CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 144 **February 26, 2024**

SGD vs GD: high noise and rank shrinkage

Mengjia Xu[1,2], Tomer Galanti[1], Akshay Rangamani[1], Lorenzo Rosasco[1,3,4], Andrea Pinto[1], 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 with small minibatch size yields for neural networks much higher asymptotic fluctuations in the updates of the weight matrices than GD. It has also been often reported that SGD in deep RELU networks shows empirically a low-rank bias in the weight matrices. A recent theoretical analysis derived a bound on the rank and linked it to the size of the SGD fluctuations [25]. 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 component $W^{\perp}$ of the weight 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. Because of the larger number of updates required to go through all the training data, the convergence rate *per epoch* of these components is much faster for SGD than for GD. In practice, SGD has a much stronger bias than GD towards solutions for weight matrices W with high fluctuations – even when the choice of mini batches is deterministic – and low rank, provided the initialization is from a random matrix. Thus SGD with non-zero regularization, shows the coupled phenomenon of asymptotic noise and a specific low-rank bias– unlike GD.

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 across updates (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, as we suggest, a specific form of low rank bias.

1.1 Related Work

Stochastic gradient descent (SGD) is a widely used method for optimizing deep learning models [3]. Despite the inherent similarities between Stochastic Gradient Descent (SGD) and Gradient Descent (GD), recent research has highlighted various distinctions between the solutions learned by both algorithms. For example, in the context of stochastic convex optimization, SGD is known to converge within $O\left(\frac{1}{\epsilon^2}\right)$ iterations to a solution with ϵ excess expected error. In contrast, GD, using an equal number of iterations, might result in overfitting [4]. On the empirical side, it was observed that SGD with smaller batches generalizes more effectively than with larger batches [5, 6], and that GD. Despite numerous studies, the subtle yet significant effects of SGD, especially in comparison to GD regarding tendencies towards large fluctuations and low rank, are not completely understood. Current literature lacks comprehensive comparisons of SGD and GD in these respects.

In an effort to understand the success of deep learning, various papers have explored the implicit regularization effects of gradient-based optimization. A major focus of significant research in recent years has been the implicit bias of linear neural networks towards rank minimization. The majority of this interest centered around the matrix factorization problem, which is equivalent to training a depth-2 linear neural network with multiple outputs with respect to the square loss. For instance, [7] conjectured and provided both empirical and theoretical evidence that, with sufficiently small step sizes and initialization close to the origin, gradient descent on a full-dimensional factorization converges to the minimum nuclear norm solution. However, this conjecture was later refuted by [8], which demonstrated that norm minimization does not occur in a wide range of matrix factorization problems. [9] postulated that the implicit regularization in matrix factorization can be explained by rank minimization and also hypothesized that some notion of rank minimization may be key to explaining generalization in deep learning. [8] provided evidence that the implicit regularization in matrix factorization acts as a heuristic for rank minimization. Beyond factorization problems, [10] demonstrated that in linear networks with an output dimension of 1, gradient flow (GF) with respect to exponentially-tailed classification loss functions converges to networks where the weight matrix of every layer has a rank of 1.

In the non-linear case, the situation is more complex. An empirical study found that during minimization SGD spans a small subspace, implying an effective bias on the rank of the weight matrices [11]. A couple of papers tried to study the inductive bias of gradient-based methods to learn low-rank weight matrices from a theoretical standpoint. For example, in [12, 13, 14] they considered a setting where the model is at the global minima of the L_2 regularization subject to fitting all of the training samples. For example, [13] showed that in this setting, the weight matrices of a two-layer network become rank 1 at the global minimum when the data is assumed to lie on a one-dimensional manifold. This result was later extended in [12] for datasets that lie on higher dimensional spaces. Additionally, [14] discovered that in sufficiently deep ReLU networks, when fitting the data, the weight matrices in the topmost layers become low-rank at the global minimum. This observation is also related to the property of Neural Collapse [15, 1]. In separate work, [16] prove that for linear multilayer networks SGD, but not GD, has a non-zero probability to jump from a higher rank minimum to a lower rank one[1].

Despite recent advancements in understanding the tendency of weight matrices to be low-rank at the global minimum, the fundamental causes of this behavior during the optimization process are not yet fully understood. Previous studies [10] have demonstrated that when univariate linear networks are trained on binary classification tasks using exponentially-tailed loss functions through gradient

[1]This work assumes a matrix completion task and additive regularization instead of the (more natural) choice of regularizing the product of the norms of the layers (as we assume here, see Appendix and see [1]).

flow (GF), the networks tend to converge to weight matrices with a rank of one, especially when the data is linearly separable. Building on this, a more recent paper [17] found that when training ReLU networks with multiple linear top layers using GF, these top layers also tend to converge to rank one weight matrices.

In this work 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 describe relevant differences between GD and SGD: the latter is characterized by asymptotic intrinsic fluctuations in the weight matrices in the bottom and middle layers – even in the absence of any explicit randomness in the algorithm – which are coupled with a bias towards shrinking the components of the weight matrices in the null space of the data – which can be described as a specific form of a bias towards small rank. For one-layer linear networks we provide an analysis of convergence of SGD vs GD in a special but important case: 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 (measured over an epoch). Thus SGD is much more effective than GD at *pruning features that are not supported by the input or output 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 never converges, across its updates of the weights, to a perfect equilibrium: there is always, generically, SGD noise. We concluded that the underlying reason is a rank constraint in the SGD updates that depends on the size of the mini-batches – an observation that, to our knowledge, seems to have escaped previous studies. The argument can be seen by considering the SGD update equations, which are given here in terms of reparametrization of the weight matrices W_k using $\rho_k V_k = W_k$, $\rho_k = \|W_k\|$, $\rho = \Pi_{k=1}^L \rho_k$ (see [1]). We also define the output of the network as $g(x) = \rho f(x)$; $\bar{f}_n = y_n f_n > 0$, $\mu = \frac{1}{N}\sum_n \bar{f}_n$ and $M = \frac{1}{N}\sum_n \bar{f}_n^2$.

The normalized weight matrices V_k and ρ are first initialized, and then iteratively updated in the following manner

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

where $\mathcal{S}'$ is selected uniformly as a subset of the training set $\mathcal{S}$ of size B, $\eta > 0$ is the learning rate. It is important to emphasize that we assume here regularization on the product of the norms of the layers (instead of the sum of the norms). Such a regularization follows from normalizing each weight matrix by weight normalization; it is a natural choice to preserve homogeneity of the network.

A study of Equations 1 is in the Appendix. A main result is the following lemma (proof in the Appendix):

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.*

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 [18]. When $\lambda = 0$, interpolation of all data points is expected: in this case, the GD equilibrium can be reached without any fluctuation since the SGD-specific noise essentially disappears, as shown by the histograms on the left and the right hand side of Figure 10 in [1]. For $\lambda > 0$, however, the solution $\{V_k\}, k = 1, \cdots, L$ 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 predictions about SGD fluctuations

The analysis of section C in the Appendix leads to a few predictions about SGD noise 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 small.

(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 large fluctuations 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 25 the norm of the update of 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.

