Predicting the Learning Rate of Gradient Descent for Accelerating Matrix Factorization

C. S. B. Nóbrega, L. B. Marinho

Federal University of Campina Grande, Brazil
caionobrega@copin.ufcg.edu.br, lbmarinho@dsc.ufcg.edu.br

Abstract. Matrix Factorization (MF) has become the predominant technique in recommender systems. The model parameters are usually learned by means of numerical methods, such as gradient descent. The learning rate of gradient descent is typically set to lower values in order to ensure that the algorithm will not miss a local optimum. As a consequence, the algorithm may take several iterations to converge. Ideally, one wants to find the learning rate that will lead to a local optimum in the first iterations, but that is very difficult to achieve given the high complexity of the search space. Starting with an exploratory analysis on several recommender systems datasets, we observed that there is an overall linear relationship between the learning rate and the number of iterations needed until convergence. Another key observation is that this relationship holds across the different recommender datasets chosen. From this, we propose to use simple linear regression models for predicting, for an unknown dataset, a good learning rate to start with. The idea is to estimate a learning rate that will get us as close as possible to a local optimal in the first iteration, without overshooting it. We evaluate our approach on 8 real-world recommender datasets and compare it against the standard learning algorithm, that uses a fixed learning rate, and adaptive learning rate strategies from the literature. We show that, for some datasets, we can reduce the number of iterations up to 40% when compared to the standard approach.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

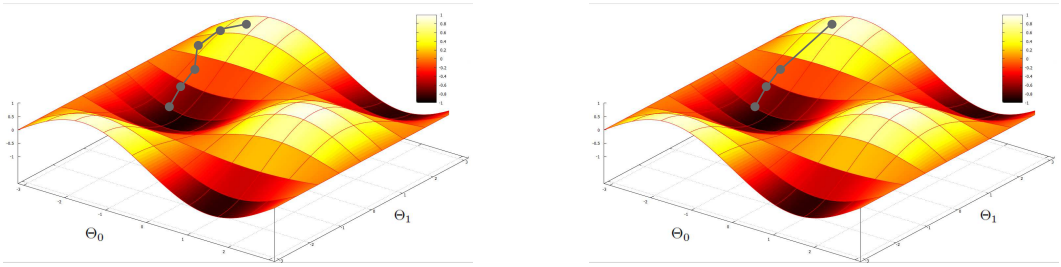
Keywords: Gradient Descent, Learning Rate, Matrix Factorization, Recommender Systems

1. INTRODUCTION

Matrix factorization is one of the most successful techniques in recommender systems nowadays [Koren et al. 2009]. MF features high accuracy, scalability, robustness against sparsity, and implementation ease, which explains its success and popularity. Moreover, the winning method of the Netflix challenge[1], a very important recommender systems challenge that provided major advances in the area, used MF as a key component. By now, there are many variations of MF [Paterek 2007] and [Gantner et al. 2010], each one handling a different aspect of recommender systems. In this article, we investigate MF for the classic problem of rating prediction, i.e., how to predict the ratings that users would give to items they still not accessed (e.g. movies still not watched).

The model parameters of MF are usually learned by means of numerical methods such as gradient descent, given that the loss function is non-convex. The learning rate of gradient descent is typically set to lower values (usually 0.01 or 0.02 [DeCoste 2006]) in order to ensure that the algorithm will not miss a local optimum and diverges. As a consequence, the algorithm may take several iterations to converge, which ends up increasing the computational costs of the training phase. [Rendle and Schmidt-Thieme 2008], for example, report 200 iterations until convergence in the Netflix dataset with the learning rate set to 0.01.

[1] http://www.netflixprize.com/



(a) Default Learning Rate (b) High Learning Rate for first iteration

Fig. 1: Illustration of gradient descent: (a) using a small learning rate over all iterations. (b) using a larger rate for the first iteration

Ideally, one wants to find the learning rate that leads to a local optimum in the first iterations, but that is very difficult to achieve given the high complexity of the search space. However, if in one shot one gets close enough to the local optimum, one will need only a few iterations until convergence, hence saving iterations and speeding up the learning process. Figure 1 illustrates an hypothetical execution of gradient descent for learning the parameters Θ_0 and Θ_1 under the loss function $J(\Theta_0, \Theta_1)$ and the idea of how to reduce the number of iterations (represented by the dots on the surface's function) with a good initial learning rate value.

