Can the Analysis of SGD on Convex Functions Guide the Choice of the Stepsize in Matrix Factorization ?

Justin Samuel Deschenaux, Guillaume Follonier, Jun Han
EPFL, Switzerland

Abstract—**In this project, we study matrix factorization, which is a standard model used for recommender systems. This problem is non-convex so the main results derived in the course do not apply in theory. However, it is possible that the behaviour of SGD could be similar, under certain setups. Therefore, we want to see if the stepsize $\frac{R}{B\sqrt{T}}$ derived for Lipschitz convex functions over closed and convex sets yields better performance than a grid-search on the learning rates. We also compare different initializations. Our result shows that although two stepsizes can lead to similar performance at last, the best stepsize chosen by grid-search converges faster.**

I. Introduction

Matrix factorization is widely used in real-world applications like recommender systems. It is used to infer unknown data associated with pairs of elements, such as ratings of users on items. Although it was popularized in 2006 (Netflix prize), this algorithm still attracts researchers nowadays, for example [1], [2], [3], [4]. There are many papers doing experiments related to initialization for matrix factorization, for example, [5], [6]. [5] discusses multiple initialization procedures and how they affect the convergence of their least-squares based algorithm. [6] studies the initialization of non-negative matrix factorization (NMF) and uses classical NMF optimization algorithms without stochastic gradient descent (SGD). [7] reviews most algorithms, which can be used for matrix factorization, such as Alternating Direction Method of Multipliers (ADMM). All papers we find do not contain similar experiments to ours. We focus on SGD because many popular optimization methods are variants of this algorithm. Moreover, it is also one of the main and most thoroughly explored topics of the course.

In this project, we want to see whether choosing the stepsize inspired by the formula $\gamma = \frac{R}{B\sqrt{T}}$ is better than performing a grid-search. To this end, we first design experiments to empirically determine a bound on the norm of the gradients as well as the distance from the starting point to the local minimum we reach. Secondly, we know that initialization partly determines how fast the algorithm converges. Therefore, we also work with different starting positions.

II. Models and Methods

A. Matrix Factorization Model

Given a set of ratings $r_{i,u} \in \Omega$ for item i by user u, the goal is to infer the score of unknown pairs. The problem is challenging because one typically knows a tiny fraction of all possible entries. With such data, one tries to enforce the following inductive bias: products and users are characterized by a small set of unknown parameters representing their personal tastes, traits or attributes. If one is able to learn those latent factors, it is possible to make educated guesses. More formally, let $X \in \mathbb{R}^{D\times N}$ represent the incomplete data matrix we want to approximate where D is the number of items and N is the number of users. Moreover, define $W \in \mathbb{R}^{D\times K}$ as the product/item matrix and $Z \in \mathbb{R}^{N\times K}$ the user one, where $K \in \mathbb{N}$ denotes the number of latent factors (hyperparameter). We tune $K = 8$ by trial and error with the consideration of computational time. With larger K, the model has more expressive power but is very prone to over-fitting. Moreover, it takes more time to train. Since our data has ~ 20 features originally (movie genres), $K = 8$ seems a good choice. Finally, rows of W and Z correspond to feature vectors of items and users respectively. The problem to solve is as follows,

$$\min_{W,Z} \sum_{(i,u)\in\Omega} (X_{i,u} - (WZ^{\top})_{i,u})^2. \tag{1}$$

B. Applying SGD for Matrix Factorization

One can see that the problem is non-convex in general by observing that when W and Z are scalar, the function is already non-convex. Therefore the convergence analyses conducted during the lectures do not hold globally. However, we also know that it is possible to prove convergence along certain trajectories. While we are not aware of any strong analysis for general matrix factorization, our objective is to empirically tackle this question. Our intuition goes as follows: it might be the case that when optimizing over W and Z using SGD, we stay in a neighbourhood where the function is locally convex, hence choosing a stepsize based on analysis in the convex case could lead to good performance. Moreover, such behaviours could depend on the starting position of the algorithm, i. e. initialization of W and Z. Therefore, we also investigate in that direction.