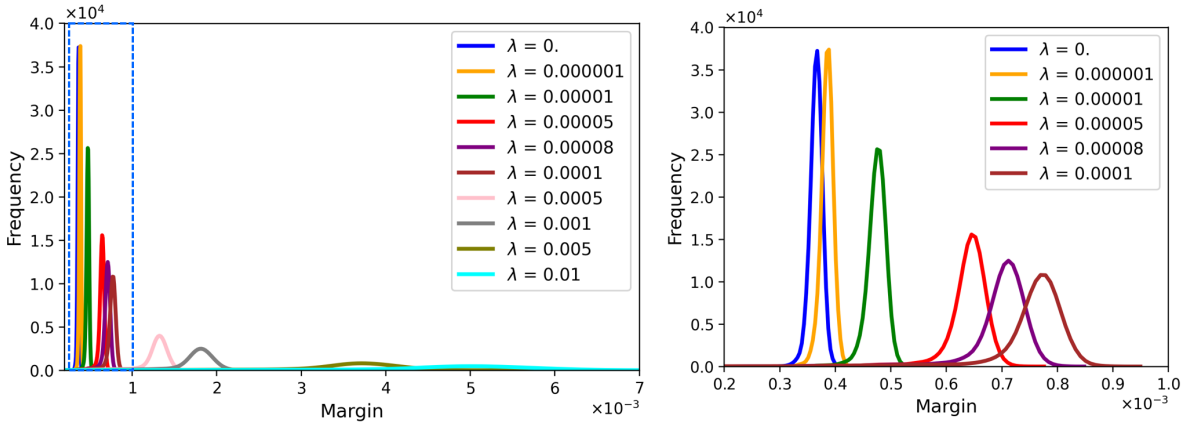
(4) Larger rank of V_k leads to larger $\|V_k(t+1) - V_k(t)\|$ as suggested by Equation 1: compare Figure 4(a) and Figure 4(b).

(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 17. 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 17 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).

2.2 Low-rank bias of SGD?

In previous work [1] we suggested that the rank constraint of the SGD updates not only implies SGD-specific fluctuations, as described in the previous sections, but also a bias towards low rank solutions[2]. This conjecture was formalized in an upper bound on the rank [18] which however is quite loose in most practical cases. The underlying intuition was based on Equations 1. In the case of $B = N$, the equations describe gradient descent for which it is well known that $\|V_k(t+1) - V_k(t)\|$ goes to zero with $t \to \infty$. In the case of $B = 1$, however, the right hand side of the V_k equation is not a gradient of the loss. Thus we cannot infer that the SGD update necessarily decreases in norm. The SGD update equations only suggests that at the end of one epoch V_k is the linear combination of rank 1 matrix updates, implying that V_k has rank $\leq N$. We will show in the next section, however, that for a linear network under the same conditions yielding SGD-specific fluctuations ($\lambda > 0$ and $B < N$) there is a SGD specific bias towards aggressively pruning the components of W that are in the null space of the data. We believe that this argument can be directly extended to multilayer, nonlinear networks, applying the linear result to each layer starting from the top layer to the bottom one and iterating.

[2] This bias reinforces a similar bias that SGD shares with GD – due to maximization of the margin under normalization (that can be inferred from [14])



(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, and $\|W\|$ the Frobenius norm of the weight matrix. The loss $\mathcal{L}$ is minimized by the gradient flow

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

The corresponding gradient descent iteration corresponds is

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

and the Stochastic Gradient descent (SGD) iteration is

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

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 where $S^t = S$ (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

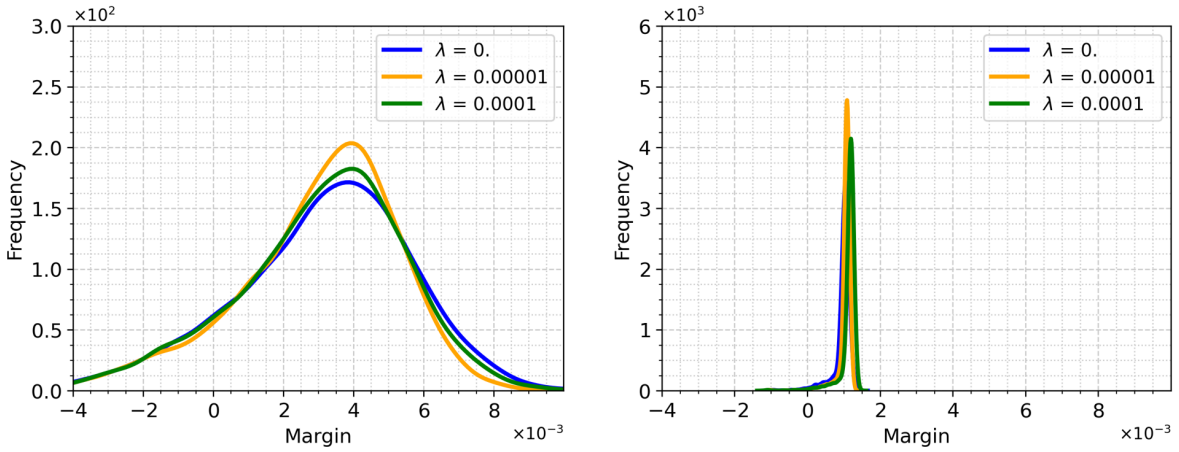


Figure 2: Margin distribution over all training data for binary classification on the CIFAR10, trained with GD using our deep ReLU model and varying λ. (a) We used the same training hyperparameters as previous experiments, but with a large batch size (B) of 10000, which matches the entire training data 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).

deterministically from 1 to N in each epoch. The key convention in the Neural Network literature is about epochs: one epoch means one pass over all the training examples; thus minibatch batch size is the number of training examples used for one update of the weights and the number of iterations per epoch is the number of minibatches per epoch. Example: with 1000 training examples, and batch size 500, then it will take 2 iterations to complete 1 epoch.

3.2 Bias of SGD towards shrinkage of the null space

Assume asymptotic equilibrium, that is $\frac{\partial \mathcal{L}}{\partial W} = 0$ (or $W(t+1) - W(t) = 0$) and assume $B = 1$.

In this case, Equation 5 for $S_t = 1$ becomes

$$\Delta W(t) = W(t+1) - W(t) = -2\eta(W(t)x_{i_t} - y_{i_t})x_{i_t}^T - 2\eta\lambda W(t). \tag{6}$$

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 for all t sufficientrly large

$$\Delta W(t) = -y_{i_t}x_{i_t}^T(\lambda I + x_{i_t}x_{i_t}^T)^{-1} \tag{7}$$

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 would be the rank of $y_i x_i^T$. Thus the assumption of equilibrium for SGD, that is zero fluctuations in W, implies that W has rank 1– – which is not consistent, in general, with a small square error in regression (for $d > 1$). Therefore, the assumption $\Delta W = 0$ *asymptotically must be wrong*. The situation is different for GD (corresponding to SGD 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 $\Delta W = 0$ does not lead to a contradiction: asymptotic equilibrium can be reached.