In this article, we investigate the intrinsic relationship between the learning rate and the number of iterations of MF applied to recommender systems datasets. Our problem is to predict a good estimation of the initial value of the learning rate in order to minimize the number of iterations, leading to rapid convergence on yet unseen recommendation datasets. We observed that, for most of the recommender systems datasets we used (7 of 8), there is a positive linear relationship between the learning rate and the number of iterations needed until convergence. Surprisingly, this relationship holds across the different datasets, with only minor variations in the regression line. Assuming that MF presents similar learning behavior when applied to similar datasets, i.e., datasets that share the same entities (users and items), domain (recommender systems), and rating scale (5 star rating scale). From this observation, we propose to use simple linear regression models for predicting, for an unknown dataset, which learning rate would lead to the minimum number of iterations.

We conduct experiments in three real world datasets and show that we can reduce the number of iterations up to 40% when compared to the standard approach which uses a fixed learning rate value. Furthermore, we compare our approach to four learning rate adaptive techniques from the literature.

This article is organized as follows. In Section 2, we present the related work. In Section 3, we formalize the research problem this article investigates. In Section 4, we investigate the relationship between the learning rate and the number of iterations needed until convergence in the most popular recommender systems datasets available to the public. In Section 5 we present the results and conclusions of the experiments conducted. Finally, in Section 6, we conclude the article and discuss opportunities for future work.

2. RELATED WORK

In general, the value of the learning rate is determined by a search among candidate values. Using a validation set, a MF model for each candidate is learned and evaluated. The candidate that leads to the best model is chosen. This approach is expensive since it trains several models, especially when the search space is large. Moreover, this is a very common approach to find other hyperparameters, such as the ones related to regularization. In fact, [Rendle 2012] proposes an adaptive approach for speeding up the learning of the regularization hyperparameters. In this article we are only interested in the learning rate of gradient descent.

Gradient descent is a generic approach used as the subroutine of many learning and optimization algorithms, e.g., artificial neural networks (ANN) [Moreira and Fiesler 1995]. In fact, some of the most recent advances on gradient descent algorithms come from different research areas other than recommender systems. A common approach is to adapt the learning rate over the iterations of the gradient descent, i.e., algorithms dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative updates [Bartlett et al. 2007] and [Auer and Gentile 2000]. The method introduced in [Duchi et al. 2011], for example, starts with a default learning rate that is divided by the norm of the previous gradients in subsequent iterations. MF models, on the other hand, have a large number of parameters which makes it difficult to grasp the geometry of the search space.

Indeed, as learning rate adaptation has proven to be an effective solution to the problem of tuning the learning rate in the ANN field, [Luo et al. 2013] proposes different versions of gradient descent to learn the parameters of the MF. They adapted 3 techniques, namely, Deterministic Step Size Adaptation (DSSA), Incremental Delta Bar Delta (IDBD) and Stochastic Meta Descent (SMD) and compare them to his own method, Gradient Cosine Adaptation (GCA). In DSSA, IDBD and SMD methods there is an independent learning rate for each parameter of MF, and in CGA method, on the other hand, one for each rating. All these adaptive strategies update the learning rate based on the gradient of the previous iterations. In DSSA, if two successive iterations update the gradient in the same direction, then the learning rate is increased to make the learning process faster, and vice-versa. In IDBD, an exponential function of learning rate is applied and all past iterations are considered rather than the last two, like DSSA. In SMD, the update rule of the learning rate takes into account the gradient of related parameters, unlike DSSA and IDBD, which considers its respective parameters. Finally, the GCA strategy is similar to DSSA, but it uses the cosine of the gradients of two successive iterations. One drawback of the aforementioned techniques, and also the main difference to our approach, is that they consider several learning rates (one for each parameter), which leads to a memory overhead since this information needs to be cached during the learning process.

