Theory III: Dynamics and Generalization in Deep Networks*

Andrzej Banburski [1], Qianli Liao[1], Brando Miranda[1], Tomaso Poggio[1],
Lorenzo Rosasco[1], Bob Liang[1], and Jack Hidary[2]

[1]Center for Brains, Minds and Machines, MIT
[1]CSAIL, MIT
[2]Alphabet (Google) X

Abstract

We review recent observations on the dynamical systems induced by gradient descent methods used for training deep networks and summarize properties of the solutions they converge to. Recent results by [1] illuminate the apparent absence of "overfitting" in the special case of linear networks for binary classification. They prove that minimization of loss functions such as the logistic, the cross-entropy and the exponential loss yields asymptotic convergence to the maximum margin solution for linearly separable datasets, independently of the initial conditions. Here we discuss the case of nonlinear multilayer DNNs near zero minima of the empirical loss, under exponential-type losses and square loss, for several variations of the basic gradient descent algorithm, including a new NMGD (norm minimizing gradient descent) version that converges to the minimum norm fixed points of the gradient descent iteration. Our main results are:

- gradient descent algorithms with weight normalization constraint achieve generalization;
- the fundamental reason for the effectiveness of existing weight normalization and batch normalization techniques is that they are approximate implementations of maximizing the margin under unit norm constraint;
- without unit norm constraints some level of generalization can still be obtained for not-too-deep networks because the balance of the weights across different layers, if present at initialization, is maintained by the gradient flow[2].

In the perspective of these theoretical results, we discuss experimental evidence around the apparent absence of "overfitting", that is the observation that the expected classification error does not get worse when increasing the number of parameters. Our explanation focuses on the implicit normalization enforced by algorithms such as batch normalization. In particular, the control of the norm of the weights is related to Halpern iterations for minimum norm solutions which are equivalent to regularization with vanishing $\lambda(t)$.

*This replaces previous versions of Theory III, that appeared on Arxiv or on the CBMM site. The basic analysis is reformulated with comments on work that appeared after the original version of our memos.

1 Introduction

In the last few years, deep learning has been tremendously successful in many important applications of machine learning. However, our theoretical understanding of deep learning, and thus the ability of developing principled improvements, has lagged behind. A satisfactory theoretical characterization of deep learning is emerging. It covers the following questions: 1) *representation power* of deep networks 2) *optimization* of the empirical risk 3) *generalization properties* of gradient descent techniques — why the expected error does not suffer, despite the absence of explicit regularization, when the networks are overparametrized? We refer to the latter as the non-overfitting puzzle, around which several recent papers revolve (see among others [3, 4, 5, 6, 7]). This paper addresses the third question.

2 Deep networks: definitions and properties

Definitions We define a deep network with K layers with the usual coordinate-wise scalar activation functions $\sigma(z) : \quad \mathbf{R} \to \mathbf{R}$ as the set of functions $f(W; x) = \sigma(W^K \sigma(W^{K-1} \cdots \sigma(W^1 x)))$, where the input is $x \in \mathbf{R}^d$, the weights are given by the matrices W^k, one per layer, with matching dimensions. We use the symbol W as a shorthand for the set of W^k matrices $k = 1, \cdots, K$. For simplicity we consider here the case of binary classification in which f takes scalar values, implying that the last layer matrix W^K is $W^K \in \mathbf{R}^{1,K_l}$. The labels are $y_n \in \{-1, 1\}$. The weights of hidden layer l are collected in a matrix of size $h_l \times h_{l-1}$. There are no biases apart form the input layer where the bias is instantiated by one of the input dimensions being a constant. The activation function in this paper is the ReLU activation.

Positive one-homogeneity For ReLU activations the following positive one-homogeneity property holds $\sigma(z) = \frac{\partial \sigma(z)}{\partial z} z$. For the network this implies $f(W; x) = \prod_{k=1}^K \rho_k \tilde{f}(V_1, \cdots, V_K; x_n)$, where $W_k = \rho_k V_k$ with the Frobenius norm $||V_k|| = 1$ (for convenience). This implies the following property of ReLU networks w.r.t. their Rademacher complexity:

$$\mathbb{R}_N(\mathbb{F}) = \rho_1 \cdots \rho_K \mathbb{R}_N(\tilde{\mathbb{F}}), \tag{1}$$

where $\mathbb{F}$ is the class of neural networks described above and accordingly $\tilde{\mathbb{F}}$ the corresponding class of normalized neural networks. This invariance property of the function f under transformations of W_k that leave the product norm the same is typical of ReLU (and linear) networks. In the paper we will refer to the norm of f meaning the product $\rho = \prod_{k=1}^K \rho_k$ of the Frobenius norms of the K weight matrices of f. Thus $f = \rho \tilde{f}$. Note that

$$\frac{\partial f}{\partial \rho_k} = \frac{\rho}{\rho_k} \tilde{f}. \tag{2}$$

Structural property The following structural property of the gradient of deep ReLU networks is sometime useful (Lemma 2.1 of [8]):

$$\sum_{i,j} \frac{\partial f(x)}{\partial W_k^{i,j}} W_k^{i,j} = f(x); \tag{3}$$

for $k = 1, \cdot, K$. Equation 3 can be rewritten as an inner product

$$\left(\frac{\partial f(x)}{\partial W}\right)^T W = f(x) \tag{4}$$

where W is the vectorized representation of the weight matrices W_k for each of the different layers (each matrix is a vector)

Gradient flow and continuous approximation We will speak of the gradient flow of the empirical risk L (or sometime of the flow of f if the context makes clear that one speaks of the gradient of L) referring to

$$\dot{W} \equiv \frac{dW}{dt} = -\gamma(t)\nabla_W(L(f)), \tag{5}$$

where $\gamma(t)$ is the learning rate. In the following we will mix the continuous formulation with the discrete version whenever we feel this is appropriate for the specific statement. We are well aware that the two are not equivalent but we are happy to leave a careful analysis – especially of the discrete case – to better mathematicians.

Maximization by exponential With $\tilde{f}$ being the normalized network (weights at each layer are normalized by the Frobenius norm of the layer matrix) and ρ being the product of the Frobenius norms, the exponential loss $L(f) = \sum_n e^{-y_n f(x_n)} = \sum_n e^{-\rho y_n \tilde{f}(x_n)}$ approximates for "large" ρ a max operation, selecting among all the data points x_n the ones with the smallest margin $\rho\tilde{f}$. Thus minimization of $L(f)$ for large ρ corresponds to margin maximization

$$\arg\min L(f) \approx \arg\max_{V_k=1} \quad \min_n y_n \tilde{f}(x_n). \tag{6}$$

A more formal argument may be developed extending theorems of [9] to the nonlinear case.

3 A semi-rigorous theory of the optimization landscape of Deep Nets: Bezout theorem and Boltzman distribution

In [10, 11] we consider Deep Networks in which each ReLU nonlinearity is replaced by a univariate polynomial approximating it. Empirically the network behaves in a quantitatively identical way in our tests. We then consider such a network in the context of regression under a square loss function. As ususal we assume that the network is over-parametrized, that is the number of weights D is larger than the number of data points N. The critical points of the gradient consist of

- global minima corresponding to interpolating networks for which $f(x_i) - y_i = 0 \;\; for \; i = 1, \cdots, N$;
- critical points which correspond to saddles and to local minima for which the loss is not zero but $\nabla_W \sum_{i=1}^N L(f(x_i), y_i) = 0$,

In the case of the global, interpolating minimizers, the function f is a polynomial in the D weights (and also a polynomial in the inputs x). The degree of each equation is determined by the degree of the univariate polynomial P and by the number of layers K. Since the system of polynomial equations, unless the equations are inconsistent, is generically underdetermined – as many equations as data points in a larger number of unknowns – Bezout theorem suggests an infinite number of *degenerate global minima*, under the form of Z *regions* of zero empirical error (the set of all solutions is an algebraically closed set of dimension at least $Z = D - N$). Notice that if an underdetermined system is chosen at random, the dimension of Z is equal to $D - N$ with probability one.

The critical points of the gradient that are not global minimizers are given by the set of equations $\nabla_W \sum_{i=1}^{N} L(f(x_i), y_i) = 0$. This is a set of D polynomial equations in D unknowns: $\sum_{i=1}^{N} (f(x_i) - y_i) \nabla_W f(x_i) = 0$. In this case, we generically expect a set of *isolated critical points*.

Thus, we have

Theorem 1 *(informal statement): There are a large number of global zero-error minimizers which are highly degenerate; the other critical points – saddles and local minima – are generically (that is with probability one) non-degenerate.*

The second part of our argument (in [11]) is that SGD concentrates on degenerate minima. The argument is based on the similarity between a Langevin equation and SGD and on the fact that the Boltzman distribution is formally the asymptotic "solution" of the stochastic differential Langevin equation and also of SGDL, defined as SGD with added white noise (see for instance [12]. The Boltzman distribution is

$$p(f) = \frac{1}{Z} e^{-\frac{L}{T}}, \tag{7}$$

where Z is a normalization constant, $L(f)$ is the loss and T reflects the noise power. The equation implies that SGDL prefers degenerate minima relative to non-degenerate ones of the same depth. In addition, among two minimum basins of equal depth, the one with a larger volume is much more likely in high dimensions as shown by the simulations in [11]. Taken together, these two facts suggest that SGD selects degenerate minimizers corresponding to larger isotropic flat regions of the loss. Then SDGL shows concentration – *because of the high dimensionality* – of its asymptotic distribution Equation 7.

Together [10] and [11] suggest the following

Theorem 2 *(informal statement): SGD selects with high probability the global minimizer of the empirical loss, which are degenerate.*

4 Related work

There are many recent papers studying optimization and generalization in deep learning. For optimization we mention work based on the idea that noisy gradient descent [13, 14, 15, 16] can

find a global minimum. More recently, several authors studied the dynamics of gradient descent for deep networks with assumptions about the input distribution or on how the lables are generated. They obtain global convergence for some shallow neural networks [17, 18, 19, 20, 21, 22]. Some local convergence results have also been proved [23, 24, 25]. The most interesting such approach is [22], which focuses on minimizing the training loss and proving that randomly initialized gradient descent can achieve zero training loss (see also [26, 27, 28]) as in section 3. In summary, there is by now an extensive literature on optimization that formalizes and refines to different special cases and to the discrete domain our results of Theory II and IIb (see section 3).

For generalization, which is the topic of this paper, existing work demonstrate that gradient descent works under the same situations as kernel methods and random feature methods [29, 30, 31]. Closest to our approach – which is focused on the role of batch and weight normalization – is the paper [32]. Its authors study generalization assuming a regularizer because they are – like us – interested in normalized margin. Unlike their assumption of an explicit regularization, we show here that commonly used techniques, such as batch normalization, in fact normalize margin without the need to add a regularizer or to use weight decay.

5 Preliminaries on Generalization

Classical *generalization bounds for regression* suggest that *bounding the complexity of the minimizer* provides a bound on generalization. Ideally, the optimization algorithm should select the smallest complexity minimizers among the solutions – that is, in the case of ReLU networks, the minimizers with minimum norm. An approach to achieve this goal is to add a vanishing regularization term to the loss function (the parameter goes to zero with iterations) that, under certain conditions, provides convergence to the minimum norm minimizer, independently of initial conditions. This approach goes back to Halpern fixed point theorem [33]; it is also independently suggested by other techniques such as Lagrange multipliers, normalization and margin maximization theorems [9].

Well-known *margin bounds for classification* suggest a similar (see Appendix 10) approach: maximization of the margin of the normalized network (the weights at each layer are normalized by the Frobenius norm of the weight matrix of the layer). The margin is the value of yf over the support vectors (the data with smallest margin $y_n f(x_n)$, assuming $y_n f(x_n) > 0, \forall n$).

In the case of nonlinear deep networks, the critical points of the gradient of an exponential-type loss include saddles, local minima (if they exist) and global minima of the loss function; the latter are generically degenerate [10]. A similar approach to the the linear case leads to minimum norm solutions, independently of initial conditions.

5.1 Regression: (local) minimum norm empirical minimizers

We recall that generalization bounds [34] apply to $\forall f \in \mathbb{F}$ with probability at least $(1-\delta)$ and have the typical form

$$|L(f) - \hat{L}(f)| \leq c_1 \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{8}$$

where $L(f) = \mathbf{E}[\ell(f(x), y)]$ is the expected loss, $\mathbb{R}_N(\mathbb{F})$ is the empirical Rademacher average of the class of functions $\mathbb{F}$ measuring its complexity; c_1, c_2 are constants that depend on properties of the Lipschitz constant of the loss function, and on the architecture of the network.

The bound together with the property Equation 1 implies that among the minimizers with zero square loss, the optimization algorithm should select the minimum norm solution. In any case, the algorithm should control the norm. Standard GD or SGD algorithms do not provide an explicit control of the norm. Empirically it seems that initialization with small weights helps – as in the linear case (see Figures and see section 7). We propose a slight modification of the standard gradient descent algorithms to provide a norm-minimizing GD update – NMGD in short – as

$$W_{n+1} - W_n = -(1 - \lambda_n)\gamma_n \nabla_w L(f) - \lambda_n W_n, \tag{9}$$

where γ_n is the learning rate and $\lambda_n = \frac{1}{n^a}$ (this is one of several choices) is the vanishing regularization-like Halpern (see Appendix 12) term.

5.2 Classification: maximizing the margin of the normalized minimizer

A typical margin bound for classification [35] is

$$|L_{binary}(f) - L_{surr}(f)| \leq b_1 \frac{\mathbb{R}_N(\mathbb{F})}{\eta} + b_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{10}$$

where η is the margin, $L_{binary}(f)$ is the expected classification error, $L_{surr}(f)$ is the empirical loss of a surrogate loss such as the logistic or the exponential. For a point x, the margin is $\eta \sim y\rho\tilde{f}(x)$. Since $\mathbb{R}_N(\mathbb{F}) \sim \rho$, the margin bound is optimized by effectively maximizing $\tilde{f}$ on the "support vectors" – that is the x_i, y_i s.t $\arg\min_n y_n \tilde{f}(x_n)$.

We show (see Appendix 10) that for separable data, maximizing the margin subject to unit norm constraint is equivalent to minimize the norm of f subject to a constraint on the margin. A regularized loss with an appropriately vanishing regularization parameter is a closely related optimization technique. For this reason we will refer to the solutions in all these cases as minimum norm. This view treats interpolation (in the regression case) and classification (in the margin case) in a unified way.

6 Gradient descent with norm constraint

In this section we focus on the classification case with an exponential loss function. The generalization bounds in the previous section are satisfied by the maximizing the margin subject to the product of the norms being equal to one:

$$\underset{\prod_k ||V_k||=1}{\arg\max} \quad \min_n y_n \rho \tilde{f}(x_n). \tag{11}$$

In words: *find the network weights that maximize the margin subject to a norm constraint.* The latter ensures a bounded Rademacher complexity and together they minimize the term $\frac{\mathbb{R}_N(\mathbb{F})}{\eta}$.

