CBMM Memo No. 144 **December 3, 2023**

The Janus effects of SGD vs GD: high noise and low rank

Mengjia Xu[1,2], **Tomer Galanti**[1], **Akshay Rangamani**[1], **Lorenzo Rosasco**[1,3,4], **Tomaso Poggio**[1,*]

[1]Center for Brains, Minds, and Machines, Massachusetts Institute of Technology, Cambridge, MA, USA
[2]Department of Data Science, New Jersey Institute of Technology, Newark, NJ, USA
[3]MaLGaCenter - DIBRIS - Universit‘a di Genova, Genoa, Italy
[4]MaLGaCenter - DIMA - Universit‘a di Genova, Genoa, Italy

Abstract

It was always obvious that SGD has higher fluctuations at convergence than GD. It has also been often reported that SGD in deep RELU networks has a low-rank bias in the weight matrices. A recent theoretical analysis linked SGD noise with the low-rank bias induced by the SGD updates associated with small minibatch sizes [1]. In this paper, we provide an empirical and theoretical analysis of the convergence of SGD vs GD, first for deep RELU networks and then for the case of linear regression, where sharper estimates can be obtained and which is of independent interest. In the linear case, we prove that the components of the matrix W corresponding to the null space of the data matrix X converges to zero for both SGD and GD, provided the regularization term is non-zero (in the case of square loss; for exponential loss the result holds independently of regularization). The convergence rate, however, is exponential for SGD, and linear for GD. Thus SGD has a much stronger bias than GD towards solutions for weight matrices W with high fluctuations and low rank, provided the initialization is from a random matrix (but not if W is initialized as a zero matrix). Thus SGD under exponential loss, or under the square loss with non-zero regularization, shows the coupled phenomenon of low rank and asymptotic noise.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

1 Introduction

Over the past few years, deep neural networks have challenged machine learning theory with several puzzles. One of them is the role and properties of minibatch SGD vs GD. It seems generally accepted that, apart from computational advantages, SGD is similar to GD in its basic properties. There are, however, clear differences. In particular, SGD updates never reach equilibrium (for fixed learning rate, small mini-batch size and weight decay $\lambda > 0$): the gradient of the loss is never zero, as shown in Figure 1 – unlike GD [1]. Hence, Neural Collapse as described by [2] does not strictly take place. This in turn implies that SGD, unlike GD, asymptotically shows both a specific "SGD noise" and a coupled low rank bias (assuming random initialization). Because of this double faced property exhibited by SGD, but not by GD, we refer to it as the "Janus effect", drawing inspiration from the ancient Roman God with two faces.

1.1 Related Work

Stochastic gradient descent (SGD) is one of the standard workhorses for optimizing deep models [3]. Though initially proposed to remedy the computational bottleneck of GD, recent studies suggest an implicit bias in SGD, which prevents the overparameterized models from converging to the minima that cannot generalize well [4, 5]. Empirical studies suggest that (i) SGD outperforms GD [6], (ii) small batch SGD generalizes better than large batch SGD [7], and (iii) GD with additional external noise cannot compete with SGD [6]. However, despite many efforts, the potentially important, yet implicit, effects induced by SGD relative to GD have not been fully understood. In particular, we are unaware of any study comparing SGD with GD with respect to biases towards large fluctuations and, especially, low rank.

One area of recent research focuses on characterizing the implicit regularization of gradient-based optimization and its relationship to generalization in deep learning. Several papers have examined the potential bias of gradient descent or stochastic gradient descent toward rank minimization. Empirically, it was shown in [8, 9, 10, 11, 12] that replacing weight matrices with low-rank approximations results in only a small drop in accuracy. This implies that the weight matrices at convergence may be close to low-rank matrices. Following this line of work, various attempts were made to understand the origins of this low-rank bias, and its potential relation with generalization.

Initially, it was believed that the implicit regularization in matrix factorization could be characterized in terms of the nuclear norm of the corresponding linear predictor [13]. This conjecture was later refuted [14]. Subsequent conjecture posits that rank minimization may play a key role in explaining generalization in deep learning. For instance, [15] conjectured that the implicit regularization in matrix factorization can be explained by rank minimization, and also hypothesized that some notion of rank minimization may be crucial to explaining generalization in deep learning. Additionally, [14] established evidence that the implicit regularization in matrix factorization is a heuristic for rank minimization.

An empirical study suggested that during minimization SGD spans a small subspace, implying an effective bias on the rank of the weight matrices [16]. Furthermore, several studies [17, 18, 19] have examined the rank of weight matrices in neural networks that globally minimize a L_2 regularized loss. Specifically, [17] demonstrated that for data on a one-dimensional manifold, the weight matrices of a two-layer network become rank one at the global minimum, a finding later extended in [18] to show that the weight matrix has rank $\leq d$ when the data lies on a d-dimensional space. Additionally, [19] discovered that for sufficiently deep ReLU networks to fit the data, the weight matrices at the top-most layers become low-rank at the global minimum.

Despite recent progress in characterizing low-rank weight matrices at the global minimum, the underlying reasons behind the bias towards low rank during optimization have remained elusive. Prior research [20] has shown that training univariate linear networks on binary classification tasks with exponentially-tailed losses via gradient flow (GF) results in the model converging to weight matrices of rank one, provided that the data is linearly separable. In a more recent study [21], the authors extended this result and showed that when successfully training a ReLU network with multiple linear layers at the top, using GF, the top layers converge to rank one weight matrices.

Here, we describe an empirical and theoretical analysis of SGD vs GD convergence, first for deep RELU networks and then for the case of linear regression. We explain the difference between GD and small minibatch SGD: the latter is characterized by asymptotic intrinsic fluctuations in the weight matrices in the bottom and middle layers which are coupled with a bias towards small rank. In the one-layer linear case we provide a complete analysis of convergence of SGD vs GD: the components of the matrix W corresponding to the null space of the data matrix X converges to zero for both SGD and GD, but the decay is much faster for SGD. Thus SGD is much more effective at pruning features that are not supported by the data.

2 Deep RELU networks

In a previous paper [1] we discussed several differences between SGD and GD. In particular, in the presence of regularization, SGD does not converge to a perfect equilibrium: there is always, at least generically, SGD noise. We concluded that the underlying reason is a rank constraint in the SGD update that depends on the size of the mini-batches – an observation that seems to have escaped previous studies. This rank constraint also implies a stronger SGD bias towards low rank solutions that reinforces a similar bias that SGD shares with GD – due to maximization of the margin under normalization (that can be inferred from [19]). The argument can be seen by considering the SGD update equations. The normalized weight matrices V_k and the product of the Frobenius norm ρ are first initialized, and then iteratively updated simultaneously in the following manner

$$\begin{aligned} \rho &\leftarrow \rho - \eta \frac{2}{B} \sum_{(x_n, y_n) \in \mathcal{S}'} (1 - \rho \bar{f}_n) \bar{f}_n - 2\eta\lambda\rho \\ V_k &\leftarrow V_k - \frac{2}{B} \rho \sum_{j=1}^{B} \left[\left(1 - \rho \bar{f}_{i_j}\right) \left(-V_k \bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k} \right) \right] \end{aligned} \tag{1}$$

where $\mathcal{S}'$ is selected uniformly as a subset of $\mathcal{S}$ of size B, $\eta > 0$ is the learning rate.

A study of Equations 1 is in the Appendix. The main result is Lemma 1. Its qualitative predictions can be understood directly from Equations 1. Observe that the first Equation shows that for large t $\rho f = 1$ when $\lambda = 0$, whereas $1 - \rho f > 0$ for $\lambda > 0$.

Lemma 1 shows that there cannot be convergence to a unique set of weights $\{V_k\}_{k=1}^L$ that satisfy equilibrium for all minibatches. More details of the argument are illustrated in [22]. When $\lambda = 0$, interpolation of all data points is expected: in this case, the GD equilibrium can be reached without any constraint on the normalized weight matrices V_k. This is also the situation in which SGD noise is expected to essentially disappear: compare the histograms on the left and the right hand side of Figure 10 in [1]. Thus, during training, the solution $\{V_k\}_{k=1}^L$ is not the same for all samples: there is *no convergence to a unique solution* but instead fluctuations between solutions during training.

2.1 Testing key qualitative predictions of the theory

The analysis of section D leads to a few predictions that we have tested in our experiments.

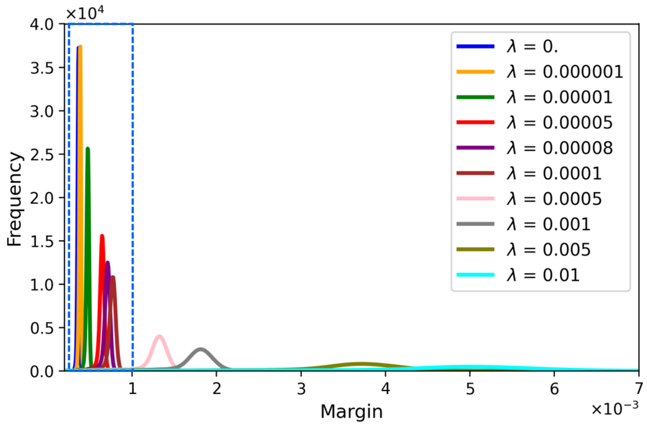
(1) Fluctuations in $\bar{f}_n$ during training should be minimal for $\lambda = 0$ – just due to the finite learning rate of gradient descent – and increase for increasing λ. Separately, μ, which is the average margin over all the training data, increases according to the theory with increasing λ because $\mu = (M + \lambda)\rho$. The corresponding margin $\bar{f}_n$ for different λ in the case of binary classification on CIFAR10 trained with SGD are shown in Figure 1. As predicted, the variance of the fluctuations is small with $\lambda = 0$ and grows with increasing λ. Notice that the asymptotic fluctuations in f_n across different n, are due to fluctuations in the V_k weight matrices for $k < L$, that is, for weight matrices that did not undergo neural collapse. From an analysis of the equations (see [1], it seems likely that the fluctuations in ρ are negligeable.

(2) According to Lemma 1 there should be no SGD specific noise when the mini-batch size is equal to the training dataset size – that when SGD becomes GD – and no dependence of these fluctuations on λ. Our experiments confirm this prediction, see Figure 2. There are fluctuations and their variance is large because, using the same hyper-parameters of the other experiments, GD does not converge to zero square loss and in fact is quite far from it with a significant percentage of incorrect classifications on the training set.