More succinctly, the GD and SGD updates have the form

$$W(t+1) - W(t) = -2\frac{\eta}{B}(W(t) - W^*)XX^T - 2\eta\lambda W(t) \tag{8}$$

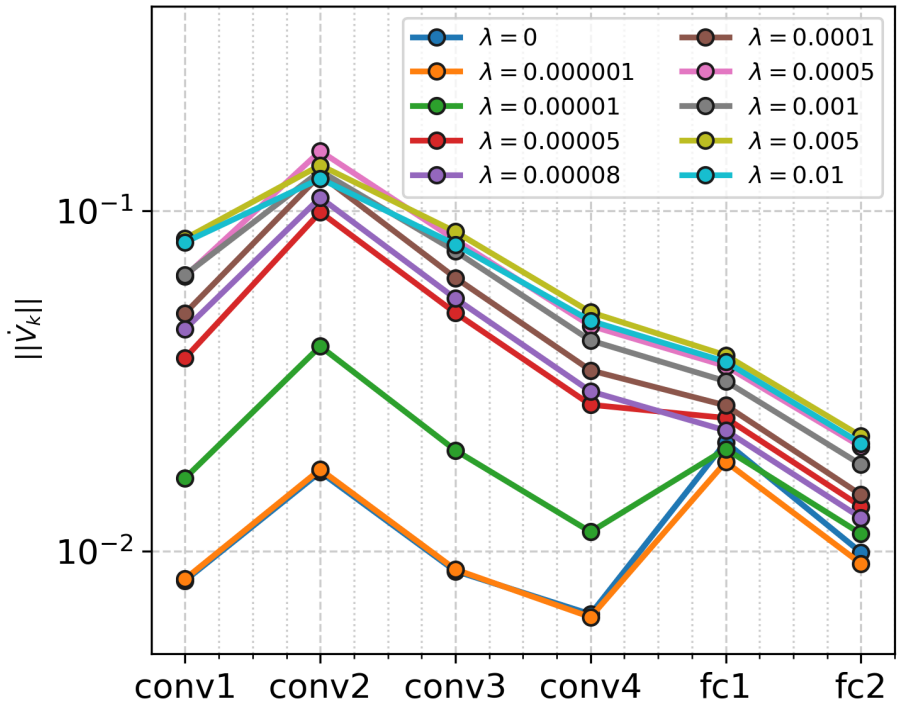


Figure 3: Layer-wise SGD noise $\|V_k(t+1) - V_k(t)\|$ in logarithmic scale by 10 different λ at "convergence". $\|V_k(t+1) - V_k(t)\|$ is small when λ is 0 or 1e-06. An increase of λ from 1e-05 to 0.01 generates larger fluctuations, especially for the first four convolution layers. The last two fully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

where X is the matrix composed of the x_i belonging to the minibatch, and $Y = W^*X$. It is important to notice that GD and SGD are quite different *wrt an epoch*: when the minibatch size is 1, W is updated N times in the case of SGD and only once in the case of GD. 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 $W^0 = 0$, and provided the step-size is chosen appropriately, both SGD and GD converge [**?**] and it if the data are generated by a linear low rank matrix, the correct rank is recovered. 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.

The formal version of the argument is the following observation:

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

Equation 8 can be rewritten as

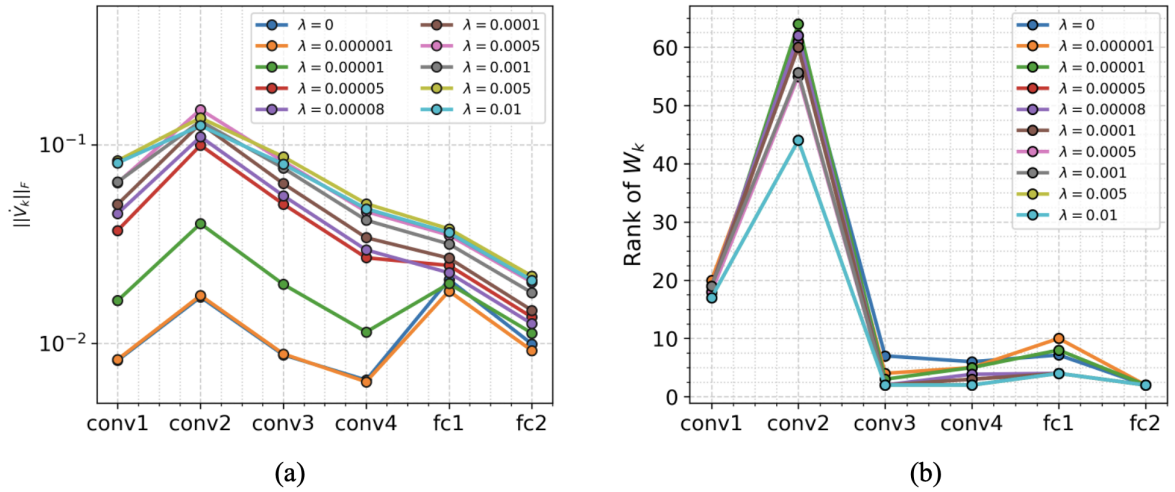


Figure 4: (a) Layer-wise asymptotic values of $\|V_k(t+1)-V_k(t)\|$ discretized and plotted on a logarithmic scale parametrized by 10 different λ. The SGD noise across layers is small when λ is set to 0 or 1e-06. An increase of λ from 1e-05 to 0.01 generates larger fluctuations, especially for the first four convolution layers. The last two fully connected layers tend to consist of low-rank matrices, corresponding to smaller $\|V_k(t+1) - V_k(t)\|$. (b) Layer-wise rank of V_k by 10 different λ 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 $\|V_k(t+1) - V_k(t)\|$ as shown in (a).

$$W^{\parallel}(t+1) = W^{\parallel}(t)[(1-2\eta\lambda)I - \frac{\eta\lambda}{B}XX^T] - 2\eta\lambda YX^T \tag{9}$$

and

$$W^{\perp}(t+1) - W^{\perp}(t) = -2\eta\lambda W^{\perp}(t) \tag{10}$$

The two equation are independent; equation 10 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

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 D) where t is epochs. It is well-known that SGD and GD converge to the regularized solutions (which is the minimum norm solution for $\lambda \to 0$) at a similar rate when W is initialized from $W_0 = 0$ (see Figure A.1 in Appendix). The situation is different when we 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 (per epoch) 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 *per epoch shrinkage of small eigenvalues* for SGD vs GD, which is correlated with increasing gradient noise

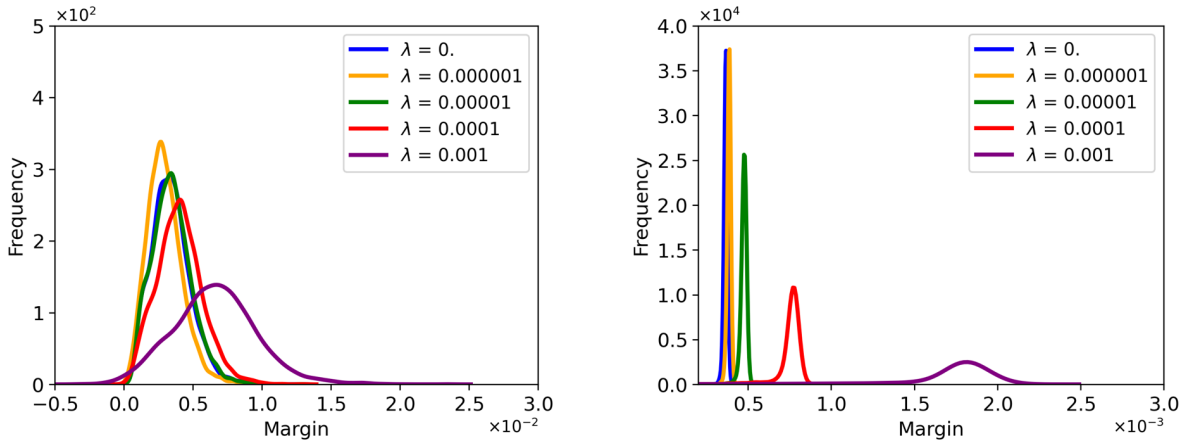


Figure 5: Margin distributions over all training data for binary classification on the CIFAR dataset trained with **cross entropy loss** in (a) and **square loss** in (b) using different λ. The results in (a) verified the prediction presented in Section 2.1, i.e., the presence of the SGD-specific noise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

– that is norm of $\|\Delta W\|$ – and with λ. This effect is qualitatively similar to a bias towards low rank[3]. SGD maintains its bias when 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 long epoch for GD and SGD) gives similar results (Figure A.4). Figure A.5 shows that the asymptotic noise in the training of SGD does not decrease (for constant 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" at the level of minibatches updates[4]: 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 normally 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 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 rank constraint in the SGD updates of W_k and the constraint of minimizing the error for each data point. An equivalent description is that SGD never finds an asymptotic W that is the best fit to all the data but instead finds a sequence of similar W, each one updated to fit the last minibatch.