In turns, this product norm constraint is ensured by a stricter unit constraint on the norm of each layer. Either constraint defines an equivalence class of networks f because of Eq. (1). A direct approach is to minimize the exponential loss function $L(f(w)) = \sum_{n=1}^{N} e^{-f(W;x_n)y_n} = \sum_{n=1}^{N} e^{-\rho\tilde{f}(W;x_n)y_n}$, subject to $||V_k||^2 = \sum_{i,j}(V_k)_{i,j}^2 = 1, \forall k$, that is under a unit norm constraint for the weight matrix at each layer. Clearly these constraints imply the constraint on the product of weight matrices in (11). As we discuss later (see Appendices and [36]), there are several ways to implement the minimization in the tangent space of $||V||^2 = 1$. The interesting observation is that they are closely related to to gradient descent techniques widely used for training deep networks, such as weight normalization (WN) [37] and batch normalization (BN) [38]. In the following we describe one of the techniques, the Lagrange multiplier method, because it enforces the constraint from the generalization bounds in a transparent way.

6.1 Lagrange multiplier method

We define the loss

$$L = \sum_{n=1}^{N} e^{-\rho\tilde{f}((x_n)y_n} + \sum_{k=1}^{K} \lambda_k ||V_k||^2 \tag{12}$$

where the Lagrange multipliers λ_k are chosen to satisfy $||V_k|| = 1$ at convergence or when the algorithm is stopped (the constraint can also be enforced at each iteration, see later).

We perform gradient descent on L with respect to ρ, V_k. We obtain for $k = 1, \cdots, K$

$$\dot{\rho_k} = \sum_n \frac{\rho}{\rho_k} e^{-\rho(t)\tilde{f}(x_n)y_n} \tilde{f}(x_n), \tag{13}$$

and for each layer k

$$\dot{V_k} = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}(t) - 2\lambda_k(t) V_k(t). \tag{14}$$

The sequence $\lambda_k(t)$ must satisfy $\lim_{t\to\infty} ||V_k|| = 1$.

Since the first term in the right hand side of Equation (14) goes to zero with $t \to \infty$ and the Lagrange multipliers λ_k also go to zero, the normalized weight vectors converge at infinity with $\dot{V}_k = 0$. On the other hand, $\rho(t)$ grows to infinity. Interestingly, as shown in section 7, the norm square of each layer grows at the same rate.

Let us assume that starting at some time t, $\rho(t)$ is large enough that the following asymptotic expansion (as $\rho \to \infty$) is a good approximation: $\sum_n e^{-\rho(t)\tilde{f}(x_n)} \sim C \max_n e^{-\rho(t)\tilde{f}(x_n)}$, where C is the multiplicity of the minimal $\tilde{f}$.

The data points with the corresponding minimum value of the margin $y_n\tilde{f}(x_n)$ are the support vectors. They are a subset of cardinality C of the N datapoints, all with the same margin η. In particular, the term $g(t) = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}$ becomes $g(t) \approx \rho(t) e^{-\rho(t)\eta} \sum_i^H \frac{\partial \tilde{f}(x_n)}{\partial V_k}$.

A rigorous proof of the argument above can be regarded as an extension of the main theorem in [9] from the case of linear functions to the case of one-homogeneous functions. In fact, while updating the present version of this paper we noticed that [39] has theorems including such an extension.

Remarks

1. If we impose the conditions $||V_k|| = 1$ at each t, $\lambda_k(t)$ must satisfy

$$||V_k(t) + \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k} - \lambda_k(t) V_k(t)|| = 1,$$

   where we redefined as λ the quantity 2λ. Thus

$$\lambda(t) = 1 - \sqrt{1 - ||g(t)||^2 + ||V_k^T g(t)||^2} + V^T(t) g(t). \tag{15}$$

   goes to zero at infinity because $g(t)$ does.

2. It is possible to add a regularization term to the equation for $\dot{\rho}$. The effect of regularization is to bound $\rho(t)$ to a maximum size ρ_{max}, controlled by a fixed regularization parameter λ_ρ: in this case the dynamics of ρ converges to a (very large) ρ_{max} set by a (very small) value of λ_ρ.

6.2 Related techniques for unit norm constraint: weight normalization, batch normalization and natural gradient

A main observation of this paper is that the Lagrange multiplier technique is very similar in its goal and implementation to other gradient descent algorithms with unit norm constraint. A review of gradient-based algorithms with unit-norm constraints [36] lists

1. the *Lagrange multiplier method* of our section 6.1,

2. the *coefficient normalization method* that is related to batch normalization, see Appendix 21.2

3. the *tangent gradient method* that corresponds to weight normalization (Appendix 21.1) and finally

4. the *true gradient method* using natural gradient.

The four techniques are equivalent for small values of λ [36]. Stability issues for numerical implementations are also characterized in [36]. Our main point here is that the four techniques are closely related and have the same goal: performing gradient descent with a unit norm constraint. It seems fair to say that in the case of GD (a single minibatch, including all data) the four techniques should behave in a similar way. In particular, as discussed in appendices 21.2 and

21.1, batch normalization enforces WN – modulo the details of the implementation and how to best enforce algorithmic stability.

This argument suggests that WN and BN implement an approximation of constrained natural gradient. Interestingly, there is a close relationship between the Fisher-Rao norm and the natural gradient [8]. In particular, the natural gradient descent is the steepest descent direction induced by the Fisher-Rao geometry.

6.3 Margin maximizers

As we mentioned, in GD with unit norm constraint there will be convergence to $\dot{V}_k = 0$ for $t \to \infty$. There may be trajectory-dependent, multiple alternative selections of the support vectors (SVs) during the course of the iteration while ρ grows: each set of SVs may correspond to a max margin, minimum norm solution without being the global minimum norm solution. Because of Bezout-type arguments [10] *we expect multiple maxima.* They should generically be degenerate even under the normalization constraints – which enforce each of the K sets of V_k weights to be on a unit hypersphere. Importantly, the normalization algorithms ensure control of the norm and thus of the generalization bound even if they cannot ensure that the algorithm converges to the globally best minimum norm solution (this depends on initial conditions for instance). In summary

Theorem 3 *(informal statement)*
The GD equations 13 and 14 converge to maximum margin solutions with unit norm.

6.4 Dynamics

In the appendices we discuss the dynamics of gradient descent in the continuous framework for a variety of losses and in the presence of regularization or normalization. Typically, normalization is similar to a vanishing regularization term.

The Lagrange multiplier case is a simple example (see Appendix 6.1). For $\dot{V}_k(t) = 0$ the following equations – *as many as the number of weights* – have to be satisfied asymptotically

$$V_k = \frac{g(t)}{2\lambda}. \tag{16}$$

where $g(t) = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}$ and $\lambda(t)$ goes to zero at infinity at the same rate as $g(t)$ (see the special case of Equation (15)). This suggests that weight matrices from W_1 to W_K should be in relations of the type $W_3 = W_2^T = \cdots$ for linear multilayer nets; appropriately similar relations should hold for the rectifier nonlinearities. In other words, *gradient descent under unit norm is biased towards balancing the weights of different layers since this is the solution with minimum norm.*

The Hessian of L w.r.t. V_k tells us about the linearized dynamics around the asymptotic critical point of the gradient. The Hessian (see Appendix 18)

$$\sum_n \left[-\left(\prod_{i=1}^{K} \rho_i^2 \right) \frac{\partial \tilde{f}(V;x_n)}{\partial V_k} \frac{\partial \tilde{f}(V;x_n)}{\partial V_{k'}}^T + \left(\prod_{i=1}^{K} \rho_i \right) \frac{\partial^2 \tilde{f}(V;x_n)}{\partial V_k \partial V_{k'}} \right] e^{-\prod_{i=1}^{K} \rho_i \tilde{f}(V;x_n)} - 2\lambda(t)\mathbf{I}. \quad (17)$$

is in general degenerate corresponding to an asymptotically degenerate hyperbolic equilibrium (biased towards minimum norm solutions if the rate of decay of $\lambda(t)$ implements correctly a Halpern iteration). The number of degenerate directions of the gradient flow corresponds to the number of symmetries of the neural network $f(x)$ as discussed in Appendix 19. In the deep linear case, these would correspond to the freedom of applying opposite general linear transformations to neighboring layers. In the case of ReLU networks the situation becomes data-dependent.

Remark

For classification with exponential-type losses the Lagrange multiplier technique, WN and BN are trying to achieve approximately the same result – maximize the margin while constraining the norm. An even higher level perspective, unifying view of several different optimization techniques including the case of regression, is to regard them as instances of Halpern iterations. Appendix 12 describes the technique. The gradient flow corresponds to an operator T which is non-expansive. The fixed points of the flow are degenerate. Minimization with a regularization term in the weights that vanishes at the appropriate rate (Halpern iterations) converges to the minimum norm minimizer associated to the local minimum. Halpern iterations are a form of regularization with a vanishing $\lambda(t)$ (which is the form of regularization used to define the pseudoinverse). From this perspective, the Lagrange multiplier term can be seen as a Halpern term which "attracts" the solution towards zero norm. This corresponds to a local minimum norm solution for the unnormalized network (imagine for instance in 2D that there is a surface of zero loss with a boundary as in Figure 1). The minimum norm solution in the classification case corresponds to a maximum margin solution for the normalized network. Globally optimal generalization is not guaranteed but generalization bounds such as Equation 10 are locally optimized. It should be emphasized however that it is not yet clear whether all the algorithms we mentioned implement the correct dependence of the Halpern term on the number of iterations. We will examine this issue in future work.

7 Generalization without unit norm constraints

Empirically it appears that GD and SGD converge to solutions that can generalize even without BN or WN or other techniques enforcing unit norm constraints. Without explicit constraints, convergence may be difficult for quite deep networks; generalization is usually not as good as with BN or WN but it still occurs. How is this possible? Recent work [2] shows that the difference between the Frobenius norms of the weights of various layers does not change during gradient descent. This implies that if the weight matrices are all small at initialization, the gradient flow corresponding to gradient descent maintains approximately equal Frobenius norms across different layers, that is maintains a minimum norm constraint (equal weight matrices norms at different layers), which is the constraint we enforce in an explicit way with the Lagrange

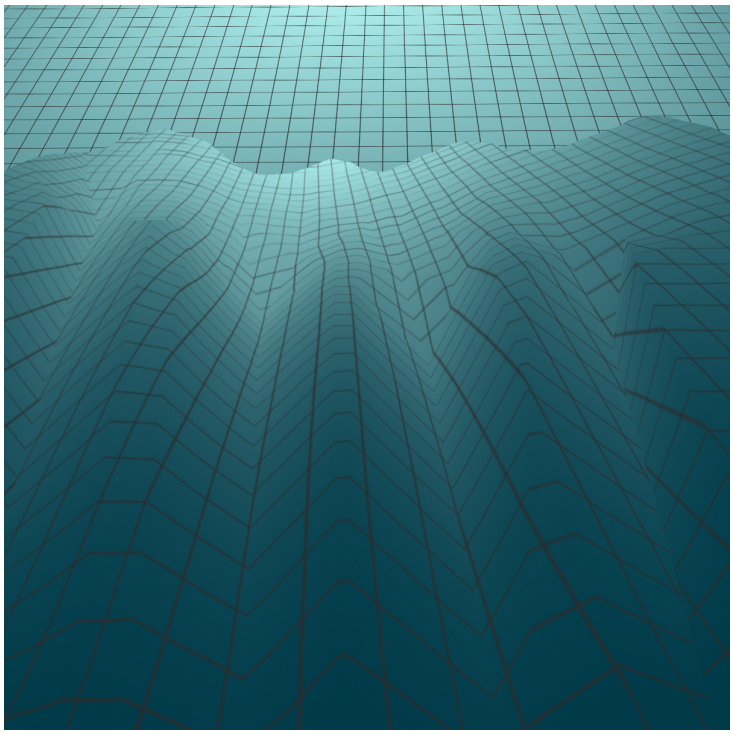


Figure 1: *Landscape of the empirical loss with unnormalized weights.* Suppose the empirical loss at the water level in the figure is $\leq \epsilon$. Then there are various global minima each with the same loss and different minimum norms. Because of the universality of deep networks from the point of view of function approximation, it seems likely that similar landscapes may be realizable (consider the approximator $\exp^{-f(w)}$ with the components of x as parameters; an example is $f(w) = w_1^2 + \frac{1}{100} w_2^2 \sin w_2$). It is however an open question whether overparametrization may typically induce "nicer" landscapes, without the many "gulfs" in the figure.

multiplier or the WN technique. It is easy to see in our framework that the Frobenius norm of each layer changes at the same rate under gradient flow. Consider Equation (12) for $\lambda_k = 0$, that is without norm constraint. Inspection of it shows that $\rho_k \dot{\rho_k}$ is independent of k. It follows that

$$\frac{d}{dt}\rho_k^2 = \frac{d}{dt}\rho_{k-1}^2, \forall k = 2, \cdots, K. \tag{18}$$

Thus if we consider two of the K layers, the following property holds: $\rho_1^2(t) = \rho_2^2(t) + \eta$ with $||V_1|| = ||V_2|| = 1$. If η is small at initialization then the norm of the two layers will remain very similar under the gradient flow. Then, the minimization problem $\min e^{-\rho_k^K \tilde{f}}$ with $||V_k||^2 = 1$, $\forall k$ is equivalent to the Lagrange multiplier problem of Equation (12).

This is the nonlinear, multi-layer equivalent of the well-known property of the linear case: GD starting from zero or from very small weights converges to the minimum L_2 norm.

Of course, other effects, in addition to the role of initialization and batch or weight normalization may be at work here, improving generalization. For instance, high dimensionality under certain conditions has been shown to lead to better generalization for certain interpolating kernels [40, 41]. Though this is still an open question, it seems likely that similar results may also be valid for deep networks.

Furthermore, notice that commonly used weight decay with appropriate parameters can induce generalization. Interestingly, typical implementations of data augmentation in SGD (e.g. PyTorch) also eliminate the overparametrization problem: at each iteration of SGD only "new" data are used and depending on the number of iterations one can easily obtain more training data than parameters. In any case, within this online framework, classical results guarantee directly convergence to the minimum of the expected risk (see Appendix 11) without the need to invoke generalization bounds.

Remarks

- For a generic loss function such as the square loss and linear networks there is convergence to the minimum norm solution by GD for zero-norm initial conditions.
- For exponential type losses and linear networks in the case of classification the convergence is independent of intial conditions. The reason is that what matters is $\frac{w}{||w||} = \frac{\rho\tilde{w}}{\rho} = \tilde{w}$.
- For exponential type losses and one-homogeneous networks in the case of classification the situation is similar since $\frac{f}{\rho} = \frac{\rho\tilde{f}}{\rho} = \tilde{f}$. With zero-norm initial conditions the norms of the K layers are appromitavely equal and $\rho = \rho_1^K$. The degeneracy of the solutions is strongly reduced but it remains an open question in terms of the generalization bounds why the (stronger) unit norm constraints for each of the layer should provide better generalization than the unit norm constraint on the product of the norms.

8 Discussion

In summary, our results imply that multilayer, nonlinear, deep networks under gradient descent with norm constraint converge to maximum margin solutions. This is similar to the situation for linear networks. The prototypical (linear) example for over-parametrized deep networks is convergence of gradient descent to weights that represent the pseudoinverse of the input matrix.

We have to distinguish between square loss regression and classification via an exponential-type loss. In the case of square loss regression, NMGD converges to the minimum norm solution independently of initial conditions – under the assumption that the global minimum is achieved.

