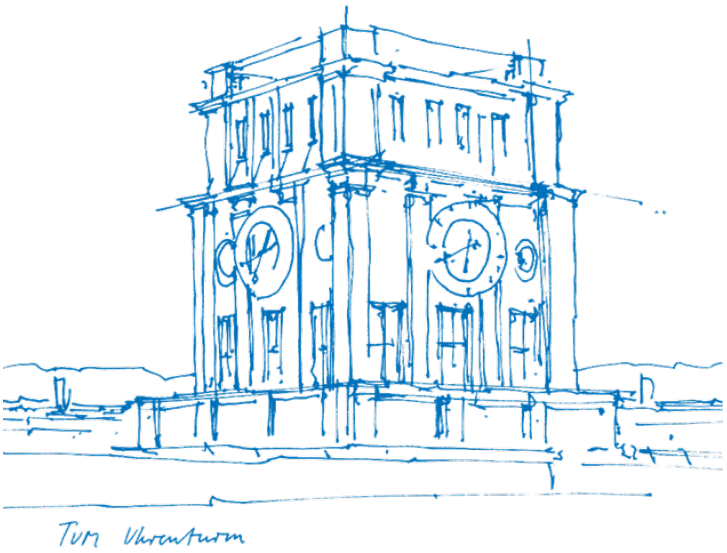
Seminar: Mathematics of Data Science
Department of Mathematics
Technical University of Munich

Gradient Descent for Deep Matrix Factorization

Dynamics and Implicit Bias towards Low Rank

Donata Buožytė



Contents

1 Introduction

The following report is based on the paper *"Gradient Descent for Deep Matrix Factorization: Dynamics and Implicit Bias towards Low Rank"* by H. Chou, C. Gieshoff, J. Maly and H. Rauhut [2].

1.1 Motivation

Numerical experiments, as conducted in for example [4] or [5], imply that models trained with the (stochastic) gradient descent algorithm are likely to have a low training error, but large validation error, which implies an overfitting. In this context overfitting indicates that the trained model rather learned the training samples, which leads to a low training error, than learning to also fit unknown data well, which leads to large errors on for example the validation set. However, models derived from deep neural networks based on a gradient decent optimization with real-world data generalize well, which is contrary to the implication of overfitting. [2]

As suggested in [2] and [5], one of the reasons why the conclusions from theoretical analyses are contrary to possible observations from realistic examples is the implicit bias towards certain solutions introduced by the chosen optimization algorithm.

Hence, in the following report the implicit bias of the (stochastic) gradient descent algorithm towards low rank matrices will be analyzed and discussed.

1.2 Linear Feed-Forward Neural Networks

A feed-forward neural network f of depth N, $N \geq 1$, is defined as a composition of N layers. Each layer $k \in \{1, \dots, N\}$ is defined as

$$g_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}, \quad g_k(x) = \sigma_k(W_k x + b_k)$$

with the weight matrix $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$, the bias $b_k \in \mathbb{R}$ and an activation function $\sigma_k : \mathbb{R} \to \mathbb{R}$, which is in general non-linear and applied element-wise. The bias term b_k is often set to 0 for simplicity or absorbed by the input x and the weight matrix W_k by adding an 1 to the input vector and accordingly the bias term to the weight matrix. [1, 2]

Therefore the overall feed-forward neural network f can be written as

$$f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_N}, \quad f(x) = g_N \circ \ldots \circ g_1(x).$$

A linear feed-forward neural network can be derived by using the identity function as the activation function in each layer and omitting all bias terms, i.e. $\sigma_k(x) = x$ and $b_k = 0$ for $k \in \{1, \dots, N\}$. Therefore

$$\begin{aligned} f_{\text{linear}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_N}, \quad f_{\text{linear}}(x) &= g_N \circ \ldots \circ g_1(x) \\ &= W_N \cdots W_1 \cdot x. \end{aligned}$$

1.3 Loss Minimization and Model Derivation

The goal in supervised machine learning is to derive a model f based on a chosen neural network architecture and some data $\{(x_i, y_i)\}_{i=1}^M$, called *training data*, so that f describes the relationship

$x_i \mapsto y_i$ as good as possible. The final model can then be used to approximate the output values, which are often called *labels*, of new input values $\tilde{x}$. As mentioned in the motivation, another set, which is called *validation set*, should also be provided. This set is similar to the training set, but is used to determine how well the model generalizes, because the model never has the possibility to learn or "memorize" this set. During the training the loss on the validation set is used to determine if an overfitting occurs, which would lead to a rising validation error, and is additionally an indicator how well the model performs on unknown data.

As aforementioned, a neural network is defined by it's layers. Since in general the activation functions are fixed in advance and the bias terms are absorbed or omitted, the only free parameters are the weight matrices. Therefore the optimization problem, that is approached in the case of supervised machine learning, is given by

$$\min_{W_1,\dots,W_N} \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(f(x_i), y_i)$$

where $\mathcal{L} : \mathbb{R}^{n_N} \times \mathbb{R}^{n_N} \to \mathbb{R}_+$ is the *loss function*. This function compares the model output $f(x_i)$ to the according ground truth y_i and is in general chosen based on the neural network architecture and the data. [1, 2]

In the case of linear neural networks f_{linear} the chosen loss function is often the squared L_2-loss $\mathcal{L}(f(x_i), y_i) = \mathcal{L}(f_{\text{linear}}(x_i), y_i) = \|f_{\text{linear}}(x_i) - y_i\|_2^2$, see [2]. Hence the according optimization problem is

$$\min_{W_1,\dots,W_N} \frac{1}{M} \sum_{i=1}^{M} \|W_1 \cdots W_N \cdot x_i - y_i\|_2^2. \tag{1.1}$$

If the training samples $\{x_i\}_{i=1}^M \subset \mathbb{R}^{n_0}$ span the whole input space $\mathbb{R}^{n_0}$, then the optimization problem (1.1) can be reformulated as follows.

$$\min_{W_1,\dots,W_N} \|W_N \cdots W_1 - \hat{W}\|_F^2 \tag{1.2}$$

where $\hat{W}$ is the *ground truth* matrix. The intuition for this reformulation is given by the fact that in this case the goal is to derive a linear map from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_N}$. Hence the multiplication of the weight matrices $W_N \cdots W_1$ should approximate the linear map, which in this case is defined by $\hat{W}$.