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

[3]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).

[4]It does in expectation but its variance is never zero, see also [19].

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.

Under the square loss the role of weight decay is key to all the results here. Interestingly, there is growing evidence that weight decay yields better generalization in state-of-the-art systems such as transformers. It is equally important to emphasize that weight decay is not necessary to yield SGD specific noise when exponential loss functions are used instead of the square loss. In this context, it may also be interesting to explore the role of *early stopping* when weight decay is absent since it is well known that early stop provides an implicit regularization effect.

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 Rademacher complexity defined using different norms may depend on rank. The main effect, however, is likely to be indirect. We conjecture that the specific bias towards small rank for random initialization 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, possibly 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 16 and 17 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

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

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:

$$\dot{V}_k = \frac{2}{N} \rho \sum_n \frac{e^{-\rho \bar{f}_n}}{1 + e^{-\rho \bar{f}_n}} (-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}). \tag{12}$$

The precise mechanism behind the specific 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 (per epoch) to zero under SGD than under GD. 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. An extension of this result to multilayer, nonlinear networks has been empirically observed in Appendix E, indicating that the bias towards low rank by SGD is not limited to linear cases. However, a comprehensive theoretical justification for this phenomenon in deep, nonlinear networks remains an open question.

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] Idan Amir, Tomer Koren, and Roi Livni. Sgd generalizes better than gd (and regularization doesn't help), 2021.

[5] 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.

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

[8] 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.

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

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

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

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

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

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

[15] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[16] Zihan Wang and Arthur Jacot. Implicit bias of sgd in l_2-regularized linear dnns: One-way jumps from high to low rank, 2023.

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

[18] 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.

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

[20] 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.

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

[22] 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.

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

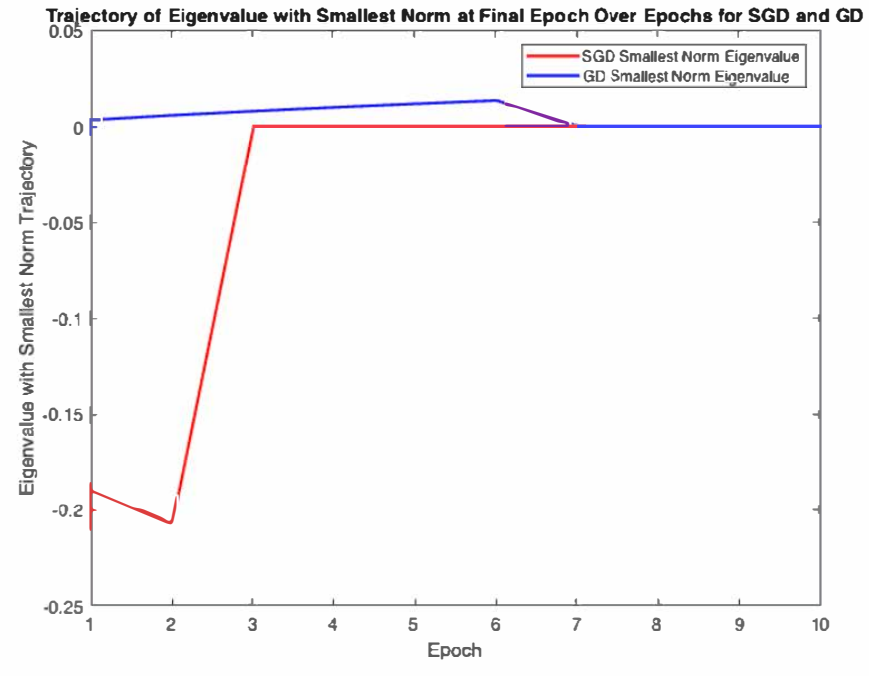


Figure A.1: Solving $Wx = y$, for a data matrix X with $d = 11$ and $N = 10$; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 10. Initialization is from $W = 0$. The final MSE for SGD is 0.879681 and for GD is 1.524118.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

A Linear experiments

All the figures in the Appendix describe key experiments with linear regression that help visualize the theoretical predictions in the main text.

B 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 [20], see the proof in [21]. 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 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 [22].

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 **??**) to solve this classification problem. For simplicity, we consider networks $f_W : \mathbb{R}^d \to \mathbb{R}^p$ of the following form $f_W(x) = W_L \sigma\left(W_{L-1} \ldots \sigma\left(W_1 x\right) \ldots\right)$, 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[5] weight matrices $V_k = \frac{W_k}{\|W_k\|}$ and define $\rho_k = \|W_k\|$

[5] We choose the Frobenius norm here.

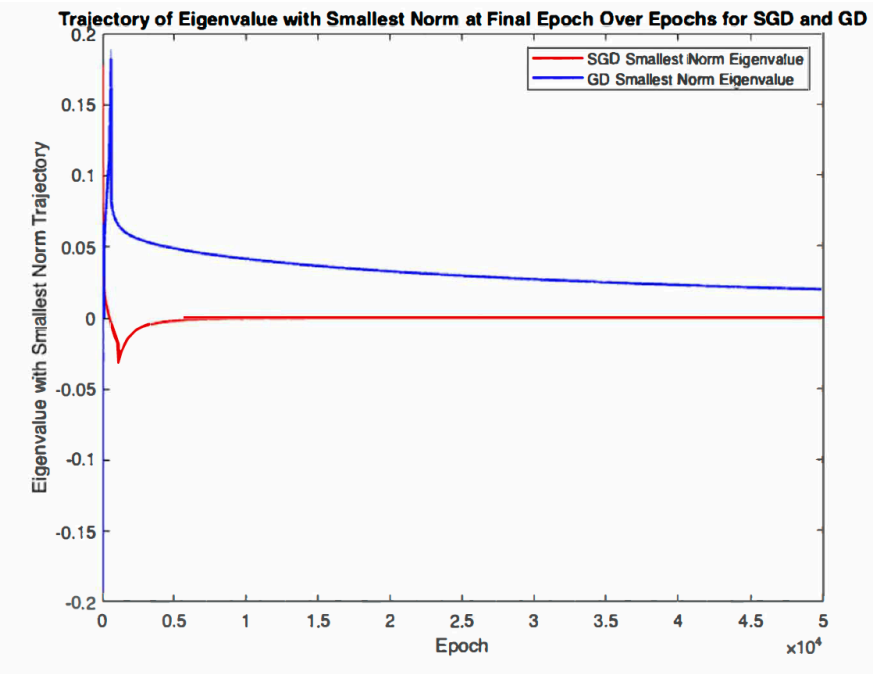


Figure A.2: Solving $Wx = y$, for a data matrix X with $d = 11$ and $N = 10$; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 50000. Initialization is from W random. The final MSE for SGD is 0.126220 and for GD is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

obtaining $f_W(x) = \rho_L V_L \sigma\left(\rho_{L-1} \dots \sigma\left(\rho_1 V_1 x\right) \dots\right)$, see Figure **??**(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 $f_W(x) = \rho f_V(x) = \rho V_L \sigma\left(V_{L-1} \dots \sigma\left(V_1 x\right) \dots\right)$, as shown in Figure **??**(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 **??**(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, 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

$$\begin{aligned} \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L}) :&= \frac{1}{N} \sum_n (\rho f_n - y_n)^2 + \sum_{k=1}^{L} \nu_k (\|V_k\|^2 - 1) + \lambda \rho^2 \\ &= \frac{1}{N} \sum_n (1 - \rho \bar{f}_n)^2 + \sum_{k=1}^{L} \nu_k (\|V_k\|^2 - 1) + \lambda \rho^2, \end{aligned} \tag{13}$$

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, $\forall\, n \in [N] :\ \bar{f}_n > 0$.