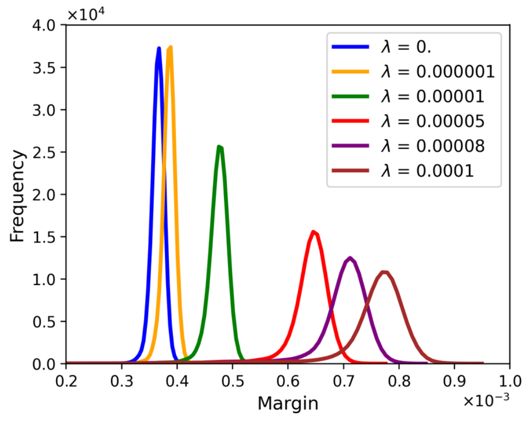
(3) According to Equation 24 the size of $\|\dot{V}_k\|$ depends on λ, because $\lambda > 0$ ensures $\ell_n > 0$. It should be minimal at the top layer, assuming that the top layer is close to converging to Neural Collapse, since the rank of the top layer is small (2 in our case). Figure 3 confirms our prediction and shows the dependency on k and λ.

(4) Larger rank of V_k leads to larger $\|\dot{V}_k\|$ as predicted by Equation 1: compare Figure 4(a) and Figure 4(b).

(5) In the case of exponential-type loss functions such as the logistic loss, the presence of the SGD-specific noise is expected, even when $\lambda = 0$, because of Equation 16. The cross-entropy loss margin results with different λ are shown in Figure 5 (a), while the square loss results are shown in (b). Even for $\lambda = 0$ there cannot be interpolation: the value of $\frac{e^{-\rho \bar{f}_n}}{1+e^{-\rho \bar{f}_n}}$ in the Equation 16 is always positive (and controlled by ρ). Of course the size of the fluctuations is expected to increase further with increasing λ, as shown in Figure 5 (a).



(a)



(b)

Figure 1: (a) Margin distributions – that is histograms of $f(x_n)$ – over 10000 training data samples for binary classification on the CIFAR10, trained with SGD and our deep ReLU networks with varying λ. (b) A zoomed-in view of the blue rectangular region in (a) reveals more detailed margin changes with λ ranging from 0 (without regularization) to 1e-04. The margins exhibit little noise (i.e., very small standard deviations) when $\lambda = 0$ and 1e-06. An increase of λ from 5e-05 to 0.01, leads to an increase in the average margin, but, more interestingly to an increase in the standard deviation of the noise distribution.

3 Linear Regression

3.1 SGD and GD

Consider the linear regression problem of finding the best linear network $W \in \mathcal{R}^{m\times d}$ that satisfies $Wx = y$ from a set of N training data $x_i \in \mathcal{R}^d$ with $i = 1, \cdots, N$, and corresponding target $y_i \in \mathcal{R}^m$ with $i = 1, \cdots, N$. We always assume to be in an overparameterized setting where $N < d$. The empirical loss/risk with weight decay is given by

$$\mathcal{L}(W) = \sum_{i=1}^{N} \|Wx_i - y_i\|^2 + \lambda \|W\|^2, \tag{2}$$

where λ denotes the weight decay regularization parameter.

The loss $\mathcal{L}$ is minimized by gradient flow

$$\dot{W} = -\frac{\partial \mathcal{L}}{\partial W} = -\frac{2}{N}\sum_{i=1}^{N}(Wx_i - y_i)x_i^T - 2\lambda W. \tag{3}$$

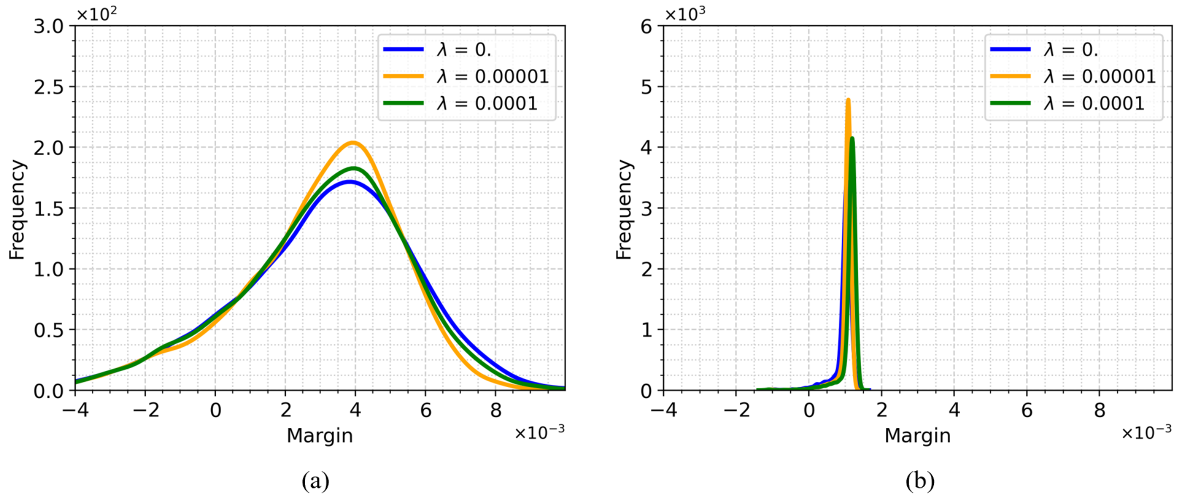


Figure 2: Margin distribution over all training data for binary classification on the CIFAR10, trained with GD using our deep ReLU model and varying λ. (a) We used the same training hyperparameters as previous experiments, but with a large batch size (B) of 10000, which matches the entire training data size (N). The fluctuations here are not SGD-specific (see text); GD here is far from zero square loss with about 20% classification errors on the training set. (b) Training our model with a constant larger learning rate $\eta = 0.05$, $B = 10000$, and 5000 epochs; GD achieved small classification errors ($< 1\%$) that is comparable to the performance achieved with SGD. The fluctuations w.r.t. different λ are much smaller than those observed in (a).

The corresponding gradient descent iteration is

$$W(t+1) = W(t) - \eta \frac{\partial \mathcal{L}}{\partial W} = -\frac{2\eta}{N} \sum_{i=1}^{N} (W^t x_i - y_i) x_i^T - 2\eta\lambda W(t), \tag{4}$$

The Stochastic Gradient descent (SGD) iteration corresponds to

$$W(t+1) = W(t) - \eta \frac{\partial \mathcal{L}}{\partial W} = -\frac{2\eta}{B} \sum_{i \in S^t} (W(t) x_i - y_i) x_i^T - 2\eta\lambda W(t), \tag{5}$$

where one minibatch $\mathcal{S}^t$ of size $B \leq N$ is selected uniformly as a subset of the training dataset $\mathcal{S}$; $\eta > 0$ is the learning rate which we assume fixed in this paper (unlike typical setups in which η decreases with iterations). Gradient descent is the special case of $S^t = S$ (i.e., minibatch size $B = N$). In the following we consider a realization of the stochastic process associated with SGD. In fact there is no difference in the analysis of this section if we just assume that the mini-batches of size 1 are selected deterministically from 1 to N.

3.2 Low rank bias of SGD

Assume asymptotic equilibrium, that is $\dot{W} = 0$ (or $W(t+1) - W(t) = 0$). For simplicity assume $B = 1$. Since we assume overparameterization, if $\lambda = 0$ this implies $Wx_i = y_i \quad \forall i$, that is exact interpolation of the training data. If $\lambda > 0$, exact interpolation is impossible (see Lemma 1 in [1]), that is $\|(Wx_i - y_i\| \neq 0 \quad \forall i$. Then $0 = -(Wx_i - y_i)x_i^T - \lambda W \quad \forall i$, which yields

$$W = -y_i x_i^T (\lambda I + x_i x_i^T)^{-1} \quad \forall i, \tag{6}$$

Since the inverse of $(\lambda I + x_i x_i^T)$ exists (see for instance the Sherman–Morrison formula) and has full rank, the rank of W is the rank of $y_i x_i^T$. Thus the assumption $\dot{W} = 0$ implies that W has rank 1– – which is not consistent, in general, with a small square error in regression (for $d > 1$). Therefore, $\dot{W}$ *cannot be zero for* $t \to \infty$ *and the assumption of* $\dot{W} = 0$ *must be wrong*: there is no Neural Collapse and there is instead a bias towards low rank. The situation is different for GD (corresponding to SGD

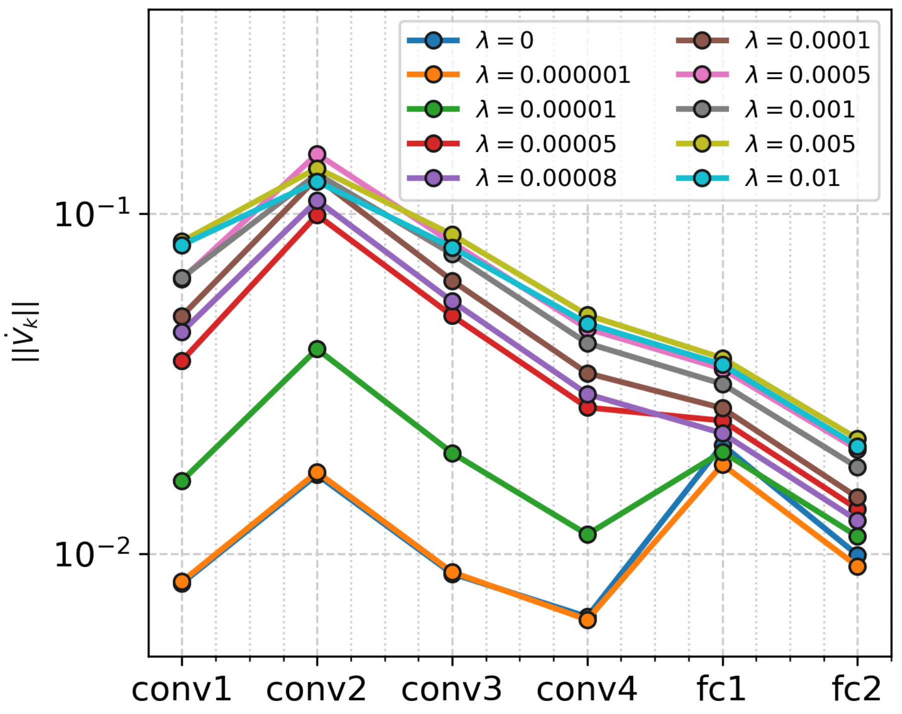


Figure 3: Layer-wise SGD noise $||\dot{V}_k||_F (\forall k \in [L])$ in logarithmic scale by 10 different λ at "convergence". $||\dot{V}_k||_F$ is small when λ is 0 or 1e-06. An increase of λ from 1e-05 to 0.01 generates larger fluctuations, especially for the first four convolution layers. The last two fully connected layers tend to consist of low-rank matrices, corresponding to smaller $\|\dot{V}_k\|$.

with $B = N$). In this case the matrix $\sum_{i=1}^{N}(Wx_i - y_i)x_i^T$ can be expected to be full rank or close to it, since it is the sum of N rank 1 matrices. In this case the assumption $\dot{W} = 0$ does not lead to a contradiction: asymptotic equilibrium can be reached and there is no low rank bias.