The norm used in the problem (1.2) is the *Frobenius* norm. For a matrix $A \in \mathbb{R}^{m \times n}$ with the singular value decomposition $A = U\Sigma V^T$, where $\Sigma_{ii} = \sigma_i$ are the singular values of A, this norm is defined as

$$\|A\|_F^2 = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij})^2} = \sqrt{\sum_{i=1}^{r} \sigma_i^2}.$$

This definition implies that in the following singular value decomposition and especially eigendecompositions will be of special interest.

1.4 Gradient Descent Algorithm

The goal of the gradient descent algorithm is to approximate the value x^*, which minimizes the function of interest h. As the name already implies, this is accomplished by updating the initial guess x_0 iteratively by stirring it slightly into the direction of the steepest descent, which is given by negative gradient of h at the current position.

The gradient descend approach can also be used to derive the solution for the optimization problem (1.2). Here it is important to mention that instead of updating one value, the algorithm will update

every weight matrix $W_1, \dots, W_N$ in each iteration step. The function of interest h is defined as the loss function

$$\mathcal{L} : \mathbb{R}^{n_N \times n_0} \times \mathbb{R}^{n_N \times n_0} \to \mathbb{R}_+, \quad \mathcal{L}(W) = \frac{1}{2}\|W - \hat{W}\|_F^2 \tag{1.3}$$

with the derivative

$$\nabla_W \mathcal{L}(W) = (W - \hat{W})$$

where $\hat{W}$ denotes the ground truth matrix. If the neural network consists of more than one layer, the matrix W will be defined by a multiplication of matrices, i.e. $W = W_N \cdots W_1$. Hence why the derivatives with respect to each weight matrix W_i have to be computed, too.

$$\nabla_{W_i} \mathcal{L}(W) = (W_N \dots W_{i+1})^T \nabla_W \mathcal{L}(W)(W_{i-1} \dots W_1)^T$$

Based on the loss function and its derivatives the gradient descent algorithm can be defined as follows. [2]

input : learning rate η, initialization matrices $W_{0,i}$ and weights α_i for $i \in \{1, \dots, N\}$
output: set of weight matrices $\{W_i^{(k)}\}_{i=1}^N$ for each iteration step k

Initialize every weight matrix: $W_i^{(0)} = \alpha_i W_{0,i} \quad \forall i \in \{1, \dots, N\}$;
for $k \in \{1, 2, \dots\}$ **do**
  $W^{(k)} = W_N^{(k)} \cdots W_1^{(k)}$;
  for $i \in \{1, \dots, N\}$ **do**
    $W_i^{(k+1)} = W_i^{(k)} - \eta \nabla_{W_i} \mathcal{L}(W^{(k)})$
  end
end

Algorithm 1: Gradient descent algorithm

It should be noted, that to implement this algorithm a termination condition has to be specified. Possible conditions are:

1. Define a certain number of steps K that the Algorithm 1 has to take, i.e. $k \in \{1, 2, \dots, K\}$.
2. Terminate as soon as $W^{(k)}$ is close enough to the ground truth matrix. Here, different interpretations of closeness, as for example $\mathcal{L}(W) = \frac{1}{2}\|W - \hat{W}\|_F^2 < \epsilon$ for an $\epsilon > 0$, can be used.
3. Terminate if the gradient $\nabla_W \mathcal{L}(W^{(k)})$ reaches 0 or is within a certain radius ϵ around 0.

Other possible termination conditions will be discussed in the following report.

2 Dynamics of the Gradient Descent with an Identical Initialization

In this chapter the dynamics of the gradient descent algorithm, which is defined in Algorithm 1, with an identical initialization is derived. Therefore the weights α_i and initialization matrices $W_{0,i}$ are given by

$$\alpha_i := \alpha, \quad W_{0,i} := W_0 = I \qquad \forall i \in \{1, \dots, N\} \tag{2.1}$$

where I represents the identity matrix. [2]

With this initialization the following Lemma can be stated.

Lemma 2.1 ([2, 3]) *Let $(W_i^{(k)})_{i=1}^N$ be the solutions at each iteration step k of Algorithm 1 with the identical initialization* (2.1) *and the loss function* (1.3) *and let $\hat{W} = V\Lambda V^T$ be the eigendecomposition of the symmetric ground truth matrix $\hat{W}$. Then for each step k the following statements hold:*

$$D^{(k)} = D_i^{(k)} := V^T W_i^{(k)} V \text{ are real, diagonal, identical for each } i \in \{1, \dots, N\} \tag{2.2}$$

$$D^{(k)} \text{ follows the dynamics } D^{(k+1)} = D^{(k)} - \eta (D^{(k)})^{N-1}((D^{(k)})^N - \Lambda) \tag{2.3}$$

An intuition why in each iteration the term $V^T W_i^{(k)} V$ results in a diagonal matrix are the chosen initial matrices, which are weighted identity matrices, and the fact that V is a unitary matrix by definition, hence

$$V^T \alpha I V = \alpha V^T V = \alpha I.$$

Additionally, the updates by the gradient descent consist of the weight matrices and the ground truth matrix, which can all be diagonalized by the unitary matrix V.

