Distance Aware Cluster Recommender

Marcel Mohler, Kilian Risse, Adrian van Schie
Group: best_recommender_in_town
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—We introduce a novel recommender system called distance aware cluster recommender (DACR** [dey-ker]**), which, given a set of users' movie ratings predicts missing ratings. This allows to recommend movies that the user might also like.**
DACR uses known techniques like k-means and singular value decomposition and predicts values based on the neighbors of known ratings.
We provide a working implementation of this approach as well as a performance analysis of the achieved error. Furthermore, we show that an iterative version of DACR reduces the error up to 6.09% compared to two baseline algorithms.
This project has been done as part of ETH Zurich's Computational Intelligence Lab.

I. Introduction

A recommender system is a system that based on your prior ratings suggests other items that you may like. In particular, online media and e-commerce websites like Netflix and Amazon make heavy use of such systems to predict which movies or items the user wants to see next.
A simple approach to implement such a recommender system is to take the training data, fill unobserved values with a constant, cluster the data, and predict missing values by the average of the observed values in the cluster. This approach works reasonably well, but is unsatisfactory as the prediction depends on the constant with which we fill the unobserved values. As a first improvement, one can follow the outlined approach but replace the unobserved values by a weighted average of the neighborhood of each sample. This improves the result considerably. DACR generalizes this approach; instead of predicting the values by calculating the average of the observed values, DACR again takes a weighted average of the neighborhood of each sample in the cluster.
We present an overview of the models and algorithms in Section II and continue with a detailed performance analysis in section III. Section IV gives a short overview over some possible improvements and Section V concludes this paper.

II. Models and Methods

A. Terminology

This section gives a short overview of the terminology used throughout this work:

- *Ratings* are the values we are interested in. A rating is associated to a *person* and a *movie*. A rating may either be *observed*, the person rated the movie, or may be *unobserved*; the rating is unknown.
- A *feature* are all the ratings of a movie.
- A *sample* are all the ratings of a person. We will usually refer to a sample as a vector $y = (y_1, \dots, y_n)$.
- O_z denotes the set of indices corresponding to the observed ratings of sample z.
- The *distance* between sample x and y is denoted by $d_k^s(x, y)$. With $O_{xy} = O_x \cap O_y$, $d_k^s(x, y)$ is defined as

$$d_k^s(x,y) = \begin{cases} \infty, & \text{if } |O_{xy}| < s \\ \sum_{i \in O_{xy}} |x_i - y_i|^k, & \text{otherwise.} \end{cases}$$

  With a slight abuse of notation, we let $d_k^s(z, X)$ be the list of differences between z and all samples in the set X.
- The *neighborhood* of a sample $y \in Y$, denoted by $\Gamma_Y(y)$, contains all samples $x \in Y$ with $x \neq y$ and $d_k^s(y, x) < \infty$. Note that Γ also depends on s and k but the subscripts are omitted for readability.

B. Algorithms

This section gives a high level overview of DACR. The training data is denoted by Y in all algorithms.
In order to prevent any misunderstandings, what follows in Algorithm 1 is a definition of the basic algorithm described in the introduction.
Note that CLUSTER denotes an arbitrary clustering algorithm that clusters given data per sample. As mentioned before, the goal is to improve how Y_{fill} is created. In DACR this is done by replacing unobserved values by a

Algorithm 1 A very simple approach

```
1: procedure PREDICTOR_BASE($Y, c$)
2:     $Y_{fill}$ ← unobserved values are set to $c$
3:     $labels$ ← CLUSTER($Y_{fill}$)
4:     for all $l \in labels$ do
5:         $C$ ← samples with label $l$ from $Y$
6:         for all unobserved values $u$ in $C$ do
7:             $u$ ← avg of observed values in $C$
8:         copy $C$ into $Y$
9:     return $Y$
```