More succinctly, the GD and SGD update *for each minibatch* has the form

$$W(t+1) - W(t) = -2\frac{\eta}{B}(W(t) - W^*)XX^T - 2\eta\lambda W(t) \tag{7}$$

where X is the matrix composed of the x_i belonging to the minibatch, and W^* is defined by $Y = W^*X$. For GD, XX^T is generically full rank, whereas it is rank deficient for SGD with $B < N$. It is well known that when W is initialized as $W^0 = 0$, both SGD and GD converge to the correct rank and to the regularized solution. However, when W is initialized as a random matrix (possibly of small norm) then the convergence of SGD and GD is quite different as shown in Figures A.1 and A.2.

SGD noise and low rank bias are two faces of the same phenomenon specific to SGD and absent for GD. The formal version of this intuition is the following obvious observation, cast here as a theorem.

Theorem 1. *Consider the linear regression problem $WX = Y$. Assume that $W \in \mathbb{R}^{m,d}$ is found by SGD with minibatch of size B or by GD, both with learning rate η and regularization parameter λ. Assume overparametrization, that is the data matrix is $X \in \mathbb{R}^{d,N}$ with $d > N$. Let π be the projection on the span of the data (the columns of X), $W^{\|} = W\pi$ the weight matrix restricted to the data span, and $W^{\perp} = W(I - \pi)$ the weight matrix W restricted to the null space of X, so that $W = W^{\|} + W^{\perp}$. Then for minibatches of size 1,*

- *if W at initialization is the zero matrix (or more generally $W\pi = W$), both SGD and GD will converge to the same regularized solution;*
- *if W at initialization is a non-zero matrix (more generally $W\pi \neq W$), such as a random matrix, SGD and GD will also converge to the regularized solution. However, the convergence of $W^{\perp}$ (that is, components of $\mathcal{W}$ that are in the null space of XX^T) to zero is much faster for SGD*

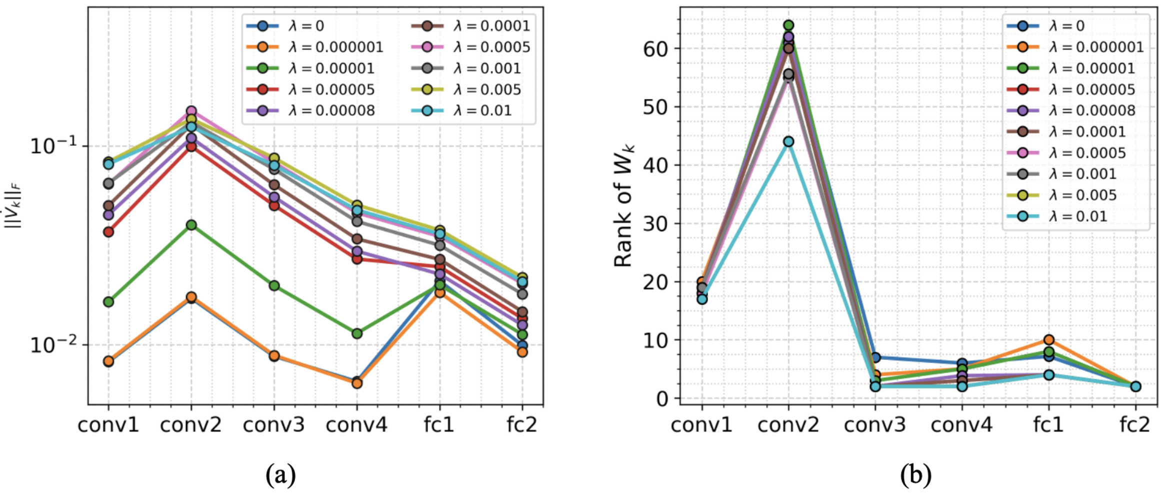


Figure 4: (a) Layer-wise $||\dot{V}_k||_F$ in logarithmic scale by 10 different λ at the convergence. The SGD noise across layers, as measured by $||\dot{V}_k||_F$, is small when λ is set to 0 or 1e-06. An increase of λ from 1e-05 to 0.01 generates larger fluctuations, especially for the first four convolution layers. The last two fully connected layers tend to consist of low-rank matrices, corresponding to smaller $||\dot{V}_k||$. (b) Layer-wise rank of V_k by 10 different λ at the convergence trained with SGD (batch size is 128). The rank of the weight matrix V_k across different layers decrease by increasing the weight decay parameter (λ). The last ("deeper") four layers achieved much smaller ranks compared to the first two layers, i.e., the top layers tend to consist of low-rank weight matrices, corresponding to smaller $||\dot{V}_k||$ as shown in (a).

than GD. The rate is $(1 - \eta\lambda)^N$ for SGD and $(1 - \eta\lambda)$ for GD (assuming η to be constant during the epoch).

Proof. Equation 7 can be rewritten as

$$W^{\parallel}(t+1) = W^{\parallel}(t)[(1 - 2\eta\lambda)I - \frac{\eta\lambda}{B}XX^T] - 2\eta\lambda YX^T \tag{8}$$

and

$$W^{\perp}(t+1) - W^{\perp}(t) = -2\eta\lambda W^{\perp}(t) \tag{9}$$

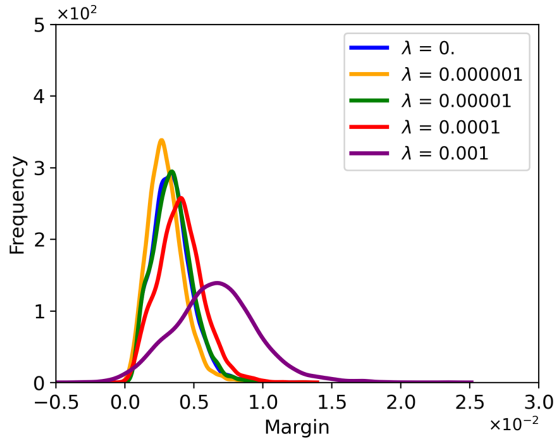
The two equation are independent; equation 9 shows that the elements of $W^{\perp}$ decay to zero for each minibatch with a factor $1 - \eta\lambda$. The theorem follows considering a full epoch. □

Remarks

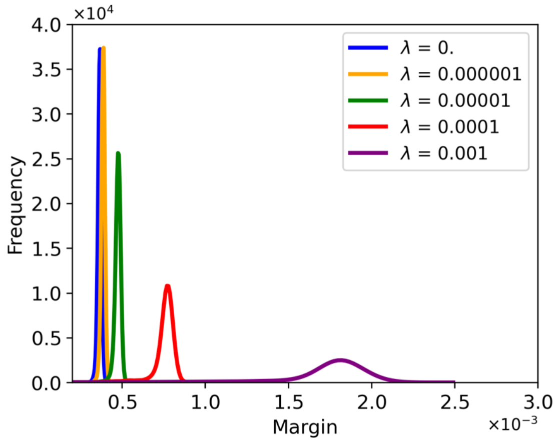
- For minibatch size $B > 1$ the convergence rate per epoch of the null space of W to zero is $(1 - \frac{\eta\lambda}{B})^{\frac{N}{B}}$.
- The SGD update equations can be fully deterministic, running through the data from 1 to N in the same order in each epoch. This is equivalent to randomly choosing a minibatch without repetition until all data are sampled. Despite the absence of any random process there is asymptotic noise in the predicted y.

3.3 Linear regression experiments: overparametrization, low rank, random initialization

We consider the case of a linear, overparametrized one layer network with input dimension $d > N$. We assume that the learning rate of GD is set $\eta = 0.5\max(svd)$. The learning rate for SGD is the same at initialization and then decays as $\approx \frac{1}{\sqrt{t}}$ (see Appendix E) where t is epochs. It is well-known that SGD and GD converge to the regularized solutions (which is the minimum norm solution for $\lambda \to 0$)



(a)



(b)

Figure 5: Margin distributions over all training data for binary classification on the CIFAR dataset trained with **cross entropy loss** in (a) and **square loss** in (b) using different λ. The results in (a) verified the prediction presented in Section 2.1, i.e., the presence of the SGD-specific noise is expected to be significant even when $\lambda = 0$, unlike the square loss case.

at the same rate when W is initialized from $W_0 = 0$ (see Figure A.1 in Appendix). We also initialize W_0 to be a random matrix of small norm. In this case (see Figure A.2), the small singular values decay to zero for SGD much faster than for GD. At the same time the norm of the gradient is almost always larger for SGD than GD – as expected from Equation 5. The small singular values are almost always much smaller for SGD than GD: there is a strong *shrinkage of small eigenvalues* for SGD vs GD, which is correlated with increasing gradient noise – that is norm of $\|\dot{W}\|$ – and with λ. This effect is qualitatively similar to a bias towards low rank[1]. SGD maintains its bias in the case the problem is to find a small number of active regressors among many useless coordinates (see Figure A.3). Online gradient descent in which each example is used only once (just one epoch for GD and SGD) gives similar results (Figure A.4). Figure A.5 shows the asymptotic noise in the training of SGD that does not decrease (for contant η). Notice that the update equations used here are deterministic running through the data from 1 to N: this is equivalent to randomly choosing a minibatch without repetition until all data are sampled.

4 Discussion

Our analysis of minibatch SGD shows, consistently with the classical analysis, that SGD with fixed learning rate does not strictly "converge"[2]: we observe sizeable fluctuations under the square loss in the overparametrized case for very small λ, when degenerate solutions, that fit the data perfectly, abound for $\lambda = 0$. This is not surprising, of course, since minibatch SGD uses random samples of the training data. Furthermore, classical stochastic gradient descent with an explicit random noise term (instead of random sampling of minibatches) is known to converge almost surely to a global minimum when the objective function is convex or pseudoconvex, and otherwise converges almost surely to a local minimum, provided that the learning rate η decrease with an appropriate rate.

As shown in Figure 2, the minibatch SGD specific noise disappears when the minibatch size increases and SGD becomes GD. This also means that Neural Collapse, as described by [2], never truly happens for the linear regression we have studied or for generic, intermediate layers in a neural network: NC1 is equivalent to all margins $f(x_n) \quad \forall n$ to be the same at convergence, which cannot strictly happen for SGD and $\lambda > 0$ (both conditions are required for NC1 to be possible[1]). On the other hand, the origin of the minibatch SGD noise is not due to an explicit noise term, but can be described as arising from a competition between a bias for small rank and the constraint of minimizing the error in

[1]Note that direct measurements of rank are fragile because of the discontinuous nature of rank its dependence on an arbitrary threshold (machine precision for the function rank in Matlab).