Rearranging the matrices in (2.2) leads to the equation $W_i^{(k)} = V D^{(k)} V^T$ for each weight matrix W_i at step k. Using that the matrix V consists the eigenvectors of the matrix $\hat{W}$ and that the matrix $D^{(k)}$ is always diagonal, it can be concluded that this equation can be interpreted as the eigendecomposition of each weight matrix at the iteration step k. Based on this result the eigendecomposition of the multiplication of the weight matrices is

$$W^{(k)} = W_N^{(k)} \cdots W_1^{(k)} = (V D^{(k)} V^T)^N = V (D^{(k)})^N V^T \quad \forall k \tag{2.4}$$

As the matrices $D^{(k)}$ are diagonal, raising them to the power of any integer is equivalent to applying this raise to each diagonal element of the matrix. Combining this with the diagonality of the matrix Λ, the dynamics in (2.3) can be simplified to

$$d_{ii}^{(k+1)} = d_{ii}^{(k)} - \eta (d_{ii}^{(k)})^{N-1}((d_{ii}^{(k)})^N - \lambda_i) \tag{2.5}$$

where $d_{ii}^{(k)}$ is the i-th diagonal element of $D^{(k)}$ and λ_i the according eigenvalue of the ground truth matrix $\hat{W}$. With this equation the dynamics of each diagonal element can be analyzed separately.

Theorem 2.1 ([2]) *Let $N \geq 2$, $\lambda \in \mathbb{R}$, $\alpha > 0$ and $\{d^{(k)}\}_k$ be the solution of an arbitrary diagonal element with the dynamics defined in* (2.5)*.*
Let $M = \max(\alpha, |\lambda|^{\frac{1}{N}})$ and the learning rate η be s.t.

$$0 < \eta < \begin{cases} \frac{1}{2NM^{2N-2}} & \text{if } \lambda \geq 0 \\ \frac{1}{(3N-2)M^{2N-2}} & \text{if } \lambda < 0. \end{cases}$$

Additionally, let $\epsilon \in (0, |\alpha - \lambda_+^{\frac{1}{N}}|)$, where $\lambda_+ \coloneqq \max\{0, \lambda\}$, be the desired error and $T = \min\{k : |d^{(k)} - \lambda_+^{\frac{1}{N}}| \leq \epsilon\}$ the minimal number of steps needed to achieve this desired error.
Then T has a finite upper bound: $T \leq T_N^{\mathrm{Id}}(\lambda, \epsilon, \alpha, \eta)$.

The finite upper bound $T_N^{\mathrm{Id}}(\lambda, \epsilon, \alpha, \eta)$ used in the Theorem 2.1 is defined as

$$T_N^{\mathrm{Id}}(\lambda, \epsilon, \alpha, \eta) \coloneqq \begin{cases} \frac{1}{\eta|\lambda|} T_N^-(\epsilon, \alpha) & \text{if } \lambda < 0 \\ \frac{1}{\eta} T_N^+(\lambda, \lambda^{\frac{1}{N}} + \epsilon, \alpha) & \text{if } 0 \leq \lambda < \alpha^N \\ \frac{\ln((\lambda^{\frac{1}{N}} - \alpha)/\epsilon)}{|\ln(1 - \eta N (c_N \lambda)^{2 - \frac{2}{N}}|} & \text{if } 0 \leq c_N \lambda < \alpha^N \leq \lambda \\ \frac{1}{\eta} T_N^+(\lambda, (c_N\lambda)^{\frac{1}{N}}, \alpha) + s_N(\lambda, \alpha) + \frac{\ln(\lambda^{\frac{1}{N}}/\epsilon) - a_N}{|\ln(1 - \eta N (c_N \lambda)^{2 - \frac{2}{N}}|} & \text{if } \alpha^N \leq c_N \lambda,\ \epsilon < (1 - c_N^{\frac{1}{N}})\lambda^{\frac{1}{N}} \\ \frac{1}{\eta} T_N^+(\lambda, \lambda^{\frac{1}{N}} - \epsilon, \alpha) + s_N(\lambda, \alpha) & \text{if } \alpha^N \leq c_N \lambda,\ \epsilon \geq (1 - c_N^{\frac{1}{N}})\lambda^{\frac{1}{N}} \end{cases} \tag{2.6}$$

with

$$c_N \coloneqq \frac{N-1}{2N-1}, \quad s_N(\lambda, \alpha) \coloneqq \left\lceil c_N^{1 - \frac{1}{N}} \left(\frac{\lambda^{\frac{1}{N}}}{\alpha} \right)^{N-1} \right\rceil,$$
$$a_N \coloneqq |\ln(1 - c_N^{\frac{1}{N}})|, \quad b_N \coloneqq |\ln(\frac{1}{2c_N} - c_N^{\frac{1}{N}})|.$$

The further definitions of $T_N^-(\alpha, \beta)$ and $T_N^+(\alpha, \beta, \gamma)$ can be found in [2, Equations (21)]. Overall the formula (2.6) estimates the number of steps in the gradient descend needed to reach the error ϵ defined in Theorem 2.1.

As the value λ in Theorem 2.1 can be chosen arbitrarily, it can also be interpreted as an eigenvalue of the ground truth matrix $\hat{W}$.

Hence, first assume than λ is an arbitrary negative eigenvalue $\lambda_i^{(-)}$ of $\hat{W}$. Then the according diagonal element $d_{ii}^{(k)}$ of the diagonal matrix $D^{(k)}$ will approximate the value $(\lambda_i^{(-)})_+^{\frac{1}{N}} = \max\{0, (\lambda_i^{(-)})^{\frac{1}{N}}\} = 0$ under the condition stated in Theorem 2.1. Now assume than λ is an arbitrary non-negative eigenvalue $\lambda_i^{(+)}$ of $\hat{W}$. Then $d_{ii}^{(k)}$ will approximate the value $(\lambda_i^{(+)})_+^{\frac{1}{N}} = \max\{0, (\lambda_i^{(+)})^{\frac{1}{N}}\} = (\lambda_i^{(+)})^{\frac{1}{N}}$ under the same conditions as aforementioned.

