Accelerating Matrix Factorization by Overparameterization

Pu Chen and Hung-Hsuan Chen
Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan
puchen.tw@gmail.com, hhchen@ncu.edu.tw

Keywords: Matrix Factorization, Collaborative Filtering, SVD, Recommender Systems, Overparameterization

Abstract: This paper studies overparameterization on the matrix factorization (MF) model. We confirm that overparameterization can significantly accelerate the optimization of MF with no change in the expressiveness of the learning model. Consequently, modern applications on recommendations based on MF or its variants can largely benefit from our discovery. Specifically, we theoretically derive that applying the vanilla stochastic gradient descent (SGD) on the overparameterized MF model is equivalent to employing gradient descent with momentum and adaptive learning rate on the standard MF model. We empirically compare the overparameterized MF model with the standard MF model based on various optimizers, including vanilla SGD, AdaGrad, Adadelta, RMSprop, and Adam, using several public datasets. The experimental results comply with our analysis – overparameterization converges faster. The overparameterization technique can be applied to various learning-based recommendation models, including deep learning-based recommendation models, e.g., SVD++, nonnegative matrix factorization (NMF), factorization machine (FM), NeuralCF, Wide&Deep, and DeepFM. Therefore, we suggest utilizing the overparameterization technique to accelerate the training speed for the learning-based recommendation models whenever possible, especially when the size of the training dataset is large.

1 INTRODUCTION

This paper studies overparameterization on matrix factorization (MF), which is arguably a fundamental technique in recommender systems. Overparameterization refers to the scenario that introduces extra learnable parameters to a model *without* increasing the expressiveness (i.e., the hypothesis space) of the model. Adding redundant parameters may seem to be a waste of computational power and space. However, this paper shows a counterintuitive result both theoretically and empirically – we can accelerate the optimization process of an MF model via overparameterization. Essentially, optimizing the overparameterized MF model by SGD is equivalent to introducing momentum and an adaptive learning rate on the original MF model, and this is probably the root cause of the acceleration. Since MF and its variants, especially those based on deep learning models, such as Wide&Deep (Cheng et al., 2016), NeuralCF (He et al., 2017), and DeepFM (Guo et al., 2017), are fundamental technologies of today's recommender systems, this discovery can be applied to a wide range of applications and scenarios.

Overparameterization may sound like overfitting,

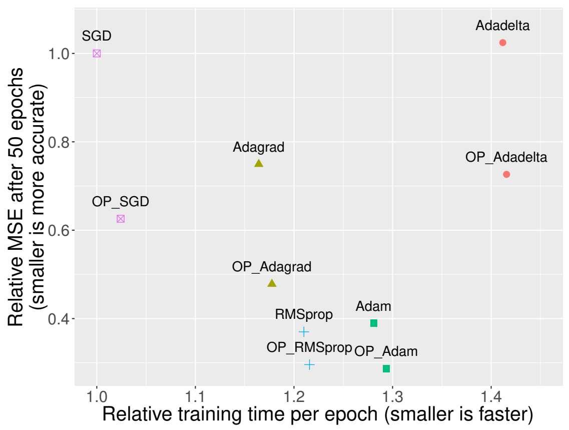

Figure 1: The relative training time and the mean squared error (MSE) scores of different optimization methods on the original MF model and the overparameterized MF model, based on the average of experimental results on several public datasets.

and indeed previous papers sometimes use the two terms interchangeably (Whittaker et al., 2010). However, in this paper, we treat the two terms differently. Specifically, the overparameterized MF model, albeit with more (redundant) parameters, has the same chance to overfit or underfit the training data (compared to the original MF model) because the overparameterized MF model has identical hypothesis space to the original MF model. Therefore, the improvement in the experiments comes from better optimization, not the overfitting or underfitting issues resulting from different hypothesis spaces of the MF and the overparameterized MF models.

Figure 1 summarizes our experimental results. We compared various optimization techniques in terms of their training time (measured by training time per epoch) and loss (measured via mean squared error score between the predicted and the actual ratings) on the original and the overparameterized MF model. We use the notation `OP_<X>` to denote applying optimizer `<X>` on the overparameterized MF model, and `<X>` to denote applying optimizer `<X>` on the original MF model. As seen, when we fix the optimizer, the overparameterized model converges much faster, with a slightly increased training time in each epoch. Note that all the reported values here are normalized by using `SGD` as the baseline. For example, `OP_SGD` is located at $(1.02, 0.63)$, meaning its mean squared error (MSE) score is 0.63 times `SGD`'s MSE score (i.e., much more accurate), and its training time is 1.02 times `SGD`'s training time (i.e., slightly slower). The numbers reported here are the average of multiple runs on multiple datasets. The details of the experimental settings and results are discussed in Section 4.