[2]It does in expectation but its variance is never zero, see also [23].

fitting the data during learning W. An equivalent description is that minibatch SGD never finds an asymptotic W that is the best fit to all the data but instead finds a sequence of very similar W, each one optimized to fit the last data point(s).

It is interesting to notice that in our simulations of the linear case the choice of the mini batches is not random but it is the same sequence going through the data from 1 to N in each epoch. We did not notice any difference wrt random choice of nonoverlapping mini batches. Despite the absence of any randomness (apart the initial choice of the sequence order), SGD is associated with significant noise-like fluctuations arising from the nonlinear dynamics we described. In this sense, we believe, the SGD noise is better described as deterministic chaos.

It is unclear whether the SGD-specific noise described here has a role in better generalization. Our tentative answer is negative, since there is an empirical evidence that good solutions can sometimes be found for $\lambda = 0$. It is possible, however, that the SGD-specific noise may help in searching for a global minimizer, especially in underparametrized situations, when we expect isolated and not degenerate global minima (see [1]).

A closely related open question is whether small rank implies better generalization. A direct effect is possible though we do not know of any result showing that smaller rank yields better generalization bounds for deep networks. In fact the standard Rademacher complexity of linear function classes with L_2 norm does not decrease with rank, though other measures with different norms may depend on rank. The main effect, however, is likely to be indirect. We conjecture that the bias towards small rank plays an important role in optimization, by eliminating "features" that are not supported by the data, as we showed in the linear case. Furthermore, this effect may be especially critical in the optimization of deep overparametrized networks, implying a significant advantage of SGD vs GD, even apart from simple computational efficiency.

As it should be clear from our analysis, the mechanism underlying the SGD-specific fluctuations we have identified is a competition between the two terms on the right side of Equations 15 and 16 in the Appendix. Consider the case of classification under the square loss. If the term $(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k})$ in the equation

$$\dot{V}_k = \frac{2}{N}\rho \sum_n \left[\left(1 - \rho \bar{f}_n\right) \left(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k} \right) \right]. \tag{10}$$

becomes small, the matrix V_k becomes closer to $\frac{\partial \bar{f}_n}{\partial V_k}$, which has rank 1. But small V_k rank implies that the network cannot interpolate the data, which means that $(1 - \rho \bar{f}_n)$ cannot be small.

The same competition arises in the case of the exponential loss, in this case independently of λ:

$$\dot{V}_k = \frac{2}{N}\rho \sum_n \frac{e^{-\rho \bar{f}_n}}{1 + e^{-\rho \bar{f}_n}} \left(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}\right). \tag{11}$$

This also means that SGD – unlike GD – is biased to find a trade-off between a small norm approximation to the weight matrices and a small regression oor classification error.

The precise mechanism behind the bias of SGD vs GD towards low rank is clear in the linear case: elements of W in the null space of the data matrix decay much more quickly to zero under SGD than under GD. Although our analysis in the case of deep RELU networks does not rely on the solution of the linear case, they are fully consistent with each other. It is thus likely that the exponential decay of the null space under SGD is the reason for the strong bias towards low rank observed in deep networks.

References

[1] Mengjia Xu, Akshay Rangamani, Qianli Liao, Tomer Galanti, and Tomaso Poggio. Dynamics in deep classifiers trained with the square loss: Normalization, low rank, neural collapse, and generalization bounds. *Research*, 6:0024, 2023.

[2] X. Y. Han, Vardan Papyan, and David L. Donoho. Neural collapse under mse loss: Proximity to and dynamics on the central path, 2021.

[3] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[4] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[5] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*, 2017.

[6] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2019.

[7] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017.

[8] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[9] Jose M. Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 856–867, Red Hook, NY, USA, 2017. Curran Associates Inc.

[10] Murad Tukan, Alaa Maalouf, Matan Weksler, and Dan Feldman. No fine-tuning, no cry: Robust svd for compressing deep networks. *Sensors*, 21(16), 2021.

[11] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 67–76, 2017.

[12] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 10–15 Jul 2018.

[13] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization, 2017.

[14] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *CoRR*, abs/2012.09839, 2020.

[15] Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *CoRR*, abs/2005.06398, 2020.

[16] Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754, 2018.

[17] Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. *arXiv preprint arXiv:2002.09773*, 2020.

[18] Greg Ongie and Rebecca Willett. The role of linear layers in nonlinear interpolating networks, 2022.

[19] Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. *CoRR*, abs/2201.12760, 2022.

[20] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *CoRR*, abs/2006.06657, 2020.

[21] Thien Le and Stefanie Jegelka. Training invariances and the low-rank phenomenon: beyond linear networks. In *International Conference on Learning Representations*, 2022.

[22] Tomer Galanti and Tomaso Poggio. Sgd noise and implicit low-rank bias in deep neural networks. Technical report, Center for Brains, Minds and Machines (CBMM), 2022.

[23] T. Poggio and Y. Cooper. Loss landscape: Sgd can have a better view than gd. *CBMM memo 107*, 2020.

[24] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[25] Tomaso Poggio, Andrzej Banburski, and Qianli Liao. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 2020.

[26] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020.

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

A Linear theory: why SGD has a stronger low-rank bias than GD (with ChatGPT)

To calculate and compare the decay rates for SGD and GD during one epoch, let's use a specific example and equations with the given parameters. We will assume the weight decay parameter is λ, the learning rate is η, and there are 2 data points with a minibatch size of 1 for SGD.

The Data Matrix X is a matrix with first column $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and second column $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$. The weight matrix W is initialized as $W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}$.

- *Gradient for the first column of* X (X_1) :

$$\nabla_W L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- *Gradient for the second column of* X (X_2) :

$$\nabla_W L_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Thus when updating with the first column, the gradient will only affect the first row of W. Similarly for the second column.

A.1 SGD with Decay Rate

When Stochastic Gradient Descent (SGD) includes weight decay, the update rule for the weight matrix W is modified to penalize large weights. The weight decay term adds a regularization term proportional to the square of the norm of the weights, scaled by a parameter λ. The update rule for SGD with weight decay becomes:

$$W_{\text{new}} = W - \eta \left(\nabla_W L + \lambda W \right)$$

where:

- η is the learning rate.
- $\nabla_W L$ is the gradient of the loss with respect to W (as calculated earlier).
- λ is the weight decay parameter.

Let's analyze the effect of this update rule on the last column of W. Consider

$$\text{Initial } W : W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}$$

From the earlier calculation, we have two gradients $\nabla_W L_1$ and $\nabla_W L_2$ based on the columns of X, while the weight decay term λW affects all elements of W, including the last column. Consider now the updates on the last column:

- *First Column of X Update*: the gradient update $\nabla_W L_1$ does not directly affect the last column of W, but the weight decay term λW does.

$$W_{\text{new, last column}} = (1 - \eta\lambda) \cdot W_{\text{last column}}$$

- *Second Column of X Update:* similarly, $\nabla_W L_2$ does not affect the last column, but the weight decay term does.

$$W_{\text{new, last column}} = (1 - \eta\lambda) \cdot W_{\text{last column}}$$

Thus

- The last column of W will change during iterations due to the weight decay term, even if the gradients based on the data X do not directly influence it.
- The effect of weight decay is a uniform scaling of all elements of W by a factor of $(1-\eta\lambda)$ in each iteration.
- Over multiple iterations, this will lead to the last column of W gradually decreasing in magnitude, moving towards zero if $\lambda > 0$ and $\eta > 0$.

A.2 SGD vs GD

In SGD, the weight matrix W is updated after each data point. Since there are two data points in our example and the minibatch size is 1, SGD will perform two updates in one epoch.

- Update rule for SGD with weight decay:
$$W_{\text{new}} = W - \eta\left(\nabla_W L_i + \lambda W\right)$$
where $\nabla_W L_i$ is the gradient of the loss for the i-th data point.
- Since we are focusing on the decay due to λ, and assuming the gradient $\nabla_W L_i$ does not affect the last column (as per our previous analysis), the update for the last column of W due to weight decay alone is: $W_{\text{new, last column}} = (1-\eta\lambda) \cdot W_{\text{last column}}$
- After 2 updates (one epoch): $W_{\text{final, last column}} = (1-\eta\lambda)^2 \cdot W_{\text{initial, last column}}$

In GD, the weight matrix W is updated once after going through the entire dataset.

- Update rule for GD with weight decay:
$$W_{\text{new}} = W - \eta\left(\nabla_W L_{\text{total}} + \lambda W\right)$$
where $\nabla_W L_{\text{total}}$ is the total gradient over the entire dataset.
- Focusing on the decay due to λ, and considering $\nabla_W L_{\text{total}}$ does not affect the last column, the update for the last column of W due to weight decay alone is: $W_{\text{new, last column}} = (1-\eta\lambda) \cdot W_{\text{last column}}$
- After one update (one epoch): $W_{\text{final, last column}} = (1-\eta\lambda) \cdot W_{\text{initial, last column}}$

Hence, the obtained decay factors for SGD and GD are as follows.

- SGD decay factor after one epoch:
$$(1-\eta\lambda)^2$$
- GD decay factor after one epoch:
$$(1-\eta\lambda)$$

Conclusion: Comparing the two, we see that the decay factor for SGD is the square of the decay factor for GD after one epoch, given the setup of 2 data points and minibatch size of 1 for SGD. It implies a faster decay rate for the last column of W in SGD compared to GD.