By combining the results above and using the result (2.4) it can be concluded that the eigenvalues $\tilde{\lambda}_i$ of the matrix $W^{(k)} = W_N^{(k)} \cdots W_1^{(k)}$ converge for $k \to \infty$ to the following values.

$$\tilde{\lambda}_i = \begin{cases} (0^{\frac{1}{N}})^N = 0 & \text{if } \lambda_i < 0 \\ (\lambda_i^{\frac{1}{N}})^N = \lambda_i & \text{if } \lambda_i \geq 0. \end{cases}$$

where λ_i is a eigenvalue of the ground truth matrix $\hat{W}$.

This concludes the following theorem.

Theorem 2.2 (Recovery of Positive Eigenvalues, [2]) *Let $N \geq 2$ and $\hat{W} = V \Lambda V^T \in \mathbb{R}^{n \times n}$ be the eigendecomposition of the symmetric ground truth matrix $\hat{W}$.*

Let $W^{(k)} = W_N^{(k)} \cdots W_1^{(k)}$, where each $W_j^{(k)}$ is defined by the Algorithm 1 with the identical initialization (2.1) *and the loss function* (1.3)*, $W_0 = I$ and $\alpha > 0$.*
Let $M = \max(\alpha, \|\hat{W}\|_2^{\frac{1}{N}})$ and the learning rate η be s.t.

$$0 < \eta < \frac{1}{(3N-2)M^{2N-2}}. \tag{2.7}$$

Then: $\lim_{k\to\infty} W^{(k)} = V\Lambda_+ V^T$, where Λ_+ is given by $(\Lambda_+)_{ij} := \max\{0, (\Lambda_+)_{ij}\}$, and the error is defined by the diagonal matrix $E^{(k)} = V^T W^{(k)} V - \Lambda_+$.

It should be noted that by the Theorem 2.1 theoretically also the eigenvalue 0 can be recovered. But as the negative eigenvalues of the ground truth matrix are approximated with 0 as well, the diagonal element representing the "recovered eigenvalue 0" can't be identified.

The Theorem 2.2 establishes the first central observation of [2]:

> The gradient descent algorithm defined by the Algorithm 1 with the identical initialization defined in (2.1) can recover all non-negative eigenvalues of the ground truth matrix $\hat{W}$ under certain conditions.

3 Dynamics of the Gradient Descent with a Perturbed Initialization

In this chapter the dynamics of the gradient descent algorithm, which is defined in Algorithm 1, with an perturbed initialization, which is based on a perturbation of the identical initialization, is derived. Therefore the weights α_i and initialization matrices $W_{0,i}$ are given as

$$\alpha_1 \coloneqq (\alpha - \beta), \quad \alpha_i \coloneqq \alpha \quad \forall i \in \{2, \dots, N\}, \quad W_{0,i} \coloneqq W_0 = I \quad \forall i \in \{1, \dots, N\} \tag{3.1}$$

where I represents the identity matrix and $0 < \beta < \alpha$. As stated in [2], the perturbation can be w.l.o.g. applied on the first weight matrix instead of an arbitrary weight matrix.

Similarly as in the previous chapter, a lemma regarding diagonal matrices based on the eigenvectors of the ground truth matrix and the weight matrices can be stated.

Lemma 3.1 ([2, 3]) *Let $N \geq 2$, $(W_i^{(k)})_{i=1}^N$ be the solutions at each iteration step k of Algorithm 1 with the perturbed initialization* (3.1) *and the loss function* (1.3) *and let $\hat{W} = V\Lambda V^T$ be the eigendecomposition of the symmetric ground truth matrix $\hat{W}$. Then for each step k the following statements hold:*

$$D_i^{(k)} \coloneqq V^T W_i^{(k)} V \text{ are real, diagonal for each } i \in \{1, \dots, N\} \tag{3.2}$$

where the diagonal matrices $D_i^{(k)}$ follow the dynamics

$$\begin{aligned} D_1^{(k+1)} &= D_1^{(k)} - \eta (D_2^{(k)})^{N-1} (D_1^{(k)} (D_2^{(k)})^{N-1} - \Lambda), \\ D_i^{(k+1)} &= D_2^{(k)} - \eta D_1^{(k)} (D_2^{(k)})^{N-2} (D_1^{(k)} (D_2^{(k)})^{N-1} - \Lambda) \quad \forall i \in \{2, \dots, N\} \end{aligned} \tag{3.3}$$

Again, the equation (3.3) concludes the eigendecomposition $W_i^{(k)} = V D_i^{(k)} V^T$ for each weight matrix and therefore

$$W^{(k)} = W_N^{(k)} \cdots W_1^{(k)} = \Pi_{i=1}^N (V D_i^{(k)} V^T) = V (\Pi_{i=1}^N D_i^{(k)}) V^T \quad \forall k. \tag{3.4}$$

As the matrices $D_i^{(k)}$ are still diagonal, the dynamics of the matrices in (3.3) can be reduced to dynamics of single elements.

$$\begin{aligned} d_1^{(k+1)} &= d_1^{(k)} - \eta (d_2^{(k)})^{N-1} (d_1^{(k)} (d_2^{(k)})^{N-1} - \lambda_1), \\ d_i^{(k+1)} &= d_2^{(k)} - \eta d_1^{(k)} (d_2^{(k)})^{N-2} (d_1^{(k)} (d_2^{(k)})^{N-1} - \lambda_i) \quad \forall i \in \{2, \dots, N\} \end{aligned} \tag{3.5}$$

Based on these element-wise dynamics a lemma regarding their convergence can be derived.