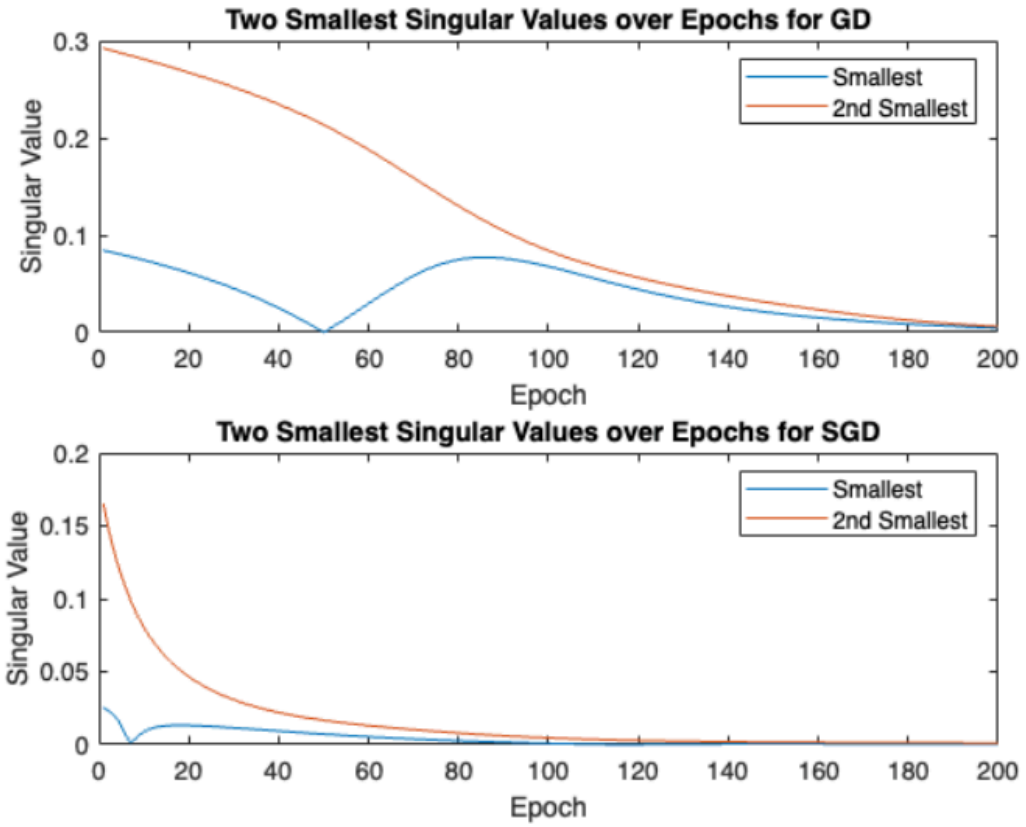


Figure A.3: Regression with sparse features. Solving $Wx = y$, for a data matrix X with $d = 10$ and $N = 5$ with only two of components of x being relevant; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. The number of epochs is 200. Initialization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

Furthermore, when $\sum_{n=1}^{N} \bar{f}_n > 0$, we say that *average separability* is satisfied. The minimum of $\mathcal{L}_S$ 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. [6] 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.

C Theoretical Analysis

C.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

[6] 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.

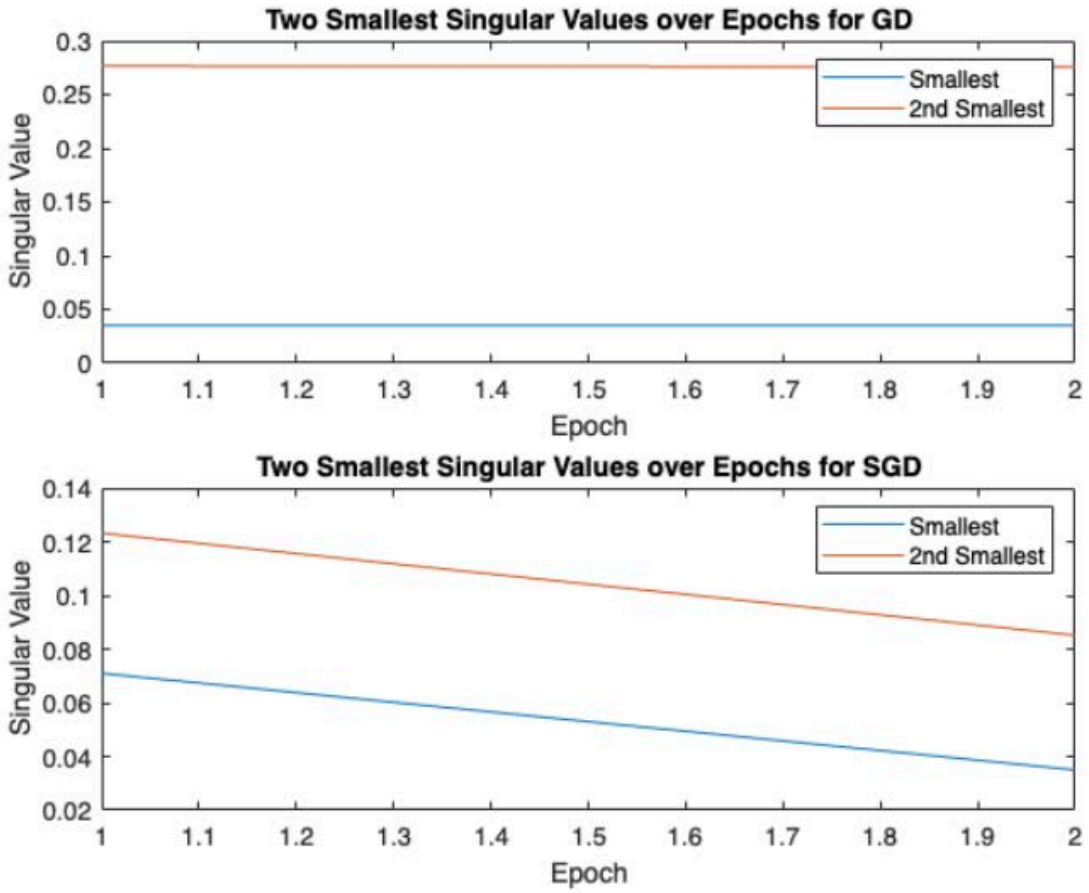


Figure A.4: Regression with sparse features in an online setting. One epoch with $N = 200$, $d = 10$. Solving $Wx = y$, for a data matrix X with $d = 10$ and $N = 5$ with only two of components of x being relevant; SGD and GD are used with regularization $\lambda = 0.01$ and optimal learning rate η as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

matrices) and V_k (the normalized weight matrices) are as follows

$$\begin{aligned}\dot{\rho} &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^L)}{\partial \rho} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho \\ \dot{V}_k &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\rho\frac{\partial \bar{f}_n}{\partial V_k} - 2\nu_k V_k,\end{aligned} \tag{14}$$