Consider now the case of *classification by minimization of exponential losses* using the Lagrange normalization algorithm. The main result is that the dynamical system in the normalized weights converges to a solution that (locally) maximizes margin. We discuss the close relations between this algorithm and weight normalization algorithms, which are themselves related to batch normalization. All these algorithms are commonly used. The fact that the solution corresponds to a maximum margin solution under a fixed norm constraint also explains the puzzling behavior

of Figure 3. The test classification error does not get worse when the number of parameters increases well beyond the number of training data because the dynamical system is constrained to maximize the *margin* under unit norm of $\tilde{f}$, without necessarily minimizing the loss.

An additional implication of our results is that *the effectiveness of batch normalization is based on more fundamental reasons* than reducing covariate shifts (the properties described in [42] are fully consistent with our characterization in terms of a regularization-like effect). Controlling the norm of the weights is exactly what generalization bounds prescribe: GD with normalization (NMGD) is the correct way to do it. Normalization is closely related to Halpern iterations used to achieve a minimum norm solution.

The theoretical framework described in this paper leaves a number of important open problems. Does the empirical landscape have multiple global minima with different minimum norms (see Figure 1)? Or is the landscape "nicer" for large overparametrization – as hinted in several very recent papers (see for instance [43] and [44])? Can one ensure convergence to the global empirical minimizer with global minimum norm? How? Are there conditions on the Lagrange multiplier term – and on corresponding parameters for weight and batch normalization – that ensure convergence to a maximum margin solution independently of initial conditions?

Acknowledgments

We thank Yuan Yao, Misha Belkin and especially Sasha Rakhlin for illuminating discussions. Part of the funding is from Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216, and part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] D. Soudry, E. Hoffer, and N. Srebro. The Implicit Bias of Gradient Descent on Separable Data. *ArXiv e-prints*, October 2017.

[2] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 384–395. Curran Associates, Inc., 2018.

[3] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *CoRR*, abs/1611.04231, 2016.

[4] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv:1706.08947*, 2017.

[5] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel Rodrigues. Robust large margin deep neural networks. *arXiv:1605.08254*, 2017.

[6] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *ArXiv e-prints*, June 2017.

[7] C. Zhang, Q. Liao, A. Rakhlin, K. Sridharan, B. Miranda, N.Golowich, and T. Poggio. Musings on deep learning: Optimization properties of SGD. *CBMM Memo No. 067*, 2017.

[8] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *CoRR*, abs/1711.01530, 2017.

[9] Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 1237–1244, 2003.

[10] T. Poggio and Q. Liao. Theory II: Landscape of the empirical risk in deep learning. *arXiv:1703.09833, CBMM Memo No. 066*, 2017.

[11] C. Zhang, Q. Liao, A. Rakhlin, K. Sridharan, B. Miranda, N.Golowich, and T. Poggio. Theory of deep learning IIb: Optimization properties of SGD. *CBMM Memo 072*, 2017.

[12] M. Raginsky, A. Rakhlin, and M. Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: A nonasymptotic analysis. *arXiv:180.3251 [cs, math]*, 2017.

[13] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. *CoRR*, abs/1703.00887, 2017.

[14] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points - online stochastic gradient for tensor decomposition. *CoRR*, abs/1503.02101, 2015.

[15] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1246–1257, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[16] Simon S. Du, Jason D. Lee, and Yuandong Tian. When is a convolutional filter easy to learn? In *International Conference on Learning Representations*, 2018.

[17] Yuandong Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 3404–3413. JMLR.org, 2017.

[18] M. Soltanolkotabi, A. Javanmard, and J. D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, Feb 2019.

[19] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 597–607, USA, 2017. Curran Associates Inc.

[20] Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 605–614, 2017.

[21] Simon Du, Jason Lee, Yuandong Tian, Aarti Singh, and Barnabas Poczos. Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima. 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 1339–1348, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[22] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *CoRR*, abs/1811.03804, 2018.

[23] Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 4140–4149. JMLR.org, 2017.

[24] Kai Zhong, Zhao Song, and Inderjit S. Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *CoRR*, abs/1711.03440, 2017.

[25] X. Zhang, Y. Yu, L. Wang, and Q. Gu. Learning One-hidden-layer ReLU Networks via Gradient Descent. *arXiv e-prints*, June 2018.

[26] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8157–8166. Curran Associates, Inc., 2018.

[27] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[28] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *CoRR*, abs/1811.08888, 2018.

[29] Amit Daniely. Sgd learns the conjugate kernel class of the network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2422–2430. Curran Associates, Inc., 2017.

[30] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *CoRR*, abs/1811.04918, 2018.

[31] Sanjeev Arora, Simon S. Du, Wei Hu, Zhi yuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *CoRR*, abs/1901.08584, 2019.

[32] Colin Wei, Jason D. Lee, Qiang Liu, and Tengyu Ma. On the margin theory of feedforward neural networks. *CoRR*, abs/1810.05369, 2018.

[33] Benjamin Halpern. Fixed points of nonexpanding maps. *Bull. Amer. Math. Soc.*, 73(6):957–961, 11 1967.

[34] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. pages 169–207, 2003.

[35] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[36] S. C. Douglas, S. Amari, and S. Y. Kung. On gradient adaptation with unit-norm constraints. *IEEE Transactions on Signal Processing*, 48(6):1843–1847, June 2000.

[37] Tim Salimans and Diederik P. Kingm. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 2016.

[38] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[39] Colin Wei, Jason D. Lee, Qiang Liu, and Tengyu Ma. On the Margin Theory of Feedforward Neural Networks. *arXiv e-prints*, page arXiv:1810.05369, Oct 2018.

[40] Tengyuan Liang and Alexander Rakhlin. Just Interpolate: Kernel "Ridgeless" Regression Can Generalize. *arXiv e-prints*, page arXiv:1808.00387, Aug 2018.

[41] Alexander Rakhlin and Xiyu Zhai. Consistency of Interpolation with Laplace Kernels is a High-Dimensional Phenomenon. *arXiv e-prints*, page arXiv:1812.11167, Dec 2018.

[42] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? *arXiv e-prints*, page arXiv:1805.11604, May 2018.

[43] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A Mean Field View of the Landscape of Two-Layers Neural Networks. *arXiv e-prints*, page arXiv:1804.06561, Apr 2018.

[44] Phan-Minh Nguyen. Mean Field Limit of the Learning Dynamics of Multilayer Neural Networks. *arXiv e-prints*, page arXiv:1902.02880, Feb 2019.

[45] Paulo Jorge S. G. Ferreira. The existence and uniqueness of the minimum norm solution to certain linear and nonlinear problems. *Signal Processing*, 55:137–139, 1996.

[46] Huan Xu and Shie Mannor. Robustness and generalization. *CoRR*, abs/1005.2243, 2010.

[47] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv:1509.01240 [cs, math, stat]*, September 2015. arXiv: 1509.01240.

[48] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[49] T. Poggio, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv:1703.09833, CBMM Memo No. 073*, 2017.

[50] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *CoRR*, abs/1611.07476, 2016.

[51] Daniel Kunin, Jonathan M. Bloom, Aleksandrina Goeva, and Cotton Seed. Loss landscapes of regularized linear autoencoders. *CoRR*, abs/1901.08168, 2019.

[52] Igor Gitman and Boris Ginsburg. Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *CoRR*, abs/1709.08145, 2017.

Appendices

9 Experiments

Summary:

- *GD starting from zero seems to work for CIFAR10 suggesting that the empirical loss landscape does not show local minima but only saddle points and global minima.*
- *Different initializations affect final result (large initialization typically induce larger final norm and larger test error). It is significant that there is dependence on initial conditions (differently from [1] linear case).*
- *Similar to initialization, perturbations of the weights increase norm and increase test error.*
- *Training loss of normalized nets predict well test performance of the same networks.*

In the computer simulations shown in this section, we turn off all the "tricks" used to improve performance such as data augmentation, weight decay, *etc.* However, we *keep batch normalization*. We reduce in some of the experiments the size of the network or the size of the training set. As a consequence, performance is not state-of-the-art, but optimal performance is not the goal here (in fact the networks we use achieve state-of-the-art performance using standard setups). The expected risk was measured as usual by an out-of-sample test set.

The puzzles we want to explain are in Figures 2 and 3.

A basic explanation for the puzzles is similar to the linear case: when the minima are degenerate the minimum norm minimizers are the best for generalization. The linear case corresponds to quadratic loss for a linear network shown in Figure 4.

In this very simple case we test our theoretical analysis with the following experiment. After convergence of GD, we apply a small random perturbation δW with unit norm to the parameters W, then run gradient descent until the training error is again zero; this sequence is repeated m times. We make the following predictions for the square loss:

- The training error will go back to zero after each sequence of GD.
- Any small perturbation of the optimum W_0 will be corrected by the GD dynamics to push back the non-degenerate weight directions to the original values. Since the components of the weights in the degenerate directions are in the null space of the gradient, running GD after each perturbation will not change the weights in those directions. Overall, the weights will change in the experiment.
- Repeated perturbations of the parameters at convergence, each followed by gradient descent until convergence, will not increase the training error but will change the parameters, increase norms of some of the parameters and increase the associated test error. The L_2 norm of the projections of the weights in the null space undergoes a random walk.

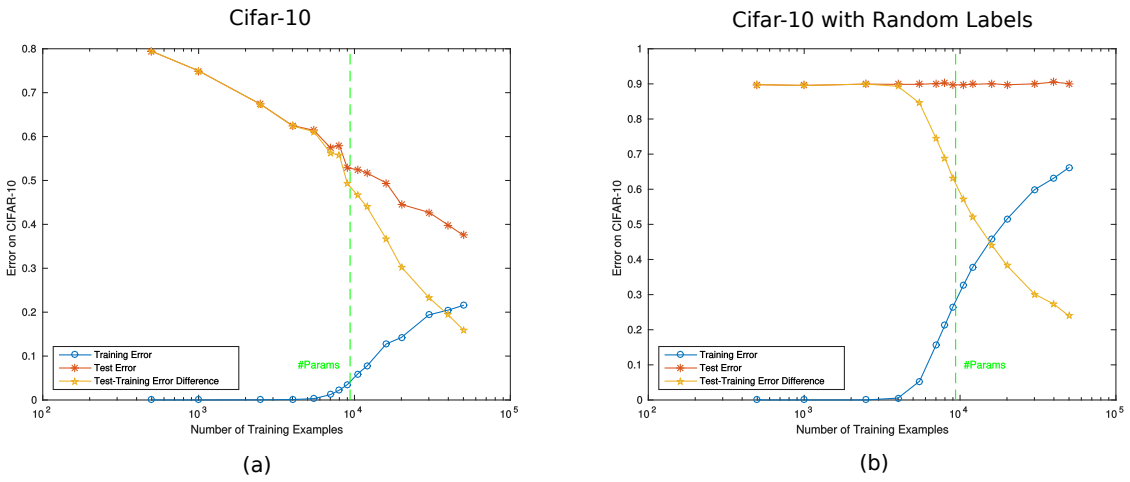


Figure 2: *Generalization for Different number of Training Examples.* (a) Generalization error in CIFAR and (b) generalization error in CIFAR with random labels. The DNN was trained by minimizing the cross-entropy loss and it is a 5-layer convolutional network (*i.e.*, no pooling) with 16 channels per hidden layer. ReLU are used as the non-linearities between layers. The resulting architecture has approximately 10000 parameters. SGD was used with batch size = 100 for 70 epochs for each point. Neither data augmentation nor regularization is performed.

The same predictions apply also to the cross entropy case with the caveat that the weights increase even without perturbations, though more slowly. Previous experiments by [10] showed changes in the parameters and in the expected risk, consistently with our predictions above, which are further supported by the numerical experiments of Figure 8. In the case of cross-entropy the almost zero error valleys of the empirical risk function are slightly sloped downwards towards infinity, becoming flat only asymptotically.

The numerical experiments show, as predicted, that the behavior under small perturbations around a global minimum of the empirical risk for a deep networks is similar to that of linear degenerate regression (compare Figure 8 with Figure 5). For the loss, the minimum of the expected risk may or may not occur at a finite number of iterations. If it does, it corresponds to an equivalent optimum (because of "noise") non-zero and non-vanishing regularization parameter λ. Thus a specific "early stopping" would be better than no stopping. The corresponding classification error, however, may not show overfitting.

Figure 9 shows the behavior of the loss in CIFAR in the absense of perturbations. This should be compared with Figure 5 which shows the case of an overparametrized linear network under quadratic loss corresponding to the multidimensional equivalent of the degenerate situation of Figure 4. The nondegenerate, convex case is shown in Figure 7.

Figure 10 shows the testing error for an overparametrized linear network optimized under the square loss.This is a special case in which the minimum norm solution is theoretically guaranteed by zero inital conditions without NMGD.

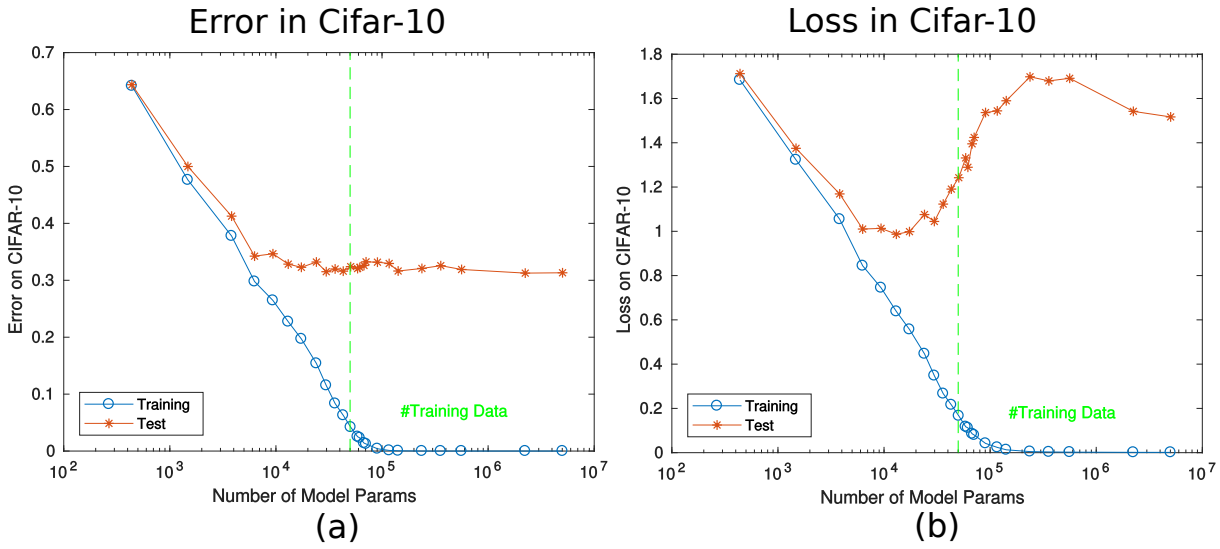


Figure 3: *Expected error in CIFAR-10 as a function of number of neurons.* The DNN is the same as in Figure 2. (a) Dependence of the expected error as the number of parameters increases. (b) Dependence of the cross-entropy risk as the number of parameters increases. There is some "overfitting" in the expected risk, though the peculiarities of the exponential loss function exaggerate it. The expected classification error does not increase here when increasing the number of parameters, because the product of the norms of the network is close to the minimum norm (here because of initialization).