This paper makes the following contributions.

1. We propose an overparameterized matrix factorization model, which intentionally introduces redundant parameters to the original matrix factorization model. The new model has an identical hypothesis space to the original model.

2. We derive theoretically that applying the vanilla SGD on the overparameterized MF model is equivalent to optimizing the original MF model based on SGD with momentum and adaptive learning rates, and each learnable parameter has a different learning rate. It is surprising that overparameterization implicitly operates these carefully designed strategies aiming at accelerating optimization, especially for deep neural networks (Sutskever et al., 2013; Behera et al., 2006; Duchi et al., 2011).

3. We apply different optimizers on the overparameterized matrix factorization model and the original matrix factorization model. We conducted experiments based on public datasets. The experimental results show that the overparameterized model converges to a small loss with fewer epochs in all our experiments.

The rest of the paper is organized as follows. In Section 2, we review the related works regarding matrix factorization, optimization, overfitting and underfitting, and overparameterization. Section 3 introduces the proposed overparameterized matrix factorization model and derives its connections to the optimization strategy. Section 4 shows the experiments. Finally, we discuss the discoveries in Section 5.

2 RELATED WORK

Matrix factorization is arguably one of the most fundamental techniques in recommender systems. Matrix factorization became popular probably because of the Netflix Prize and Simon Funk (Piatetsky, 2007), and therefore sometimes it is called Simon Funk's SVD (Kurucz et al., 2007). Essentially, MF treats each user and each item as a vector, and a user's rating on an item is predicted by the inner product of the two vectors. MF highly influences the recommendation community and many of the following works, such as factorization machines (Rendle, 2010), SVD++ (Koren et al., 2009), nonnegative matrix factorization (Luo et al., 2014), NeuralCF (He et al., 2017), Wide & Deep model (Cheng et al., 2016), Prod2Vec (Grbovic et al., 2015), and Behavior2Vec (Chen, 2018). Instead of proposing another learning-based recommender system that takes user and item interaction as the input, this paper focuses on a more fundamental issue – better optimizing the MF model and, more generally, the learning-based recommendation model based on overparameterization. Therefore, the discoveries in this paper may help many of the learning-based recommendation models to obtain better parameters in fewer epochs.

One popular method for evaluating machine learning algorithms is to divide the labeled instances into training and test data, and the evaluation is conducted based on the test data. Many papers discussing recommender systems also adopt such a method. However, it has been shown that such a simple method may obtain biased training and test instances when evaluating recommender systems (Chen et al., 2017). For the evaluation metrics, one common method is to use the root mean squared score (RMSE) to compare the predicted and the observed rating (Piatetsky, 2007; Chen, 2017; Harper and Konstan, 2016). However, it has been found that a low RMSE score does not necessarily represent a good recommendation ranking (Steck, 2013). Therefore, ranking-based evaluation metrics, such as the discounted cumulative gain (DCG) and normalized discounted cumulative gain (nDCG), are probably better evaluation metrics (Järvelin and Kekäläinen, 2002). Despite the issues of biased training and test data and RMSE metric in recommender systems, this paper still uses these settings, because our goal is to accelerate the optimization process of the existing approaches rather than proposing another recommendation algorithm.

A supervised model learns the parameters to minimize the objective function, which is often defined as a weighted sum of the training loss and the generalization terms. If only the training loss is included in the objective function, it is likely to overfit the training data. Overfitting typically occurs when a learning model is too complex to overinterpret the relationship between the features and the target variable. An overcomplex model has a larger hypothesis space, so the model tends to "memorize" the training data, including the outliers, instead of discovering the patterns from the data. As a result, an overfitted model usually performs excellently on the training data but unsatisfactory on the unseen data. To prevent overfitting, not only the training loss but also model generalization needs to be considered, which can be achieved through various techniques, such as limiting the L1-norm or the L2-norm of the learnable parameters (Tibshirani, 1996), dropout (Srivastava et al., 2014), early stopping (Prechelt, 1998), and data augmentation (Wong et al., 2016). However, if a model is too simple such that it fails to capture the relationship between the features and the target variables, the model may have poor performance as well. This is often called underfitting. Recently, it was found that for recommender systems, the regularization weights should be applied "itemwise". Specifically, it is better to assign smaller regularization weights (i.e., lower constraints) to the latent factors associated with the frequent items (i.e., the items that reveal more information in the training data) (Chen and Chen, 2019).