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

$$\nu_k = \frac{1}{N}\sum_n (\rho\bar{f}_n - \rho^2 f_n^2) = \frac{1}{N}\sum_n \rho\bar{f}_n(1-\rho f_n). \tag{15}$$

Thus the gradient flow is the following dynamical system

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

The gradient flow for the logistic loss would result in

$$\dot{\rho} = \frac{2}{N}\sum_n \frac{e^{-\rho\bar{f}_n}}{1+e^{-\rho\bar{f}_n}}\bar{f}_n - 2\lambda\rho \quad \text{and} \quad \dot{V}_k = \frac{2}{N}\rho\sum_n \frac{e^{-\rho\bar{f}_n}}{1+e^{-\rho\bar{f}_n}}(-V_k\bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}). \tag{17}$$

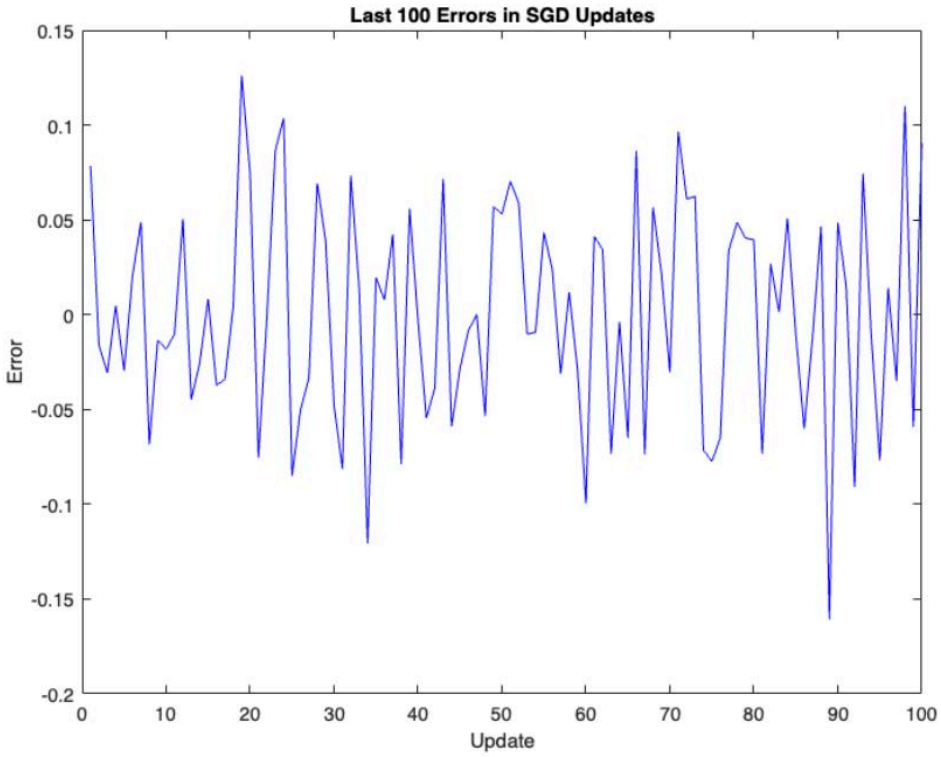


Figure A.5: Regression in an online setting. One epoch with $N = 100000$, $d = 10$. Solving $Wx = y$, for a data matrix X with $d = 10$; SGD is used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. Initialization is from random W. The updates are fully deterministic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

C.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

$$\begin{aligned}
\rho &\leftarrow \rho - \eta \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial \rho} = \rho - \eta \frac{2}{B} \sum_{(x_n, y_n) \in \mathcal{S}'} (1 - \rho \bar{f}_n) \bar{f}_n - 2\eta\lambda\rho \\
V_k &\leftarrow V_k - \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = V_k - \eta \frac{2}{B} \sum_{(x_n, y_n) \in \mathcal{S}'} (1 - \rho \bar{f}_n)(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}),
\end{aligned} \tag{18}$$

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 $\eta > 0$.

C.3 No equilibrium

The Lemma 1 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. The Lemma is

Lemma 2. *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.*

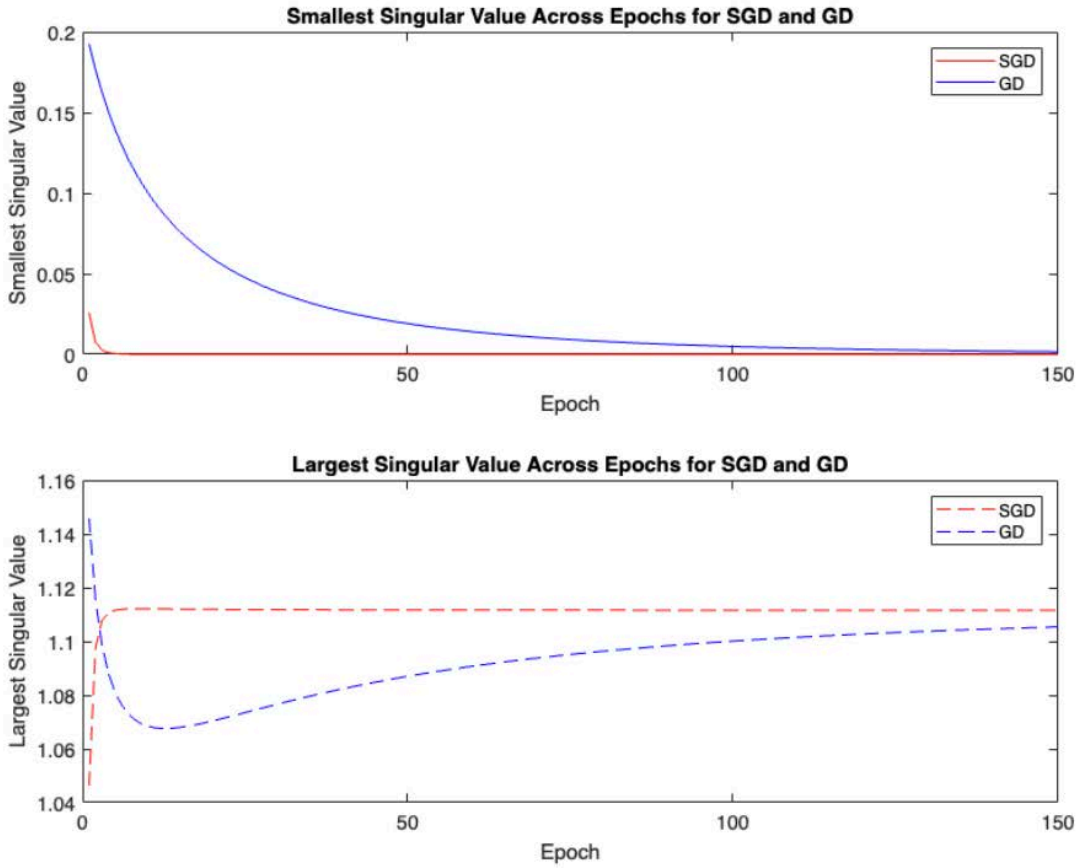


Figure A.6: Sparse regression with $N = 10$, $d = 2$; of the two components of y, one does not depend on the data. Solving $Wx = y$; GD and SGD are used with regularization $\lambda = 0.01$, over 150 epochs, and an optimal learning rate η as described in the text.