10 Minimal norm and maximum margin

We discuss the connection between maximum margin and minimal norms problems in binary classification. To do so, we reprise some classic reasonings used to derive support vector machines. We show they directly extend beyond linearly parametrized functions as long as there is a one-homogeneity property, namely, for all $\alpha > 0$,

$$f(\alpha W; x) = \alpha f(W; x)$$

Given a training set of N data points $(x_i, y_i)_{i=1}^N$, where labels are ± 1, the functional margin is

$$\min_{i=1,\dots,N} y_i f(W; x_i). \tag{19}$$

If there exists W such that the functional margin is strictly positive, then the training set is separable. We assume in the following that this is indeed the case. The maximum (max) margin problem is

$$\max_{W} \min_{i=1,\dots,N} y_i f(W; x_i), \qquad \text{subj. to} \quad \|W\| = 1. \tag{20}$$

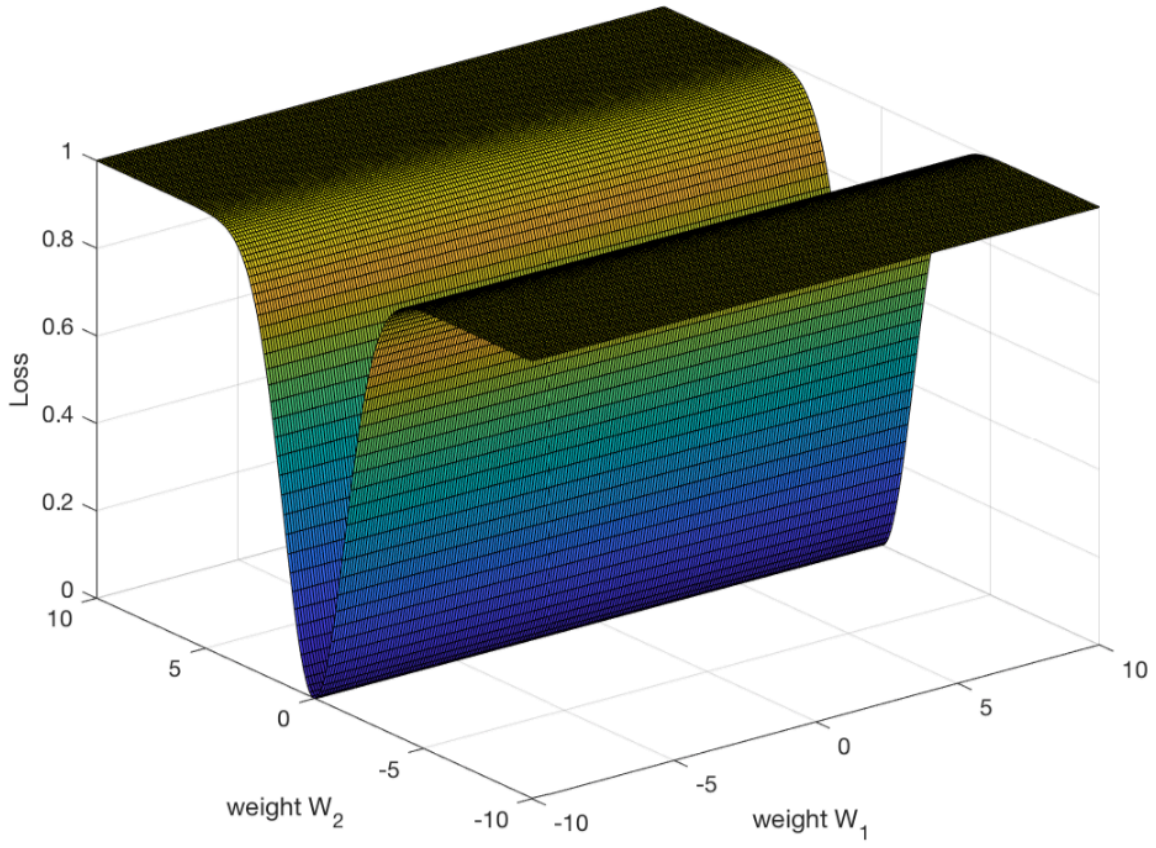
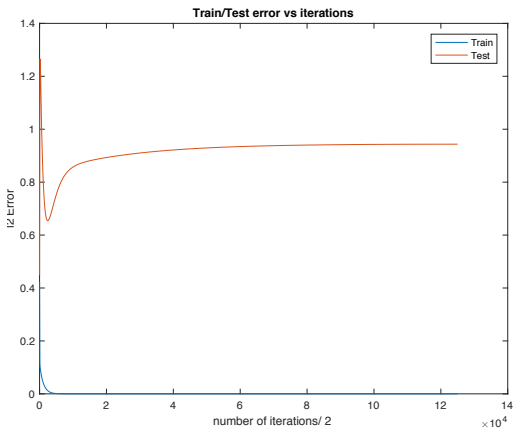


Figure 4: *A quadratic loss function in two parameters w_1 and w_2. The minimum has a degenerate Hessian with a zero eigenvalue. In the proposition described in the text, it represents the "generic" situation in a small neighborhood of zero minimizers with many zero eigenvalues – and a few positive eigenvalues – of the Hessian of a nonlinear multilayer network. In multilayer networks the loss function is likely to be a fractal-like surface with many degenerate global minima, each similar to a multidimensional version of the degenerate minimum shown here. For the crossentropy loss, the degenerate valleys are sloped towards infinity.*

The latter constraint is needed to avoid trivial solutions in light of the one-homogeneity property. We next show that Problem (20) is equivalent to

$$\min_W \frac{1}{2}\|W\|^2, \qquad \text{subj. to} \quad y_i f(W; x_i) \geq 1, \qquad i = 1, \ldots, N. \tag{21}$$

To see this, we introduce a number of equivalent formulations. First, notice that functional



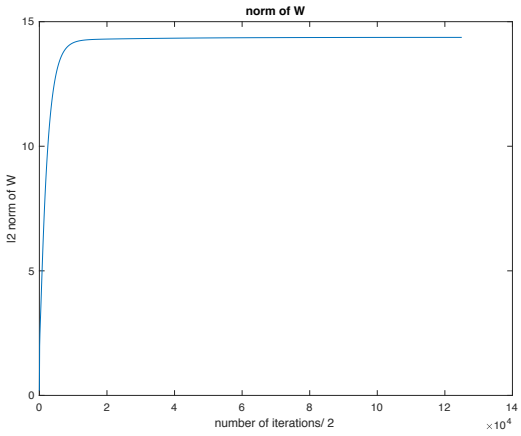


Figure 5: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 4. The feature matrix* $\phi(X)$ *is a polynomial with degree 30. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training point are* 9 *while the number of test points are* 100*. The training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. The weights were not perturbed in this experiment. The* L_2 *norm of the weights is shown on the right. Note that training was repeated 30 times and what is reported in the figure is the average train and test error as well as average norm of the weights over the 30 repetitions. There is overfitting in the test error.*

margin (19) can be equivalently written as

$$\max_{\gamma>0} \gamma, \qquad \text{subj. to} \quad y_i f(W; x_i) \geq \gamma, \qquad i = 1, \ldots, N.$$

Then, the max margin problem (20) can be written as

$$\max_{W,\gamma>0} \gamma, \qquad \text{subj. to} \quad \|W\| = 1, \quad y_i f(W; x_i) \geq \gamma, \qquad i = 1, \ldots, N. \tag{22}$$

Next, we can incorporate the norm constraint noting that using one-homogeneity,

$$y_i f(W; x_i) \geq \gamma \Leftrightarrow y_i f(\frac{W}{\|W\|}; x_i) \geq \gamma' \Leftrightarrow y_i f(W; x_i) \geq \|W\|\gamma = \gamma'$$

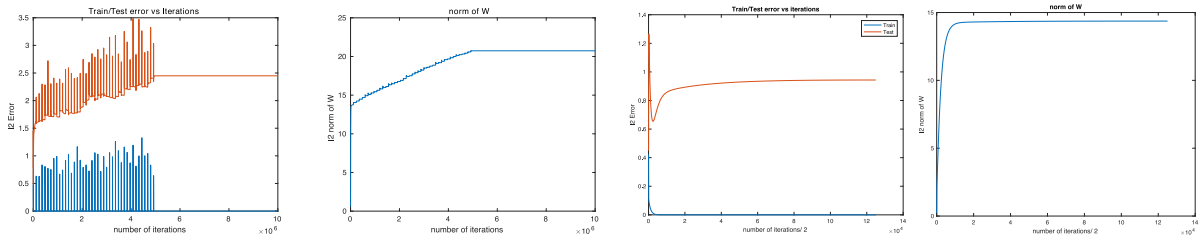


Figure 6: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 4. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points is* 9 *while the number of test points is* 100*. For the first pair of plots the feature matrix* $\phi(X)$ *is a polynomial with degree 39. For the first pair had points were sampled to according to the Chebyshev nodes scheme to speed up training to reach zero on the train error. Training was done with full Gradient Descent step size* 0.2 *for* $10,000,000$ *iterations. Weights were perturbed every* $120,000$ *iterations and Gradient Descent was allowed to converge to zero training error (up to machine precision) after each perturbation. The weights were perturbed by addition of Gaussian noise with mean* 0 *and standard deviation* 0.45*. The perturbation was stopped half way at iteration* $5,000,000$*. The* L_2 *norm of the weights is shown in the second plot. Note that training was repeated 29 times figures reports the average train and test error as well as average norm of the weights over the repetitions. For the second pair of plots the feature matrix* $\phi(X)$ *is a polynomial with degree 30. Training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. The* L_2 *norm of the weights is shown in the fourth plot. Note that training was repeated 30 times figures reports the average train and test error as well as average norm of the weights over the repetitions. The weights were not perturbed in this experiment.*

so that Problem (22) becomes

$$\max_{W,\gamma'>0} \frac{\gamma'}{\|W\|}, \qquad \text{subj. to} \qquad y_i f(W; x_i) \geq \gamma', \qquad i = 1, \ldots, N. \tag{23}$$

Finally, using again one-homogeneity, without loss of generality, we can set $\gamma' = 1$ and obtain the equivalent problem

$$\max_{W} \frac{1}{\|W\|}, \qquad \text{subj. to} \qquad y_i f(W; x_i) \geq 1, \qquad i = 1, \ldots, N. \tag{24}$$

The result is then clear noting that

$$\max_{W} \frac{1}{\|W\|} \Leftrightarrow \min_{W} \|W\| \Leftrightarrow \min_{W} \frac{\|W\|^2}{2}.$$

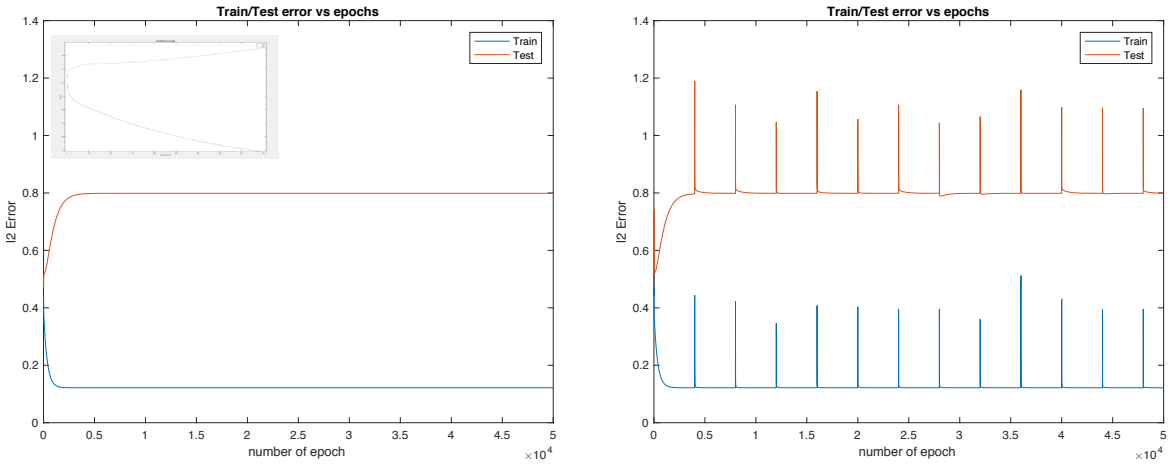


Figure 7: *The graph on the left shows training and testing loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) in the nondegenerate quadratic convex case. The feature matrix* $\phi(X)$ *is a polynomial with degree 4. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training point are* 9 *while the number of test points are* 100*. The training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. The inset zooms in on plot showing the absense of overfitting. In the plot on the right, weights were perturbed every* 4000 *iterations and then gradient descent was allowed to converge to zero training error after each perturbation. The weights were perturbed by adding Gaussian noise with mean* 0 *and standard deviation* 0.6*. The plot on the left had no perturbation. The* L_2 *norm of the weights is shown on the right. Note that training was repeated 30 times and what is reported in the figure is the average train and test error as well as average norm of the weights over the 30 repetitions.*

11 Data augmentation and generalization with "infinite" data sets

In the case of batch learning, *generalization guarantees* on an algorithm are conditions under which the empirical error $I_{S_N}(f)$ on the training set converges to the expected error $I(f)$, ideally with bounds that depend on the size N of the training set. The practical relevance of this guarantee is that the empirical error is then a measurable proxy for the unknown expected error and its error can be bound . In the case of "pure" online algorithms such as SGD – in

which the samples z_i are drawn i.i.d. from the unknown underlying distribution – there is no training set per se or equivalently the training set has infinite size S_∞. Under usual conditions on the loss function and the learning rate, SGD converges to the minimum of the expected risk. Thus, the proof of convergence towards the minimum of the expected risk bypasses the need for generalization guarantees. With data augmentation most of the implementations – such as the PyTorch one – generate "new" examples at each iteration. This effectively extends the size of the finite training set S_N for $N \to \infty$ guaranteeing convergence to the minimum of the expected risk. *Thus existing proofs of the convergence of SGD provide the guarantee that it converges to the "true" expected risk when the size of the "augmented" training set S_N increases with $N \to \infty$.*

Notice that while there exists unique $I(f_K)$, f_K does not need to be unique: the set of f_K which provide global minima of the expected error is an equivalence class.

12 Halpern iterations: selecting minimum norm solution among degenerate minima

In this section we summarize a modification of gradient descent that we apply to the various problems of optimization under the square and exponential loss for one-layer and nonlinear, deep networks.

We are interested in the convergence of solutions of gradient descent dynamics and their stability properties. In addition to the standard dynamical system tools we also use closely related elementary properties of non-expansive operators. A reason is that they describe the step of numerical implementation of the continuous dynamical systems that we consider. More importantly, they provide iterative techniques that converge (in a convex set) to the minimum norm of the fixed points, even when the operators are not linear, *independently of initial conditions.*

Let us define an operator T in a normed space X with norm $||\cdot||$ as *non expansive* if $||Tx - Ty|| \leq ||x - y||, \quad \forall x, y \in X$. Then the following result is classical ([45, 33])

Theorem 4 *[45] Let X be a strictly convex normed space. The set of fixed points of a non-expansive mapping $T : C \to C$ with C a closed convex subset of X is either empty or closed and convex. If it is not empty, it contains a unique element of smallest norm.*

In our case $T = (I - \gamma(t)\nabla_w L(f))$. To fix ideas, consider gradient descent on the square loss. As discussed later and in several papers, the Hessian of the loss function ($E = L(f(\cdot))$) of a deep networks with ReLUs has eigenvalues bounded from above (see for instance [46] and [47]) because the network is Lipschitz continuous and bounded from below by zero at the global minimum. Thus with an appropriate choice of $\gamma(t)$ the operator T is non-expanding and its fixed points are not an empty set, see Appendix 20. If we assume that the minimum is global and that there are no local minima but only saddle points then the null vector is in C. Then the element of minimum norm can be found by iterative procedures (such as Halpern's method, see Theorem 1 in [33]) of the form

$$x_{t+1} = (1 - s_t)Tx_t \tag{25}$$

where the sequence s_t satisfies conditions such as $\lim_{n\to\infty} s_n = 0$ and $\sum_{n=1}^{\infty} = \infty$ [1].

