An Experimental Study of Balance in Matrix Factorization

Jennifer Hsia
Department of Computer Science
Princeton University
Princeton, United States of America
jhsia@princeton.edu

Peter J. Ramadge
Department of Electrical Engineering
Princeton University
Princeton, United States of America
ramadge@princeton.edu

2021 55th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-1268-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/CISS50987.2021.9400232

***Abstract*—We experimentally examine how gradient descent navigates the landscape of matrix factorization to obtain a global minimum. First, we review the critical points of matrix factorization and introduce a balanced factorization. By focusing on the balanced critical point at the origin and a subspace of unbalanced critical points, we study the effect of balance on gradient descent, including an initially unbalanced factorization and adding a balance-regularizer to the objective in the MF problem. Simulations demonstrate that maintaining a balanced factorization enables faster escape from saddle points and overall faster convergence to a global minimum.**

***Index Terms*—matrix factorization, gradient descent, balance, saddle points, non-convex optimization.**

I. Introduction

Matrix factorization is a well-known technique for representational learning. In basic matrix factorization, a real $m \times n$ matrix X is approximately factorized into the product of (non-unique) real matrices W and S by solving

$$\min_{W \in \mathbb{R}^{m\times k}, S \in \mathbb{R}^{k \times n}} 1/2 \|X - WS\|_F^2. \tag{1}$$

The inner dimension k of the product WS is a pre-specified integer parameter. When $k < r = \operatorname{rank}(X)$, the new low-rank representation of X forces a careful selection of which aspects of X to encode in W and S to the maximize the fidelity of the representation. This saves space, simplifies downstream computations involving X, and can give insight into latent factors giving rise to the data.

A motivating application for the present work is the analysis of multi-subject functional magnetic resonance imaging (fMRI) of brain activity. The top diagram in Figure 1 shows a simplified example where the data matrix on the left is constructed by stacking two subjects' time-synchronized fMRI data matrices. By solving the constrained matrix factorization

$$\begin{aligned} \min_{W_1, W_2, S} J(W,S) = \quad & \frac{1}{2} \left\| \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} - \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} S \right\|_F^2 \\ \text{s.t.} \quad & {W_i}^T W_i = I_k, \quad i = 1, 2, \end{aligned} \tag{2}$$

The first author thanks the School of Engineering and Applied Science at Princeton University for a Essig-Enright Fund summer research award. The second author acknowledges support from NSF MRI Award: 1919452.

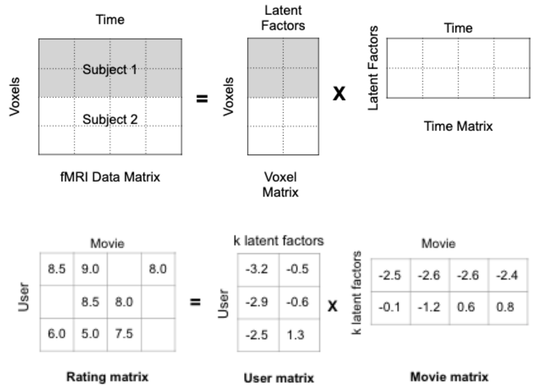

Fig. 1. Top: Example of matrix factorization applied to fMRI data for low-rank representation. Below: Example of matrix factorization applied to a recommender system [1]

we obtain a voxel matrix W and a time matrix S of rank at most k. The voxel matrix identifies k latent factors describing how each subject's brain responds to the experimental stimulus (e.g. audiobook, movie). The time matrix identifies k latent factors describing the shared stimulus presented to the subjects. Together, these matrices reveal functional response differences across subjects and the components of the stimulus eliciting a shared response.

One can also add regularization to problem (1). For example, a solution of (1) is balanced if $W^T W - SS^T = 0$ [2]. This implies $\|W\|_F^2 = \|S\|_F^2$. A balanced solution can be encouraged by adding a regularizer to (1):

$$\min_{W,S} 1/2 \|X - WS\|_F^2 + \rho(\|W\|_F^2 + \|S\|_F^2), \tag{3}$$

where $\rho > 0$ is a regularization parameter. This problem can be solved in place problem (2) to yield a distinct but nevertheless informative factorization. We will demonstrate that balance has advantages when (1) and (3) are solved using gradient descent.

Much is known about the landscape of the matrix factorization problem. In 1989, Baldi and Hornik [3] showed that every critical point of problem (1) is either a global minimum or a strict saddle point. A strict saddle point (W,S) is a critical point at which $\lambda_{\min}(\nabla^2 J(W,S)) < 0$. Such saddle points guarantee at least one escape direction from a neighborhood of the saddle. Since the critical points of matrix factorization can only be global minima or strict saddles, this greatly simplifies the landscape of the problem. Nevertheless, the MF landscape is still complex. Valavi et al. examine the strict saddle points of (1) and derive a bound for the minimum eigenvalue of the Hessian at critical points [2]. These bounds show that despite being strict saddles, the saddle points can become very flat.