Proof. Let $f_V(x) = V_L\sigma(V_{L-1}\dots\sigma(V_1x)\dots)$ 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 $\sigma(v) = \mathrm{diag}(\sigma'(v)) \cdot v$ (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,

$$\begin{aligned} f_V(x) &= V_L\sigma(V_{L-1}\dots\sigma(V_1x)\dots) \\ &= V_L \cdot D_{L-1}(x;V)\cdots D_1(x;V)\cdot V_1 \cdot x, \end{aligned} \tag{19}$$

where $D_l(x;V) = \mathrm{diag}[\sigma'(u_l(x;V)))]$ and $u_l(x;V) = V_l\sigma(V_{l-1}\dots\sigma(V_1x)\dots)$. 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 $\mathcal{V}$.

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 $b = D_{k-1}(x)\cdot V_{k-1}\cdots V_1x$. 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 $\frac{\partial a^\top Xb}{X} = ab^\top$, 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.

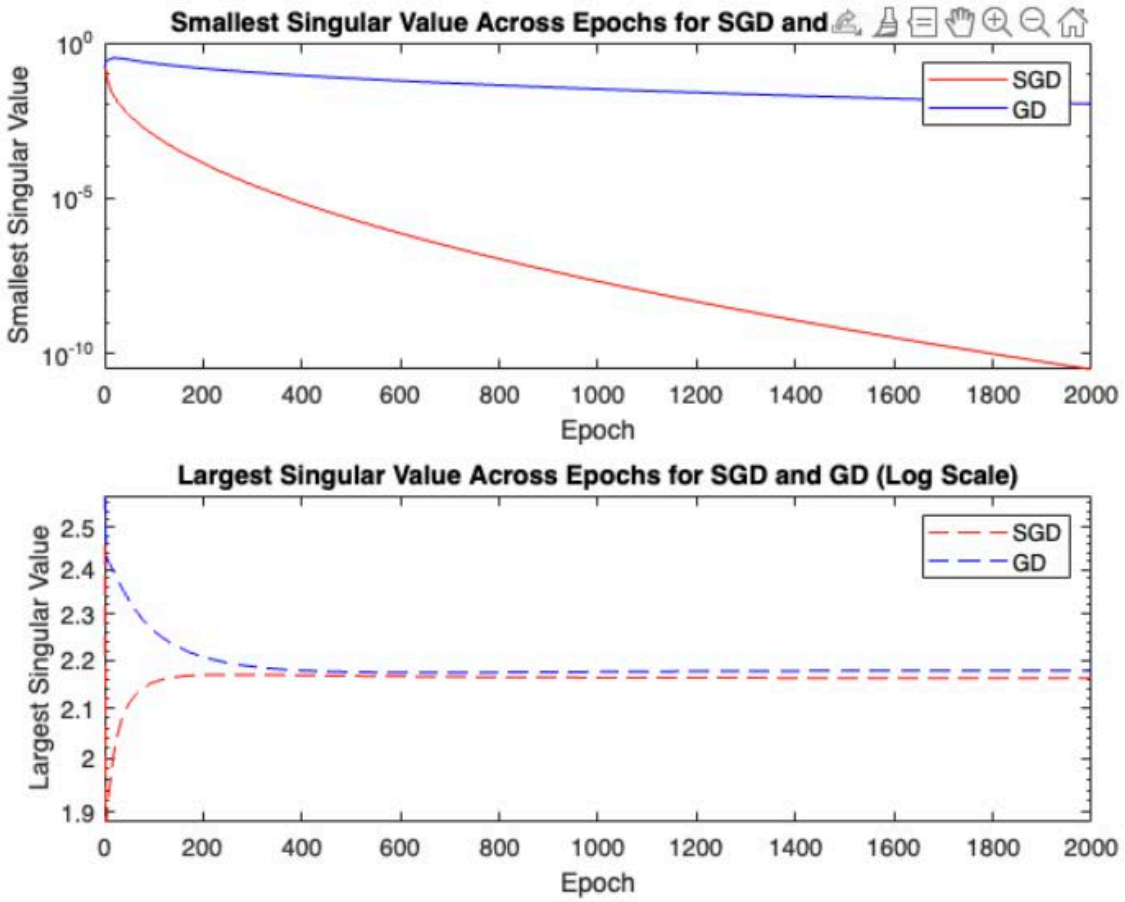


Figure A.7: Sparse regression $Wx = y$ with $N = 6$, $d = 6$; of the two components of y, one does not depend on the data. GD and SGD are used with regularization $\lambda = 0.01$ and optimal learning rate η as described in the text. **Top:** The scale for the singular values is *logarithmic* to check 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.

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

$$\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = \frac{2}{B}\rho \sum_{j=1}^{B} \left[\left(1 - \rho \bar{f}_{i_j}\right) \left(-V_k \bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k} \right) \right] = 0. \tag{20}$$

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

$$\begin{aligned} \forall\ i, j \in [N]:\ 0 &= \frac{\partial \mathcal{L}_{\mathcal{S}_i'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} - \frac{\partial \mathcal{L}_{\mathcal{S}_j'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} \\ &= \frac{2}{B} \cdot \rho \left[\left(1 - \rho \bar{f}_i\right) \left(-V_k \bar{f}_i + \frac{\partial \bar{f}_i}{\partial V_k} \right) - \left(1 - \rho \bar{f}_j\right) \left(-V_k \bar{f}_j + \frac{\partial \bar{f}_j}{\partial V_k} \right) \right]. \end{aligned} \tag{21}$$

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

$$V_k = \frac{\left[(1 - \rho \bar{f}_i) \cdot \frac{\partial \bar{f}_i}{\partial V_k} + (1 - \rho \bar{f}_j) \cdot \frac{\partial \bar{f}_j}{\partial V_k} \right]}{\left[(1 - \rho \bar{f}_i)\bar{f}_i - (1 - \rho \bar{f}_j)\bar{f}_j \right]}. \tag{22}$$

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

$$\left(1 - \rho \bar{f}_i\right) \cdot \frac{\partial \bar{f}_i}{\partial V_k} = \left(1 - \rho \bar{f}_j\right) \cdot \frac{\partial \bar{f}_j}{\partial V_k} = U. \tag{23}$$

Therefore, since $\alpha = (1 - \rho \bar{f}_i)\bar{f}_i = (1 - \rho \bar{f}_j)\bar{f}_j$, by Equation 20,

$$0 = \frac{2}{B}\rho \sum_{j=1}^{B} \left[\left(1 - \rho \bar{f}_{i_j}\right) \left(-V_k \bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k} \right) \right] = -2\rho\alpha V_k + 2\rho U. \tag{24}$$

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.

□

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 16 implies

$$\|\dot{V}_k\| = \|\frac{2\rho}{N} \sum_{n \in B} \ell_n (\frac{\partial \bar{f}_n}{\partial V_k} - f_n V_k)\|, \tag{25}$$

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

C.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[7].

C.5 Experiments

In our experiments, we conducted binary classification with the CIFAR10 dataset [23] using the deep ReLU networks (see Figure **??**(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, 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 **??**(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 **??** (b). we used the equivalent weight normalization (WN) algorithm, freezing the weights of the WN parameter "g" [20] 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).