In particular, the following holds

Theorem 5 *[33] For any $x_0 \in B$ the iteration $x_n = kTx_{n-1}$ with $|k| < 1$ converges to one of the fixed points y_k of T. The sequence $w_{n+1} = k_{n+1}Tw_n$ with $k_n = 1 - \frac{1}{n^a}$ and $0 < a < 1$ converges to the fixed point of T with minimum norm.*

The norm-minimizing GD update – NMGD in short – has the form

$$w_{n+1} - w_n = -(1 - \lambda_n)\gamma_n \nabla_w L(f) - \lambda_n w_n \tag{26}$$

where γ_n is the learning rate and $\lambda_n = \frac{1}{n^a}$ (this is one of several choices).

It is an interesting question whether convergence to the minimum norm is independent of initial conditions and of perturbations. This may depend among other factors on the rate at which the Halpern term decays.

13 Network minimizers under square and exponential loss

We consider one-layer and multilayer networks under the square loss and the exponential loss. Here are the main observations and results

1. *One-layer networks* The Hessian is in general degenerate. Regularization with arbitrarily small λ ensures independence from initial conditions for both the square and the exponential loss. In the absence of explicit regularization, GD converge to the minimum norm solution for zero initial conditions. With NMGD-type iterations GD converge to the minimum norm *independently of initial conditions* (this is similar to the result of [1] obtained with different assumptions and techniques). For the exponential loss NMGD ensures convergence to the normalized solution $\tilde{f}$ that maximizes the margin (and that corresponds to the overall minimum norm solution), see Appendix 14.3. In the exponential loss case, weight normalization GD is degenerate since the data (support vectors) may not span the space of the weights.

2. *Deep networks, square loss* The Hessian is in general degenerate, even in the presence of regularization (with fixed λ). NMGD-type iterations lead to convergence not only to the fixed points – as vanilla GD does – but to the (locally) minimum norm fixed point.

[1] Notice that these iterative procedures are often part of the numerical implementation (see [48] and section 4.1) of discretized method for solving a differential equation whose equilibrium points are the minimizers of a differentiable convex subset of a function L. Note also that proximal minimization corresponds to backward Euler steps for numerical integration of a gradient flow. Proximal minimization can be seen as introducing quadratic regularization into a smooth minimization problem in order to improve convergence of some iterative method in such a way that the final result obtained is not affected by the regularization.

3. *Deep networks, exponential loss* The Hessian is in general degenerate, even in the presence of regularization. NMGD-type iterations lead to convergence to the minimum norm fixed point $\tilde{f}$ associated with the global minimum.

4. *Implications of minimum norm for generalization in regression problems* NMGD-based minimization ensures minimum norm solutions.

5. *Implications of minimum norm for classification*

   For classification a typical margin bound is

$$|L_{binary}(f) - L_{surr}(f)| \leq b_1 \frac{\mathbb{R}_N(\mathbb{F})}{\eta} + b_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{27}$$

   which depends on the margin η. $L_{binary}(f)$ is the expected classification error; $L_{surr}(f)$ is the empirical loss of a surrogate loss such as the logistic. For a point x the margin is $\eta \sim y\rho\tilde{f}(x)$. Since $\mathbb{R}_N(\mathbb{F}) \sim \rho$, the margin bound is optimized by effectively maximizing $\tilde{f}$ on the "support vectors". As shown in Appendix 10 maximizing margin under the unit norm constraint is equivalent to minimizing the norm under the separability constraint.

Remarks

- NMGD can be seen as a variation of regularization (that is weight decay) by requiring λ to decrease to zero. The theoretical reason for NMGD is that NMGD ensures minimum norm or equivalently maximum margin solutions.

- Notice that one of the definitions of the pseudoinverse of a linear operator corresponds to NMGD: it is the regularized solution to a degenerate minimization problem in the square loss for $\lambda \to 0$.

- The failure of regularization with a fixed λ to induce hyperbolic solutions in the multi-layer case was surprising to us. Technically this is due to contributions to non-diagonal parts of the Hessian from derivatives across layers and to the shift of the minimum.

14 One-layer networks

14.1 Square loss

For linear networks under square loss GD is a non-expansive operator. There are fixed points. The Hessian is degenerate. Regularization with arbitrarily small λ ensures independence of initial conditions. Even in the absence of explicit regularization GD converge to the minimum norm solution for zero initial conditions. Convergence to the minimum norm holds also with NMGD-type iterations but now independently of initial conditions.

We consider linear networks with one layer and one scalar output that is $w_k^{i,j} = w_1^{1,j}$ because there is only one layer. Thus $f(W; x) = w^T x$ with $w_1^{1,j} = w^T$.

Consider

$$L(f(w)) = \sum_{n=1}^{N} (y_n - w^T x_n)^2 \tag{28}$$

where y_n is a bounded real-valued variable. Assume further that there exists a d-dimensional weight vector that fits all the n training data, achieving zero loss on the training set, that is $y_n = w^T x_n \quad \forall n = 1, \cdots, N$.

1. *Dynamics* The dynamics is

$$\dot{w} = -F(w) = -\nabla_w L(w) = 2 \sum_{n=1}^{N} E_n x_n^T \tag{29}$$

with $E_n = (y_n - w^T x_n)$.

The only components of the the weights that change under the dynamics are in the vector space spanned by the examples x_n; components of the weights in the null space of the matrix of examples X^T are invariant to the dynamics. Thus w converges to the minimum norm solution *if* the dynamical system starts from zero weights.

2. The Jacobian of $-F$ – and Hessian of $-L$ – for $w = w^0$ is

$$J_F(w) = H = -\sum_{n=1}^{N} (x_n^i)(x_n^j) \tag{30}$$

This linearization of the dynamics around w^0 for which $L(w^0) = \epsilon_0$ yields

$$\dot{\delta w} = J_F(w^0)\delta w. \tag{31}$$

where the associated L is convex, since the Jacobian J_F is minus the sum of auto-covariance matrices and thus is semi-negative definite. It is negative definite if the examples span the whole space but it is degenerate with some zero eigenvalues if $d > n$ [49].

3. *Regularization* If a regularization term λw^2 is added to the loss the fixed point shifts. The equation

$$\dot{w} = -\nabla_w (L + \lambda |w|^2) = 2 \sum_{n=1}^{N} E_n x_n^T - \lambda w \tag{32}$$

gives for $\dot{w} = 0$

$$w_0 = \frac{2}{\lambda} \sum_{n=1}^{N} E_n . x_n^T \tag{33}$$

The Hessian at w_0 is with

$$H(w) = -\sum_{n=1}^{N}(x_n^i)(x_n^j) - \lambda \tag{34}$$

which is always negative definite *for any arbitrarily small fixed* $\lambda > 0$. Thus the dynamics of the perturbations around the equilibrium is given by

$$\dot{\delta w} = H(w^0)\delta w. \tag{35}$$

and is hyperbolic. Explicit regularization ensures the existence of a hyperbolic equilibrium for any $\lambda > 0$ at a finite w^0. In the limit of $\lambda \to 0$ the equilibrium converges to a minimum norm solution.

4. *NMGD* The gradient flow corresponds to $w_{t+1} = Tw_t$ with $T = I - \nabla_w L$. The gradient is non-expansive (see Appendix 20). There are fixed points (w satisfying $E_n = 0$) that are degenerate. Minimization using the NMGD method converges to the minimum norm minimizer.

14.2 Exponential loss

Linear networks under exponential loss and GD show growing Frobenius norm. On a compact domain ($||w|| \leq R$) the exponential loss is L-smooth and corresponds to a non-expansive operator T. Regularization with arbitrarily small λ ensures convergence to a fixed point independent of initial conditions. GD with normalization and NMGD-type iterations converge to the minimum norm, maximum margin solution for separable data with degenerate Hessian.

Consider now the exponential loss. Even for a linear network the dynamical system associated with the exponential loss is nonlinear. While [1] gives a rather complete characterization of the dynamics, here we describe a different approach.

The exponential loss for a linear network is

$$L(f(w)) = \sum_{n=1}^{N} e^{-w^T x_n y_n} \tag{36}$$

where y_n is a binary variable taking the value $+1$ or -1. Assume further that the d-dimensional weight vector $\tilde{w}$ separates correctly all the n training data, achieving zero classification error on the training set, that is $y_i(\tilde{w})^T x_n \geq \epsilon, \forall n = 1, \cdots, n \quad \epsilon > 0$. In some cases below (it will be clear from context) we incorporate y_n into x_n.

1. *Dynamics* The dynamics is

$$\dot{w} = F(w) = -\nabla_w L(w) = \sum_{n=1}^{N} x_n^T e^{-x_n^T w} \tag{37}$$

thus $F(w) = \sum_{n=1}^{N} x_n^T e^{-x_n^T w}$.

It is well-known that the weights of the networks that change under the dynamics must be in the vector space spanned by the examples x_n; components of the weights in the null space of the matrix of examples X^T are invariant to the dynamics, exactly as in the square loss case. Unlike the square loss case, the dynamics of the weights diverges but the limit $\frac{w}{|w|}$ is finite and defines the classifier. This means that if a few components of the gradient are zero (for instance when the matrix of the examples is not full rank – which is the case if $d > n$) the associated component of the vector w will not change anymore and the corresponding component in $\frac{w}{|w|}$ will decrease to zero because the norm is increasing. This is one intuition of why in Srebro result there may not be dependence on initial conditions, unlike the square loss case.

2. Though there are no equilibrium points at any finite w, we can look at the Jacobian of F – and Hessian of $-L$ – for a large but finite w (this is the case if we have a small regularization term $\lambda\rho$ in the dynamics (see 21). The Hessian is

$$H = -\sum_{n=1}^{N}(x_n^i)(x_n^j)e^{-(w^T x_n)}. \tag{38}$$

The linearization of the dynamics around any finite w yields a convex but not strictly convex L, since H is the negative sum of auto-covariance matrices. The Hessian is semi-negative definite in general. It is negative definite if the examples span the whole space but it is degenerate with some zero eigenvalues if $d > n$.

The dynamics of perturbation around some w^0 is given by

$$\dot{\delta w} = H(w^0)\delta w. \tag{39}$$

3. *Regularization* If an arbitrarily small regularization term such as $\lambda w^2 = \lambda\rho$ is added to the loss, the gradient will be zero for finite values of w – as in the case of the square loss. Different components of the gradient will be zero for different v w_i. At this equilibrium point the dynamic is hyperbolic and the Hartman-Grobman theorem directly applies to nonlinear dynamical system:

$$\dot{w} = -\nabla_w(L + \lambda|w|^2) = \sum_{n=1}^{N} y_n x_n^T e^{-y_n(x_n^T w)} - \lambda w. \tag{40}$$

The minimum is given by $\sum_n \boldsymbol{x}_n e^{-x_n^T w} = \lambda \boldsymbol{w}$, which can be solved by $\boldsymbol{w} = \sum_n k_n \boldsymbol{x}_n$ with $e^{-k_n x_n \cdot \sum_j x_j} = k_n\lambda$ for $n = 1, \ldots, N$.

The Hessian of $-L$ in the linear case for w^0 s.t. $\sum_n y_n(x_n)e^{-y_n(x_n^T w^0)} = \lambda(w^0)$ is given by

$$-\sum_{n=1}^{N} x_n^T x_n e^{-y_n(x_n^T w^0)} - \lambda \tag{41}$$

which is always negative definite, since it is the negative sum of the coefficients of positive semi-definite auto-covariance matrices and $\lambda > 0$. This means that the minimum of L is hyperbolic and linearization gives the correct behavior for the nonlinear dynamical system.

As before for the square loss, explicit regularization ensures the existence of a hyperbolic equilibrium, independent of initial conditions and of perturbations. This result has been confirmed by numerical simulations.

In this case the equilibrium exists for any $\lambda > 0$ at a finite value of w which increases to ∞ for $\lambda \to 0$. In the limit of $\lambda \to 0$ the equilibrium converges to a maximum margin solution for $\tilde{w} = \frac{w}{||w||}$.

14.3 Weight normalization, linear case

Assume that all x_n are normalized vectors in $\mathbf{R}^{d+1}$ (that is they are on $\mathbf{S}^d$) with one of the components set to 1, corresponding to a bias term. Suppose the data are linearly separable (that is, there exists a $\tilde{w}$ such that $\tilde{w}^T x_n > 0 \quad \forall n = 1, \cdots, N$; this is always true if $N \leq d + 1$). The dynamical system is

$$\dot{\tilde{w}} = \frac{\sum_n e^{-\rho \tilde{w}^T x_n}}{\rho}(x_n - \tilde{w}\tilde{w}^T x_n). \tag{42}$$

and

$$\dot{\rho} = \sum_n e^{-\rho \tilde{w}^T x_n} \tilde{w}^T x_n \tag{43}$$

with the following properties: $\tilde{w} = \frac{w}{\rho}$ with $\rho = ||w||$ and $||\tilde{w}|| = 1$; $\rho \geq 0$.

The dynamics imply $\dot{\tilde{w}} \to 0$ for $t \to \infty$, while $||\tilde{w}|| = 1$. Unlike the square loss case for w, the degenerate components of $\dot{\tilde{w}}$ are updated by the gradient equation. Thus the dynamics for these components is independent of the initial conditions. Note that the constraint $||\tilde{w}|| = 1$ is automatically enforced by the definition of $\tilde{w}$.

Suppose there are degenerate, that is zero, components of the gradient vector $\sum_n e^{-||w||f_n} x_n$, because for instance the data are not full rank. Under the dynamic these components will be driven to zero as long as $w^T x_n > 0$.

In Equation (85) $\dot{\tilde{w}}$ is orthogonal to $\tilde{w}$. This implies that the update $d\tilde{w}$ to $\tilde{w}$ is orthogonal to $\tilde{w}$ and thus does not change its norm. Furthermore, the equation is stable w.r.t. perturbations of $\tilde{w}$, implying that unit normalization is stable under the dynamics (consider $\dot{\tilde{w}} = x_n - \tilde{w}\tilde{w}^T x_n$; $\tilde{w}^T \dot{\tilde{w}} = 0$; if $||\tilde{w}|| > 1$ then the dynamics implies that $\tilde{w}^T \dot{\tilde{w}} < 0$).

Consider the dynamical system $\dot{\tilde{w}} = x_n - \tilde{w}\tilde{w}^T x_n$ for a single, generic x_n. The condition $\dot{\tilde{w}} = 0$ gives $\tilde{w} = \tilde{x_n}$.

With WN one can prove that the normalized weights converge a minimum norm minimizer, which is identical to the support vector machine solution for hard margin. The basic result is the same already obtained by [1] but our approach is different and relies on the use of GD with unit constraint.