One essential building block of the supervised learning models is the optimizer, i.e., the optimization algorithm. The simplest optimizer is probably stochastic gradient descent (SGD), which iteratively updates the values of the parameters based on the opposite direction of the gradients. To accelerate the optimization speed, especially for deep neural networks, many variations of SGD have been proposed. For example, momentum is included to consider not only the current gradient but also previous update speeds (Qian, 1999). The adaptive subgradient method (i.e., AdaGrad) adapts the learning rate of the parameters based on the appearance frequency of the parameters – the frequent parameters with smaller learning rates and the infrequent parameters with larger learning rates (Duchi et al., 2011). Adadelta further modifies AdaGrad by including only a limited number of previous gradients (Zeiler, 2012). RMSprop is another method that updates each parameter with adaptive learning rates to reduce the oscillation across the slopes of different dimensions (Graves, 2013). Adam integrates past gradients (with exponential decay) and squares of past gradients (with exponential decay) (Kingma and Ba, 2014). These optimizers accelerate the training speed, especially for deep neural networks. We compared many of these optimizers on the original and the overparameterized MF in this paper.

It is commonly believed that a model with too many learnable parameters tends to overfit the training data, so the success of deep learning probably originates in the rich expressiveness of the model and a large quantity of available training data. However, recent studies found that increasing parameters may not only increase the expressiveness of the model but also accelerate the optimization under certain conditions (Arora et al., 2018; Du et al., 2019). Our paper studies the effect of overparameterization on the matrix factorization problem without increasing the hypothesis space. We also found that overparameterization accelerates the optimization for the MF method. The closest work to our study is probably the weighted-SVD (WSVD) model (Chen, 2017), which also overparameterizes the matrix factorization model. However, the WSVD model was derived from a different perspective and did not provide a formal analysis of overparameterization and its connection to optimizers and optimization speed.

3 METHODOLOGY

3.1 Preliminary

The matrix factorization model states that r_{ij}, a user i's rating score on item j, is influenced by the interaction of user i's latent factor vector $\boldsymbol{p}_i$ and the item's latent factor vector $\boldsymbol{q}_j$. The biased matrix factorization model, also known as the Simon Funk SVD model, incorporates the concept of latent factors along with user biases and item biases. As a result, the biased matrix factorization model predicts a user i's rating score $\hat{r}_{ij}$ on an item j based on Equation 1. In the following, we use matrix factorization to denote biased matrix factorization.

$$\hat{r}_{ij} = \mu + b_i + c_j + \boldsymbol{p}_i \cdot \boldsymbol{q}_j, \tag{1}$$

where μ is the average of all known ratings, b_i is the bias of user i, i.e., user i tends to overrate or underrate an item (compared to the average), c_j is the bias of item j, i.e., item j tends to receive a lower or higher score (compared to the average), and $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ are the vectors of the latent factors of user i and item j, respectively.

The loss function is usually defined as the weighted sum of two terms – (1) the sum of squared score between the known ratings and the predicted ratings and (2) the L2-norm of the learned parameters. Equation 2 shows the loss function.

$$\mathcal{L} = \frac{1}{2} \sum_{\forall(i,j)\in\boldsymbol{\kappa}} (\hat{r}_{ij} - r_{ij})^2 + \frac{\lambda}{2} \|\Theta\|_2^2, \qquad (2)$$

where $\boldsymbol{\kappa}$ is the set of all given ratings from user i to item j, λ is a hyperparameter to decide the relative importance between the training loss and regularization power, and $\boldsymbol{\Theta}$ is the set of all the parameters to learn, i.e., all the b_is, c_js, and all the entries in $\boldsymbol{p}_i$s and $\boldsymbol{q}_j$s.

The learning process finds the parameters to minimize the objective function (Equation 2). Various optimizers can be used in practice.

3.2 Overparameterized MF

Here we propose an overparameterized matrix factorization model, which has the same hypothesis space as the MF model shown in Equation 1. However, we demonstrate later that our new model converges faster than the original MF model, both theoretically and empirically.

The predicting equation of the overparameterized MF is shown in Equation 3.

$$\hat{r}_{ij} = \mu + b_i + c_j + (\boldsymbol{w}_1 \odot \tilde{\boldsymbol{p}}_i) \cdot (\boldsymbol{w}_2 \odot \tilde{\boldsymbol{q}}_j), \qquad (3)$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are vectors whose lengths equal the length of $\tilde{\boldsymbol{p}}_i$ and the length of $\tilde{\boldsymbol{q}}_j$, and $\odot$ represents the Hadamard product (i.e., elementwise product).