There are many other variations and applications of matrix factorization. See for example, the recent survey paper by Chi et al. [4]. This paper gives an overview of matrix factorization, including a comparison of tailored initialization and initialization-free algorithms and a discussion of canonical matrix factorization problems.

To illustrate the breadth of these applications we mention a few specific examples. In recommender systems (lower diagram in Fig. 1), each row of X represents a user, each column represents an item, and the entries are user ratings of items. In general, X is sparse since each user rates only a few items. The goal is to predict the empty entries of X. Factoring X into the product of a low-rank user matrix and item matrix allows approximation of the unknown entries:

$$\min_{W,S} J(W,S) = \frac{1}{2}\left\|K \otimes (X - WS)\right\|_F^2 . \tag{4}$$

Here $\otimes$ denotes the Schur product, and matrix K selects the known entries of X when minimizing the loss function.

In other versions of matrix factorization, constraints are very important. For example, [5] uses a non-negative matrix factorization algorithm on multimodal attention-deficit/hyperactivity disorder (ADHD) data to idenitfy biomarkers of the disorder. Another example, uses a rank 1 factorization problem with a linear constraint for online matrix factorization [6]:

$$\begin{aligned} \min_{W,S} J(W,S) = \quad & \frac{1}{2}\left\|X - ws^T\right\|_F^2 \\ \text{s.t.} \quad & a^T w + b = 0. \end{aligned} \tag{5}$$

Our goal is to better understand the role and limitations of gradient descent in solving MF problems. Above, we have conveyed the broad range of such problems. In the present paper, we focus on problems of the form (1) and (3). We investigate the use of (3) as a surrogate for problems of the form (2). Specifically, we explore the consequences of moving away from the strict saddle point at the origin $(W,S) = (\mathbf{0},\mathbf{0})$ into a subspace of strict saddles. These saddles can become increasingly flat as one moves away from the origin. We show how to construct such strict saddles so as to attain precise control over the minimum eigenvalue of the Hessian. This allows us to examine the impact of these saddle points on gradient descent as the saddle point moves further away from the balanced saddle at the origin. In detail, we use the above parameterization together with an fMRI dataset to empirically demonstrate the impact of strict saddles on gradient descent and on the balance ($W^TW - SS^T$ or its norm version $\|W\|_F^2 - \|S\|_F^2$) of the factorization. We perform the same investigation for problem (3) to demonstrate how and why improving balance improves gradient descent convergence.

II. Preliminaries

A. Singular Value Decomposition and Balanced Solutions

Assume $X \in \mathbb{R}^{m\times n}$ with $m \leq n$ has rank $r \leq m$. The singular value decomposition (SVD) of X yields $X = U\Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T$ giving r pairs of singular vectors u_i, v_i and singular values $\sigma_i > 0$. One solution of (1) is $(W^*, S^*) = (U_k, \Sigma_k V_k^T)$, where U_k is the first k columns of U, Σ_k is the first $k \times k$ submatrix of Σ, and V_k is the first k columns of V. To verify this claim, recall that $\|A - B\|_F^2 \geq \sum_{i=1}^q (\sigma_i(A) - \sigma_i(B))^2$ [7, Corollary 7.4.1.3]. Then note that W^*S^* achieves the lower bound

$$\|X - W^*S^*\|_F^2 = \|\sum_{i=1}^r \sigma_i u_i v_i^T - \sum_{i=1}^k \sigma_i u_i v_i^T\|_F^2 = \sum_{i=k+1}^r \sigma_i^2.$$

This solution, however, requires time complexity $O(mn^2)$ [8].

In general, the solution given above is not balanced. On the other hand, $(W^*, S^*) = (U_k\sqrt{\Sigma_k}, \sqrt{\Sigma_k}V_k^T)$ is a balanced solution. To verify this note that ${W^*}^T W^* - S^* {S^*}^T = \Sigma_k U_k^T U_k - \Sigma_k V_k^T V_k = \Sigma_k - \Sigma_k = 0$. Balanced solutions have additional useful properties [9], [10].

B. Gradient Descent

Gradient descent (GD) provides an alternative approach for solving matrix factorization (MF) problems, with the potential for improved scaling with the size of the problem. After setting an initial value (W_0, S_0), GD iterates:

$$(W_{t+1}, S_{t+1}) = (W_t, S_t) - \lambda \nabla J(W_t, S_t),\ t \geq 0, \tag{6}$$

until either the accuracy is satisfactory or a maximum iteration is reached. Here $\nabla J(W,S) = ((WS-X)S^T, W^T(WS-X))$ is the gradient of $J(W,S)$ and λ is the learning rate.

To gauge the time complexity of GD, we first break down the gradient into individual terms. For two matrices $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{r\times n}$, $(AB)_{ij}$ is a dot product of row i of A and column j of B. This requires a sum over r products with a time complexity $O(r)$. There are mn elements to compute since $AB \in \mathbb{R}^{m\times n}$, so the time complexity of computing the product is $O(mnr)$. Similarly, the time complexity of calculating XS^T and W^TX is $O(mnr)$, and that of WSS^T and W^TWS is $O(\max(m,n)r^2)$. The max operator arises since SS^T requires $O(nr^2)$ while $W(SS^T)$ requires $O(mr^2)$. The time complexity is dominated by the larger term, hence the runtime is $O(\max(m,n)r^2)$. Since $r \leq \min(m,n)$, it follows that $O(\max(m,n)r^2) \leq O(mnr)$. So computing XS^T and W^TX dominates the total time. The time complexity per update is $O(mnr)$.