Consider the full dynamics

$$\dot{\tilde{w}} = \frac{\sum_n e^{-||w||\tilde{w}^T x_n}}{||w||}(x_n - \tilde{w}\tilde{w}^T x_n).$$

For large t and corresponding large $||w||$ the terms in the summation which have the smallest (positive) value of the dot product (in general more terms than one) $\tilde{w}^T x_n$ dominate. This is because for large $||\alpha||$, the term $\sum_n e^{-\alpha x_n} \approx e^{-\alpha x_*}, x_* = min_n x_n$. Thus $\dot{\tilde{w}} \approx \frac{e^{-||w||\tilde{w}^T x_*}}{||w||} \sum_*(x_* - \tilde{w}\tilde{w}^T x_*)$, where $\sum_*$ indicates a sum over the support vectors. This converges to

$$\tilde{w} = \sum_* \alpha_* x_*,$$

where x_* can be considered normalized, though this is not a restriction. This is the hard margin SVM solution.

15 Deep networks: square loss

$$L(f(w)) = \sum_{n=1}^{N}(y_n - f(W; x_n))^2 \tag{44}$$

Here we assume that the function $f(W)$ achieves zero loss on the training set, that is $y_n = f(W; x_n) \quad \forall n = 1, \cdots, N$.

1. *Dynamics*

   The dynamics now is

   $$\dot{(W_k)}_{i,j} = -F_k(w) = -\nabla_{W_k} L(W) = 2\sum_{n=1}^{N} E_n \frac{\partial f}{\partial (W_k)_{i,j}} \tag{45}$$

   with $E_n = (y_n - f(W; x_n))$.

2. The Jacobian of $-F$ – and Hessian of $-L$ – for $W = W_0$ is

   $$\begin{aligned} J(W)_{kk'} &= 2\sum_{n=1}^{N}(-(\nabla_{W_k} f(W; x_n))(\nabla_{W^{k'}} f(W; x_n)) + E_n \nabla^2_{W_k, W^{k'}} f(W; x_n)) \\ &= -2\sum_{n=1}^{N}(\nabla_{W_k} f(W; x_n))(\nabla_{W^{k'}} f(W; x_n)), \end{aligned} \tag{46}$$

   where the last step is because we assume perfect interpolation of the training set, that is $E_n = 0 \forall n$. Note that the Hessian involves derivatives across different layers, which

introduces interactions between perturbations at layers k and k'. The linearization of the dynamics around W_0 for which $L(W_0) = 0$ yields a convex L, since the Hessian of $-L$ is semi-negative definite. In general we expect several zero eigenvalues because the Hessian of a deep overparametrized network under the square loss is degenerate as shown by the following theorem in Appendix 6.2.4 of [49]:

Theorem 6 *(K. Takeuchi) Let H be a positive integer. Let $h_k = W_k\sigma(h_{k-1}) \in \mathbb{R}^{N_k,n}$ for $k \in \{2,\dots,H+1\}$ and $h_1 = W_1X$, where $N_{H+1} = d'$. Consider a set of H-hidden layer models of the form, $\hat{Y}_n(w) = h_{H+1}$, parameterized by $w = \mathrm{vec}(W_1,\dots,W_{H+1}) \in \mathbb{R}^{dN_1+N_1N_2+N_2N_3+\cdots+N_HN_{H+1}}$. Let $L(w) = \frac{1}{2}\|\hat{Y}_n(w) - Y\|_F^2$ be the objective function. Let w^* be any twice differentiable point of L such that $L(w^*) = \frac{1}{2}\|\hat{Y}_n(w^*) - Y\|_F^2 = 0$. Then, if there exists $k \in \{1,\dots,H+1\}$ such that $N_kN_{k-1} > n \cdot \min(N_k, N_{k+1},\dots,N_{H+1})$ where $N_0 = d$ and $N_{H+1} = d'$ (i.e., overparametrization), there exists a zero eigenvalue of Hessian $\nabla^2 L(w^*)$.*

3. *Regularization* Explicit quadratic regularization adds terms like $\lambda_k||W_k||^2$ to the loss, shifting the minima. Unfortunately this also means that $E_n \neq 0$. Thus the the Hessian cannot be guaranteed to be negative definite for any $\lambda > 0$ and in general is expected to be degenerate.

4. *NMGD* The gradient flow corresponds to $w_{t+1} = Tw_t$ with $T = I - \nabla_w L$ and the operator T is not non-expansive.

16 Deep networks: exponential loss

Consider the exponential loss

$$L(f(W)) = \sum_{n=1}^{N} e^{-f(W;x_n)y_n} \tag{47}$$

with definitions as before. We assume that $f(W;x)$, parametrized by the weight vectors W_k, separates correctly all the n training data x_i, achieving zero classification error on the training set for $W = W^0$, that is $y_i f(W^0;x_n) > 0, \forall n = 1,\cdots,N$. Observe that if f separates the data, then $\lim_{a\to\infty} L(af(W^0)) = 0$ and this is where gradient descent converges [1].

There is no critical point for finite t. Let us linearize the dynamics around a large W^0 by approximating $f(W^0 + \Delta W_k)$ with a low order Taylor approximation for small ΔW_k.

1. *Dynamics*

   The gradient flow is not zero at any finite $(W^0)_k$. It is given by

$$\dot{W}_k = \sum_{n=1}^{N} y_n[\frac{\partial f(W;x_n)}{\partial W_k}]e^{-y_n f(x_n;W)} \tag{48}$$

where the partial derivatives of f w.r.t. W_k can be evaluated in W_0.

Let us consider a small perturbation of W_k around W^0 in order to linearize F around W^0.

2. The linearized dynamics of the perturbation are $\delta \dot{W}_k = J(W)\delta W$, with

$$J(W)_{kk'} = -\sum_{n=1}^{N} e^{-y_n f(W_0;x_n)} \left(\frac{\partial f(W;x_n)}{\partial W_k} \frac{\partial f(W;x_n)}{\partial W_{k'}} - y_n \frac{\partial^2 f(W;x_n)}{\partial W_k \partial W_{k'}} \right) \Bigg|_{W^0}. \tag{49}$$

Note now that the term containing the second derivative of f does not vanish at a minimum, unlike in the square loss case.

3. *Regularization*

Adding a regularization term of the form $\sum_{i=1}^{K} \lambda_k ||W_k||^2$ yields for $i = 1, \cdots, K$

$$\dot{W}_k = -\nabla_w (L + \lambda |W_k|^2) = \sum_{n=1}^{n} y_n \frac{\partial f(W;x_n)}{\partial W_k} e^{-y_n f(x_n;W)} - \lambda_k W_k \tag{50}$$

For compactness of notation, let us define

$$g_k^{(n)} = y_n \frac{\partial f}{\partial W_k} e^{-y_n f(W;x_n)}, \tag{51}$$

with which we have a transcendental equation for the minimum.

$$\lambda_k (W_k)_{min} = \sum_n g_k^{(n)}. \tag{52}$$

The negative Hessian of the loss is then

$$H_{kk'} = \sum_n \frac{\partial g_k^{(n)}}{\partial W_{k'}} - \lambda_k \delta_{kk'} \mathbb{I}. \tag{53}$$

17 Deep Networks, weight normalization

Using Appendix 21, we obtain the dynamics for the normalized weights

$$\dot{\tilde{W}}_k^{i,j} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \frac{\rho}{\rho_k^2} \left(\frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_k^{i,j}} - \tilde{W}_k^{i,j} \tilde{W}_k^{a,b} \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_k^{a,b}} \right). \tag{54}$$

In Equation (54) for large enough $||W|| = \rho$, the sum is equivalent to a max operation, choosing the j such that $f(x_j) = \max_n f(x_n)$. Notice that the components of $\tilde{W}_k^{i,j}$ which are not changed by gradient descent near the minimum will nevertheless change under this normalized dynamics.

The solution of $\dot{\tilde{W}}_k^{i,j} = 0$ is then given by (using non-zero α_n associated with the support vecors)

$$0 = \sum \alpha_n \left(\frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_k^{i,j}} - \tilde{W}_k^{i,j} \tilde{W}_k^{a,b} \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_k^{a,b}} \right). \tag{55}$$

Denoting $\sum \alpha_n \tilde{f}(x_n) = \hat{f}$ we obtain a set of coupled equations for the $\tilde{W}_k^{i,j}$ as

$$\tilde{W}_k^{i,j} = \frac{1}{\hat{f}} \frac{\partial \hat{f}}{\partial \tilde{W}_k^{i,j}}. \tag{56}$$

Near zero exponential loss the Hessian in positive semidefinite. However, it seems impossible to control the non-diagonal part of the Hessian to ensure its positive definitenss. Empirically the Hessian is typically found to have a few positive (stable) eigenvalues and many zero eigenvalues.

Remarks

- At the stationary points $f(x) = (f'_k)^T f'_k$ and $f(x) = \frac{1}{K+1} \sum_{k=0}^{k=K} (f'_k)^T f'_k$ because $W_k = f'_k$.

18 Dynamics of Lagrange multiplier approach

Notice that $\dot{V}_k(t) = 0$ implies the following constraints – *as many as the number of weights*:

$$V_k = \frac{g(t)}{2\lambda}. \tag{57}$$

We call the constraints Equations 57 the MN equations. For the special case of a linear, one layer network we know that such a solution exist and is unique.

Notice that $V_k^T \frac{\partial \tilde{f}(x_n)}{\partial V_k} = f(x)$ because of Equation 3 implying that $\alpha_k \sum_i f(x_i) = 1$.

The intuition is that among all the solutions V_k that separate the data – that is weights such that $f(x_n) y_n > 0, \forall n$ – the MN equations select the simplest one with equal weights across layers.

Linear case Consider a linear network $f(x) = OW_2W_1x$ where O is a fixed vector $O = \frac{1}{d}(1, .., 1)$ summing the outputs of W_2 into a single scalar. Let us consider normalized weights. Then $\frac{\partial \tilde{f}(x)}{\partial V_1} = O^T V_2^T x^T$. In detail

$$f(x) = \sum_{i,jk} O_i V_2^{i,j} V_1^{j,k} x^k \tag{58}$$

gives

$$\frac{\partial \tilde{f}(x)}{\partial V_1^{q,h}} = \sum_{i,j,k} O^i V_2^{i,j} \delta^{j,q} \delta^{k,h} x^k = \sum_i O^i V_2^{i,q} x^h \tag{59}$$

which via previous Equations yields $V_1^{j,k} = \alpha_1 \sum_k O^i V_2^{i,j} x^k$

Now replace in the network V_1 with the expression above. This yields $f(x) = \alpha_1 O V_2 O^T V_2^T x^T x = \alpha_1 O V_2 O^T V_2^T$ assuming x is also normalized. Also $\frac{\partial \tilde{f}(x)}{\partial V_2} = O^T x^T V_1^T$ which substituted in f gives $f(x) = O O^T x^T V_1^T V_1 x$. Using indeces, we have $f(x) = \alpha_1 \sum_{i,j,k,l} O_i V_2^{i,j} O^k V_2^{k,j} x^l x^l$.

Nonlinear case Consider a nonlinear network $f(x) = O\sigma(W_2\sigma(W_1 x))$ where O is as before. Let us normalize weights as before. We use Lemma 3.1 in [8] to rewrite f as

$$f(x) = OD_2W_2D_1W_1x \tag{60}$$

where D_i are diagonal matrices with elements that are either 0 or 1 and represent $\frac{\partial\sigma(z)}{\partial z}$ using the property $\sigma(z) = \frac{\partial\sigma(z)}{\partial z}z$. Equation 60 is a linear equation that can be used with care to check consistency and meaning of the NM conditions in the nonlinear case similarly to the linear case.

The Hessian of L wrt V_k tells us about the linearized dynamics around a minimum where the gradient is zero. The Hessian is

$$\sum_n \left[-\left(\prod_{i=1}^K \rho_i^2\right) \frac{\partial \tilde{f}(V;x_n)}{\partial V_k} \frac{\partial \tilde{f}(V;x_n)}{\partial V_{k'}}^T + \left(\prod_{i=1}^K \rho_i\right) \frac{\partial^2 \tilde{f}(V;x_n)}{\partial V_k \partial V_{k'}} \right] e^{-\prod_{i=1}^K \rho_i \tilde{f}(V;x_n)} - 2\lambda \mathbf{I}. \tag{61}$$

19 Degeneracy of the Hessian for deep networks

As we have seen previously, adding L2 regularization to the loss of a linear network, be it for square loss or exponential, has the effect of providing stability to gradient descent. This is because the Hessian of a non-regularized linear network is positive semi-definite everywhere, meaning that there exist direction in which perturbations do not diminish over time. Adding the term λw^2 however forces the Hessian to be positive definite everywhere.

One might suspect that similar behavior might be exhibited by deep networks too. However, as seen above, away from the critical points the Hessian of deep nets can have eigenvalues of all signatures. Close to critical points obtained by GD however, numerical studies [50] show that eigenvalues of non-regularized networks are non-negative, though many of them are 0.

Naively, adding a quadratic term should make the previously degenerate point have a positive definite Hessian. Notice however, that adding the regularization term shifts the position of the critical point and there are no a priori guarantees that the new minimum should be non-degenerate. In fact, the result below shows us it is not true.

Theorem 7 *For deep neural networks with exponential type losses, adding an L2 regularization term $\lambda||W||_F^2$ does not guarantee non-degenerate critical points.*

Proof It suffices to show that there exists a network with an exponential loss that has degenerate critical points independently of the value of λ. Consider the simplest case of a 2-layer network with 4 weights $w_1, \dots w_4$ and one training example $x = (1, 1)$. The loss is then

$$L(w) = e^{-w_1w_2-w_3w_4} + \lambda(w_1^2 + w_2^2 + w_3^2 + w_4^2) \tag{62}$$

Note first that with $\lambda = 0$, this loss has a minimum at infinity and a saddle point at the origin. It is easy to verify that at the critical points we have

$$w_1^* = \frac{e^{-f_*}}{2\lambda} w_2^*, \qquad w_2^* = \frac{e^{-f_*}}{2\lambda} w_1^*, \tag{63}$$

where $f = w_1 w_2 + w_3 w_4$. These imply that there are two sets of critical points: the origin and the points defined by $e^{-w_1 w_2 - w_3 w_4} = 2\lambda$. The determinant of Hessian of this loss is given by

$$\det H = e^{-4f_*} \left(4\lambda^2 e^{-2f_*} - 1\right) \left(4\lambda^2 e^{-2f_*} + 2\lambda e^{f_*}(w_1^2 + w_2^2 + w_3^2 + w_4^2) + 2f_* - 1\right). \tag{64}$$

At the origin we find a local minimum with a positive definite Hessian, but at $e^{-f_*} = 2\lambda$ the determinant is 0. Thus for arbitrary λ, the global minimum is degenerate. This degeneracy stems from a freedom of reparametrization provided by two circles in the w_1-w_2 and w_3-w_4 planes[2]. This example works not only for the exponential loss, but also for the logistic loss without any modifications. We thus find that, at least for exponential type losses, adding regularization might not provide stabilization of gradient descent. While the counter-example is very simple, it is not isolated – simple numerical checks show that the situation is generic.

