IMPROVED R IMPLEMENTATION OF COLLABORATIVE FILTERING FOR RECOMMENDER SYSTEMS

In this work we present an R implementation of memory-based collaborative filtering which is:

- **SIGNIFICANTLY FASTER (EXECUTION TIME IS DECREASED BY A FACTOR OF 100X)**
- **APPLICABLE TO LARGE DATASETS ON WHICH CLASSIC IMPLEMENTATION MIGHT RUN OUT OF MEMORY.**

### WHAT ARE OUR GOALS?

Memory-based collaborative filtering (CF) is a common technique that makes recommendations by using information about similar users. However, to predict how user i will rate item j, we take into account both similar items.

Here are the user formulas for unsorted CF. In order to calculate predictions for the unsorted method, we need to find the average of the ratings of users similar to user i who have rated item j.

\[
\hat{r}_{ij} = \frac{1}{k} \sum_{u \in \text{neighbors}} r_{uij}
\]

Predictions are usually not calculated in a loop, but rather using matrix multiplication since it is much faster operation. Here is an example for predicting how user 123 will rate item 245:

\[
\begin{pmatrix}
123 & 245
\end{pmatrix}
\begin{pmatrix}
0.5 & 0.7
\end{pmatrix}
\]

By multiplying two matrices we get instant results for all predictions (not only for UGCF). Obviously, in case there is a lot of users or items, matrices might become large. Thus, the calculation of all predictions requires a lot of memory and processing time. Therefore, the main issues of Memory-based CF are related to:

- **EXECUTION TIME**
- **COMPUTATIONAL COMPLEXITY**

We address these issues using new approach and compare it to commonly used R packages. In recommendation:

### EXECUTION TIME IMPROVEMENT

The main issue is the implementation of user-based CF, as all the same approach applies for item-based CF:

1. **Take a ratings matrix and optimally normalize its ratings.**
2. **Calculate similarities between users.**
3. **Find users similar to the target user.**
4. **Calculate predictions and exponentiate them in case normalization was performed in the first step.**

### SPARSITY OPTIMIZATION

The characteristic of rating matrices is that they are very sparse (users typically rate only few items, if any). For calculating similarity, the key optimization was achieved by using functions optimized for sparse data, as well as using efficient hash techniques.

### FUTURE WORK

Another optimization we made was regarding filtering on a nearest neighbors. We grouped all the values from the similarity matrix by column and applied a function that finds the k highest values per column. This was implemented using R’s `apply` function.

### EVALUATION

The comparison of our implementation vs. ‘recommendations’ was performed using the following scenario:

- **Center normalization and cosine measure to calculate similarities.**
- **10-fold cross validation.**
- **The evaluation was performed on a single machine with 16 GB RAM.**

---

www.SmartCat.io