B Linear experiments

C Multilayer RELU networks and training

We introduce a model of the training procedure that uses square loss for binary classification, Lagrange multipliers (LM) for normalizing the weights and a regularization term controlled by λ. The normalization technique we use is completely equivalent to the Weight Normalization [24], see the proof in [25]. In the paper, we assume the network is overparametrized, so that there is convergence to global minima with appropriate initialization, parameter values, and data. Under the assumption of

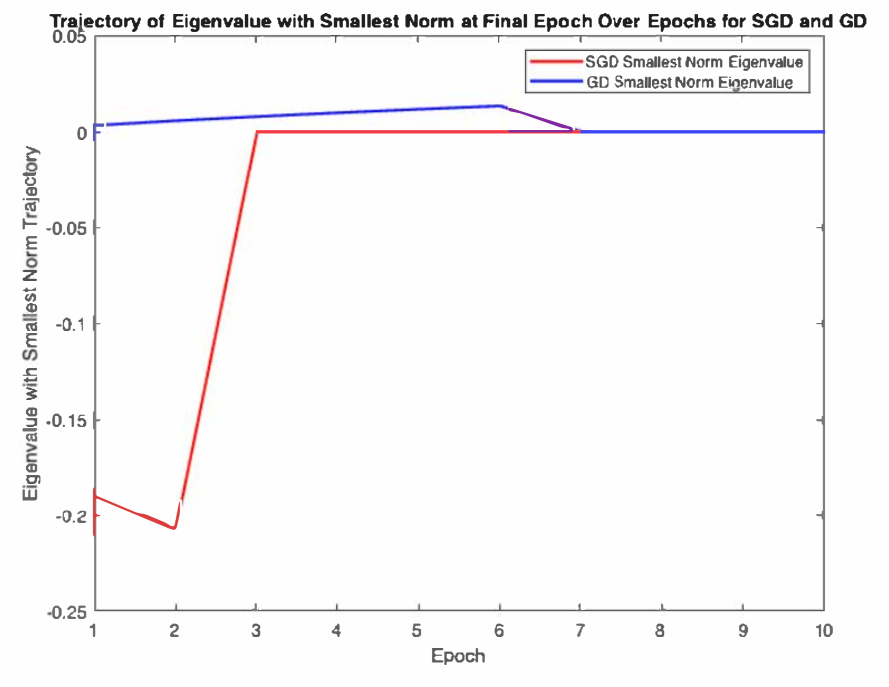


Figure A.1: Solving $Wx = y$, for a data matrix X with $d = 11$ and $N = 10$; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 10. Initialization is from $W = 0$. The final MSE for SGD is 0.879681 and for GD is 1.524118. At the final epoch, the eigenvalues with the smallest norm for both SGD and GD have a norm of 0.000000.

overparameterization, we also expect interpolation of all training data when $\lambda = 0$ [1]. In the presence of weight decay (i.e., $\lambda > 0$), perfect interpolation of all data points cannot occur and is replaced by "quasi-interpolation" of the labels (y_n). In the special case of binary classification where $y_n = \pm 1$, quasi-interpolation is defined as $\forall\ n: \ |f(x_n) - y_n| \leq \epsilon$, where $\epsilon > 0$ is small. Our experiments and analysis of the training dynamics show that the presence of regularization leads to a weaker dependence on initial conditions, as has been observed in [26].

In this study, we consider a binary classification problem given a training dataset $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^N$ of N samples, where $x_n \in \mathbb{R}^d$ are the inputs (normalized such that $\|x_n\| \leq 1$) and $y_n \in \{\pm 1\}$ are the labels. We use deep rectified homogeneous networks with L layers (see Figure A.8) to solve this classification problem. For simplicity, we consider networks $f_W : \mathbb{R}^d \to \mathbb{R}^p$ of the following form $f_W(x) = W_L \sigma(W_{L-1} \ldots \sigma(W_1 x) \ldots)$, where $x \in \mathbb{R}^d$ is the input to the network and $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(x) = \max(0, x)$ is the rectified linear unit (ReLU) activation function that is applied coordinate-wise at each layer. The last layer of the network is linear.

Due to the positive homogeneity of ReLU (i.e., $\sigma(\alpha x) = \alpha\sigma(x)$ for all $x \in \mathbb{R}$ and $\alpha > 0$), one can reparametrize $f_W(x)$ by considering normalized[3] weight matrices $V_k = \frac{W_k}{\|W_k\|}$ and define $\rho_k = \|W_k\|$ obtaining $f_W(x) = \rho_L V_L \sigma(\rho_{L-1} \ldots \sigma(\rho_1 V_1 x) \ldots)$, see Figure A.8(a). Because of the homogeneity of the ReLU, it is possible to pull out the product of the layer norms as $\rho = \prod_{k=1}^L \rho_k$ and write $f_W(x) = \rho f_V(x) = \rho V_L \sigma(V_{L-1} \ldots \sigma(V_1 x) \ldots)$, as shown in Figure A.8(b). Notice that the two networks – $f_W(x)$ and $\rho f_V(x)$ – are equivalent reparameterizations of the same function but their optimization generally differ. We define $f_n := f_V(x_n)$.

Our definitions follow the convention used in [1] that the norm ρ_k of the convolutional layers is defined as *the norm of their filters* rather than the norm of their associated Toeplitz matrices. The ρ calculated in this way is the quantity that enters the generalization bounds.

In the model described in Figure A.8(b), we assume that all layers are normalized, except for the last one. Thus, the weight matrices $\{V_k\}_{k=1}^L$ are constrained by the LM term to be close to, and eventually converge to, unit norm matrices (in fact to fixed norm matrices); notice that normalizing V_L and then multiplying the output by ρ, is equivalent to letting $W_L = \rho V_L$ be unnormalized. Hence,

[3]We choose the Frobenius norm here.

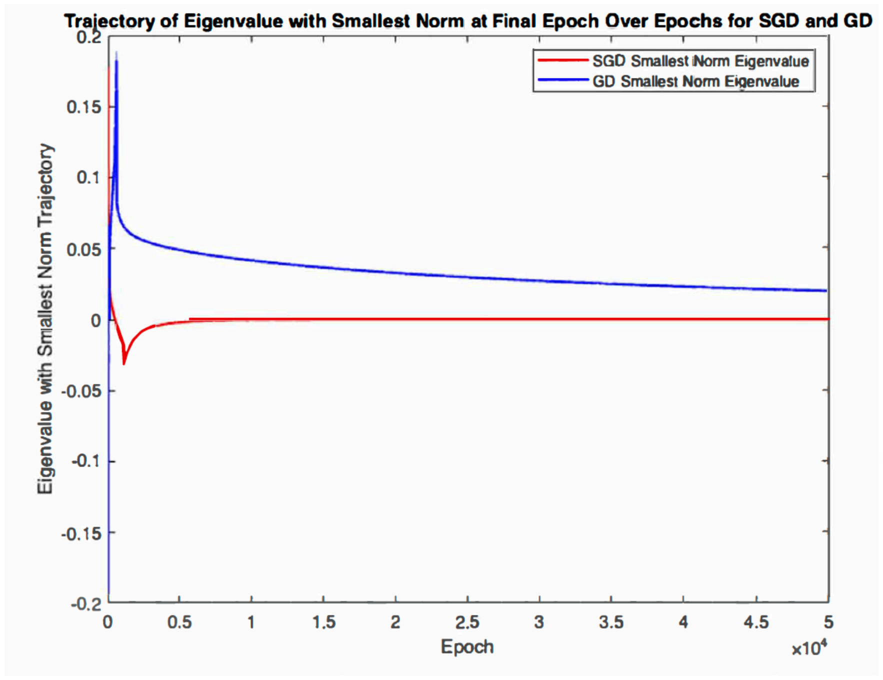


Figure A.2: Solving $Wx = y$, for a data matrix X with $d = 11$ and $N = 10$; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 50000. Initialization is from W random. The final MSE for SGD is 0.126220 and for GD is 0.138375. At the final epoch, the eigenvalues with the smallest norm for SGD has norm 0.000000 and for GD has norm 0.020154.

f_V is the network that at convergence has $L-1$ normalized layers.

Based on the aforementioned definitions, we can write the Lagrangian corresponding to the minimization of the regularized loss function under the constraint $\|V_k\|^2 = 1$ in the following manner

$$\begin{aligned}\mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L}) :&= \frac{1}{N}\sum_n (\rho f_n - y_n)^2 + \sum_{k=1}^{L} \nu_k(\|V_k\|^2 - 1) + \lambda\rho^2 \\ &= \frac{1}{N}\sum_n (1 - \rho \bar{f}_n)^2 + \sum_{k=1}^{L} \nu_k(\|V_k\|^2 - 1) + \lambda\rho^2,\end{aligned} \tag{12}$$

where ν_k are the Lagrange multipliers and $\lambda > 0$ is a predefined parameter.

Separability and Margins. Two of the most important aspects of classification are *separability* and *margins*. Given an input training or test data sample and its label pair (x, y) and the model f_W, we say that f_W correctly classifies x if $\bar{f}_n = y_n f_n > 0$. Moreover, for a given dataset $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^N$, *separability* is defined as the condition in which all training samples are classified correctly, $\forall\, n \in [N] :\ \bar{f}_n > 0$. Furthermore, when $\sum_{n=1}^N \bar{f}_n > 0$, we say that *average separability* is satisfied. The minimum of $\mathcal{L}_{\mathcal{S}}$ for $\lambda = 0$ is usually zero under the assumption of overparametrization. This corresponds to separability.

Notice that if f_W is a zero loss solution of the regression problem, then $\forall\, n :\ f_W(x_n) = y_n$, which is also equivalent to $\rho f_n = y_n$, where we denote $y_n f_n = \bar{f}_n$ *the margin for* x_n. [4] By multiplying both sides of this equation by y_n, and summing both sides over $n \in [N]$, we obtain that $\rho \sum_n \bar{f}_n = N$. Thus, the norm ρ of a minimizer is inversely proportional to its average margin μ in the limit of $\lambda = 0$, with $\mu = \frac{1}{N}\sum_n \bar{f}_n$. It is also useful to define *the margin variance* $\sigma^2 = M - \mu^2$ *with* $M = \frac{1}{N}\sum_n \bar{f}_n^2$. Notice that $M = \frac{1}{N}\sum_n \bar{f}_n^2 = \sigma^2 + \mu^2$ and that both M and σ^2 are not negative.

[4]Notice that the term "margin" is usually defined as $\min_{n\in[N]} \bar{f}_n$. Instead, we use the term "margin for x_n" to distinguish our definition from the usual one.