Recall from the course that for a convex function f, supposing $||\mathbf{x}_0 - \mathbf{x}^*|| \leq R$ and $\mathbb{E}[\|\nabla f(x_t)\|^2] \leq B^2$ where $\mathbf{x}^*$ is the global minimum, we have convergence of SGD in $O(1/\varepsilon^2)$ steps with the stepsize $\gamma = \frac{R}{B\sqrt{T}}$. Since we cannot rely on theoretical properties, we are forced to estimate R and B empirically.

C. Initializations

We start with three choices from [5], [6]: *Random A-col*, *SVD-centroid initialization*, and *spherical k-means clustering*. In these cases, W is defined using the aforementioned procedures, while Z is found by performing least squares regression. The exact formula is

$$Z^\top := (W^\top W + \lambda I_K)^{-1} W^\top X, \tag{2}$$

where $\lambda = 0.01$ is a regularizer tackling ill-conditioned problems. We choose a small λ because other initializations we present are not regularized. X is the data matrix, where unknown item/user pairs are set to zero. *Random Acol* defines each column of W by selecting uniformly at random p columns (without replacement) from X and averaging them. *SVD-centroid initialization* generates W by computing the SVD decomposition of $X = U\Sigma V^\top$ and taking the top K eigenvectors of U before multiplying them by their associated eigenvalue. Finally, the third initialization solves a k-means clustering problem on the columns of X with K centroids as an initialization of W.

We also implement three other initializations based on statistics of the data. Let $\bar{r}_u$ be the average of user u's ratings, $\bar{r}_i$ the average rating of item i, and $\bar{r}$ the average over all known $r_{i,u}$. Our personal initializations are such that the predictions made with the original W and Z are equal to the aforementioned means. More formally, we define the parameters based on $\bar{r}_u$, $\bar{r}_i$ and $\bar{r}$ as follows. First, we set $Z_{u,k} = \frac{\bar{r}_u}{K}$ and $W_{i,k} = 1$. Similarly, working with the empirical item means $\bar{r}_i$ yields an initialization with $W_{i,k} = \frac{\bar{r}_i}{K}$ and $Z_{u,k} = 1$. Finally, by using $\bar{r}$, we initialize all elements of W and Z as $\sqrt{\frac{\bar{r}}{K}}$. Note that $\bar{r} \geq 0$, hence the square root is well defined. By the definition of matrix multiplication, we see that initially, the prediction of a user on a movie is equal to the averages $\bar{r}_u$, $\bar{r}_i$ or $\bar{r}$ respectively. We call these three initializations as *user mean*, *movie mean* and *global mean*. So far, we have presented 6 different ways to choose the original parameters. For completeness, we also consider the default used in PyTorch, namely, drawing each entry $W_{i,k}$ and $Z_{u,k}$ as i.i.d Gaussian white noise and we call this method as *normal* or iid $\mathcal{N}(0, 1)$.

III. Results

A. Preliminary Experiment for Initializations

To begin with, we train the model for 500 epochs with all seven initializations and choose the best learning rate among 1, 0.1, 0.01 and 0.001. We use a batch size of 256 samples and apply unregularized MSE loss on the rating prediction of user/item pairs of the batch. Moreover, we use SGD without momentum for all our experiments.

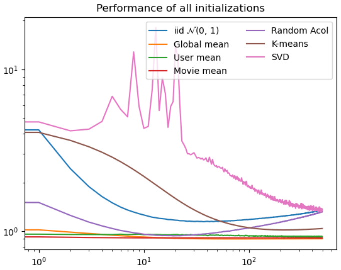

Fig. 1. Performance of all initializations optimized with SGD and the best learning rate by grid-search on values 1, 0.1, 0.01, 0.001.