value derived in relation to the neighborhood Γ instead of replacing them by a constant. This is shown in the procedure FILL.DISTANCE_WEIGHTED. This procedure

```
1: procedure FILL.DISTANCE_WEIGHTED($Y, s, l$)
2:     $Y_{res} = Y$
3:     for $0 \le i < |Y|$ do
4:         $z \leftarrow Y[i]$
5:         if $|\Gamma_Y(z)| < l$ then
6:             continue
7:         $D \leftarrow d_4^s(z, \Gamma_Y(z))$
8:         $d_{sum} \leftarrow \sum_{d \in D} d$
9:         $D \leftarrow \frac{D}{d_{sum}}$
10:        for all $j \notin O_z$ do
11:            $Y_{res}[i][j] = \sum_{k=0}^{|\Gamma_Y(z)|} \Gamma_Y(z)[k][j] \cdot D[k]$
12:    return $Y_{res}$
```

will also be used to predict values in the cluster called FILL.CLUSTER_DIST, even though with different constants. A choice of constant is given in Section III-B. Further, if we do not have enough observed values in the cluster to predict a value, we use the value from Y_{fill}.
A vital observation is that each user has their own way of rating movies; e.g., users might have different opinions on what it takes for a movie to get the highest rating. In statistical terms this means that ratings of different users have different means and different deviations. In order to be able to compare the ratings to one another, DACR normalizes the observed ratings before the first usage; after normalization all samples have the same mean and deviation.
As a cluster algorithm DACR uses K-MEANS as described in the CIL lecture[1] with the extension of K-MEANS++ [2], which yields an efficient and effective way to choose the initial clustering centers.
DACR also includes an SVD decomposition along with a rank reduction in the end in order to reduce the error.
An algorithmic outline of DACR can be found in Algorithm II-B.
To provide a faster run time some components of the DACR implementation have been parallelized. This includes the FILL.DISTANCE_WEIGHTED procedure, which works on parts of the input in parallel. Furthermore, K-MEANS employs multiple executions simultaneously, which is the same as restarting K-MEANS to hope for better clusters. Additionally, the SVD rank reduction is started with different ranks. In the case of the simultaneous K-MEANS and SVD, DACR continues with the results that yield the smallest root mean square error (RMSE).
DACR allows many configurable parameters. In order to train our model, we first split Y into a training set and a test set. We then trained the model by running the algorithm with different choices of parameters and choosing the ones that returned the smallest validation error. Section III-B gives more insight into these parameters.
Lastly, DACR includes an extension called iterative distance aware cluster recommender (ITERDACR). Instead of one single execution of DACR as seen in Algorithm II-B, ITERDACR chains t iterations. t can be defined by the user and Section III-F presents the influence of this parameter. Also note that instead of calling FILL.DISTANCE_WEIGHTED on each iteration only gets called once in the beginning and then replaced by a simpler version.

Algorithm 2 Distance Aware Cluster Recommender

```
1: procedure DACR($Y$)
2:     $Y \leftarrow$ NORMALIZE(Y)
3:     $Y_{fill} \leftarrow$ FILL.DISTANCE_WEIGHTED($Y, 1, 4$)
4:     $labels \leftarrow$ CLUSTER($Y_{fill}$)
5:     for all $l \in labels$ do
6:         $C \leftarrow$ samples with label $l$
7:         $C \leftarrow$ FILL.DISTANCE_WEIGHTED($C, 1, 40$)
8:         for all ratings $c \in C$ not predicted do
9:             $c \leftarrow$ value from $Y_{fill}$
10:        copy $C$ into $Y$
11:    $Y \leftarrow$ SVD_REDUCE(Y)
12:    $Y \leftarrow$ DENORMALIZE(Y)
13:    return $Y$
```

C. Further Approaches

We tried several other ideas that did not end up in the final prototype. What started as a completely different approach was to use a low rank matrix factorization method using gradient descent for optimization [3]. In this approach two low rank matrices X and Θ with k columns are computed, such that the matrix product $X \cdot \Theta^T$ approximates the data Y reasonably well. More formally, the goal is to minimize the objective function $J = \frac{1}{2}\left\|Y - X \cdot \Theta^T\right\|^2 + \frac{\lambda}{2}\|X\|^2 + \frac{\lambda}{2}\|\Theta\|^2$. The second and third terms in the objective function are regularization terms to cope with over-fitting. Gradient descent is used to minimize the objective function. X and Θ are initialized with random values sampled from a uniform distribution over $[0, 1)$. The parameters in this approach are the rank k that determines the dimensions of the matrices X and Θ, λ in the objective function and the learning rate α in the gradient descent algorithm. The data has been split into training data (90%) and test data (10%).
We will provide a comparison in Section III.
Furthermore we attempted to combine these two approaches by using DACR for large clusters while sticking to gradient descent for sparsely filled clusters. However,