Except [Luo et al. 2013], none of the related works deal with MF or recommender systems. Moreover, the aforementioned works follow a bottom-up approach trying to adapt the learning rate based on previous values, while we follow a top-down approach, trying to predict, from the beginning, the learning rate that will lead to the minimum number of iterations.

Another line of research worth mentioning concerns distributed versions of gradient descent, as presented in [Zinkevich et al. 2010] for the domain of artificial neural networks, and in [Gemulla et al. 2011] for recommender systems.

3. PROBLEM SETTING

In this article, we tackle the problem of accelerating matrix factorization, estimating a good value to initialize the learning rate of gradient descent. The idea is to use a learning rate that gets as close as possible to a local optimum. As pointed out in Section 1, the learning rate is a low constant value determined by cross-validation (usually 0.01 or 0.02 [DeCoste 2006]). Before formalizing our problem, we briefly recall MF and gradient descent.

Let U denote the set of users and I the set of items, and S the sparse matrix, where rows represent users, columns represent items, and values represent the ratings that users gave to items. As the name suggests, MF attempts to decompose $S \in \mathbb{R}^{|U|\times|I|}$ into the product of the two lower rank matrices $P \in \mathbb{R}^{|U|\times k}$ and $Q \in \mathbb{R}^{|I|\times k}$, such that $S \approx PQ^T$. Here k denotes the number of latent factors used, i.e., the latent factors that explain the interactions between users and items. After the factorization, users and items are represented by vectors lying in the same space of k dimensions. Predictions are now done by simple dot products between user and items vectors, i.e., rows of P and columns of Q^T respectively. MF is cast as an optimization problem (cf. equation 1). For the rating prediction

problem the loss function err is the well known squared error. Since matrix S is sparse, the error is computed only w.r.t. to the observed values.

$$\underset{P,\ Q}{\operatorname{argmin}}\, err(P \times Q^T; S) \tag{1}$$

As mentioned before, the model parameters, here represented by the vector $\Theta = (P,\ Q)$, are usually learned by gradient descent. Gradient descent uses the partial derivative of the loss function w.r.t. to each parameter to guide the search for the local optimum. In each iteration gradient descent gives another step towards it, where the size of the step is given by the learning rate α. The equation 2 depicts, in high level, the update step of gradient descent.

$$\Theta = \Theta - \alpha \left(\frac{\partial}{\partial \Theta} err(P \times Q^T; S) \right) \tag{2}$$

In the following, we define the overall problem we investigate in this article. Let β denote the learning rate used in the first iteration of equation 2 and α the learning rate of the remaining iterations. Let $iter(\beta, \alpha, P, Q)$ be a function that returns the number of iterations needed to learn P and Q (by means of formula 1) with a given configuration of β and α. In this Article, we assume that α is fixed and focus on finding the β that minimizes the number of iterations until convergence. The basic idea is to start with a β that gets as close as possible to a local optimum and then use a fixed α for the remaining iterations until convergence. In all, our problem is formalized as follows:

$$\underset{\beta}{\operatorname{argmin}}\, iter(\beta, \alpha, P, Q) \tag{3}$$

4. LEARNING RATE PREDICTION

Ideally, we would like to find a learning rate capable of reaching a local optimum in the first iterations of the gradient descent. As we do not know what value yields this result, we do a search. In this search, we start with a given learning rate to be used only in the first iteration, here called β, and then, after the very first iteration, we set it back to a conservative (or default) value called α. What we want to do is to get as close as possible to a local optimum, without overshooting it. The rest of the way we do it as before, i.e., using small step sizes. Doing that, we expect to decrease the overall number of iterations needed until convergence.