Lemma 3.2 ([2]) *Let $N \geq 2$, $\lambda \in \mathbb{R}$, $0 < \beta < \alpha$ and $\{d_i^{(k)}\}_{i=1}^N$ be the solution of the dynamics defined in* (3.5)*.*
Let $M = \max(\alpha, |\lambda|^{\frac{1}{N}})$, $c \in (1, 2)$ be the maximal real solution of the equation $1 = (c-1)c^{N-1}$ and the learning rate η be s.t.

$$0 < \eta < \frac{1}{9N(cM)^{2N-2}}.$$

Then $\lim_{k \to \infty} d_1^{(k)} (d_i^{(k)})^{N-1} = \lambda$ for any $i \in \{2, \dots, N\}$.

This lemma and especially the proof of it, which can be found in [2, Proof of Lemma 2.11], suggest that there are two types of dynamics for the diagonal elements. If the chosen value λ is positive, then the dynamics behave in a similar way as in the case of the identical initialization. Here the value stays at it's initial value and starts to increase and approximate λ after a certain number of steps. If however λ is negative, then the dynamics are at first similar to those in the positive case. They only start to differ when a diagonal element changes it's sign and takes a value below 0. After this point the dynamics are again similar as in the first type, except the values decrease instead of increase to approximate λ.

By combining the Lemma 3.2 and the eigendecomposition derived in (3.4), the following theorem can be deduced.

Theorem 3.1 (Recovery of Arbitrary Eigenvalues, [2]) *Let $N \geq 2$ and $\hat{W} = V\Lambda V^T \in \mathbb{R}^{n\times n}$ be the eigendecomposition of the symmetric ground truth matrix $\hat{W}$.*
Let $W^{(k)} = W_N^{(k)} \cdots W_1^{(k)}$, where each $W_j^{(k)}$ is defined by the Algorithm 1 with the perturbed initialization (3.1) *and the loss function* (1.3) *and $W_0 = I$.*
Let $M = \max(\alpha, \|\hat{W}\|_2^{\frac{1}{N}})$, $0 < \frac{\beta}{c-1} < \alpha$, $c \in (1,2)$ with c being the maximal real solution of $1 = (c-1)c^{N-1}$ and the learning rate η be s.t.

$$0 < \eta < \frac{1}{9N(cM)^{2N-2}}. \tag{3.6}$$

Then: $\lim_{k\to\infty} W^{(k)} = V\Lambda V^T = \hat{W}$ and the error is defined by the diagonal matrix $E^{(k)} = V^T W^{(k)} V - \Lambda$.

The Theorem 3.1 establishes the second central observation of [2]:

> The gradient descent algorithm defined by the Algorithm 1 with the perturbed initialization defined in (3.1) can recover all eigenvalues of the ground truth matrix $\hat{W}$ under certain conditions.

4 Implicit Bias of the Gradient Descent Algorithm

The previously derived Theorems 2.2 and 3.1 establish that the gradient descent algorithm is able to recover certain or even all eigenvalues of the ground truth matrix when using certain intializations. Additionally, more dominant eigenvalues are recovered in less iterations than less dominant eigenvalues, where the dominance of an eigenvalue is influenced by the sign and its absolute value. When examining for example small positive eigenvalues and their negative counterparts, then the positive values are likely recovered faster. Compared to however a negative value with large absolute value, the small positive values could be recovered slower.

Considering the PCA approach for dimensionality reduction, as described in for example [6], then one of the key ideas is to compute the k largest eigenvalues and the corresponding eigenvectors of the covariance matrix. These eigenvectors span a k-dimensional space and reduce the dimensionality of the representation as the data can be projected onto this subspace.

Due to the order, in which the eigenvalues of the ground truth matrix are recovered, the same idea can be applied to the gradient descent algorithm. Stopping the algorithm after a certain number of steps n_k will result in an overall weight matrix $W^{(k)} = W_N^{(k)} \cdots W_1^{(k)}$, which has the same eigenvectors and the same k most dominant eigenvalues as $\hat{W}$. The remaining eigenvalues represent the unrecovered eigenvalues and are therefore 0 or nearly 0, depending on the initialization. Hence the rank of $W^{(k)}$ is definitely lower than the rank of $\hat{W}$ and $W^{(k)}$ can be considered a low rank approximation of the ground truth matrix.

In the paper [2] it is additionally suggested, that if the effective rank of the overall weight matrix is considered, then these dynamics exhibit an interesting behaviour. The effective rank drops at first to 1, plateaus at this level and then increases monotonically, while plateauing at effective ranks, which correspond to the effective ranks of certain low rank approximations of the ground truth matrix. This result can also be used to derive a new stopping criterion for the gradient descent algorithm, in case the computation of the effective rank of the overall weight matrix is simpler than the computation of its eigenvalues. One could for example determine when the effective rank plateaus and the iteration can be stopped as soon as k plateaus were reached. The resulting matrix is then a low rank approximation of rank $\approx k$.

This establishes the third central observation of [2]:

> The gradient descent algorithms has an implicit bias towards low rank approximations of the ground truth matrix $\hat{W}$.

4.1 Effective Rank

The effective rank of a matrix $A \in \mathbb{R}^{n \times m}$ is defined as follows. [2, 7]

$$1 \leq \operatorname{rank}_{\text{effective}}(A) \coloneqq \frac{\|A\|_*}{\|A\|_2} = \frac{\sum_i \sigma(A)_i}{\|A\|_2} \leq \operatorname{rank}(A)$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\|\cdot\|_2$ denotes the spectral norm and $\sigma(A)_i$ corresponds to the i-th singular value of A.

In contrast to the normal matrix rank, which describes the dimension of the space spanned by it's columns, the effective rank describes the "true" rank of a matrix by including the geometry of the space spanned by its columns.

Example:

for $A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$ the rank and the effective rank are

$$\operatorname{rank}(A) = 3 \neq 1.4 = \operatorname{rank}_{\text{effective}}(A).$$

5 Numerical Results

In this chapter the dynamics of the gradient descent algorithm with the two different initializations, which are discussed in the previous chapters, are analyzed based on two different examples.