Table I
CHOICE OF CONSTANT PARAMETERS

Algorithm	parameter name	value
K-MEANS	number of clusters	18
K-MEANS	number of restarts	4
K-MEANS	local iterations	1
FILL.DISTANCE_WEIGHTED	s	1
FILL.DISTANCE_WEIGHTED	n	4
FILL.CLUSTER_DIST	s	3
FILL.CLUSTER_DIST	n	72

this did not reduce the error significantly but imposed a way longer run time, which is why we decided against it.

III. Results

A. Experiment Setup

All measurements were carried out on ETH Zurich's HPC Cluster called *Euler* [4]. Each measurement has been executed on 16 processor cores with 1024 MB RAM per core and repeated 5 times to provide statistical guarantees. Note that in order to provide reproducible results the error measurements do not include any statistical deviation (except in the case of gradient descent). However, timing results were annotated with $+/-$ one standard deviation. Reproducibility is also the reason why the amount of parallel working threads for the K-MEANS step have been reduced to one and a static randomness seed was used.

B. Learning the Parameters

DACR has been evaluated in two execution modes: non-iteratively and iteratively. We continue to call the first one DACR and the second one ITERDACR.
A series of experiments were run beforehand to obtain good choices of parameters which minimize the RMSE (see III-D). An overview of the choice of parameters can be found in Table I.

C. Baseline Algorithms

We compare DACR against two different algorithms which we call *baselines*.
Baseline one is a low rank matrix factorization method using gradient descent for optimization. For more information on the implementation see Section II-C. We will refer to this as BASELINE1 or GRADIENTDESCENT.
The second baseline algorithm performs a simple k-means on 8 clusters followed by a SVD rank-20 reduction. To initialize the data we simply use the average of the data. This approach is denoted as BASELINE2 or SIMPLE-K-MEANS.

Table II
RMSE, LOWER IS BETTER

Algorithm	Error	relative to BL1	rel. to BL2
BASELINE1	1.00990 ± 0.000072	-	-4.36%
BASELINE2	1.05592 ± 0	4.56%	-
DACR	0.99368 ± 0	-1.61%	-5.89%
ITERDACR5	0.99156 ± 0	-1.82%	-6.09%

D. Root Mean Square Error

One of the most important metrics for a recommendation system is the error. In particular we use the root mean square (RMSE) as an indication how close our predictions were from the known verification data. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{r}\sum_{i=1}^{r}(\hat{y}_i - y_i)^2}$$

where $\hat{y}_i$ denotes the predicted rating for the known rating y for position i. r is the total number of predicted ratings.
The error has been calculated with k-fold cross-validation by splitting the *kaggle* training data[5] into 8 equally sized sets. We measure the error of each set and return the average error.
Table II gives an overview of these measurements. For layout reasons BASELINE has been abbreviated to BL.
We see that BASELINE1 and BASELINE2 behave significantly worse than DACR. More interestingly, the iterative DACR with 5 iterations (denoted by ITERDACR5) is able to reduce the error even further at the minimum of 0.99156 (minus 0.21% compared to DACR). We will take a closer look at how the number of iterations influences the error in Section III-F.

E. Execution Time

While a low error is usually appreciated, it usually comes at a price. In particular, we are interested in how long it takes to make a prediction.
In contrast to the previous experiment, for these series of measurements we run all algorithms on the complete *kaggle* test set[5]. We then let the system predict the *kaggle sampleSubmission*[5] and measure the end-to-end running time. The results can be found in Table III.
We see that DACR takes about 7 times as long as SIMPLE-K-MEANS while achieving significantly better results. We take a more detailed look at ITERDACR in Section III-F.
In general, the gain in time is a lot higher than the decrease of the error. This shows that basic approaches like K-MEANS already provide surprisingly good results while decreasing the error further and further gets costly.