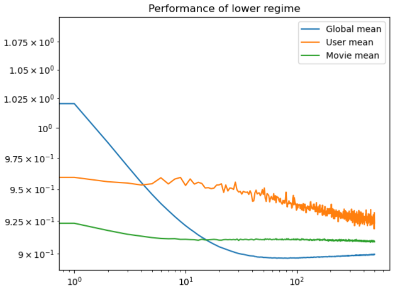

Fig. 2. Performance of three initializations achieving a lower objective, optimized with SGD and the best learning rate found by grid-search.

Table I summarizes the minimal loss for different values of γ. We see that the best result is attained with $\gamma = 0.1$ in most cases. For other cases, the loss with $\gamma = 0.1$ is very close to that with the best learning rate. We also observe that for $\gamma = 0.001$, some cases do not converge for 500 epochs, which is simply because this stepsize may be too small to make enough change towards the local minimum in a relatively short period. The resulting performance with the best respective learning rate is displayed in Fig. 1. To present the difference among *user mean*, *movie mean* and *global mean* more clearly, we show Fig. 2.

We observe that the behaviour is very different depending on the initialization. *Random Acol* and *normal* exhibits overfitting, which is expected since the model is not regularized. Our custom methods based on the first-order statistics of data achieve less error (lower regime). As shown in Fig. 2, the performance of *global mean* is the best. Three methods *global mean*, *user mean*, and *movie mean* have a very low loss even before starting to optimize with SGD. Despite of the appealing performance, choosing initializations depends on characteristics of the specific dataset. For this dataset, these three initializations are close to some local minima of our model. Especially for *movie mean*, the loss does not change much. Thus we may only draw a conclusion regarding which method is better with more data and thorough experiments.

B. Convex-case and Grid-search γ

We select three initializations, namely *global mean*, *normal*, and *random Acol*, which behave differently in the preliminary assessment. Each case is treated in the same way. As already mentioned, the convergence result with the stepsize $\frac{R}{B\sqrt{T}}$ does not hold anymore. In order to test the performance of this

Initialization	$\gamma = 1$	$\gamma = 0.1$	$\gamma = 0.01$	$\gamma = 0.001$
Normal	**1.146**	1.170	1.665	15.780
Global mean	0.905	**0.897**	**0.897**	0.956
User mean	0.925	**0.919**	0.921	0.933
Movie mean	0.990	0.922	**0.909**	0.910
Random Acol	**0.937**	0.938	1.079	3.798
Kmeans	1.020	**1.019**	1.178	3.082
SVD	Failed (NaN)	**1.312**	2.093	2.699

TABLE I
PERFORMANCE OF INITIALIZATIONS WITH DIFFERENT LEARNING RATES (LOWER IS BETTER, MSE LOSS).

stepsize, our first task is to find an estimate of R and B. To achieve this, we run 3 simulations with a fixed stepsize of 0.1 as it is the best value in the majority of cases for the previous experiment. We train for 500 epochs with a batch size of 256 and compute the norm of the gradients with respect to W and Z after each batch. We sum both values to obtain an estimate for the current iteration, which makes sense since we work with the Froebenius norm.

After all runs, we estimate $\hat{B}$ by the median of maximal norm over all training phases. Regarding $\hat{R}$, we simply calculate $\sqrt{\|W_0 - W_T\|^2 + \|Z_0 - Z_T\|^2}$ to obtain an estimate. Finally, we also pick $\hat{R}$ as the median over all simulations. The last value needed is the time horizon T, which is the number of SGD steps performed in Eq. 1. Since we work with minibatches (for computational efficiency), we update our parameters less often than with plain SGD. Hence picking $T = \#\text{samples} \cdot \#\text{epochs}$ would make the stepsize too small. Therefore, we choose $T = \#\text{batch} \cdot \#\text{epochs}$, which is the effective number of times we update our parameters. We stick with 500 epochs since all other experiments are done this way. We could also choose T by looking at the first time the loss goes below some threshold (early-stopping), but it may induce biases in the experiment and make comparisons uneven.