If the maximum number of GD iterations is T and the factorization dimension is k, then the total time complexity $O(Tmnk)$ [11]. Recall the time complexity of SVD is $O(mn^2)$. If we choose T, k wisely, GD can be faster than SVD. GD also only requires storage for the current gradient.

C. Critical Points

A point (W, S) is a critical point of J if $\nabla J(W, S) = (\mathbf{0}, \mathbf{0})$. Here $\mathbf{0}$ denotes a matrix of all zeros. The first zero matrix above has the same dimensions as W ($m \times k$), while the second has the same dimensions as S ($k \times n$).

Basic matrix factorization is a non-convex problem. It is known, however, that every critical point of the basic MF problem is either a global minimum or a strict saddle point [3], [2]. Since GD uses the gradient to update (W, S), critical points are fixed points under GD. If GD enters a neighborhood of a global minimum, the gradient becomes very small, but GD is still on track to find an optimal solution (W^*, S^*). In contrast, if it enters a neighborhood of a saddle point, the gradient becomes very small, but (W_t, S_t) is not in the neighborhood of an optimal solution. Recall a strict saddle has the property that the Hessian of J at the saddle has a negative eigenvalue. Hence there is an escape direction from the saddle that allows (noisy) GD to further decrease the loss.

Below we examine the balanced critical point $(W, S) = (\mathbf{0}, \mathbf{0})$ and the subspace of critical points $\mathcal{V}_0 = \{(\mathbf{0}, C_0^T V_0^T)\}$ where $C_0 \in \mathbb{R}^{(n-r)\times k}$ is a free parameter and $V_0 = \begin{bmatrix} v_{r+1} & \dots & v_n \end{bmatrix} \in \mathbb{R}^{n\times(n-r)}$ is the trailing subset of X's right singular vectors.

We need two results from prior work that provide the setting for our experimental investigation. We state these as lemmas and in each case give a sketch of the proof. This is important since we need to use elements of the proof in our investigation.

It is easy to see that $(W, S) = (\mathbf{0}, \mathbf{0})$ is a balanced critical point of J. For nontrivial X, it is a strict saddle.

Lemma 2.1 (After [2]): The minimum and maximum eigenvalues of $\nabla^2 J(\mathbf{0}, \mathbf{0})$ are $\pm\sigma_1$, where σ_1 is the largest singular value of $X = \sum_{i=1}^r \sigma_i u_i v_i^T$.

Proof. (Sketch) Let $G_{i,j} = u_i e_j^T$, $H_{j,i} = e_j v_i^T$, $i \in [1 : m]$, $j \in [1 : k]$. Substituting the vector $(G_{i,j}, \delta H_{j,i})$, with $\delta \in \mathbb{R}$, into the Hessian map $\nabla^2 J(\mathbf{0}, \mathbf{0})[(G, H)] = (-XH^T, -G^T X)$ yields:

$$\begin{aligned} \nabla^2 J(W, S)[(G_{i,j}, \delta H_{j,i})] &= (-\delta\sigma_i G_{i,j}, -\sigma_i H_{j,i}) \\ &= \rho(G_{i,j}, \delta H_{j,i}). \end{aligned}$$

The last line holds if and only if $\rho = -\delta\sigma_i = -\frac{\sigma_i}{\delta}$. Multiply both expressions for ρ by δ yields $\delta^2 = 1$. So $\delta = \pm 1$, and $\rho = \pm\sigma_i$. This yields $2mk$ eigenvalues of the Hessian. The final $(n - m)k$ eigenvalues result by considering the vectors $(\mathbf{0}, H_{j,i})$, $i \in [m + 1 : n], j \in [1 : k]$. The Hessian map above acts on these vectors to yield $\nabla^2 J(\mathbf{0}, \mathbf{0})[(\mathbf{0}, H_{i,j})] = (\mathbf{0}, \mathbf{0}) = 0(\mathbf{0}, H_{j,i})$. Hence these are eigenvectors with eigenvalue 0.□

Now consider the subspace $\mathcal{V}_0 = \{(W, S) = (\mathbf{0}, C_0^T V_0^T)\}$. It is easy to verify that each point in $\mathcal{V}_0$ is a critical point (i.e., $\nabla J(\mathbf{0}, C_0^T V_0^T) = (\mathbf{0}, \mathbf{0})$. Hence $\mathcal{V}_0$ is a continuum of critical points. By the lemma below every point $\mathcal{V}_0$ is a strict saddle.