To investigate the impact of different β values in the learning phase of MF, we first define a metric that indicates the percentage of iterations' reduction in comparison to the standard procedure using a fixed and small α. For example, given a dataset, assume that MF needs 20 iterations to achieve convergence in the validation set with the default learning rate α, and 12 iterations with a given β as initial learning rate. Now, the reduction is of 0.4 or 40%, if compared to the original number of iterations, and is calculated as $1 - (12/20)$.

For further investigating the relationship between β and the percentage of iterations' reduction, we varied the β values from 0.01 to 0.1 with increments of 0.005 and used the default $\alpha = 0.01$ to proceed until convergence. The choice for 0.01 is based on the literature, in which it was used in several experiments with different datasets as a conservative value. We have chosen this range, because we noticed that, for most of the datasets, when $\beta > 0.1$ the gradient descent diverges, which might indicate that we are overshooting a local optimum. In Figure 2 we plot the β values used (x axis) versus the percentage of iteration reduction (y axis) for all datasets.

In fact, for all datasets we have used in our experiments, we observed a positive linear relationship between the β and the percentage of iterations' reduction, i.e., the higher the β, the higher is the

Table I: Evaluation of linear models based on statistic R^2. Max. reduction indicates the greater percentage of iterations reduced when compared to the #iterations using default α (learning rate)

Dataset	R^2	**% max. reduction**	**#iter. with default** $\alpha = 0.01$
Movielens-100K	0.9801	38.1%	21
Movielens-1M	0.8184	39.2%	51
Movielens-10M	0.662	21.6%	37
Netflix	0.7525	0%	35
Yahoo-Movies	0.9694	41.6.5%	24
Epinions	0.9838	40.9%	22
Amazon	0.9779	43.7%	16
Dating	0.9514	36.8%	19

iterations reduction. To confirm this relationship, we fitted a simple linear regression model to each dataset and observed the goodness of fit (R^2 values). The R^2 varies between 0 and 1, indicating the percentage of the observations explained by the model. In this case, R^2 explains the proportion of the reduction of iterations that can be explained by the variation in β. The higher the R^2, the better the model fits the data. The R^2 values are close to 1, which indicates strong linear correlation (cf. Table I). In the last column of Table I we show the number of iterations needed by the standard MF using the default $\alpha = 0.01$.

The exception is the Netflix dataset that do not follow the same pattern of the other datasets. We do not have a concrete explanation for that yet, but one guess is that 0.01 already represents a good initial learning rate, and thus does not leave much room for improving upon.

Furthermore, the linear pattern remains when the number of latent factors varies as shown in Figure 3. The results with 50 and 100 latent factors have a slightly better percentage of reduction, but the error after convergence is also slightly higher. We show the results for one dataset only, but