Equation 3 is an overparameterized version of the original matrix factorization model (Equation 1). In other words, although the new model appears to have extra learnable parameters $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, the hypothesis spaces of the original MF model and the overparameterized MF model are identical. Specifically, for every $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ in Equation 1, we can find $\boldsymbol{w}_1$, $\tilde{\boldsymbol{p}}_i$, $\boldsymbol{w}_2$, and $\tilde{\boldsymbol{q}}_j$ in Equation 3 such that $\boldsymbol{p}_i = \boldsymbol{w}_1 \odot \tilde{\boldsymbol{p}}_i$ and $\boldsymbol{q}_j = \boldsymbol{w}_2 \odot \tilde{\boldsymbol{q}}_j$ (e.g., by setting $\boldsymbol{w}_1 = \boldsymbol{w}_2 = [1,...,1]^T$, $\boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i$, and $\boldsymbol{q}_j = \tilde{\boldsymbol{q}}_j$). Likewise, for every $\boldsymbol{w}_1$, $\tilde{\boldsymbol{p}}_i$, $\boldsymbol{w}_2$, and $\tilde{\boldsymbol{q}}_j$ in Equation 3, we can find the corresponding $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ in Equation 1 by merely setting $\boldsymbol{p}_i$ as $\boldsymbol{w}_1 \odot \tilde{\boldsymbol{p}}_i$ and $\boldsymbol{q}_j$ as $\boldsymbol{w}_2 \odot \tilde{\boldsymbol{q}}_j$. Therefore, the hypothesis spaces of the original MF model and of our proposed overparameterized MF model are identical.

3.3 Overparameterization accelerates matrix factorization training

Since the new model does not increase the expressiveness of the MF model, if the new model converges faster, the improvement is likely to be the result of better optimization. Below, we theoretically show that optimizing the overparameterized MF model by the vanilla SGD implicitly implies optimizing the original MF model based on SGD with the momentum and adaptive learning rate on different parameters. The proof follows the discussion in (Arora et al., 2018) for the linear regression model with ℓ_p loss.

Below, we use superscript (t) to denote the value of a variable at the t^{th} epoch. Let $\ell_{ij}^{(t)}$ denote the difference between the predicted rating and the real rating from user i to item j at epoch t (i.e., $\ell_{ij}^{(t)} := (\hat{r}_{ij}^{(t)} - r_{ij})$), and for simplicity, we ignore the generalization terms in Equation 2 (i.e., set λ to zero). If we apply the overparameterized MF model, the partial derivatives of the loss function to the variables $\boldsymbol{w}_1$ and to the variable $\tilde{\boldsymbol{p}}_i$ are shown in Equation 4 and Equation 5, respectively.

$$\nabla_{\boldsymbol{w}_1} = \ell_{ij} (\boldsymbol{w}_2 \odot \tilde{\boldsymbol{p}}_i \odot \tilde{\boldsymbol{q}}_j) \qquad (4)$$

$$\nabla_{\tilde{\boldsymbol{p}}_i} = \ell_{ij} (\boldsymbol{w}_1 \odot \boldsymbol{w}_2 \odot \tilde{\boldsymbol{q}}_j) \qquad (5)$$

The values of $\boldsymbol{p}_i$ in Equation 1 can be reconstructed by $\boldsymbol{w}_1 \odot \tilde{\boldsymbol{p}}_i$ in Equation 3. We use vanilla SGD to update $\boldsymbol{w}_1$ and $\tilde{\boldsymbol{p}}_i$, as shown below.

$$\begin{aligned}
\boldsymbol{p}_i^{(t+1)} &= \boldsymbol{w}_1^{(t+1)} \odot \tilde{\boldsymbol{p}}_i^{(t+1)} \\
&= \left(\boldsymbol{w}_1^{(t)} - \eta \nabla_{\boldsymbol{w}_1^{(t)}}\right) \odot \left(\tilde{\boldsymbol{p}}_i^{(t)} - \eta \nabla_{\tilde{\boldsymbol{p}}_i^{(t)}}\right) \\
&= \boldsymbol{w}_1^{(t)} \odot \tilde{\boldsymbol{p}}_i^{(t)} \\
&\quad - \eta \left(\boldsymbol{w}_1^{(t)} \odot \nabla_{\tilde{\boldsymbol{p}}_i^{(t)}} + \tilde{\boldsymbol{p}}_i^{(t)} \odot \nabla_{\boldsymbol{w}_1^{(t)}}\right) + O(\eta^2) \\
&\approx \boldsymbol{w}_1^{(t)} \odot \tilde{\boldsymbol{p}}_i^{(t)} - \eta \boldsymbol{w}_1^{(t)} \odot \ell_{ij}^{(t)} \left(\boldsymbol{w}_1^{(t)} \odot \boldsymbol{w}_2^{(t)} \odot \tilde{\boldsymbol{q}}_j^{(t)}\right) \\
&\quad - \eta \tilde{\boldsymbol{p}}_i^{(t)} \odot \ell_{ij}^{(t)} \left(\boldsymbol{w}_2 \odot \tilde{\boldsymbol{p}}_i^{(t)} \odot \tilde{\boldsymbol{q}}_j^{(t)}\right) \\
&= \boldsymbol{p}_i^{(t)} - \boldsymbol{\beta}^{(t)} \odot \nabla_{\boldsymbol{p}_i^{(t)}} - \boldsymbol{\gamma}^{(t)} \odot \boldsymbol{p}_i^{(t)},
\end{aligned} \qquad (6)$$