Moreover, the choice of performing 3 simulations for each initialization is for computational fairness. When performing grid-search, we train the model for 4 times with different learning rates. In the end, we obtain 4 tuned models and pick one with the best performance on the test set. When dealing with the "Lipschitz" γ, after estimating the gradient norm and the distance to local minimum for 3 times, and defining $\gamma = \frac{\hat{R}}{\hat{B}\sqrt{T}}$, we train one more time. Therefore, we also have 4 training phases, so that we use similar computational resources in both cases.

Fig. 3 shows the main result. It implies that with both learning rates, we reach similar performance on the lowest point of the curve. However, with the learning rate found by grid-search, we converge at least as fast as the other one in all three cases. The global mean plot has two similar curves because in both cases we have very close learning rates. This result suggests that looking for the learning rate by estimating B and R is not a better approach than grid-search. It is reasonable because our model is not convex and it is likely that two other conditions, i.e., $\mathbb{E}[\|\nabla f(x_t)\|^2] \leq B^2$ and $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$, do not hold globally. However, we believe that it strongly depends on the model and data under study. For example, when working with linear models, the procedure might work. Therefore, it would be interesting to see if such behaviour generalizes to other methods. Secondly, we have a very small performance gap between the choices of learning rates and it might not reflect in the actual predictions, e.g. a recommender system that would predict the top 5 items to recommend to a certain user might be the same in both cases.

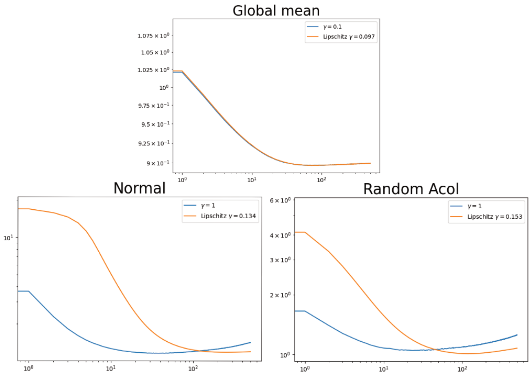

Fig. 3. Comparison of performance with grid-search (blue) and Lipschitz-guided γ (orange) for three initializations.

IV. SUMMARY

In this report, we show results of two main experiments. We compare the performance of different initializations. Although our method *global mean* achieves the best performance in our dataset, we believe that more experiments should be done to study behaviors of different initializations thoroughly. Then to answer our main question, we compare the stepsize $\gamma = \frac{R}{B\sqrt{T}}$ with that using grid-search. We find that with our fixed computational budget, the losses for two learning rates are similar. However, by using γ found by grid-search, the algorithm converges faster. It suggests that our intuition on the local convexity of the problem does not work for this dataset of matrix factorization and that we must look closely at the conditions of the theorem. If they do not hold, we cannot expect to obtain the same convergence rate or the best performance in general.

ACKNOWLEDGEMENT

We thank teaching assistants for useful advice of this project. We are also grateful for great lectures given by the professors this semester.

References

[1] D. Zhou, Y. Cao, and Q. Gu, "Accelerated factored gradient descent for low-rank matrix factorization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 4430–4440.

[2] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin, "Global optimality in low-rank matrix optimization," *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3614–3628, 2018.

[3] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.

[4] C. Jin, S. M. Kakade, and P. Netrapalli, "Provable efficient online matrix completion via non-convex stochastic gradient descent," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4527–4535.

[5] A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling, "Algorithms, initializations, and convergence for the nonnegative matrix factorization," *arXiv preprint arXiv:1407.7299*, 2014.

[6] S. Wild, "Seeding non-negative matrix factorizations with the spherical k-means clustering," 2003.

[7] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li, "A survey of matrix completion methods for recommendation systems," *Big Data Mining and Analytics*, vol. 1, no. 4, pp. 308–323, 2018.