Lemma 2.2 (After [2]): Let ω_k denote the minimal singular value of $C_0^T C_0$. Then

$$\lambda_{\min}(\nabla^2 J(\mathbf{0}, C_0^T V_0^T)) = \tfrac{\omega_k}{2} - \sqrt{\sigma_1^2 + (\tfrac{w_k}{2})^2} < 0. \quad (7)$$

Proof. (Sketch) Let $C_0^T C_0$ have SVD $Z\Omega Z^T$, where $Z \in \mathbb{R}^{k\times k}$ is orthogonal and $\Omega \in \mathbb{R}^{k\times k}$ is a diagonal matrix of eigenvalues $\omega_j \geq 0$ in descending order. Let $G_{i,j} = u_i z_j^T$ and $H_{j,i} = z_j v_i^T$, $i \in [1 : m], j \in [1 : k]$. The Hessian acts on the vectors $(G_{i,j}, \delta H_{j,i})$ to give:

$$\nabla^2 J(\mathbf{0}, C_0^T V_0^T)[(G_{i,j}, \delta H_{j,i})] = ((\omega_j - \delta\sigma_i)G_{i,j}, -\sigma_i H_{j,i}).$$

This equals $\rho(G_{i,j}, \delta H_{j,i})$ if and only if $\rho = \omega_j - \delta\sigma_i = -\frac{\sigma_i}{\delta}$. Assume $\sigma_i > 0$. Multiplying both expressions for ρ by δ/σ_i yields $\delta^2 - \frac{\omega_j}{\sigma_i}\delta - 1 = 0$. Solving this equation, gives two solutions $\delta = \frac{\omega_j}{2\sigma_i} \pm \frac{1}{\sigma_i}\sqrt{\sigma_i^2 + (\frac{w_j}{2})^2}$. Hence $\rho = \frac{\omega_j}{2} \pm \sqrt{\sigma_i^2 + (\frac{w_j}{2})^2}$. This gives $2rk$ eigenvalues of the Hessian ($r = \operatorname{rank}(X)$), the least of which is given by (7). Finally, we need to show that none of the remaining Hessian eigenvalues is more negative than (7). The proof is similar to that used in the proof of Lemma 2.1. □

By Lemma 2.1, $(\mathbf{0}, \mathbf{0})$ is a strict saddle point with $\lambda_{\min}(\nabla^2 J(W, S)) = -\sigma_1$. The smaller the value of $\lambda_{\min}(\nabla^2 J(\mathbf{0}, \mathbf{0}))$, the steeper the best escape direction from the saddle point, and the faster GD can continue its descent. Note that $(\mathbf{0}, \mathbf{0})$ is a balanced strict saddle point.

By Lemma 2.2, every point in $\mathcal{V}_0$ is also a strict saddle. However, when $\omega_k \gg \sigma_1$, $\lambda_{\min}(\nabla^2 J)$ can become arbitrarily close to zero. This means the fastest escape direction can become increasingly flat, and it can take longer for GD to escape a neighborhood the saddle. We also note that except for the strict saddle at $(\mathbf{0}, \mathbf{0})$, the critical points in $\mathcal{V}_0$ are unbalanced: $W^T W = \mathbf{0}$ and $SS^T = C_0^T C_0$. Some of these unbalanced critical points (those with ω_k large) give rise to the very flat saddles discussed above.

Constructing Saddle Points: Precise Control of $\lambda_{\min}(\nabla^2 J)$

In Section III, we need to construct strict saddle points in $\mathcal{V}_0$ with precise control of $\lambda_{\min}(\nabla^2 J)$. These factorizations will be unbalanced, but balance alone does not control $\lambda_{\min}(\nabla^2 J)$. For example, if $\omega_1 \gg 0$ and $\omega_k = 0$, then $(\mathbf{0}, C_0^T V_0^T)$ is unbalanced but $\lambda_{\min}(\nabla^2 J(\mathbf{0}, C_0^T V_0^T)) = -\sigma_1$ remains invariant. To control $\lambda_{\min}(\nabla^2 J)$, we use $\omega_k > 0$ and ensure that $\sigma_{\min}(C_0^T C_0) = \omega_k$. Then $\lambda_{\min}(\nabla^2 J(\mathbf{0}, C_0^T V_0^T))$ is given by (7). This suggests selecting

$$C_0 = \arg \min_{C \in \mathbb{R}^{(n-r)\times k}} \|C\|_F^2, \quad \text{s.t. } \sigma_k(C) = \sqrt{\omega_k}. \quad (8)$$

One solution is $C_* = \sqrt{\omega_k} \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_k \end{bmatrix}$ where $\mathbf{e}_i$ is the i-th standard basis vector in $\mathbb{R}^{n-r}$. All other solutions arise by right and left multiplication of C_* by orthogonal matrices.

III. Experimental Study

The goal is to better understand how balance and gradient descent interact in the context of matrix factorization. To do so, we use three forms of initialization. The first initializes (W, S) randomly (each component $\sim N(0, \sigma^2)$) about the balanced saddle point $(\mathbf{0}, \mathbf{0})$. The second initializes (W, S) randomly about an unbalanced saddle point $(\mathbf{0}, C_*^T V_0^T)$. The third, has the same initialization as case 2, but a balance regularizer is added to the MF objective function in (1).