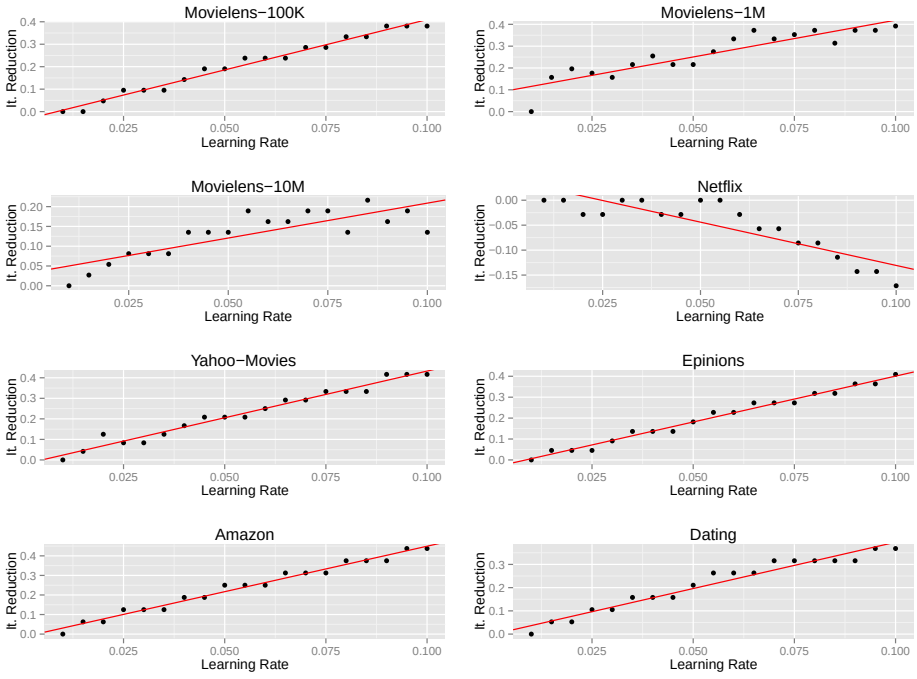


Fig. 2: Relationship between β and the percentage of iterations' reduction. The red lines show a linear model fitted to this data.

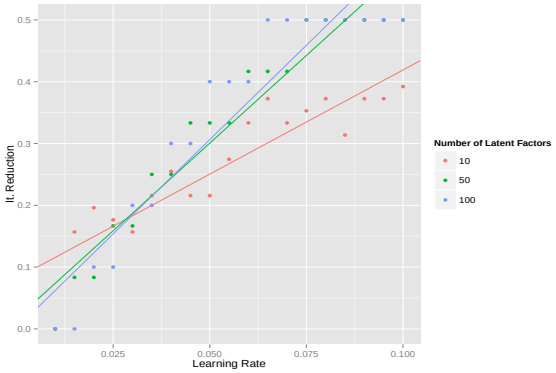


Fig. 3: Relationship between β and the percentage of iterations' reduction when the number of latent factors varies

this behavior holds for all datasets used.

5. EXPERIMENTS

In this Section, we describe how to use the prediction model for accelerating the learning phase of MF and compare this approach with other learning rate adaptive strategies introduced in [Luo et al. 2013].

5.1 Experimental Reproducibility

For the sake of experimental reproducibility, we have chosen well-known recommender systems datasets, all of them public and available online[2]. We employ 5-fold cross-validation and use the MF default setup values of the recommender system library MyMediaLite [Gantner et al. 2011]. The number of latent factors k, regularization term and stop criterion are 10, 0.015 and 0.001 respectively. We also implemented, on top of MyMediaLite, the learning rate adaptive strategies described in [Luo et al. 2013] with the following meta-parameter settings (and using the same original nomenclature): $\alpha = 0.0005$ and $\beta = 0.0005$ for DSSA, $\theta = 0.002$ for IDBD, $K = 0.9$ and $\theta = 0.005$ for SMD. With the exception of the Dating dataset, whose ratings are in the $[1, 10]$ range, the ratings of all the datasets are in the standard $[1, 5]$ range. Similarly to [Luo et al. 2013], we normalized the Dating dataset to the range $[1, 5]$ so that all the algorithms can be compared in a common ground. The normalization consisted in assigning ratings 1 and 2 to 1, 3 and 4 to 2, 5 and 6 to 3, 7 and 8 to 4 and 9 and 10 to 5. Table II presents the data characteristics of all the datasets used.

5.2 Learning Rate Prediction

As discussed in the previous Section, MF presents very similar learning behavior across different recommender datasets concerning the choices of β for the range $[0.01, 0.1]$ (cf. Figure 2). For most of the datasets, this behavior is highly explainable by a simple linear regression model. This suggests that we can use the linear model fitted to one dataset for making predictions for others, yet unseen, datasets. Given a new recommender systems dataset, the question we want to answer is this: "What learning rate should I use if I want to reduce the number of iterations in x% in relation to the standard algorithm?"

For evaluating our approach we devised the following experimental protocol. We have held one dataset for testing, and used the other datasets, individually, for prediction. We then compare the

[2]The Dating dataset is available at http://www.occamslab.com/petricek/data; the Yahoo-Movies dataset is available at http://webscope.sandbox.yahoo.com/catalog.php; the other datasets used are available at http://konect.uni-koblenz.de

Table II: Dataset characteristics: number of users, items, ratings (instances), rating range and domain in datasets.