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

D Learning rate for SGD

Consider the differential equation

$$\frac{dx}{dt} + \gamma(t)x = 0 \tag{26}$$

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 $\frac{dx}{dt} + \gamma_n x = dW(t)$.

E Support Selection: Extension to Deep Nonlinear Networks

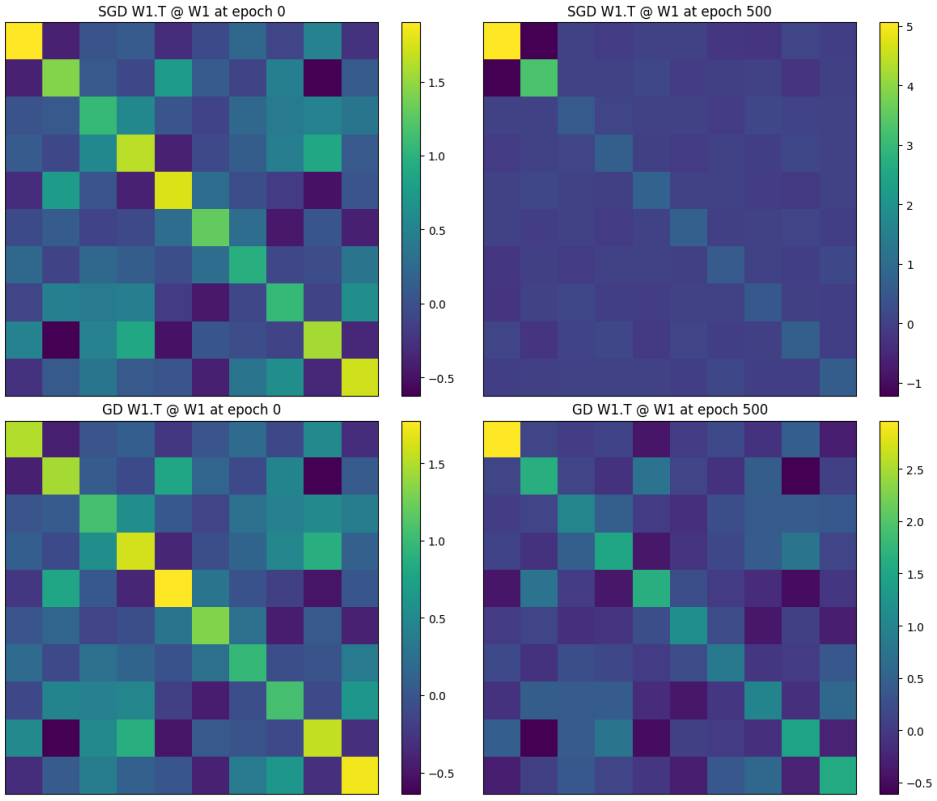


Figure A.8: **Comparative Analysis of Feature Selection Efficiency in SGD vs. GD:** This figure presents heatmaps of the Gram matrix from the first weight layer of a deep neural network, both at the start (epoch 0) and end (epoch 500) of training. The transition from dense to sparse matrices illustrates the network's learning process, highlighting SGD's superior efficiency in identifying and focusing on the most relevant features for predicting a sparse polynomial target function. The marked contrast in sparsity levels between SGD and GD underscores SGD's capability in discerning critical features quickly and accurately, a key factor in its faster convergence and improved optimization performance.

To complement the statement of Observation 1 1, we explore the implication of bias toward low-rank and the ability to identify support features in multilayer, nonlinear networks. Our empirical findings suggest a marked efficiency in SGD's ability to identify relevant features in deep neural network architectures, as compared to its GD counterpart.

Specifically, our experiment involved training a deep feed-forward neural network, composed of five layers with 20 neurons each, equipped with ReLU activation functions. The network was tasked with learning a highly sparse polynomial target function $f : \mathbb{R}^{10} \mapsto \mathbb{R}$, wherein only the first two out of ten

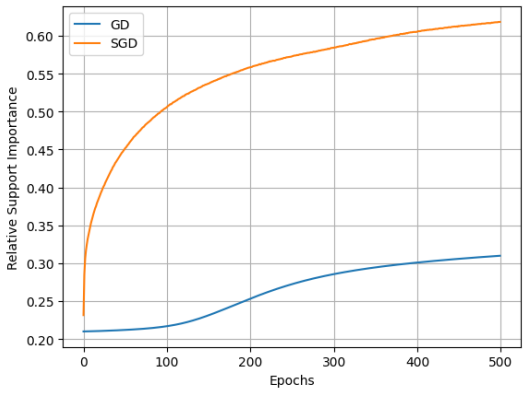


Figure A.9: **Visualization of the relative feature importance over epochs**: This figure highlights SGD's faster and more precise identification of critical support features compared to GD. This plot quantifies the efficiency of SGD in discerning the significance of features, showcasing its accelerated capability in pinpointing relevant features.

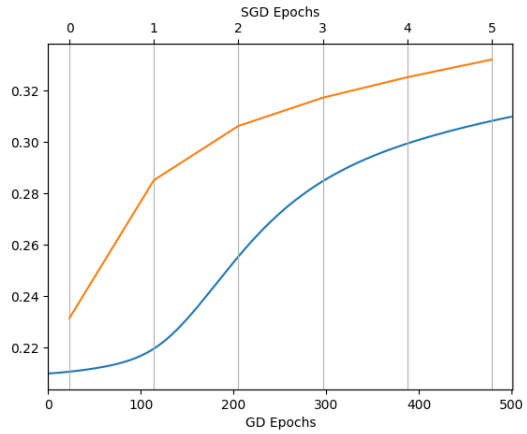


Figure A.10: **Comparison of GD and SGD in identifying support features within a comparable number of update steps**: This figure illustrates the similar efficiency despite the stark difference in epochs required (500 for GD vs. 6 for SGD). This plot underscores the optimization's strong dependency on update frequency.

features significantly contributed to the function. Utilizing a dataset of 10,000 randomly generated samples $(\mathbf{x}, y)$ from f, both GD and SGD were applied over 500 epochs, with a batch size of 64, a learning rate of 10^{-2} and a weight decay of 10^{-4}.

An initial analysis involved plotting a heatmap of the Gram matrix of the first weight layer for GD/SGD at epochs 0 and 500 A.9. The Gram matrix, which reflects the correlation between the 10 features of the target function f within the first weight matrix W_1, transitioned from a densely populated at initialization to a sparser structure, post-training. We also note that the degree of sparsity is much more pronounced with SGD than with GD; indicating that the SGD trained network figured out more confidently which are the relevant features for prediction.

We also quantifies relative feature importance across both optimization strategies over epochs A.10. To do so we compute $\sum_s \text{diag}_s / \sum_i \text{diag}_i$ where diag_s are the diagonal elements of the Gram matrix representing auto-correlation of the support features, and diag_i are the diagonal elements for all features. This metric demonstrates the accelerated and more precise identification of critical features by SGD compared to GD. The empirical observations reveal that SGD not only identifies the relative importance of support features much more rapidly than GD but also does so with greater certainty.

Lastly, we plot a comparison of the efficiency of GD and SGD in identifying support features within a similar number of update steps A.10. While GD achieves 500 parameter updates over 500 epochs, whereas SGD requires only 6 epochs to accomplish the same number of updates, utilizing a batch size of 64 from a dataset of 10,000 samples. The analysis reveals that the rate and strength with which the network identifies relevant support features are remarkably similar between the two methods. This observation emphasizes the critical role of update frequency in the optimization process.