We examine the performance of gradient descent with these three setups using an fMRI dataset [12]. The dataset measures the BOLD response of voxels in the ventral temporal cortex of 10 subjects as they watch the movie "Raiders of the Lost Ark". Data for subject j is organized as a matrix $X_j \in \mathbb{R}^{500\times 2203}$ where the first dimension indexes 500 selected voxels and the second indexes time samples (TRs).

A. Initialization

All single phase GD machine learning algorithms require selection of initial parameter values. It is common to initialize the parameters randomly in a neighborhood of the origin. For basic MF, this means random initialization about the strict saddle $(\mathbf{0},\mathbf{0})$. This is reasonable since we know this point has a strong descent direction with eigenvalue $-\sigma_1$. However about this point (and almost all strict saddles), there are unbalanced saddles some of which can become very flat. If the GD trajectory is initialized poorly, or wanders into unbalanced regions, it can encounter these very flat saddles. Our goal is to prevent GD from moving near these flat saddle points and being slowed by the lack of strong descent direction. To do so, we investigate if preserving balance keeps the GD trajectory away from such saddles.

In passing, note that related but distinct issues arise in training neural networks, e.g., the vanishing/exploding gradient problem. To combat these issues, Glorot and Bengio [13] and He et al. [14] have proposed specialized initialization procedures. The problem for basic MF differs in that there are no nonlinearities to deal with. Instead the balance of the factors plays a more central role.

B. Effect of Balance on Convergence

Generally, the more balanced a saddle point, the faster GD can escape from a neighborhood of the saddle. To investigate this, we consider three scenarios: starting/passing near (i) the balanced saddle $(W_0, S_0) = (\mathbf{0},\mathbf{0})$, (ii) an unbalanced saddle $(W_0, S_0) = (\mathbf{0}, C_*^T V_0^T)$, and (ii) the same unbalanced saddle in (ii) but with the balance-regularized objective (3). We select $\omega_k = 2\sigma_1$ and $C_* = \sqrt{2\sigma_1}[\mathbf{e}_1, \dots, \mathbf{e}_k]$. This ensures the unbalanced saddle is flatter than the saddle at $(\mathbf{0},\mathbf{0})$.

The results of these three experiments are shown in Fig. 2. GD with initialization near $(\mathbf{0},\mathbf{0})$ converges the fastest; then balance-regularized GD initialized near the unbalanced point $(\mathbf{0}, C_*^T V_0)$, and finally initialization at the same point but without balance regularization. The convergence rates accord with the levels of balance in each case. We observe that trajectories starting from a neighborhood of the unbalanced saddle $(\mathbf{0}, C_*^T V_0^T)$ take much longer to escape from the saddle than those starting near the $(\mathbf{0},\mathbf{0})$ — trajectories starting near $(\mathbf{0},\mathbf{0})$ escape their initial saddle point at around 35 iterations, and trajectories starting near $(\mathbf{0}, C_*^T V_0^T)$ without regularization escape at around 75 iterations. This is a reflection of the less negative value of $\lambda_{\min}(\nabla^2 J(\mathbf{0}, C_*^T V_0^T))$. The results also indicate that this can be ameliorated by adding a balance

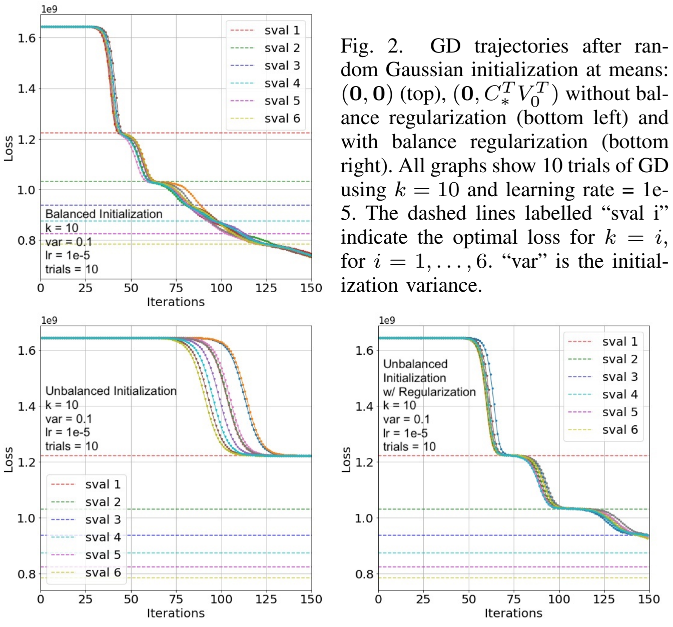

Fig. 2. GD trajectories after random Gaussian initialization at means: $(\mathbf{0},\mathbf{0})$ (top), $(\mathbf{0}, C_*^T V_0^T)$ without balance regularization (bottom left) and with balance regularization (bottom right). All graphs show 10 trials of GD using $k = 10$ and learning rate = 1e-5. The dashed lines labelled "sval i" indicate the optimal loss for $k = i$, for $i = 1, \dots, 6$. "var" is the initialization variance.

regularizer to the MF objective. When a balance regularizer is added, the trajectories initialized near the same saddle point escape faster at around 50 iterations.