where $\boldsymbol{\beta}^{(t)} := \eta \boldsymbol{w}_1^{(t)} \odot \boldsymbol{w}_1^{(t)}$, $\boldsymbol{\gamma}^{(t)} := \eta \ell_{ij}^{(t)} \left(\boldsymbol{w}_2^{(t)} \oslash \boldsymbol{w}_1^{(t)}\right) \odot \tilde{\boldsymbol{p}}_i^{(t)} \odot \tilde{\boldsymbol{q}}_j^{(t)}$ ($\oslash$ denotes the Hadamard division, i.e., elementwise division), and η is the learning rate. $O(\eta^2)$ is ignored since it is close to zero.

Following the discussion in (Arora et al., 2018), if the values of $\boldsymbol{w}_1^{(0)}$ and $\tilde{\boldsymbol{p}}_i^{(0)}$ are initialized to close to zero vectors, then $\boldsymbol{p}_i^{(0)}$ would also be close to a zero vector. Thus, $\boldsymbol{p}_i^{(t)}$ is close to a weighted combination of the previous gradients. Therefore, we may further rewrite Equation 6 in the following form.

$$\boldsymbol{p}_i^{(t+1)} = \boldsymbol{p}_i^{(t)} - \boldsymbol{\beta}^{(t)} \odot \nabla_{\boldsymbol{p}_i^{(t)}} - \gamma^{(t)} \odot \left(\sum_{\tau=1}^{t-1} \mu^{(t,\tau)} \nabla_{\boldsymbol{p}_i^{(\tau)}} \right), \tag{7}$$

where $\mu^{(t,\tau)}$ is regarded as a time-varying and adaptive momentum coefficient.

As shown in Equation 7, the update of $\boldsymbol{p}_i$ corresponds to SGD with momentum, and the learning rate $\boldsymbol{\beta}^{(t)}$ and the momentum coefficient $\mu^{(t,\tau)}$ are epoch-varying and adaptive. Additionally, each element of $\boldsymbol{p}_i$ has a different learning rate in the same epoch.

Similarly, we can show that the update of $\boldsymbol{q}_j$ follows Equation 8.

$$\boldsymbol{q}_j^{(t+1)} = \boldsymbol{q}_j^{(t)} - \boldsymbol{\delta}^{(t)} \odot \nabla_{\boldsymbol{q}_j^{(t)}} - \zeta^{(t)} \odot \left(\sum_{\tau=1}^{t-1} \xi^{(t,\tau)} \nabla_{\boldsymbol{q}_j^{(\tau)}} \right), \tag{8}$$

where $\boldsymbol{\delta}^{(t)} := \eta \boldsymbol{w}_2^{(t)} \odot \boldsymbol{w}_2^{(t)}$, $\zeta^{(t)} := \eta \ell_{ij}^{(t)} \left(\boldsymbol{w}_1^{(t)} \oslash \boldsymbol{w}_2^{(t)} \right) \odot \tilde{\boldsymbol{p}}_i^{(t)} \odot \tilde{\boldsymbol{q}}_j^{(t)}$, and $\xi^{(t,\tau)}$ is a time-varying and adaptive momentum coefficient.

In other words, the update of $\boldsymbol{q}$ shares similar properties to the update of $\boldsymbol{p}$ we showed above – momentum, adaptive learning rate, and different learning rates for different parameters within the same epoch. Since these modern optimization techniques are implicitly applied, it is likely to converge faster than the standard matrix factorization model,

4 EXPERIMENTS

4.1 Experimental datasets

We use 5 public datasets with different rating scales and different densities as the experimental data, including Epinions product review dataset (Massa et al., 2008), MovieLens-1M rating dataset (ml-1m) (Harper and Konstan, 2016), FilmTrust dataset (Guo et al., 2013), Yahoo! Movies rating dataset (ymovies) (Marlin and Zemel, 2009; Marlin et al., 2012), and Amazon Musical Instruments product rating dataset (AMI) (He and McAuley, 2016). Table 1 shows the statistics of these datasets, including the number of unique users, the number of unique items, the number of users' ratings on the items, the rating scales, and the density (i.e., the number of ratings divided by the product of the number of users and the number of items).

