**
User-User Cosine	1.08486
User-User Pearson	1.01468
Item-Item Cosine	1.14565
Item-Item Pearson	0.94104
Item-Item threshold (θ=0.9)	0.93448
Item-Item top-N (k=17)	0.94104
Top-N + gender biases	0.94104
Threshold + gender biases	0.93453

Table 3: RMSE of different LF-models

Model	**RMSE**
Latent Factors ($\alpha = 0.2$, $\lambda = 0.2$)	0.92001
Latent Factors ($\alpha = 0.05$, $\lambda = 0.1$)	0.88273
Latent Factors ($\alpha = 0.1$, $\lambda = 0.1$)	0.88173
Latent Factors ($\alpha = 0.2+$, $\lambda = x$)	overflow
Latent Factors ($\alpha = 0.2$, $\lambda = 0.1$)	0.87634
Latent Factors ($\alpha = 0.2$, $\lambda = 0.1$) + adaptive learning rate	0.87592
Ensemble Learning (2 CF-models, 3 LF-models)	0.86857

PDF-XChange Editor
Click to BUY NOW!
www.pdf-xchange.com

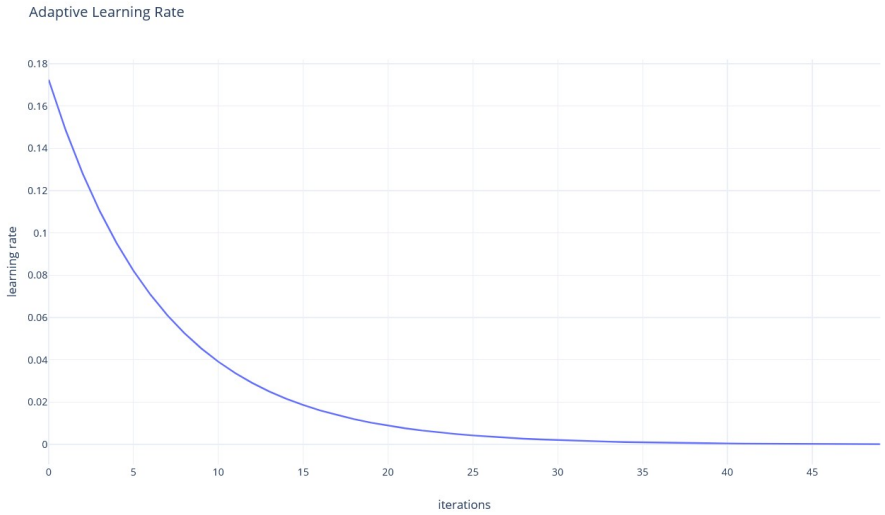

Figure 3: Adaptive learning rate over time

References

[1] Stanford. CS231n lecture. Convolutional neural networks for visual recognition. https://cs231n.github.io/neural-networks-3/#sgd.