Dataset	$\|U\|$	$\|I\|$	# rating instances	Original rating range	Domain
Movielens-100k	943	1,682	100,000	[1, 5]	Movies
Movielens-1M	6,040	3,706	1,000,209	[1, 5]	Movies
Movielens-10M	69,878	10,677	10,000,054	[1, 5]	Movies
Netflix	480,189	17,770	100,480,507	[1, 5]	Movies
Yahoo-Movies	7,642	11,916	221,364	[1, 5]	Movies
Epinions	40,163	139,738	664,824	[1, 5]	General Products
Amazon	2,146,276	1,231,018	5,744,088	[1, 5]	General Products
Dating	135,359	168,791	17,359,346	[1, 10]	Dating

number of iterations needed by MF using the standard gradient descent with the number of iterations achieved with the β predicted by each of the other datasets. We excluded the Netflix since it presents a negative linear correlation, as pointed out in Section 4.

We used the datasets Amazon, Movielens-10M and Dating as test datasets. The choice is supported by the fact that, although both are recommender systems datasets, they represent different domains (cf. Table II). For the test datasets, we try to predict the β that achieves 40% of reduction in the number of iterations, which is close to the empirical upper bound. Notice that the empirical upper bound of reduction for the Amazon is 47.3%, for MovieLens-10M is 21.6% and 36.8% for Dating.

Table III shows the results. For the Amazon dataset, the best predicting dataset was Movielens-10M. Surprisingly, this prediction led to a reduction of 62.5%, which is more than the empirical upper bound. It was an exception, since values above 0.1 led to divergence. For the Movielens-10M, the best predicting dataset was the Movielens-1M. In this case, the reduction was 21.6% which is far from 40%, but this is expected since the empirical upper bound suggests that 21.6% is already the best we can get. For the Dating dataset, the best predicting datasets were Movielens-100k and Epinions, again returning the empirical upper bound.

Figure 4 illustrates the reduction of iterations in Amazon when using the β predicted by Movielens-10M for a 40% input value. MF with the standard gradient descent reached convergence in 13 iterations, while it drops to 6 with our approach. It is interesting to notice that different local minimum are achieved, where our approach reaches an even better local minimum.

We can see that all the results are very similar. In fact, the confidence interval of the percentage of iteration's reduction for the Amazon dataset is $[36.2, 51.5]$, $[14.1, 19.1]$ for MovieLens, and $[26.9, 36.1]$ for the Dating dataset, with significance level of 0.05, which proves that all models provided similar good prediction. We can speculate that this relationship stems from features shared by the datasets e.g. all of these datasets have the same range of ratings $[1, 5]$. However, despite similar predictions, these datasets differ in the domain. This finding is interesting because this approach tends to happen regardless the domain considered.

5.3 Comparing to Learning Rate Adaptive Strategies

We compared our approach, Learning Rate Estimation (LRE), against four learning rate adaptive strategies from the literature, namely, DSSA, IDBD, SMD and GCA.

DSSA strategy assigns a specific learning rate to each parameter $p_{u,k} \in P$ and $q_{i,k} \in Q$, and updates each learning rate according to the signs of two successive updating directions of the respective parameter. The directions are the partial derivative of the loss function described in Section 3. The idea is to penalize the learning rate if the sign the successive directions are distinct, multiplying it by a value slightly lower than 1, which decreases the step size, thus making the learning of this parameter slower. On the other hand, if successive updates of gradients are made in the same direction, then the learning rate is multiplied by a value slightly larger than 1, thus increasing the step size and