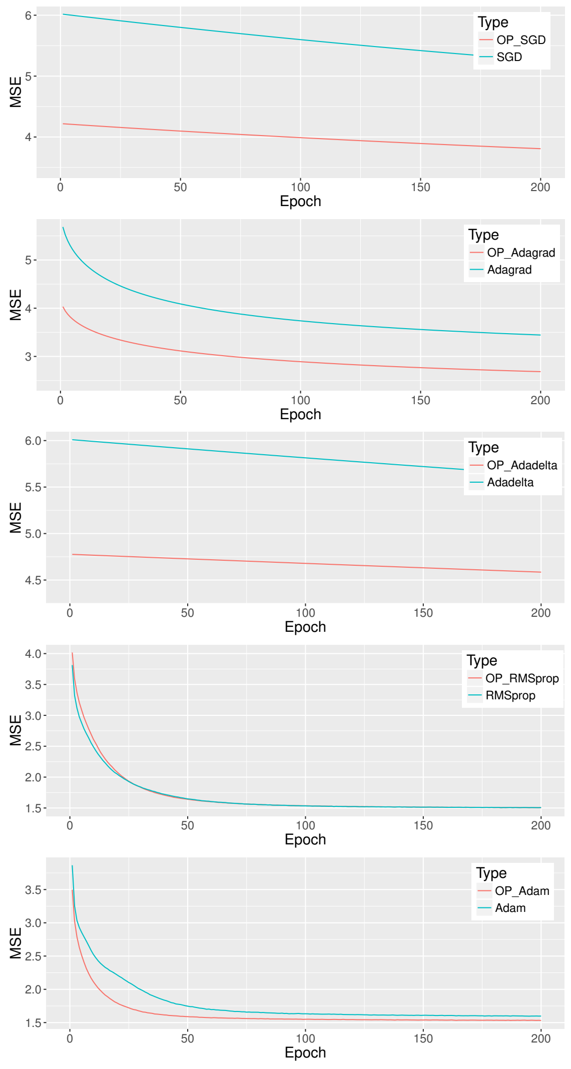

Figure 2: Mean squared error values vs epoch counts of different optimizers on the original and the overparameterized MF on the Epinions dataset

4.2 Experimental design

For each of the experimented datasets, we randomly selected 80% of the ratings as the training data and the remaining 20% as the test data.

We used the following optimizers to update the parameters of the original matrix factorization model: vanilla SGD, AdaGrad, Adadelta, RMSprop, and Adam, which were labeled `SGD`, `AdaGrad`, `Adadelta`, `RMSprop`, and `Adam`, respectively. Additionally, we used the same set of optimizers to update the parameters of the overparameterized matrix factorization model. We labeled them as `OP_SGD`, `OP_AdaGrad`, `OP_Adadelta`, `OP_RMSprop`, and `OP_Adam`.

Table 1: Statistics of the benchmark datasets

Dataset	# users	# items	# ratings	Density	Rating scale
Epinions	40,163	139,738	664,824	0.0118%	[1, 2, 3, 4, 5]
MovieLens-1M	6,040	3,706	1,000,209	4.4684%	[1, 2, 3, 4, 5]
FilmTrust	1,508	2,071	35,497	1.1366%	[0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4]
Yahoo! Movies	7,642	11,916	221,367	0.2431%	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
AMI	339,231	83,046	500,176	0.0018%	[1, 2, 3, 4, 5]

Table 2: Test MSE of different models and different optimizers at 50th and 200th epochs. For each `OP_<X>` and `<X>` pair, we highlight the one that better predicts the ratings (i.e., lower mean squared error score)

	Epinions		MovieLens-1M		FilmTrust		Yahoo! Movies		AMI	
Epoch count	50	200	50	200	50	200	50	200	50	200
OP_SGD	**4.6533**	**4.5395**	**4.1771**	**3.8116**	**3.3948**	**3.2023**	**21.1163**	**18.2237**	**4.6698**	**4.5319**
SGD	5.8939	5.5520	5.1323	4.0176	5.0633	4.7147	35.9105	35.1064	5.9421	5.6456
OP_AdaGrad	**4.2649**	**4.0729**	**3.5258**	2.9000	**2.9252**	**2.6941**	**18.4577**	**16.3162**	**4.3916**	**4.2499**
AdaGrad	5.0220	4.5538	3.6354	**2.8018**	4.5552	4.0493	34.9579	34.6572	5.3836	5.0689
OP_Adadelta	**4.6590**	**4.5886**	**4.2465**	**4.0167**	**3.5449**	**3.4816**	**23.7097**	**22.0213**	**4.7233**	**4.6365**
Adadelta	5.9347	5.7533	5.4530	4.7247	5.2721	5.1980	36.8563	36.2128	6.0604	5.8919
OP_RMSprop	**1.7400**	1.5227	1.2523	**1.2497**	**1.2619**	1.0156	**13.0354**	**12.9759**	**2.9023**	**2.1157**
RMSprop	1.7423	**1.5187**	**1.2512**	1.2505	1.3179	**0.9876**	34.3407	34.3407	3.2816	2.1889
OP_Adam	1.7607	1.6335	1.2549	1.2508	**1.3451**	1.1072	**13.0758**	**12.9767**	**2.9313**	**2.5787**
Adam	**1.6948**	**1.5934**	**1.2541**	**1.2505**	1.4430	**1.0867**	34.3789	34.3789	3.2248	2.6011