Notice that trajectories with an unbalanced initialization and without regularization are less uniform than the other cases. These 10 trajectories reach their second saddle point over a range of 25 iterations, while the 10 trajectories in the other cases do so over a range of 5 iterations. Compared to the more balanced setups, an unbalanced initialization introduces more variance in its trajectories and stalls longer before escaping from the initial saddle point. In the next section, we examine the balance of the points at each iteration for these scenarios to see how balance fluctuates along the GD trajectory.

C. Preservation of Balance

An important property of balance is its invariance under gradient flow: if (W_0, S_0) is balanced, then the solution of the o.d.e. $\dot{W}_t = W_t - \nabla_W J(W_t, S_t)$ and $\dot{S}_t = S_t - \nabla_S J(W_t, S_t)$, $t \geq 0$ remains in balance [9], [10], [15]. This suggest that if GD for MF starts near the balanced saddle $(\mathbf{0},\mathbf{0})$ and uses a very small learning rate, balance is approximately preserved. To explore this, we use Gaussian initialization with mean $(\mathbf{0},\mathbf{0})$ and a small learning rate (10^{-5}) to emulate gradient flow. The results are shown in Fig. 3.

In the top left plot of Fig. 3, the absolute imbalance starts near 0 as expected, increases slightly, but still remains on the order of 10^{-3}. The top right plot shows the relative imbalance by normalizing the absolute imbalance with the norms of W and S. Both absolute and relative imbalance of (W, S) remain on the order of 10^{-3}.

We now perform the same experiment with the same Gaussian initialization except with mean $(W_0, S_0) = (0, C_*^T V_0^T)$. The results are shown in the bottom plots of Fig. 3). Initial norm imbalance is large by construction. On an absolute

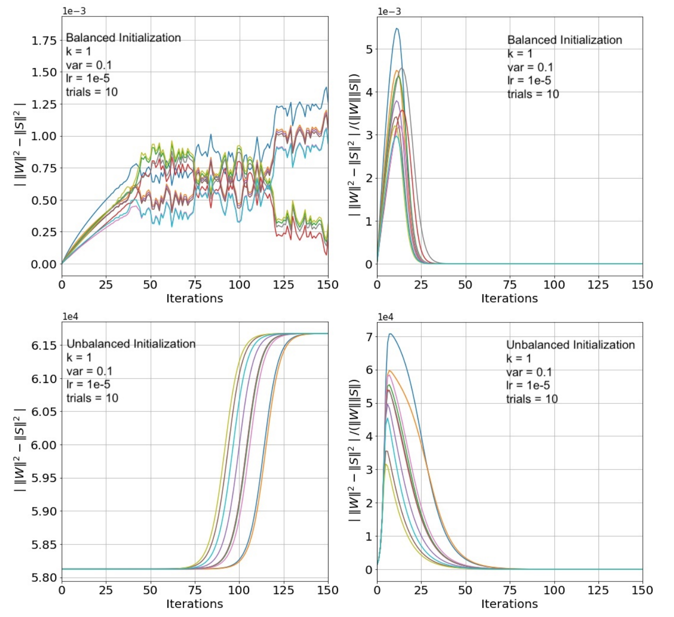

Fig. 3. Random Gaussian initialization with mean: $(\mathbf{0}, \mathbf{0})$ (top), and $(\mathbf{0}, C_*^T V_0^T)$ (bottom). Left plots display absolute norm imbalance v. iterations. Right plots display the relative norm imbalance (normalized by $\|W\|_F \|S\|_F$) vs. iterations. Each graph displays 10 trials of gradient descent with $k = 1$ and learning rate = 1e-5. "var" is the initialization variance.

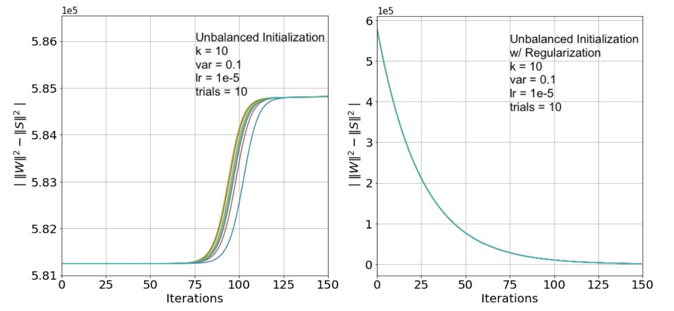

Fig. 4. Random Gaussian initialization with mean $(0, C_0^T V_0^T)$ followed by GD without regularization (left) and with balance regularization (right). Each graph reflects 10 trials with $k = 10$ and learning rate = 1e-5. lr could be relaxed here as gradient flow is not required here.

scale, the imbalance increases rapidly and is on the order of 10^4. The relative imbalance peaks around $7 * 10^4$ gradually and decreases towards 0. If (W_0, S_0) is unbalanced, (W_t, S_t) remains unbalanced. This result was also tested with $k = 10$.