Table III
End-to-End execution time, lower is better

Algorithm	**time [s]**	**relative to bl1**	**rel. to bl2**
BASELINE1	302.4 ± 1.5	-	245.21%
BASELINE2	87.6 ± 0.9	-71.03%	-
DACR	626.2 ± 5.0	107.08%	614.84%
ITERDACR5	1001.8 ± 9.7	231.28%	1043.61%

F. *Iterations*

As mentioned previously, ITERDACR decreases the error by 0.21% compared to DACR. We repeated the error measurements from Section III-D with different values for t, the number of iterations.
The results are drawn in Figure III-F and show that up to $t = 5$ the error decreases. Once we hit $t = 6$ the additional iterations seem to over-fit the data and the result is a larger error.
On the other hand, the end-to-end execution time simply increases linearly by around 200 seconds per iteration as seen in Figure III-F.

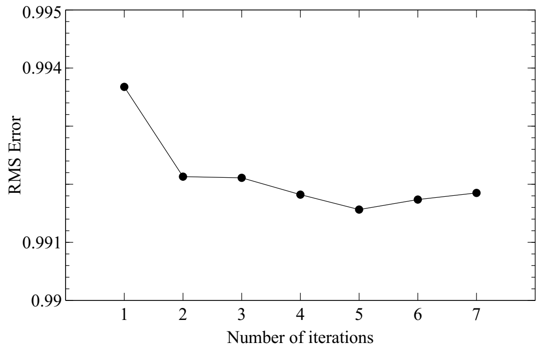

Figure 1. Number of iterations vs error

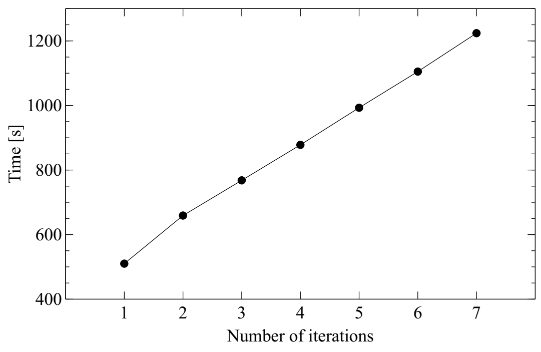

Figure 2. Number of iterations vs time

IV. Further Work

There are multiple areas on which the presented solution can be improved. As mentioned in Section II-C one could try to improve the predictions of the sparsely filled clusters with a different technique. Furthermore, while the prototype is meant to predict movie recommendation the system can simply be adapted to support any kind of ratings and should work reasonably well.

V. Summary

We presented a movie recommender system called DACR.
We provided a detailed description of the algorithm used and explained the reasons for its design.
Furthermore, we conducted an extensive performance analysis to show that DACR is a very efficient and fast way to come up with decent recommendations. A comparison with two baseline algorithms concluded that DACR performs significantly better in terms of error, while keeping similar run time constraints. ITERDACR yields even better results at a linear run time increase with the number of iterations.

References

[1] Prof. Thomas Hofmann, "CIL 2016 Lecture 04," http://www.da.inf.ethz.ch/teaching/2016/CIL/material/lecture/lecture04.pdf.

[2] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[3] Prof. Thomas Hofmann, "CIL 2016 Lecture 09," http://www.da.inf.ethz.ch/teaching/2016/CIL/material/lecture/lecture09.pdf.

[4] ETH Zurich, "Euler Cluster," https://www1.ethz.ch/id/services/list/comp_zentral/cluster/index_EN.

[5] Kaggle, "Data and Sample Submission," https://inclass.kaggle.com/c/cil-collab-filtering/data.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Distance Aware Cluster Recommender

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Mohler	Marcel
Risse	Kilian
Van Schie	Adrian

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 01.07.2016	[illegible]
	[illegible]
	Adrian van Schie

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.