4.3 Convergence speed

Here, we show the comparison of the convergence speed between the original MF model and the overparameterized MF model based on fixed optimizers. The top subfigure of Figure 2 shows the relationship between the training mean squared error (MSE) and the epoch when using SGD as the optimizer for the original MF model and the overparameterized MF model. We report the training MSE score here because we want to compare the optimization speed but not the generalizability of the models. As seen, the overparameterized MF model obtained better parameters with fewer epochs. We also showed similar comparisons on other optimizers, including AdaGrad, Adadelta, RMSprop, and Adam. As shown in the other subfigures of Figure 2, in almost all cases, the overparameterized MF model requires significantly fewer epochs to reach decent results. The only exception is RMSprop, in which the overparameterized MF model requires slightly more epochs than the standard MF model initially.

To validate the observation, we conducted similar experiments on other datasets, including Amazon Musical Instruments, FilmTrust, MovieLens-1M, and Yahoo! Movies. We observed similar results on all these datasets. To save space, we do not include these figures. However, from Figure 1, it is clear that, on average, the overparameterized MF converged much faster in the first 50 epochs.

4.4 Model accuracy

We have shown that when fixing the optimizer, the overparameterized MF model converged faster than the standard MF model. In this section, we show the comparison of the generalizability for the standard MF model and the overparameterized MF model when they used the same optimizer.

We generated the models based on the training data and computed the MSE scores based on the test data. Table 2 shows the result at the 50th and the 200th epoch. As seen, in nearly all cases, the overparameterized MF better predicts the testing rating scores (i.e., smaller MSE score) compared to the standard MF model, when both the overparameterized and the standard MF used the identical optimizer. This implies that the overparameterized model, albeit with redundant parameters, did not seem to overfit the training data. We believe this is because both models have the same hypothesis space.

4.5 Training time

Although the overparameterized model required much fewer epochs to obtain decent parameters, the overparameterized model needed to update more parameters within one epoch. This may raise the following concern – if the overparameterized model requires more computation time to accomplish one epoch, does the overparameterized model truly use a shorter amount of time to converge?

To validate this, we empirically tested the com-

Table 3: The required training seconds per epoch for different models and different optimizers

	Epinions	MovieLens-1M	FilmTrust	Yahoo! Movies	AMI
OP_SGD	7.6626	5.9525	0.2729	1.5940	7.4439
SGD	7.6017	5.7957	0.2552	1.5619	7.4621
OP_AdaGrad	9.5333	6.2044	0.2776	1.7297	10.4498
AdaGrad	9.5743	6.1554	0.2689	1.6902	10.4668
OP_Adadelta	12.7689	6.5984	0.2857	1.8528	16.7022
Adadelta	12.7604	6.5121	0.2772	1.8310	17.4254
OP_RMSprop	10.1062	6.1898	0.2774	1.7688	11.3391
RMSprop	9.8776	6.1778	0.2758	1.7213	11.7359
OP_Adam	11.2647	6.3480	0.2818	1.8020	13.0728
Adam	11.0878	6.2927	0.2651	1.7836	13.6885

putation time of one epoch for each method on the 5 tested datasets. We conducted these experiments on a standard desktop computer with an Nvidia RTX 2080Ti video card. The results are shown in Table 3. If we compare each pair of `OP_<X>` and `<X>`, it appears that the overparameterized model required slightly longer training time on average.

To quantify the relative training time of different methods for every dataset, we used `SGD` as the compared baseline. We computed the relative training time per epoch of method x on the dataset d by Equation 9.

$$r_x^{\langle d\rangle} = \frac{t_x^{\langle d\rangle}}{t_{SGD}^{\langle d\rangle}}, \tag{9}$$

where $t_x^{\langle d\rangle}$ denotes the running time per epoch of method x on the dataset d. For example, $t_{OP\_SGD}^{\langle Epinions\rangle} = 7.6626$, and so $r_{OP\_SGD}^{\langle Epinions\rangle} = 7.6626/7.6017 \approx 1.0080$, which means `SGD` was 1.0080 times faster than `OP_SGD` to finish one epoch.