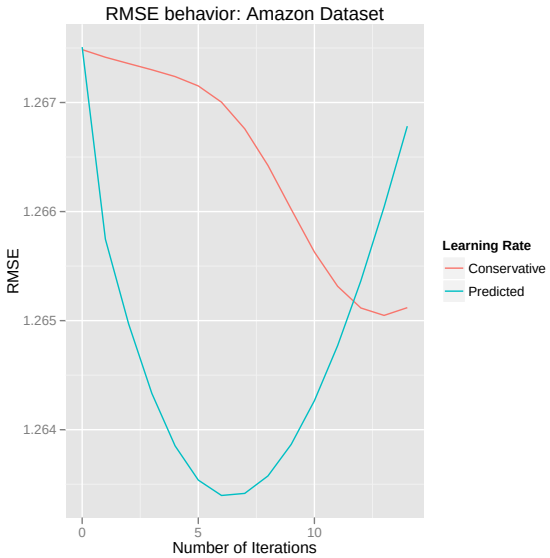


Fig. 4: RMSE (Root Mean Squared Error) relative to each iteration of gradient descent using the standard gradient descent versus our approach.

making the learning of this parameter faster. So, there is two other auxiliary matrices to keep the past gradients of each parameter to be compared with the current gradients. The meta-parameters mentioned in Section 5.1 controls how larger or lower than 1 the aforementioned penalty values will be.

In the IDBD strategy, there is a specific learning rate for each parameter. But all learning rates are in the exponential form, which leads to two advantages. First, it assures that the learning rate will always be positive. Second, it is a mechanism for making exponential steps, which is desirable because some learning rates must become very small while others remain large. The central idea of

Table III: Prediction learning rate for Amazon, MovieLens-10M and Dating datasets with linear models extracted from others datasets.

Dataset Test	***Dataset Train***	**predicted β**	**% iter. reduction**	**#iter. with predicted β**
Amazon	Ml-100K	0.097	43.7%	9
	Ml-1M	0.087	37.5%	10
	Ml-10M	0.159	62.5%	6
	Yahoo-Movies	0.091	37.5%	10
	Epinions	0.099	43.7%	9
	Dating	0.098	38.4%	6
Movielens-10M	Ml-100k	0.097	18.9%	30
	Ml-1M	0.087	21.6%	29
	Yahoo-Movies	0.091	16.2%	31
	Epinions	0.099	13.5%	32
	Amazon	0.088	16.2%	31
	Dating	0.098	13.5%	32
Dating	Ml-100k	0.097	36.8%	12
	Ml-1M	0.087	31.5%	13
	Ml-10M	0.159	21%	15
	Yahoo-Movies	0.091	31.5%	13
	Epinions	0.099	36.8%	12
	Amazon	0.088	31.5%	13

IDBD is to consider the effect of all past step-size values on the current parameters. This is made by the two auxiliary matrices that keep all the past gradients of each parameter.

Like DSSA and IDBD, in the SMD strategy there is a specific learning rate for each parameter. All learning rates are adjusted in log-space and are optimized over an exponentially decaying trace of gradients, similarly to the IDBD strategy. However, differently from IDBD where each learning rate takes into account the previous values, SMD takes into consideration the effects of one specified learning rate on the other related parameters (related parameters are those that share the same user or same item). In this case, there are also two auxiliary matrices to keep the gradients of each parameters and its related parameters. The meta-parameters mentioned in Section 5.1 denotes a discounting factor to penalize past iterations.

In the GCA strategy, there is a specific learning rate for each rating. The update of the learning rate is based on the cosine of the angle between the learning directions of two successive iterations. Note that this is similar to the DSSA strategy, but here, if the cosine is close to one (gradients pointing to the same direction) the learning step increases, while it decreases for lower values of the cosine (gradients pointing to different directions). The meta-parameters mentioned in Section 5.1 control the weight of the cosine on the update rule.

In the evaluation, we used our best predicted learning rate, estimated in the last Section, and set the adaptive strategies to the default learning rate (0.01), in order to make the results comparable against the traditional approach and LRE. Table IV shows the results. For almost datasets, our approach needed fewer iterations than the methods compared, keeping similar RMSE values, i.e., our approach reached convergence faster and kept equivalent accuracy. The exception is GCA in Movilens-10M dataset, that reaches the convergence with 16 iterations, while our reach with 29.