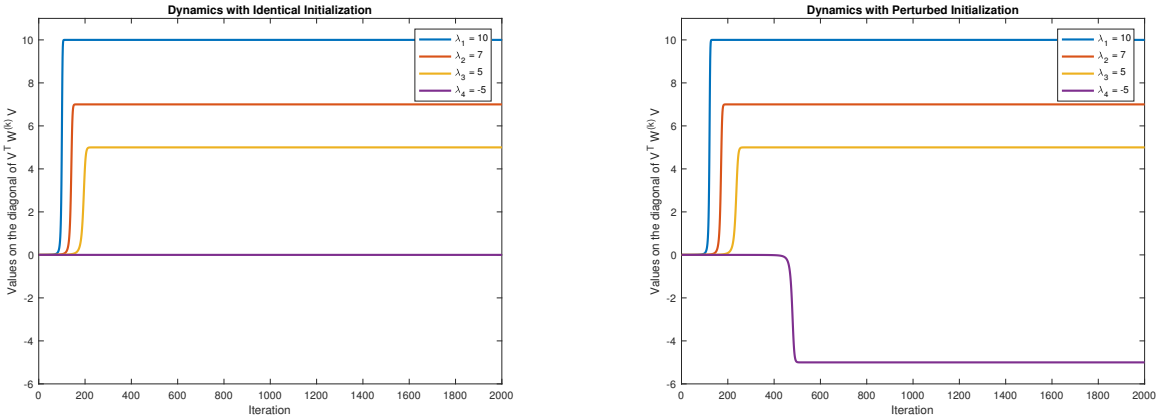
5.1 Recovery of Eigenvalues

The first example is based on the following ground truth matrix $\hat{W}$.

$$\hat{W} = \begin{pmatrix} 7/3 & 10/3 & -4 & 2/3 \\ 10/3 & 10/3 & 5 & 5/3 \\ -4 & 5 & 4 & 1 \\ 2/3 & 5/3 & 1 & -5 \end{pmatrix}; \quad \Lambda = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & -5 \end{pmatrix}; \quad V = (1/\sqrt{3}) \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix} \tag{5.1}$$

where $V\Lambda V^T$ is the eigendecomposition of $\hat{W}$. The neural network has the depth $N = 3$, while the learning rate η is 0.001. The parameters initializing the weight matrices are $\alpha = 0.1$ and $\beta = 0.05$.

The Figure 5.1 visualize how the chosen neural network recovers the respective eigenvalues in a descending order with respect to the dominance of the eigenvalues. As previously derived, if an identical initialization is used, then only non-negative eigenvalues can be recovered, see Figure 5.1a.



(a) Dynamics with an Identical Initialization

(b) Dynamics with a Perturbed Initialization

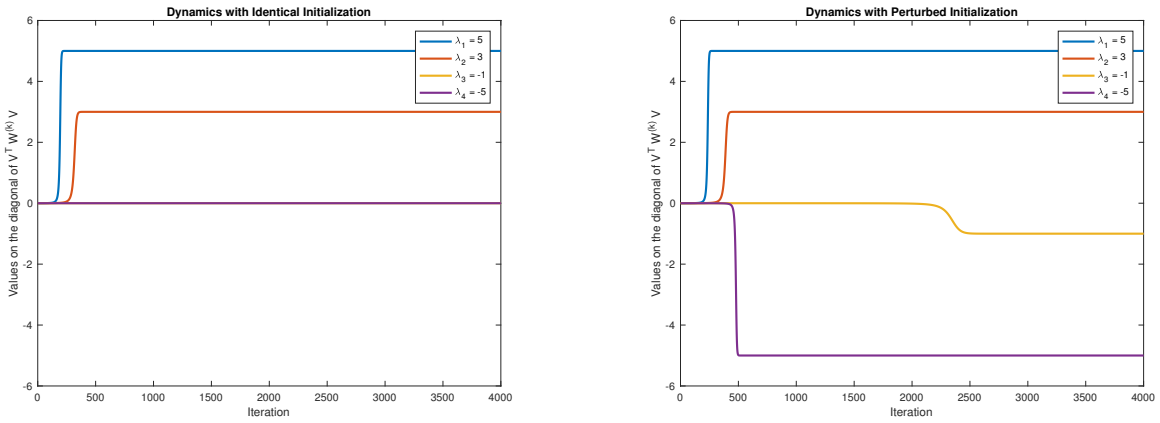
Figure 5.1 Example 1 - Dynamics of the gradient descent with different initializations. Each line represents one value of the diagonal matrix $V^T W^{(k)} V$ at each iteration step k.

In the second example the ground truth matrix $\hat{W}$ is defined by

$$\hat{W} = \frac{1}{3} \begin{pmatrix} -3 & 4 & -8 & 4 \\ 4 & -1 & 10 & 6 \\ -8 & 10 & 3 & 2 \\ 4 & 6 & 2 & 7 \end{pmatrix}; \quad \Lambda = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -5 \end{pmatrix}; \quad V = (1/\sqrt{3}) \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix} \tag{5.2}$$

where $V\Lambda V^T$ is the eigendecomposition of $\hat{W}$. The neural network and further parameters are defined as in the first example.

The dynamics of the gradient descent for this example can be seen in the Figure 5.2. Again, the previously derived recovery of eigenvalues can be observed in both cases.



(a) Dynamics with an Identical Initialization

(b) Dynamics with a Perturbed Initialization

Figure 5.2 Example 2 - Dynamics of the gradient descent with different initializations. Each line represents one value of the diagonal matrix $V^T W^{(k)} V$ at each iteration step k.

5.2 Recovery of an Eigenvalue depending on the Network Depth

In the previous two examples only one network architecture and therefore only one depth was considered. In the Figure 5.3 the dynamics of the recovery of the eigenvalue $\lambda = 7$, based on the first example presented above and networks with different depths, are displayed.