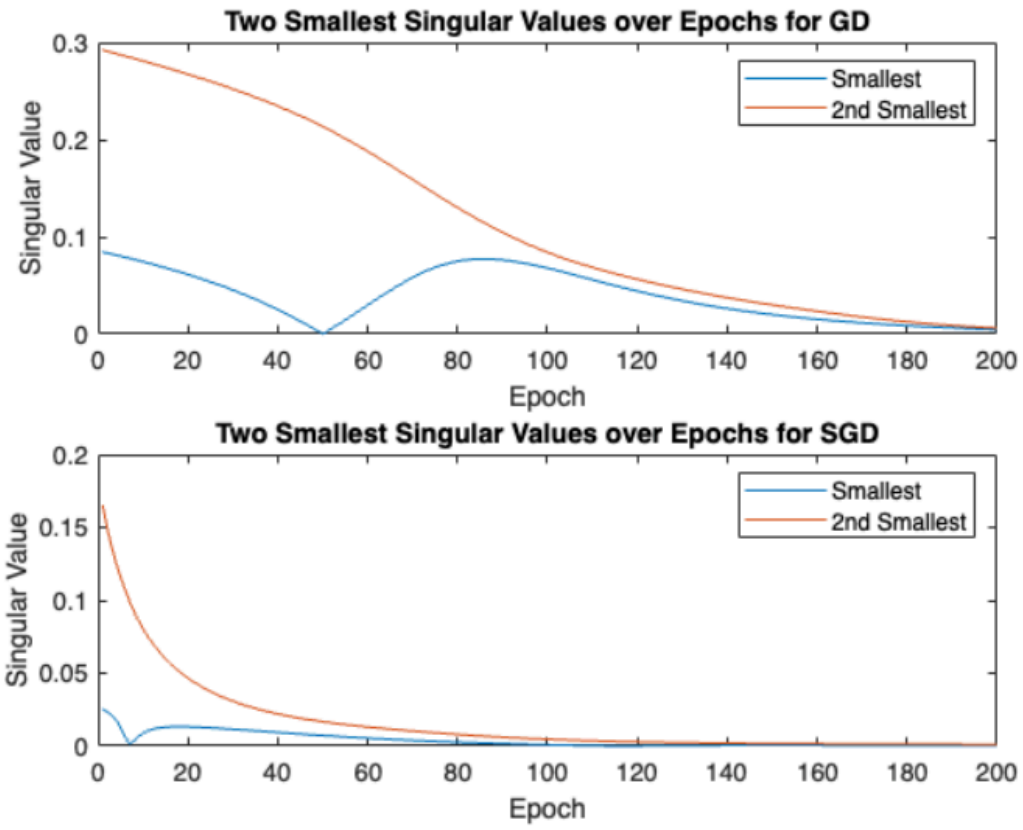


Figure A.3: Regression with sparse features. Solving $Wx = y$, for a data matrix X with $d = 10$ and $N = 5$ with only two of components of x being relevant; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 200. Initialization is from random W.

D Theoretical Analysis

D.1 Gradient flow equations

We assume the deep networks with the ReLU units and weight normalization (in the Frobenius norm) at each layer, enforced via Lagrange multipliers. We also assume square loss and binary classification. The gradient flow equations in ρ (the product of the Frobenius norms of the unnormalized weight matrices) and V_k (the normalized weight matrices) are as follows

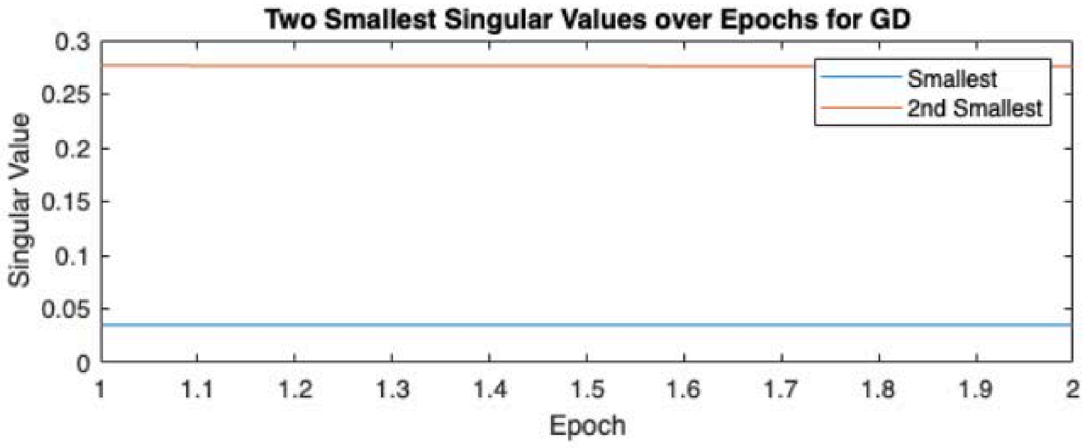
$$\begin{aligned} \dot{\rho} &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L})}{\partial \rho} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho \\ \dot{V}_k &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L})}{\partial V_k} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\rho\frac{\partial \bar{f}_n}{\partial V_k} - 2\nu_k V_k, \end{aligned} \tag{13}$$

where $\bar{f}_n = y_n f(x_n)$, $y_n = \pm 1$ and $n \in [N]$. In the equation of $\dot{V}_k$, we can use the unit norm constraint on the $\|V_k\|$ to determine the Lagrange multipliers (ν_k). Using a structural property of the gradient, the constraint $\|V_k\|^2 = 1$ implies $\frac{\partial \|V_k\|^2}{\partial t} = V_k^T \dot{V}_k = 0$, which gives

$$\nu_k = \frac{1}{N}\sum_n (\rho\bar{f}_n - \rho^2 f_n^2) = \frac{1}{N}\sum_n \rho\bar{f}_n(1-\rho f_n). \tag{14}$$

Thus the gradient flow is the following dynamical system

$$\dot{\rho} = \frac{2}{N}\left[\sum_n \bar{f}_n - \sum_n \rho(\bar{f}_n)^2\right] - 2\lambda\rho \quad \text{and} \quad \dot{V}_k = \frac{2}{N}\rho\sum_n \left[(1-\rho\bar{f}_n)\left(-V_k\bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}\right)\right]. \tag{15}$$



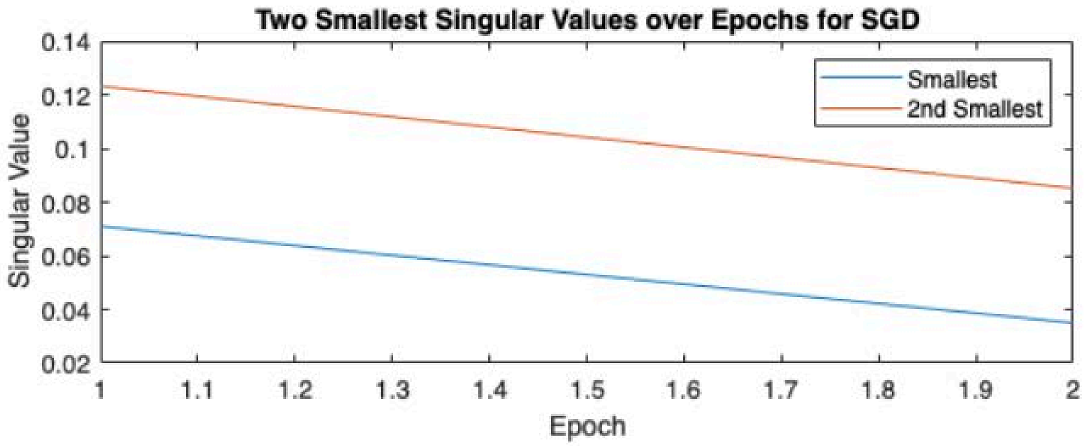


Figure A.4: Regression with sparse features in an online setting. One epoch with $N = 200$, $d = 10$. Solving $Wx = y$, for a data matrix X with $d = 10$ and $N = 5$ with only two of components of x being relevant; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. Initialization is from random W.

The gradient flow for the logistic loss would result in

$$\dot{\rho} = \frac{2}{N}\sum_n \frac{e^{-\rho\bar{f}_n}}{1+e^{-\rho\bar{f}_n}}\bar{f}_n - 2\lambda\rho \quad \text{and} \quad \dot{V}_k = \frac{2}{N}\rho\sum_n \frac{e^{-\rho\bar{f}_n}}{1+e^{-\rho\bar{f}_n}}(-V_k\bar{f}_n + \frac{\partial\bar{f}_n}{\partial V_k}). \tag{16}$$

D.2 SGD

In the previous section, we derived the gradient flow equations of ρ and V_k. In order to iteratively train these parameters over mini-batches, we consider a setting where V_k and ρ are trained as follows

$$\begin{aligned}\rho &\leftarrow \rho - \eta\frac{\partial\mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial\rho} = \rho - \eta\frac{2}{B}\sum_{(x_n,y_n)\in\mathcal{S}'}(1-\rho\bar{f}_n)\bar{f}_n - 2\eta\lambda\rho \\ V_k &\leftarrow V_k - \frac{\partial\mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = V_k - \eta\frac{2}{B}\sum_{(x_n,y_n)\in\mathcal{S}'}(1-\rho\bar{f}_n)(-V_k\bar{f}_n + \frac{\partial\bar{f}_n}{\partial V_k}),\end{aligned} \tag{17}$$

where one minibatch $\mathcal{S}'$ of size $B < |\mathcal{S}|$ is selected uniformly as a subset of the training dataset $\mathcal{S}$ and the learning rate $\eta > 0$.

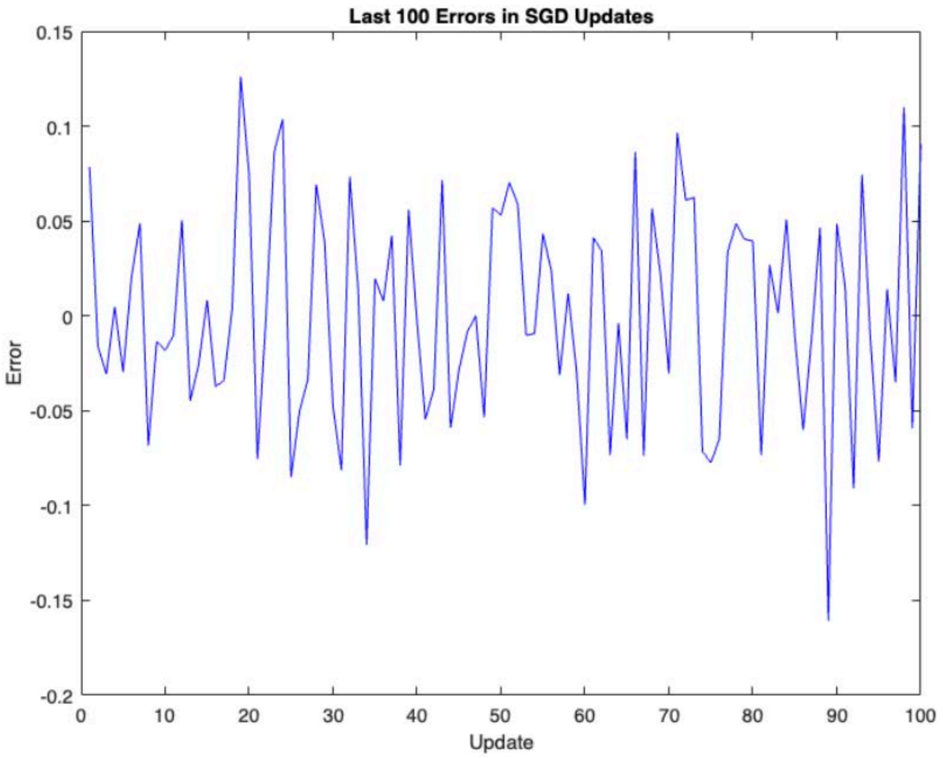


Figure A.5: Regression in an online setting. One epoch with $N = 100000$, $d = 10$. Solving $Wx = y$, for a data matrix X with $d = 10$; SGD is used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. Initialization is from random W. The updates are fully deterministic without any random component. At convergence there is asymptotic noise that does not decrease (unless learning rate goes to zero).