It is worth mentioning that the compared methods are more complex than ours in terms of space since they have to store additional matrices to keep the learning rates and the gradients, one for each parameter. While our approach has space complexity of $O((|U| + |I|) \times k + |T|)$, where $|T|$ is the number of ratings, the DSSA, IBDB and SMD approaches have space complexity in the order of $O((|U| + |I|) \times 3k + |T|)$. The GCA approach requires extra storage for maintaining each rating specified step-size values and the last gradient on the corresponding user/item feature vectors. So, GCA has space complexity of $O((|U| + |I|) \times k + |T| \times (3 + 2k))$.

6. CONCLUSIONS AND FUTURE WORK

In this work, we introduced an approach for predicting the learning rate of matrix factorization's gradient descent. The main idea is to use this prediction to get as close as possible possible to a local optimum in the first iteration.

Starting from an exploratory investigation on different recommender datasets, we observed that there is an overall linear relationship between the learning rate and the number of iterations needed until convergence. From this, we propose to use simple linear regression for predicting, for an unknown dataset, which learning rate leads to the minimum number of iterations. We have tested this hypothesis on 8 real-world recommender datasets from different domains, e.g. Movielens and Netflix recommend movies, while Epinions and Amazon recommend products. We show that, for some datasets, we can reduce the number of iterations up to 40% when compared to the standard approach.

Moreover, we compared our approach against four learning rate adaptive strategies from the literature and showed that our method outperforms almost all of them in all the evaluated datasets.

As future work, we plan to combine our approach with adaptive algorithms in order to further reduce the number of iterations needed until convergence. So, in addition to the initial value, we would have a dynamic learning rate capable to adapt itself during the model learning. Finally, we plan to investigate this behavior in another types of datasets, such as implicit feedback recommendation datasets.

Table IV: Compare our approach with learning rate adaptive strategies.

Dataset Test	*Approach*	**RMSE**	**#it**
Amazon	MF-LRE	1.2620	6
	MF-DSSA	1.2680	16
	MF-IDBD	1.2686	16
	MF-SMD	1.2679	16
	MF-GCA	1.2671	16
Movielens-10M	MF-LRE	0.8025	29
	MF-DSSA	0.7980	36
	MF-IDBD	0.7960	36
	MF-SMD	0.7981	41
	MF-GCA	0.8007	16
Dating	MF-LRE	0.8731	12
	MF-DSSA	0.873	19
	MF-IDBD	0.8728	19
	MF-SMD	0.8723	19
	MF-GCA	0.8758	13

REFERENCES

Auer, P. and Gentile, C. Adaptive and self-confident on-line learning algorithms. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. San Francisco, CA, USA, pp. 107–117, 2000.

Bartlett, P., Hazan, E., and Rakhlin, A. Adaptive online gradient descent. Tech. Rep. UCB/EECS-2007-82, 2007.

DeCoste, D. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA, pp. 249–256, 2006.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* vol. 12, pp. 2121–2159, July, 2011.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. MyMediaLite: A free recommender system library. In *5th ACM International Conference on Recommender Systems (RecSys 2011)*. Chicago, USA, 2011.

Gantner, Z., Rendle, S., and Schmidt-Thieme, L. Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*. New York, NY, USA, pp. 14–19, 2010.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA, pp. 69–77, 2011.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* 42 (8): 30–37, 2009.

Luo, X., Xia, Y., and Zhu, Q. Applying the learning rate adaptation to the matrix factorization based collaborative filtering. *Know.-Based Syst.* vol. 37, pp. 154–164, Jan., 2013.

Moreira, M. and Fiesler, E. Neural networks with adaptive learning rate and momentum terms. Idiap-RR Idiap-RR-04-1995, IDIAP, Martigny, Switzerland. 10, 1995.

Paterek, A. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup Workshop at the 13th ACM International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA, pp. 39–42, 2007.

Rendle, S. Learning recommender systems with adaptive regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*. New York, NY, USA, pp. 133–142, 2012.

Rendle, S. and Schmidt-Thieme, L. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*. New York, NY, USA, pp. 251–258, 2008.

Zinkevich, M., Weimer, M., Smola, A., and Li, L. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.). pp. 2595–2603, 2010.