In section III-B we empirically demonstrated that under unbalanced initialization, GD with a balance regularizer converges faster than GD without such a regularizer. We posit that the regularizer quickly decreases imbalance and this leads to avoiding unbalanced strict saddles with the potential to slow the progress of GD. To explore this idea, we randomly initialize (W_0, S_0) with mean $(\mathbf{0}, C_*^T V_0^T)$. Then apply GD to minimize the standard MF objective. From the same initial condition, we also apply GD to minimize the balance-regularized objective (3). The results are shown in Fig. 4.

Comparing the left and right plots in Fig. 4, we see that the

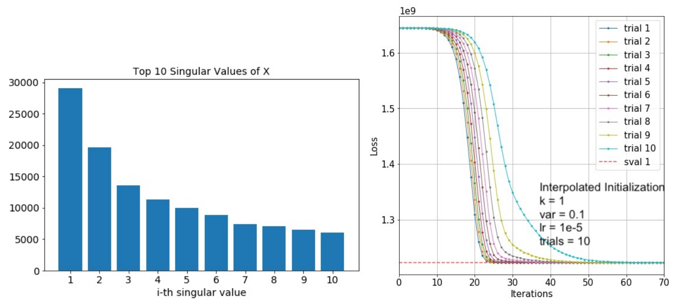

Fig. 5. Left: Top 10 Singular Values of X. Right: (W, S) initialized as an interpolation between (u_1, v_1) and (u_2, v_2). The graph plots the loss v. iterations of 10 GD trials with learning rate = 1e-5, and (W, S) as rank = 1. "sval 1" is the optimal loss with $k = 1$.

absolute imbalance of GD increases rapidly then levels out without decreasing. For balance-regularized GD, imbalance decreases exponentially. This supports our idea that balance regularization aids the rate of convergence of GD by restoring and maintaining the balance of (W_t, S_t). In effect, the increasingly more balanced trajectory encounters only strict saddles with strong descent directions, thus speeding convergence.

D. Capturing Principal Components

An interesting observation of the empirical study concerns how GD captures the principal components of X. In Fig. 2, GD initially decreases the loss slowly because it starts near the saddle point $(\mathbf{0}, \mathbf{0})$. Once it escapes, the loss drops rapidly, then slows near the dotted lines labeled sval 1 and sval 2. At these points the GD trajectory is passing near other strict saddles. The values of the loss at these strict saddles is important: sval 1 is the optimal loss when $k = 1$ and sval 2 is the optimal loss when $k = 2$. This suggests that at these iteration points, GD has captured the first and then the second principal components of X. This trend of capturing singular values in descending order of magnitude is also observed in deep linear networks [16]. However, this trend is not apparent for high order principal components. A possible explanation is that the first few singular values have large differences in consecutive values ($\sigma_1 \gg \sigma_2 \gg \sigma_3$) while the trailing singular values are much closer in value (Fig. 5). Since the first singular value is much larger than the subsequent values, it dominates the attention of GD. After the first two or three components are captured, GD seems to capture the remaining components in parallel as no one singular value dominates the others.

A principal component has two parts: the singular vectors and singular value. To test whether the singular vectors or the values are captured first. we set up 10 trials of GD with $k = 1$ and (W, S) initialized as an interpolation between (u_1, v_1) and (u_2, v_2) with trial 1 closest to the first pair of singular vectors and trial 10 closest to the second pair of singular vectors. All the trials start equidistant from the origin.

The results are shown in the right plot of Fig. 5. There is a direct correlation between how fast GD escapes from the initial saddle point and how close the initialization direction is aligned with the optimal (W, S). Trial 1 converges the

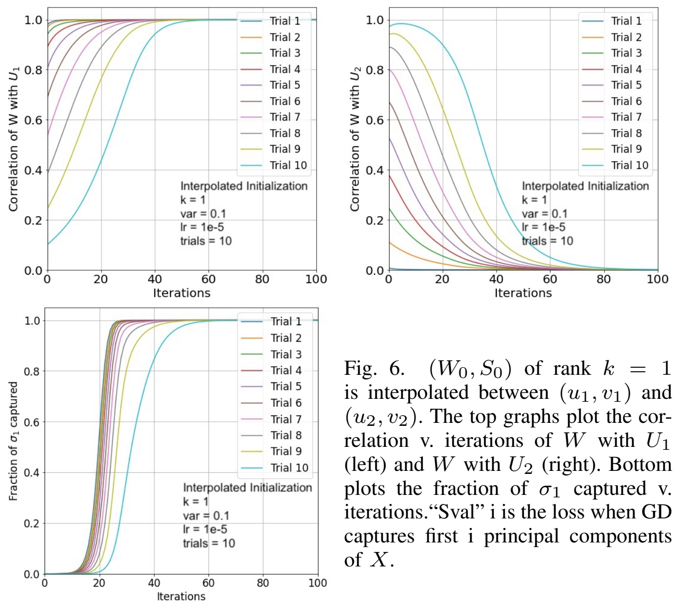

Fig. 6. (W_0, S_0) of rank $k = 1$ is interpolated between (u_1, v_1) and (u_2, v_2). The top graphs plot the correlation v. iterations of W with U_1 (left) and W with U_2 (right). Bottom plots the fraction of σ_1 captured v. iterations."Sval" i is the loss when GD captures first i principal components of X.