D.3 No equilibrium

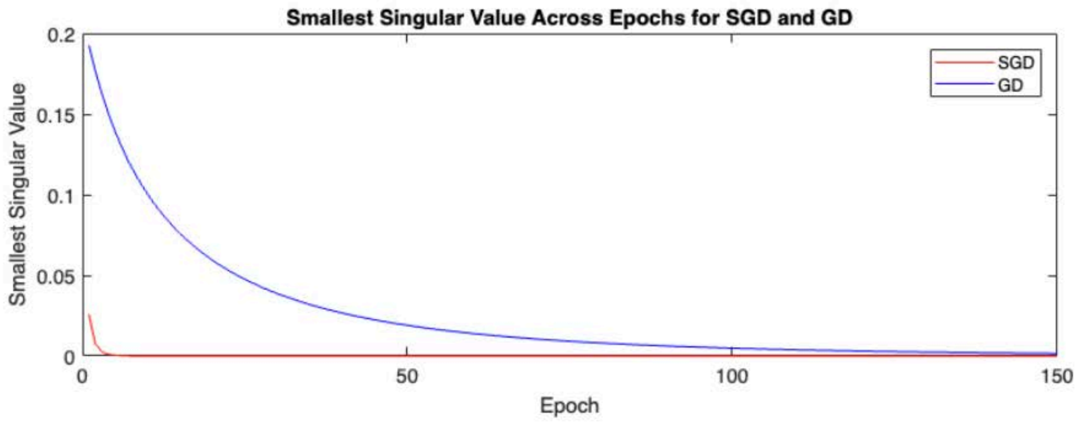
The Lemma below shows that the SGD cannot achieve equilibrium for all the mini-batches of size $B < N$, because otherwise all the weight matrices would have very small rank which is incompatible, for generic data sets, with quasi-interpolation.

Lemma 1. *Let f_W be a neural network. Assume that we iteratively train ρ and $\{V_k\}_{k=1}^L$ using the process described above with weight decay $\lambda > 0$. Suppose that training converges, that is $\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial \rho} = 0$ and $\forall\ k \in [L]$: $\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = 0$ for all mini-batches $\mathcal{S}' \subset \mathcal{S}$ of size $B < |\mathcal{S}|$. Assume that $\forall\ n \in [N] :\ \bar{f}_n \neq 0$. Then, the ranks of the matrices V_k are at most ≤ 2.*

Proof. Let $f_V(x) = V_L\sigma(V_{L-1} \dots \sigma(V_1 x) \dots)$ be the normalized neural network, where $V_l \in \mathbb{R}^{d_{l+1}\times d_l}$ and $\|V_l\| = 1$ for all $l \in [L]$. We would like to show that the matrix $\frac{\partial f_V(x)}{\partial V_k}$ is of rank ≤ 1. We note that for any given vector $z \in \mathbb{R}^d$, we have $\sigma(v) = \operatorname{diag}(\sigma'(v)) \cdot v$ (where σ is the ReLU activation function). Therefore, for any input vector $x \in \mathbb{R}^n$, the output of f_V can be written as follows,

$$\begin{aligned} f_V(x) &= V_L\sigma(V_{L-1} \dots \sigma(V_1 x) \dots) \\ &= V_L \cdot D_{L-1}(x;V) \cdots D_1(x;V) \cdot V_1 \cdot x, \end{aligned} \tag{18}$$

where $D_l(x;V) = \operatorname{diag}[\sigma'(u_l(x;V)))]$ and $u_l(x;V) = V_l\sigma(V_{l-1} \dots \sigma(V_1 x) \dots)$. We denote by $u_{l,i}(x;V)$ the i'th coordinate of the vector $u_l(x;V)$. We note that $u_l(x;V)$ are continuous functions of V. Therefore, assuming that none of the coordinates $u_{l,i}(x;V)$ are zero, there exists a sufficiently small ball around V for which $u_{l,i}(x;V)$ does not change its sign. Hence, within this ball, $\sigma'(u_{l,i}(x;V))$ are constant. We define a set $\mathcal{V} := \{V \mid \forall l \leq L :\ \|V_l\| = 1\}$ and $\mathcal{V}_{l,i} = \{V \in \mathcal{V} \mid u_{l,i}(x;V) = 0\}$. We note that as long as $x \neq 0$, the set $\mathcal{V}_{l,i}$ is negligible within $\mathcal{V}$. Since there is a finite set of indices l, i, the set $\bigcup_{l,i} \mathcal{V}_{l,i}$ is also negligible within $\mathcal{V}$.



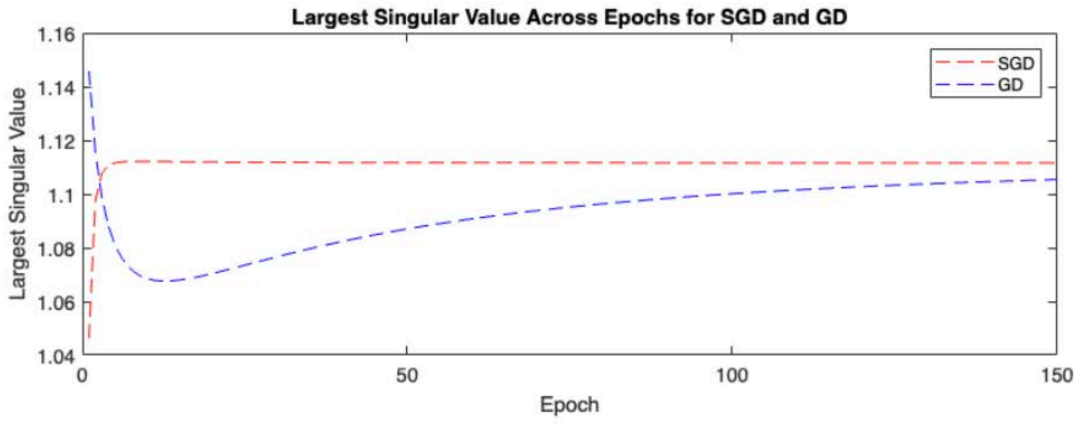


Figure A.6: Sparse regression with $N = 10$, $d = 2$; of the two components of y, one does not depend on the data. Solving $Wx = y$; GD and SGD are used with regularization $\lambda = 0.01$, over 150 epochs, and an optimal learning rate η as described in the text.

Let V be a set of matrices for which none of the coordinates $u_{l,i}(x;V)$ are zero. Then, the matrices $\{D_l(x;V)\}_{l=1}^{L-1}$ are constant in the neighborhood of V, and therefore, their derivative with respect to V_k are zero. Let $a^\top = V_L \cdot D_{L-1}(x;V)V_{L-1}\cdots V_{k+1}D_k(x;V)$ and $b = D_{k-1}(x)\cdot V_{k-1}\cdots V_1 x$. We can write $f_V(x) = a(x;V)^\top \cdot V_k \cdot b(x;V)$. Since the derivatives of $a(x;V)$ and $b(x;V)$ with respect to V_k are zero, by applying $\frac{\partial a^\top X b}{X} = ab^\top$, we have $\frac{\partial f_V(x)}{\partial V_k} = a(x;V)\cdot b(x;V)^\top$ which is a matrix of rank at most 1. Therefore, $\frac{\partial \bar{f}_n}{\partial V_k} = y_n \frac{\partial f_V(x_n)}{\partial V_k}$ is a matrix of rank at most 1. Therefore, for any input $x_n \neq 0$, with measure 1, $\frac{\partial \bar{f}_n}{\partial V_k}$ is a matrix of rank at most 1.

Since $\forall\ k \in [L]:\ \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = 0$ for all mini-batches $\mathcal{S}' = \{(x_{i_j}, y_{i_j})\}_{j=1}^B \subset \mathcal{S}$ of size $B < |\mathcal{S}|$, we have

$$\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = \frac{2}{B}\rho\sum_{j=1}^{B}\left[(1-\rho\bar{f}_{i_j})\left(-V_k\bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k}\right)\right] = 0. \tag{19}$$

Since interpolation is impossible when training with $\lambda > 0$, there exists at least one $n \in [N]$ for which $\rho\bar{f}_n \neq 1$. We consider two batches $\mathcal{S}'_i$ and $\mathcal{S}'_j$ of size B that differ by sample, (x_i, y_i) and (x_j, y_j). We have

$$\begin{aligned}\forall\ i,j \in [N]:\ 0 &= \frac{\partial \mathcal{L}_{\mathcal{S}'_i}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} - \frac{\partial \mathcal{L}_{\mathcal{S}'_j}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} \\ &= \frac{2}{B}\cdot\rho\left[(1-\rho\bar{f}_i)\left(-V_k\bar{f}_i + \frac{\partial \bar{f}_i}{\partial V_k}\right) - (1-\rho\bar{f}_j)\left(-V_k\bar{f}_j + \frac{\partial \bar{f}_j}{\partial V_k}\right)\right].\end{aligned} \tag{20}$$