In fact, simple analysis at the level of symmetries of the regularized loss, in the vein of [51], gives us the number of zero eigenvalues in the deep linear case. Consider neural networks $f(x, W_k) = W_L W_{L-1} \cdots W_2 W_1 x$. If $W_k \in \mathbb{R}^{d_k, d_{k-1}}$ and $x \in \mathbb{R}^{d_0}$, then the neural network is invariant under the action of the group $\mathrm{GL}_{d_{L-1}}(\mathbb{R}) \times \mathrm{GL}_{d_{L-2}}(\mathbb{R}) \times \cdots \times \mathrm{GL}_{d_1}(\mathbb{R})$, that is invertible $d_k \times d_k$ matrices between the layers acting as $(W_k, W_{k+1}) \mapsto (GW_k, W_{k+1}G^{-1})$. The Hessian of an unregularized loss of this neural network will thus have number of zero eigenvalues equal to the dimensionality of this group.

What happens when we add a regularizer $\sum_k \lambda ||W_k||_F^2$? The regularized loss is no longer invariant under the action of this large group. Note, however, that the Frobenius norm of a matrix is left invariant under a multiplication by an orthogonal (rotation) matrix. Hence the regularized loss is invariant under the action of the group $\mathrm{O}_{d_{L-1}}(\mathbb{R}) \times \mathrm{O}_{d_{L-2}}(\mathbb{R}) \times \cdots \times \mathrm{O}_{d_1}(\mathbb{R})$. While this is a smaller group, it still provides zero eigenvalues to the Hessian.

The situation becomes more complicated and data-dependent when we move to the nonlinear case – ReLU activations can vary between layers, and remove many of these symmetries. If there are however small regions in the network which can be rotated into each other, then we would still expect zero eigenvalues in the Hessian. We plan, using tools from Random Matrix Theory, to investigate this question in future work.

20 $T = I - \nabla L(f)$ is a non-expanding operator

Definition 8 *A function $g(\cdot) : \mathbf{R}^n \to \mathbf{R}$ is L-smooth if its gradients are Lipschitz continuous that is $\forall x, y \in \mathbf{R}^n$*

$$||\nabla g(x) - \nabla g(y) \leq L||x - y|| \tag{65}$$

The definition above is equivalent to say that $T = I - \nabla_W L(f)$ is a non-expanding operator.

Lemma 9 *A sufficient condition for L-smoothness is that the eigenvalues of $H = \nabla^2 g$ are bounded from above and from below by L.*

[2]This construction actually stems from reparametrizing a Gaussian in a quadratic potential well.

Proof The 2-norm of the Hessian $||H||_2$ is equal to the absolute value of its highest eigenvalue $\lambda_{max} \leq L$. Let us consider the function $v^T \nabla g\left(z_\beta y\right)$, where v is an arbitrary vector for now and $z_\beta = (\beta x + (1-\beta)$. By the mean value theorem, there exists $\beta \in (0,1)$, such that

$$v^T(\nabla g(x) - \nabla g(y)) = v^T \nabla^2 g(z_\beta)(x-y).$$

Notice that we have

$$||\nabla g(x) - \nabla g(y)|| = \sup_{||v||=1} v^T(\nabla g(x) - \nabla g(y)) = \sup_{||v||=1} v^T \nabla^2 g(z_\beta)(x-y).$$

Applying the Cauchy-Schwarz theorem and the assumption we obtain

$$\sup_{||v||=1} v^T H(x-y) \leq L||x-y||,$$

which completes the proof.

Lemma 10 *$\nabla g(x) = 0$ if and only if $x \in \mathbf{R}^n$ is a fixed point of the operator $T^\gamma : \mathbf{R}^n \to \mathbf{R}^n$ with $T^\gamma x = x - \gamma \nabla g(x)$, that is $T^\gamma x = x$ for non-zero γ.*

Theorem 11 *For convex $g(x)$, the operator T^γ defined as $T^\gamma x = x - \gamma \nabla g(x)$ is non-expanding that is*

$$||T^\gamma x - T^\gamma y|| \leq ||x-y|| \tag{66}$$

Proof The standard proof uses the mean value theorem to write

$$T^\gamma x - T^\gamma y = (x-y)(I - \gamma \nabla^2 g(z)) \tag{67}$$

with $z = \beta x + (1-\beta)y$ for a certain $\beta \in [0,1]$. Then submultiplicativity of norms yields

$$||T^\gamma x - T^\gamma y|| \leq ||x-y||\ ||(I - \gamma \nabla^2 g(z))||. \tag{68}$$

The last term on the right is the norm of the matrix $I - H$ where H is the Hessian we consider for various verions of GD (in the square and exponential loss cases). For weight normalization, for instance, the smallest eigenvalue of H is 0 and the largest is 1. In this case $||T^\gamma x - T^\gamma y|| \leq ||x-y||$ for any $0 < \gamma \leq 1$.

Notice that the usual assumption in analyzing gradient descent methods from the point of view of fixed-points is that is T is *contractive*. This means that H must be positive definite with positive eigenvalues. In our analysis here we only need H to be *positive semidefinite*, in particular degenerate as in the case of weight normalization (and others of our GD cases).

Theorem 12 *Assume that gradient descent starts from w_0 with g which is L-smooth*

$$w_{t+1} = w_t - \gamma \nabla g(w_t) \tag{69}$$

and converges to a minimum w_*. If $\gamma L < 1$, then
(a)

$$||w_{t+1} - w_*||^2 \leq (1 - \gamma L)^{2(t+1)} ||w_0 - w_*||^2 \tag{70}$$

(b) Additionally, if g is μ-strongly convex, then this can be strengthened to

$$||w_{t+1} - w_*||^2 \leq (1 - \gamma \mu)^{t+1} ||w_0 - w_*||^2 \tag{71}$$

Proof (a) L-smoothness gives us that

$$L||w_t - w_*||^2 \geq -(\nabla g(w_t) - \nabla g(w_*), w_t - w_*) \geq -L||w_t - w_*||^2. \tag{72}$$

We then have

$$\begin{aligned}
||w_{t+1} - w_*||^2 &= ||w_t - w_* - \gamma \nabla g(w_t)||^2 \\
&= ||w_t - w_*||^2 - 2\gamma(\nabla g(w_t), w_t - w_*) + \gamma^2 ||\nabla g(w_t)||^2 \\
&= ||w_t - w_*||^2 - 2\gamma(\nabla g(w_t) - \nabla g(w_*), w_t - w_*) + \gamma^2 ||\nabla g(w_t) - \nabla g(w_*)||^2 \\
&\leq (1 - 2\gamma L)||w_t - w_*||^2 + \gamma^2 ||\nabla g(w_t) - \nabla g(w_*)||^2 \\
&\leq (1 - 2\gamma L + \gamma^2 L^2)||w_t - w_*||^2
\end{aligned}$$

where we have used the fact that $\nabla g(w_*) = 0$ in third equality, the result (72) in the first inequality and finally the definition of L-smoothness.

(b) μ-strong convexity gives us

$$(\nabla g(w_t), w_* - w_t) \leq g(w_*) - g(w_t) - \frac{\mu}{2}||w_t - w_*||^2$$

It follows similarly to the previous case that

$$\begin{aligned}
||w_{t+1} - w_*||^2 &= ||w_t - w_* - \gamma \nabla g(w_t)||^2 \\
&= ||w_t - w_*||^2 - 2\gamma(\nabla g(w_t), w_t - w_*) + \gamma^2 ||\nabla g(w_t)||^2 \\
&\leq (1 - \gamma\mu)||w_t - w_*||^2 - 2\gamma(g(w_t) - g(w_*)) + \gamma^2 ||\nabla g(w_t)||^2 \\
&\leq (1 - \gamma\mu)||w_t - w_*||^2 - 2\gamma(g(w_t) - g(w_*)) + 2\gamma^2 L(g(w_t) - g(w_*)) \\
&\leq (1 - \gamma\mu)||w_t - w_*||^2 - 2\gamma(1 - \gamma L)(g(w_t) - g(w_*)) \\
&\leq (1 - \gamma\mu)||w_t - w_*||^2
\end{aligned}$$

since $-2\gamma(1 - \gamma L) < 0$.

21 Weight normalization

- *We introduce a GD technique to be used under the exponential loss which is very similar to the "classical" weight normalization*
- *We show almost equivalence between them.*

- *We also prove that weight normalization does not converge to the normalized vector obtained by running GD on w and normalizing it at the end.*

We consider here the dynamics of the normalized network with normalized weight matrices $\tilde{W^k}$ induced by the gradient dynamics of W^k, where W^k is the weight matrix of layer k. Normalization is equivalent to changing coordinates from W_k to $\tilde{W}_k$ and $\rho = ||W_k||$. For simplicity of notation we consider here for each weight matrix W^k the corresponding "vectorized" representation in terms of vectors $W^k_{i,j} = (w)_i$ that we denote as w (dropping the indices k, j for convenience).

We use the following definitions and properties:

- Define $\frac{w}{||w||} = \tilde{w}$; thus $w = ||w||\tilde{w}$ with $||\tilde{w}|| = 1$.
- The following relations are easy to check:
  1. $\frac{\partial ||w||}{\partial w} = \tilde{w}$
  2. $\frac{\partial \tilde{w}}{\partial w} = \frac{I - \tilde{w}\tilde{w}^T}{||w||} = S$. S has at most one zero eigenvalue since $\tilde{w}\tilde{w}^T$ is rank 1 with a single eigenvalue $\lambda_1 = 1$.
  3. $Sw = S\tilde{w} = 0$
  4. $||w||S^2 = S$
  5. $\frac{\partial ||\tilde{w}||^2}{\partial w} = 0$
- We assume $f(w) = f(||w||, \tilde{w}) = ||w||f(1, \tilde{w}) = ||w||\tilde{f}$.
- Thus $\frac{\partial f}{\partial w} = \tilde{w}\tilde{f} + |w|S\frac{\partial \tilde{f}}{\partial \tilde{w}}$

The gradient descent dynamic system used in training deep networks for the exponential loss of Equation 47 is given by

$$\dot{w} = -\frac{\partial L}{\partial w} = \sum_{n=1}^{N} y_n \frac{\partial f(x_n; w)}{\partial w_i} e^{-y_n f(x_n; w)} \tag{73}$$

with a Hessian given by (assuming $y_n f(x_n) > 0$ and dropping y_n)

$$H = \sum_{n=1}^{N} e^{-f(x_n;w)} \left(\frac{\partial f(x_n; w)}{\partial w} \frac{\partial f(x_n; w)}{\partial w}^T - \frac{\partial^2 f(x_n; w)}{\partial w^2}\right) \tag{74}$$

The dynamics above for w induces the following dynamics for $||w||$ and $\tilde{w}$:

$$\dot{||w||} = \frac{\partial ||w||}{\partial w}\dot{w} = \tilde{w}\dot{w} \tag{75}$$

and

$$\dot{\tilde{w}} = \frac{\partial \tilde{w}}{\partial w}\dot{w} = S\dot{w} \tag{76}$$

Thus

$$||\dot{w}|| = \tilde{w}^T \dot{w} = \frac{1}{||w||} \sum_{n=1}^{N} w^T \frac{\partial f(x_n; w)}{\partial w_i} e^{-f(x_n;w)} = \sum_{n=1}^{N} e^{-||w|| \tilde{f}(x_n)} \tilde{f}(x_n) \tag{77}$$

where, assuming that w is the vector corresponding to the weight matrix of layer k, we obtain $(w^T \frac{\partial f(w;x)}{\partial w}) = f(w;x)$ because of Lemma 1 in [8]. We assume that f separates all the data, that is $y_n f(x_n) > 0 \ \ \forall n$. Thus $\frac{d}{dt}||w|| > 0$ and $\lim_{t\to\infty} ||\dot{w}|| = 0$. In the 1-layer network case the dynamics yields $||w|| \approx \log t$ asymptotically. For deeper networks, this is different. In Section 22 we show that the product of weights at each layer diverges faster than logarithmically, but each individual layer diverges slower than in the 1-layer case. By defining

$$\sum_n e^{-|||w|| \tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} = \tilde{B}, \tag{78}$$

the Equation above becomes

$$\dot{\tilde{w}} = \frac{I - \tilde{w}\tilde{w}^T}{||w||} \tilde{B} = \frac{\sum_n e^{-||w|| \tilde{f}(x_n)}}{||w||} (\frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} - \tilde{w} \tilde{f}(x_n)) \tag{79}$$

or alternatively

$$\dot{\tilde{w}} = \frac{\sum_n e^{-||w|| \tilde{f}(x_n)}}{||w||} (\frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} - \tilde{w}\tilde{w}^T \frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}}). \tag{80}$$

The dynamics above for w induces the following dynamics for $||w||$ and $\tilde{w}$:

$$||\dot{w}|| = \frac{\partial ||w||}{\partial w} \dot{w} = \tilde{w} \dot{w} \tag{81}$$

and

$$\dot{\tilde{w}} = \frac{\partial \tilde{w}}{\partial w} \dot{w} = S \dot{w} \tag{82}$$

Thus

$$||\dot{w}|| = \tilde{w}^T \dot{w} = \frac{1}{||w||} \sum_{n=1}^{N} w^T \frac{\partial f(x_n; w)}{\partial w} e^{-f(x_n;w)} = \sum_{n=1}^{N} e^{-||w|| \tilde{f}(x_n)} \tilde{f}(x_n) \tag{83}$$

where, assuming that w is the vector corresponding to the weight matrix of layer k, we obtain $(w^T \frac{\partial f(w;x)}{\partial w}) = f(w;x)$ because of Lemma 1 in [8]. We assume that f separates all the data, that is $y_n f(x_n) > 0 \ \ \forall n$. Thus $\frac{d}{dt}||w|| > 0$ and $\lim_{t\to\infty} ||\dot{w}|| = 0$.

For $\dot{\tilde{w}}$ we obtain

$$\dot{\tilde{w}} = \frac{I - \tilde{w}\tilde{w}^T}{||w||} \sum_{n=1}^{N} y_n \frac{\partial f(x_n; w)}{\partial w} e^{-y_n f(x_n;w)} \tag{84}$$

which gives (incorporating the labels y_n)

$$\dot{\tilde{w}} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} (\frac{1}{\rho} (\frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} - \tilde{w}\tilde{w}^T \frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}}). \tag{85}$$

A version of the algorithm to be used with NM iterations can also have a regularization term in the ρ dynamics with a very small λ_ρ of the form

$$\dot{\rho} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \tilde{f}(x_n) - \lambda_\rho \rho \tag{86}$$

to constrain ρ to be large but finite.

For large ρ the sum over exponentials become close to a max operations, effectively selecting the x_n with the smallest margin $\tilde{f}(x_n)$.

Deep Networks

The results generalize to weight matrices

- $\frac{W_k^{i,j}}{||W_k||} = \tilde{W}_k^{i,j}$; thus $||\tilde{W}_k|| = 1$.
- $\frac{\partial ||W_k||}{\partial W_k^{i,j}} = \tilde{W}_k^{i,j}$
- $S_{k,k'}{}^{i,j,a,b} = \frac{\partial \tilde{W}_{k'}^{i,j}}{\partial W_k^{a,b}} = \delta_{k,k'} \frac{\delta^{ia}\delta^{jb} - \tilde{W}_k^{ij}\tilde{W}_k^{ab}}{||W_k||}$.