Both Figures 5.3a and 5.3b show that the depth of the network has a significant impact on the number of steps needed to recover an eigenvalue. As real-world scenarios are in general vastly more complex than the examples considered in this report, more complex and therefore deeper network architectures are used. Considering the significant increase of steps needed to recover one eigenvalue based on a networks of depth 4 and of depth 5, it can be concluded that for deep neural networks an early-stopping criterion for the gradient descent algorithm is needed, although the whole ground truth matrix could be recovered in theory.

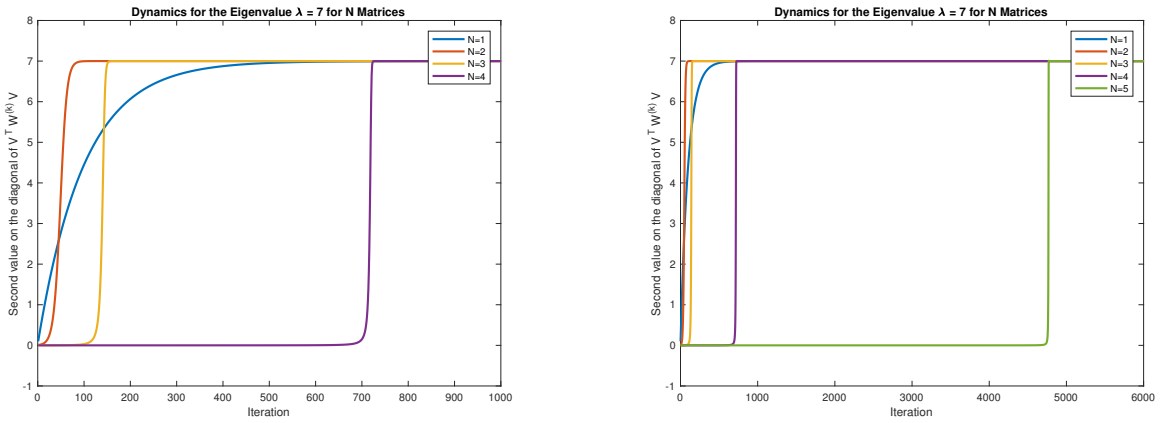
Another difference in the dynamics can be observed in the Figure 5.3a. Comparing the recovery for the network of depth 2 with the one of depth 4 shows that, although the recovery starts sooner if the network has less layers, the number of steps from the start of the recovery, i.e. from where the eigenvalue starts to differ from 0, until the value is fully recovered declines with the depth.

In this context, one should always keep in mind that deriving a final model does not only consist of training the network once, but includes a hyperparameter training or tuning. Hyperparameters are parameters such as the number of maximal steps, the depth of the network or the learning rate, which are updated if a model does not perform as expected. To tune them the network is in general trained multiple times with different hyperparameter settings to find the combination, that leads to for example the smallest approximation error. Therefore a smaller number of iterations per training is desirable, to reduce the overall time needed to derive the final model. Together with the complexity in real-world scenarios the need to restrict the iteration steps and therefore the number of recovered eigenvalues is emphasised once more.

5.3 Dynamics of the Effective Rank

Based on the first example, which is defined by the matrices introduced in (5.2), the effective rank of each approximation matrix $W^{(k)}$ develops as can be seen in the Figure 5.4.

Primarily, the aforementioned behaviour regarding the drop of the effective rank to 1 with the following step-wise increase and plateaus can be observed. By comparing the iteration steps where a



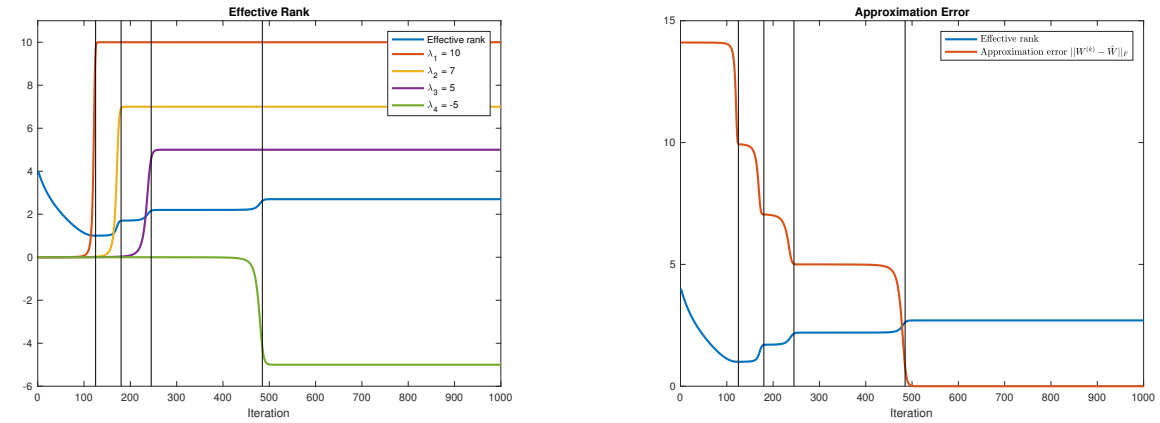
(a) Dynamics for Networks of Depth $N = 1, 2, 3, 4$ **(b)** Dynamics for Networks of Depth $N = 1, 2, 3, 4, 5$

Figure 5.3 Example 1 - Dynamics of the gradient descent with a perturbed initialization of one diagonal element of the diagonal matrix $V^T W^{(k)} V$ at each iteration step k. The networks used to approximate the ground truth matrix have the depths $N = 1, 2, 3, 4, 5$.

plateau is reached to the recovery of eigenvalues, it can be concluded that the effective rank plateaus when an eigenvalue is fully recovered and increases again when the next eigenvalue starts to be recovered. Therefore stopping the gradient descent algorithm as soon as the effective rank reached a certain number n of plateaus is similar or even equivalent to only recovering the n most dominant eigenvalues, see Figure 5.4a.