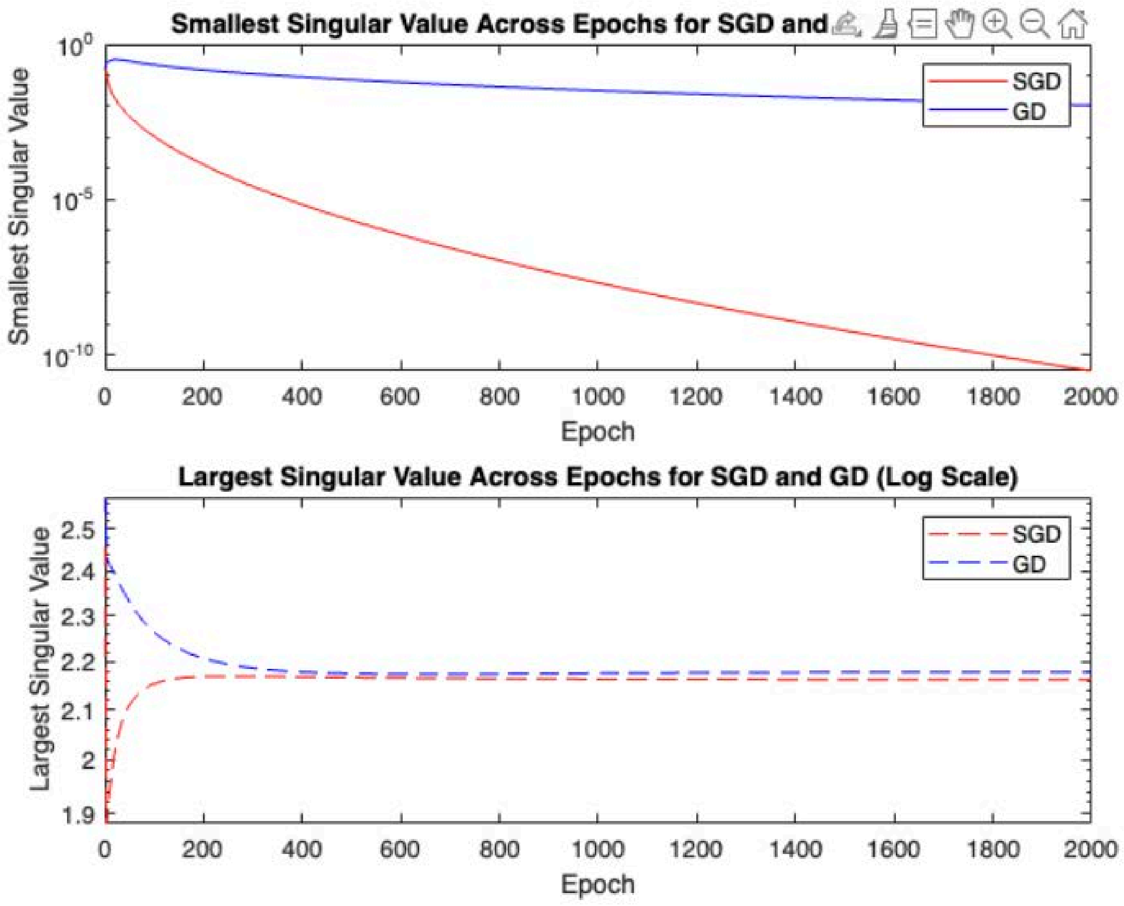


Figure A.7: Sparse regression $Wx = y$ with $N = 6$, $d = 6$; of the two components of y, one does not depend on the data. GD and SGD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. **Top:** The scale for the singular values is *logarithmic* to check the prediction for a slope that should be N times higher for SGD vs GD in the case of the smallest singular value in the null space of the data. Here the slope for the best linear fit is -0.00153 for GD and -0.00893 for GD with a goodness of fit (R-squared) above 0.9. The ratio of the slopes is ≈ 6 as predicted. **Bottom:** The largest singular value is in the span of the data, does not decay to zero and yields very similar rates of convergence for SGD and GD.

Assume that there exists a pair $i, j \in [N]$ for which $(1 - \rho\bar{f}_i)\bar{f}_i \neq (1 - \rho\bar{f}_j)\bar{f}_j$. Then, we can write

$$V_k = \frac{\left[(1 - \rho\bar{f}_i) \cdot \frac{\partial \bar{f}_i}{\partial V_k} + (1 - \rho\bar{f}_j) \cdot \frac{\partial \bar{f}_j}{\partial V_k}\right]}{[(1 - \rho\bar{f}_i)\bar{f}_i - (1 - \rho\bar{f}_j)\bar{f}_j]}. \tag{21}$$

Since $\frac{\partial \bar{f}_i}{\partial V_k}$ and $\frac{\partial \bar{f}_j}{\partial V_k}$ are matrices of rank ≤ 1 (see the analysis above), we obtain that V_k is of rank ≤ 2. Otherwise, assume that for all pairs $i, j \in [N]$, we have $\alpha = (1 - \rho\bar{f}_i)\bar{f}_i = (1 - \rho\bar{f}_j)\bar{f}_j$. In this case we obtain that for all $i, j \in [N]$, we have

$$(1 - \rho\bar{f}_i) \cdot \frac{\partial \bar{f}_i}{\partial V_k} = (1 - \rho\bar{f}_j) \cdot \frac{\partial \bar{f}_j}{\partial V_k} = U. \tag{22}$$

Therefore, since $\alpha = (1 - \rho\bar{f}_i)\bar{f}_i = (1 - \rho\bar{f}_j)\bar{f}_j$, by Equation 19,

$$0 = \frac{2}{B}\rho \sum_{j=1}^{B} \left[(1 - \rho\bar{f}_{i_j})\left(-V_k\bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k}\right)\right] = -2\rho\alpha V_k + 2\rho U. \tag{23}$$

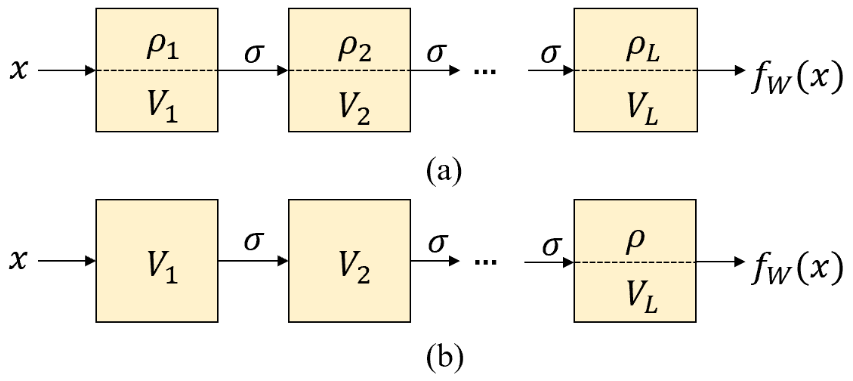


Figure A.8: An illustration of two equivalent parametrizations of our deep ReLU networks $f_W(x)$. (a) Each layer's weight matrix W_k was decomposed into its norm ρ_k and its normalized version V_k, where the layer index $k \in [L]$. σ represents the ReLU activation function. (b) Each layer's weight matrix was normalized with the LM, except for the top layer's weight matrix W_L (that is decomposed into a global ρ and the normalized weight matrix V_L). Normalizing the weight matrices with Lagrange Multipliers, as weight normalization (equivalent to LN) does, is different from the Batch Normalization, although both normalization techniques capture the relevant property of normalization – to make the dot product invariant to scaling.

Since the network cannot perfectly fit the dataset when trained with $\lambda > 0$, we obtain that there exists $i \in [N]$ for which $(1 - \rho \bar{f}_i) \neq 0$. Since $\bar{f}_i \neq 0$ for all $i \in [N]$, this implies that $\alpha \neq 0$. We conclude that V_k is proportional to U which is of rank ≤ 1.

$\square$

All gradient descent methods try to converge to points in parameter space that have zero or very small gradient, in other words they try to minimize $\|\dot{V}_k\|, \forall k$. Assuming separability (that is $\bar{f}_n > 0$) and $\lambda > 0$, $\ell_n = (1 - \rho \bar{f}_n) > 0, \forall n$, then Equation 15 implies

$$\|\dot{V}_k\| = \|\frac{2\rho}{N} \sum_{n \in B} \ell_n (\frac{\partial \bar{f}_n}{\partial V_k} - f_n V_k)\|, \tag{24}$$

which predicts that the norm of the SGD minibatch update should depend on the rank of V_k.

D.4 Origin of SGD noise

Lemma 1 shows that there cannot be convergence to a unique set of weights $\{V_k\}_{k=1}^L$ that satisfy equilibrium for all minibatches. When $\lambda = 0$, interpolation of all data points $(1 - \rho \bar{f}_n = 0, \forall n)$ is expected in the overparametrized case we consider: in this case, equilibrium can be reached without any constraint on the weight matrices. In this situation, the SGD noise is expected to disappear. Thus, during training with $\lambda > 0$, the solution $\{V_k\}_{k=1}^L$ is not the same for all samples: there is *no convergence to a unique solution* but instead fluctuations during training[5].

D.5 Experiments

In our experiments, we conducted binary classification with the CIFAR10 dataset [27] using the deep ReLU networks (see Figure A.8(b)). Specifically, we extracted images with class labels "1" and "2" from the CIFAR10 dataset, 10000 32×32 colour images were used for training and 2000 colour images for testing. The specific model architecture and implementation details are described below.

Model architecture. Our deep ReLU networks consists of four convolutional layers and two fully connected layers ($L = 6$), without biases in any of the layers. The four convolutional layers apply 3×3 convolutions with stride 2 and padding 0, the corresponding output channel numbers are 32,

[5]The absence of convergence of SGD to a unique solution is not surprising, in general, when the landscape is not convex.

64, 128, and 128. The final two fully connected layers project the 3200-dimensional output of the last convolutional layer to a 1024-dimensional vector before mapping it to 2 outputs. At the top layer, there is a global learnable parameter ρ that is the product of the Frobenius norms of weight matrices in all layers (see Figure A.8(b)). We used the ReLU activation function in all layers except for the last layer. The total number of model parameters is $3,519,335$.

Training and optimization. To implement the model introduced in Section A.8 (b). we used the equivalent weight normalization (WN) algorithm, freezing the weights of the WN parameter "g" [24] and normalized the $\{V_k\}_{k=1}^{L-1}$ matrices at each layer using their Frobenius norm. We trained our networks with the SGD optimizer (momentum 0.9), and tested two different types of loss functions (i.e., square loss and exponential loss). The hyperparameters included an initialization scale of the weight matrix at each layer set to 0.1, an initial learning rate (η) of 0.03 with a cosine annealing learning rate scheduler, a batch size (B) of 128, and 2000 training epochs. Our experiments were run on the NVIDIA RTX A6000 GPU (48GB VRAM).

E Learning rate for SGD

Consider the differential equation

$$\frac{dx}{dt} + \gamma(t)x = 0 \tag{25}$$

with solution $x(t) = x_0 e^{-\int \gamma(t)dt}$. The condition $\int \gamma(t)dt \to \infty$ corresponds to $\sum \gamma_n = \infty$. Conditions of this type are needed for asymptotic convergence to the minimum of the process $x(t)$. Consider now the "noisy" case $\frac{dx}{dt} + \gamma(t)(x + \epsilon(t)) = 0$: we need $\gamma(t)\epsilon(t) \to 0$ to eliminate the effect of the "noise" $\epsilon(t)$, implying at least $\gamma_n \to 0$. The need for $\sum(\gamma_n)^2 = 0$ may be seen considering the SDE $\frac{dx}{dt} + \gamma_n x = dW(t)$.