We computed the average relative training time by the geometric mean of the relative training time on different datasets, as defined by Equation 10. We used the geometric mean instead of the arithmetic mean because the geometric mean is appropriate for the average of relative proportion measures.

$$r_x = \left(\prod_d r_x^{\langle d\rangle}\right)^{1/n} \tag{10}$$

The r_x values are the x-axis of each method in Figure 1. Similarly, for each method, we computed the average relative MSE scores using `SGD` as the baseline. The results are the y-axis in Figure 1. As a result, this figure summarizes our experimental results. First, applying an optimizer on the overparameterized MF model requires fewer epochs to obtain decent parameters, compared to using the same optimizer on the standard MF model. Second, if we compare the running time of one epoch, applying an optimizer on the standard MF model is usually faster but only slightly faster. Overall, using the overparameterized model is still highly beneficial.

5 DISCUSSION

This paper studied overparameterization on the matrix factorization model, both theoretically and empirically. We proposed one possible method for overparameterizing the matrix factorization model. We mathematically showed that applying the vanilla SGD optimizer on the overparameterization MF implies using the SGD with momentum and adaptive learning rate on each learnable parameter. We conducted experiments on many public datasets to show that an overparameterized MF model with popular optimizers (e.g., Adam and RMSprop) converges faster than the standard MF model with the same optimizers. Both the theoretical analysis and the empirical results indicate that overparameterization accelerates the optimization of the matrix factorization model. Therefore, whenever matrix factorization is needed, we suggest using the overparameterized version proposed in this paper to accelerate the training process. Such a technique can be applied to a wide range of applications and systems that leverage MF or its variants as the core algorithm.

Although we have shown an overparameterized MF model, this is not the only method for overparameterizing the MF model. For example, the model proposed in (Chen, 2017) is also an overparameterized MF model. However, that work was motivated from a different perspective and did not formally discuss the connection among overparameterization, optimization, and the model per se. Since there can be multiple methods for overparameterizing the MF model, it could be interesting to study the connection between different overparameterized MF models, and further validate whether other overparameterized models also accelerate the optimization process. This is one of the topics we are interested in continuing.

Another possible future work is to overparameterize other supervised learning-based recommendation models, such as factorization machine, NeuralCF, and Wide & Deep. We are interested in investigating, both theoretically and empirically, the optimization speed of these models when these models are overparameterized.

ACKNOWLEDGEMENTS

We acknowledge partial support by the Ministry of Science and Technology in Taiwan under Grant No.: MOST 107-2221-E-008-077-MY3.

REFERENCES

Arora, S., Cohen, N., and Hazan, E. (2018). On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*.

Behera, L., Kumar, S., and Patnaik, A. (2006). On adaptive learning rate that guarantees convergence in feedforward networks. *IEEE Transactions on Neural Networks*, 17(5):1116–1125.

Chen, H.-H. (2017). Weighted-SVD: Matrix factorization with weights on the latent factors. *arXiv preprint arXiv:1710.00482*.

Chen, H.-H. (2018). Behavior2Vec: Generating distributed representations of users' behaviors on products for recommender systems. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):43.

Chen, H.-H. and Chen, P. (2019). Differentiating regularization weights–a simple mechanism to alleviate cold start in recommender systems. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1).

Chen, H.-H., Chung, C.-A., Huang, H.-C., and Tsui, W. (2017). Common pitfalls in training and evaluating recommender systems. *ACM SIGKDD Explorations Newsletter*, 19(1):37–45.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019). Gradient descent provably optimizes over-parameterized neural networks. In *6th International Conference on Learning Representations, ICLR 2018*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818. ACM.

Guo, G., Zhang, J., and Yorke-Smith, N. (2013). A novel bayesian similarity measure for recommender systems. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19.

He, R. and McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517. International World Wide Web Conferences Steering Committee.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Kurucz, M., Benczúr, A. A., and Csalogány, K. (2007). Methods for large scale svd with missing values. In *Proceedings of KDD Cup and Workshop*, volume 12, pages 31–38.

Luo, X., Zhou, M., Xia, Y., and Zhu, Q. (2014). An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284.

Marlin, B., Zemel, R. S., Roweis, S., and Slaney, M. (2012). Collaborative filtering and the missing at random assumption. *arXiv preprint arXiv:1206.5267*.

Marlin, B. M. and Zemel, R. S. (2009). Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM Conference on Recommender systems*, pages 5–12. ACM.

Massa, P., Souren, K., Salvetti, M., and Tomasoni, D. (2008). Trustlet, open research on trust metrics. *Scalable Computing: Practice and Experience*, 9(4).

Piatetsky, G. (2007). Interview with Simon Funk. *ACM SIGKDD Explorations Newsletter*, 9(1):38–40.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Rendle, S. (2010). Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Steck, H. (2013). Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender systems*, pages 213–220.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Whittaker, G., Confesor, R., Di Luzio, M., Arnold, J., et al. (2010). Detection of overparameterization and overfitting in an automatic calibration of SWAT. *Transactions of the ASABE*, 53(5):1487–1499.

Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6. IEEE.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.