Comparing the effective rank to the approximation error $\|W^{(k)} - \hat{W}\|_F$, which in this setting describes how well the algorithm was able to recover the eigenvalues of the ground truth matrix $\hat{W}$, shows that an increase of the effective rank correlates with a drop or decrease of the error, see Figure 5.4b.

As the plateaus of the rank and the error are located at similar iteration steps, it can be concluded that if a certain number of eigenvalues is recovered, i.e. if a certain plateau in the effective rank is reached, making further steps in the gradient descent algorithm won't improve the approximation error significantly. To reach the next significant improvement, the next eigenvalue has to be recovered, which implies more iteration steps.



(a) Effective rank compared to the eigenvalue recovery **(b)** Effective rank compared to the approximation error

Figure 5.4 Example 1 - Effective rank of the iteratively computed matrix $W^{(k)}$ compared to the dynamics of the eigenvalue recovery and the dynamics of the approximation error.

This result combined with the observations in the Figures in 5.1 and 5.2 strengthen the results stated in the previous chapter regarding the implicit bias of the gradient descent algorithm. Additionally, the Figures in 5.3 emphasise the need of an early-stopping criterion, such as the recovery of a certain number of eigenvalues or the effective rank, as otherwise the runtime could explode.

6 Summary and Discussion

The paper [2] is able to state the following three significant conclusions, which are supported by proofs and numerical results.

1. The gradient descent algorithm defined by the Algorithm 1 with the identical initialization defined in (2.1) can recover all non-negative eigenvalues of the ground truth matrix $\hat{W}$ under certain conditions.
2. The gradient descent algorithm defined by the Algorithm 1 with the perturbed initialization defined in (3.1) can recover all eigenvalues of the ground truth matrix $\hat{W}$ under certain conditions.
3. The gradient descent algorithms has an implicit bias towards low rank approximations of the ground truth matrix $\hat{W}$.

Especially the result regarding the similarity of the eigendecomposition of the ground truth matrix to the eigendecompositions of the weight matrices at each iteration step is not intuitive, but very useful. It is additionally remarkable that the dynamics of the recovery of each eigenvalue are seemingly smooth and stable, as can be seen in the numerical results. Lastly, the inferred implicit bias of the gradient descent is very helpful to gain a better understanding on the reason why the resulting models are performing well on real data, which is contrary to theoretical studies.

However, especially the number and complexity of the constraints could lead to some problems with the generalization of the results.

Considering for example the constraints on the ground truth matrix and the initial weight matrices lead to the following conclusions. On the one hand the dimensions of the ground truth matrix are restricted to $n \times n$, i.e. quadratic matrices, due to the symmetry condition. This results in neural networks, where the input dimension is the same as the output dimension. Therefore for example architectures used for general classifications, where the output space is restricted to a small number of classes, such as "cats" and "dogs", while the inputs might be pictures with significantly more pixels than the number of classes, can't be considered here. On the other hand the choice for the initial weight matrices is very restricted as only the multiplicative constants α and β can be changed. Hence the basic diagonal structure of the initial matrices cannot be changed. But the choice of the initial value used for the optimization algorithm is often considered as very crucial as it can heavily influence the results. Based on the currently proven theorems, only two types of initializations can be used. Even perturbing more than one weight matrix can for example affect the stability of the recovery of the eigenvalues negatively. Therefore possible knowledge about good initializations or random initializations, which are often used, cannot be applied without further analysis of the settings discussed in this report.

It should be mentioned that in the numerical simulations in [2] the restricting constraints on the ground truth matrix and the initializations are dropped. The presented results show that the derived conclusions still hold in this case and imply that these constraints could be weakened or even dropped completely under certain circumstances. Nevertheless the stability of the recovery has to be discussed when for example the symmetry condition is dropped. During the numerical experiments concerning this report different conditions were violated to observe the behaviour of the gradient descent algorithm in those cases. Violating the symmetry of the ground truth matrix for example led to instabilities in the recovery of the eigenvalues. In that case all eigenvalues were fully recovered, even

with an identical initialization, but certain recovered eigenvalues started to oscillate for a few thousand steps before approximating again the previously recovered eigenvalue.

Another aspect that has to be examined is that the three main conclusions are derived for very specific neural network architectures, as introduced in the first chapter. The fact that the network is linear and that therefore the weight matrices can be condensed to an "overall weight matrix" is a key feature used in the lemmas, theorems and proofs. Hence one might question, how these results could be generalized to also hold for neural networks with non-linear activation functions, which are used in real-world applications.

Since the choice of the loss function heavily controls the dynamics of the gradient descent, it could be of interest to analyse how the choice of the loss function influences the dynamics and especially the existence of the diagonal matrices, which represent the matrices containing the eigenvalues of the weight matrices.

Although there are many open questions, such as those mentioned above or in the discussion in [2], the results leading to the implicit bias of the gradient descent toward low rank approximations of the ground truth matrix provide a good intuition on how this bias develops. Furthermore the main two theorems regarding the recovery of positive and arbitrary eigenvalues provide first results, which can be possibly used to conclude the implicit bias for more general settings.

Bibliography

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Hung-Hsu Chou, Carsten Gieshoff, Johannes Maly, and Holger Rauhut. Gradient descent for deep matrix factorization: Dynamics and implicit bias towards low rank. *arXiv preprint arXiv:2011.13772*, 2020.

[3] Daniel Gissin, Shai Shalev-Shwartz, and Amit Daniely. The implicit bias of depth: How incremental learning drives generalization. *arXiv preprint arXiv:1909.12051*, 2019.

[4] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[5] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[6] G Thippa Reddy, M Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8:54776–54788, 2020.

[7] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.