fastest and trial 10 the slowest. This suggests that trial 10 is "distracted" by the second principal component and acts to capture it. To test this hypothesis, we plot the correlation $\frac{W^T u_1}{\|W\|}$ and the same quantity for u_2. In Fig. 6, the correlation of W with u_1 in trial 1 reaches 1 within 15 iterations while the correlation in trial 10 takes around 50 iterations to reach 1. On the other hand, the correlation of W with u_2 in trial 1 never exceeds 0.05 while the correlation in trial 10 starts high and increases slightly before slowly dropping. This accords with our hypothesis that trial 10 initially pursues u_2 as the optimal W since σ_2 is large despite being smaller than σ_1 (Fig. 5).

To answer whether GD captures the singular vectors or value first, we compare the correlation with U_1 with fraction of σ_1 captured (Fig. 6). For most trials, the correlation with u_1 reaches 1 before σ_1 is fully captured. This suggests GD is searching for the right direction (singular vectors) before tuning for the right magnitude (singular value).

IV. Conclusion

We have shown the benefits of maintaining balance in the factors (W_t, S_t) of matrix factorization computed by gradient descent. We did so by first showing how to construct unbalanced saddle points with $\lambda_{\min}(\nabla^2 J) < 0$ but arbitrarily close to 0. These points are unbalanced strict saddles for which the best escape direction from the saddle can be made very flat. We then demonstrated the comparatively worse performance of starting or passing near such a strict saddle. However, this poor performance can be ameliorated by adding a balance regularizer to MF's objective function. This exponentially restores balance and hence steers the GD trajectory away from unbalanced flat saddles. We also observed that gradient descent captures large principal components of X sequentially when the separation of consecutive singular values is large. When GD captures a principal component, it also passes one of infinitely many corresponding saddle points. A secondary observation is that when GD captures the leading principal components (e.g., $k = 1$), it first captures the corresponding pair of singular vectors, then the corresponding singular value. More experimentation is needed to to confirm this in general.

The experimental results use the data from a typical subject in a fMRI dataset. We are now working on a multi-subject extension of this work to learn a joint shared response matrix factorization over all 10 subjects. To allow for rapid experimentation for different forms of balance regularizers, we are working with auto-differentiation to automatically compute matrix gradients for numerous complex regularizers and with GD accelerators. The use of GD accelerators may add weight to the need for balance regularizers, since accelerators can significantly exacerbate imbalance.

References

[1] S. Ghosh, "Simple matrix factorization example on the movielens dataset using pyspark." [Online]. Available: https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536

[2] H. Valavi, S. Liu, and P. J. Ramadge, "The landscape of matrix factorization revisited," *arXiv preprint arXiv:2002.12795*, 2020.

[3] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

[4] Y. Chi, Y. M. Lu, and Y. Chen, "Nonconvex optimization meets low-rank matrix factorization: An overview," *CoRR*, vol. abs/1809.09573, 2018. [Online]. Available: http://arxiv.org/abs/1809.09573

[5] A. Anderson, P. K. Douglas, W. T. Kerr, V. S. Haynes, A. L. Yuille, J. Xie, Y. N. Wu, J. A. Brown, and M. S. Cohen, "Non-negative matrix factorization of multimodal mri, fmri and phenotypic data reveals differential changes in default mode subnetworks in adhd," *NeuroImage*, vol. 102, pp. 207–219, 2014.

[6] H. Lyu, D. Needell, and L. Balzano, "Online matrix factorization for markovian data and applications to network dictionary learning," *arXiv preprint arXiv:1911.01931*, 2019.

[7] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge University Press, 2013.

[8] A. K. Cline and I. S. Dhillon, *Computation of the Singular Value Decomposition*. CRC Press, 2006.

[9] S. Arora, N. Cohen, and E. Hazan, "On the optimization of deep networks: Implicit acceleration by overparameterization," *arXiv preprint arXiv:1802.06509*, 2018.

[10] S. S. Du, W. Hu, and J. D. Lee, "Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced," in *Advances in Neural Information Processing Systems*, 2018, pp. 384–395.

[11] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[12] J. V. Haxby, J. S. Guntupalli, A. C. Connolly, Y. O. Halchenko, B. R. Conroy, M. I. Gobbini, M. Hanke, and P. J. Ramadge, "A common, high-dimensional model of the representational space in human ventral temporal cortex," *Neuron*, vol. 72, no. 2, pp. 404–416, 2011.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[15] H. Valavi, S. Liu, and P. Ramadge, "Revisiting the landscape of matrix factorization," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. Online: PMLR, 26–28 Aug 2020, pp. 1629–1638. [Online]. Available: http://proceedings.mlr.press/v108/valavi20a.html

[16] A. K. Lampinen and S. Ganguli, "An analytic theory of generalization dynamics and transfer learning in deep linear networks," 2019.