Thus the generalization of Equation 85 is straightforward, with the one nontrivial step being that we need to be careful about the product of weights $\rho = \prod_{k=1}^{K} \rho_k$. Notice above, that the definition of the projector S depends on the layer now, so by switching from $\partial f / \partial W_k$ to $\partial \tilde{f} / \partial \tilde{W}_k$ we we gain additional ratios of norms:

$$S_k^{i,j,a,b} \frac{\partial f}{\partial W_k^{ab}} = \frac{\rho}{\rho_k} S_k^{i,j,a,b} \frac{\partial \tilde{f}}{\partial \tilde{W}_k^{a,b}} \tag{87}$$

Weight norm and gradient descent trajectory

It is important to note that the dynamics of gradient descent with weight normalization are different to those of standard gradient descent. This is because the matrix S depends on the weights w, so running standard gradient descent and normalizing at the end leads to different trajectories. This can be simply seen from the integration of the dynamical system. For weight normalization we have for any $T > 0$:

$$\begin{aligned} \tilde{w}(T) &= \tilde{w}(0) + \sum_{n=1}^{N} y_n \int_0^T dt \frac{I - \tilde{w}\tilde{w}^T}{||w||} \frac{\partial f(x_n; w)}{\partial w} e^{-y_n f(x_n; w)} \\ &\neq \frac{w(0) + \sum_n y_n \int_0^T dt \frac{\partial f(x_n;w)}{\partial w} e^{-y_n f(x_n;w)}}{||w(0) + \sum_n y_n \int_0^T dt \frac{\partial f(x_n;w)}{\partial w} e^{-y_n f(x_n;w)}||} = \frac{w(T)}{||w(T)||}. \end{aligned} \tag{88}$$

In general, normalizing and integration are noncommutative operations[3]

21.1 Standard weight normalization

Standard weight normalization [37] is a reparametrization of the weight vectors in a neural network as follows

$$w = \frac{g}{||v||} v. \tag{89}$$

Thus $\tilde{w} = \frac{v}{||v||}$ and $\rho = ||w||$ g. The gradient descent equations for the $\rho, \tilde{w}$ parametrization are

$$\frac{\partial L}{\partial \rho} = \tilde{w}^T \frac{\partial L}{\partial w} \tag{90}$$

and

$$\frac{\partial L}{\partial \tilde{w}} = S \frac{\partial L}{\partial w} \tag{91}$$

with

$$S = \frac{I - \tilde{w}\tilde{w}^T}{\rho} \tag{92}$$

The gradient descent equations for the g, v parametrization are

$$\frac{\partial L}{\partial g} = \frac{v}{||v||} \frac{\partial L}{\partial w} \tag{93}$$

and

$$\frac{\partial L}{\partial v} = \frac{g}{||v||} S' \frac{\partial L}{\partial w} \tag{94}$$

with

$$S' = I - \frac{vv^T}{||v||^2} = \frac{v^T v - vv^T}{v^T v} \tag{95}$$

The dynamics is qualitatively identical to the dynamics of the previous section.

[3]For example, consider the vector $v = (t, \sqrt{t}, 1)$. We have here $\int dt \dot{\tilde{v}}(t) = \sqrt{t^2 + t + 1} - 1/2 \sinh^{-1}((1+2x)/\sqrt{3})$. Normalizing after the integration would give us obviously $v_1(t)/||v(t)|| = t/\sqrt{t^2 + 16t/9 + 4}$.

21.2 Standard batch normalization

Over the last few wild years in the explosion of deep learning applications, many variations of SGD have been proposed to improve its performance – including Dropout, weight decay etc. One that survived especially well is *batch normalization*. The original paper describes it as a computationally more efficient version of an ideal whitening of a layer of activities by computing the covariance matrix and generating $x_{new} = (x - E[x])[Cov(x)^{-1/2}]$. The following remark seems interesting. Seen in terms of the activations $x_k = Wx_{k-1}$ weight normalization assumes $E[x] = 0$ and then approximates $[Cov(x)^{-1/2}]$ with $\frac{I-WW^T}{|W||}$ assuming x_{k-1} to be whitened. This approximation connects BN and WN.

Furthermore, consider the reparametrization defined in WN. As discussed in the previous section, the weights w are replaced by $w = g\frac{v}{||v||}$. Optimization by SGD induces a dynamics in g and v. How does this compare to Batch Normalization (BN)? In the case of BN, normalization is applied to activations of each unit, rather than the weights themselves. If the input into the BN layer is $i = \mathrm{ReLU}(Wx + b)$, then the activations per channel c are transformed into

$$o_c^{BN} = \gamma_c \frac{i_c - \mathbb{E}_B[i_c]}{\sqrt{\mathrm{Var}_B[i_c]}} + \beta_c, \tag{96}$$

where B denotes the batch of data over which the epectations are evaluated. In the linear case, this replaces Wx with $\gamma\frac{Wx}{\sigma} + \beta$. Now optimization by SGD uses the dynamics of γ and β.

In order to carry on the comparison, let us rewrite the per-channel WN algorithm as performing

$$o_c^{WN} = \gamma_c \frac{W_c x}{||W_c||} + \beta_c. \tag{97}$$

Let us assume that we are evaluating full gradients, that is, there is only a single batch of data. Additionally, assume that the biases are set to zero and for simplicity set $\beta_c = 0$ (which empirically does not have significant impact). While WN focuses on normalizing weights, BN normalizes the product of weight with the outputs of the previous layer. Hence, we expect

$$||o_c^{BN}|| \to \text{const} \qquad \text{but} \qquad ||o_c^{WN}|| \to ||x||.$$

If we however properly normalize the inputs into the network, for example setting $||x|| = 1$, these two goals can be made compatible. In this case, we would expect that $\sigma_c \to ||W_c||$ and the gradient for γ_c to behave like g in WN. In the single batch case of GD, it thus seems that the two algorithms are qualitatively very similar. As shown in [52] however, BN is numerically more stable in the presence of ReLU nonlinearities because it is applied to activations, rather than weights. This is due to the fact that dividing by $||W||$ in the presence of some neurons set to 0 can effectively lead to norms smaller than one. Notice that the problems of the current implementations of WN wrt BN may be eliminated taking into account the stabilizing steps for a numerical implementation suggested in [36]. We plan to explore this question in future work.

Remarks

- Consider WN and BN in terms of coordinate transformations. Then, neglecting biases for simplicity of notation, WN uses a transformation A, defined as $w_{new} = Aw$ such that $||w_{new}|| = 1$. On the other hand the ideal BN uses a transformation C, defined as $w_{new} = Cw$, such that $w_{new}w_{new}^T = \frac{1}{D}I$, where D is the dimensionality of w_{new}. Thus the trace of the matrix $w_{new}w_{new}^T$ is one implying that $||w_{new}|| = 1$. In practice, the current implementation of BN only enforces that the diagonal of $w_{new}w_{new}^T$ should be equal to $\frac{1}{D}I$.

- As discussed above, consider the goal of solving the equation $[Cov(x)]x_{new} = x$. Assume for simplicity of notation that $E[x] = 0$. One of several iterative techniques is the *Jacobi method*, which provides a solution x_{new} as

$$x_{new}^{(t+1)} = D^{-1}(x - Rx_{new}^{(t)}) \tag{98}$$

  where $Cov[x] = D + R$ with D the matrix with the diagonal of $Cov[x]$ and R the matrix equal to $Cov[x]$ apart from a zero diagonal. It may be possible to use this technique to improve the existing batch normalization.

22 Rate of growth of weights

In linear 1-layer networks the dynamics of gradient descent yield $||w|| \sim \log t$ asymptotically. For the validity of the results in the previous section, we need to show that the weights of a deep network also diverge at infinity. In general, the K nonlinearly coupled equations are not easily solved analytically. For simplicity of analysis, let us consider the case of a single training example $N = 1$, as we expect the leading asymptotic behavior to be independent of N. In this regime we have

$$\rho_k\dot{\rho}_k = \tilde{f}(x)\left(\prod_{i=1}^{k}\rho_i\right)e^{-\prod_{i=1}^{K}\rho_i\tilde{f}(x)} \tag{99}$$

Keeping all the layers independent makes it difficult to disentangle for example the behavior of the product of weights $\prod_{i=1}^{K}\rho_i$, as even in the 2-layer case the best we can do is to change variables to $r^2 = \rho_1^2 + \rho_2^2$ and $\gamma = e^{\rho_1\rho^2\tilde{f}(x)}$, for which we still get the coupled system

$$\dot{\gamma} = \tilde{f}(x)^2r^2, \qquad r\dot{r} = 2\frac{\log\gamma}{\gamma}, \tag{100}$$

from which reading off the asymptotic behavior is nontrivial.

As a simplifying assumption let us consider the case when $\rho := \rho_1 = \rho_2 = \ldots = \rho_k$. This gives us the single differential equation

$$\dot{\rho} = \tilde{f}(x)K\rho^{K-1}e^{-\rho_k\tilde{f}(x)}. \tag{101}$$

This implies that for the exponentiated product of weights we have

$$\left(e^{\rho_k\tilde{f}(x)}\right)^{\cdot} = \tilde{f}(x)^2K^2\rho^{2K-2}. \tag{102}$$

Changing the variable to $R = e^{\rho_k \tilde{f}(x)}$, we get finally

$$\dot{R} = \tilde{f}(x)^{\frac{2}{K}} K^2 \left(\log R\right)^{2-\frac{2}{K}} . \tag{103}$$

We can now readily check that for $K = 1$ we get $R \sim t$, so $\rho \sim \log t$. It is also immediately clear that for $K > 1$ the product of weights diverges faster than logarithmically. In the case of $K = 2$ we get $R(t) = \mathrm{li}^{-1}(\tilde{f}(x)K^2 t + C)$, where $\mathrm{li}(z) = \int_0^z dt/\log t$ is the logarithmic integral function. We show a comparison of the 1-layer and 2-layer behavior in the left graph in Figure 11. For larger K we get faster divergence, with the limit $K \to \infty$ given by $R(t) = \mathcal{L}^{-1}(\alpha_\infty t + C)$, where $\alpha_\infty = \lim_{K\to\infty} \tilde{f}(x)^{\frac{2}{K}} K^2$ and $\mathcal{L}(z) = \mathrm{li}(z) - \frac{z}{\log z}$.

Interestingly, while the product of weights scales faster than logarithmically, the weights at each layer diverge slower than in the linear network case, as can be seen in the right graph in Figure 11.

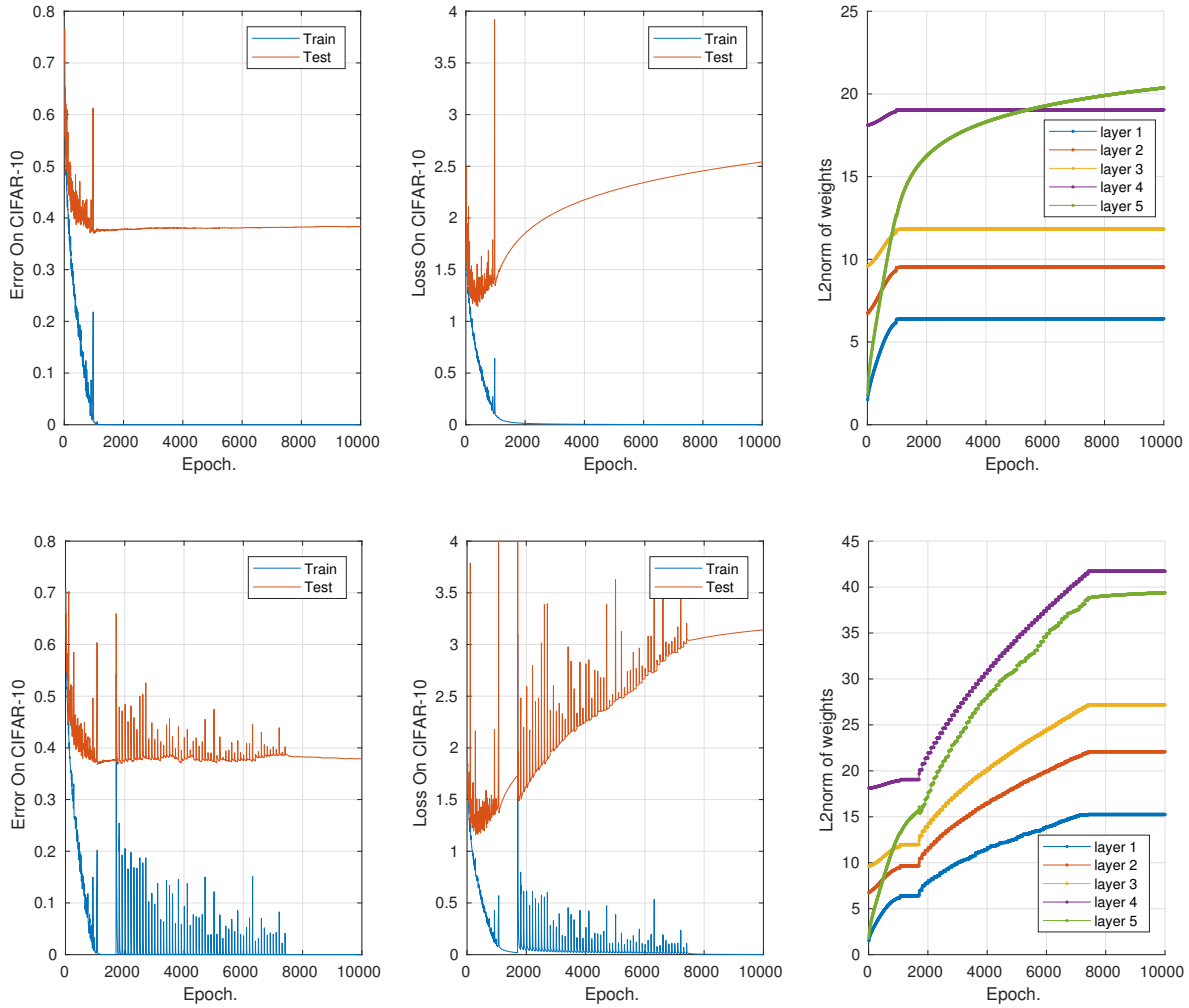


Figure 8: *We train a 5-layer convolutional neural networks on CIFAR-10 with Gradient Descent (GD) on cross-entropy loss with and without perturbations. The main results are shown in the 3 subfigures in the bottom row. Initially, the network was trained with GD as normal. After it reaches 0 training classification error (after roughly 1800 epochs of GD), a perturbation is applied to the weights of every layer of the network. This perturbation is a Gaussian noise with standard deviation being* $\frac{1}{4}$ *of that of the weights of the corresponding layer. From this point, random Gaussian noises with such standard deviations are added to every layer after every 100 training epochs. The empirical risk goes back to the original level after the perturbation, but the expected risk grows increasingly higher. As expected, the* L_2*-norm of the weights increases after each perturbation step. After 7500 epochs the perturbation is stopped. The left column shows the classification error. The middle column shows the cross-entropy risk on CIFAR during perturbations. The right column is the corresponding L2 norm of the weights. The 3 subfigures in the top row shows a control experiment where no perturbation is performed at all throughout training, The network has 4 convolutional layers (filter size* 3×3*, stride 2) and a fully-connected layer. The number of feature maps (i.e., channels) in hidden layers are 16, 32, 64 and 128 respectively. Neither data augmentation nor regularization is performed.*

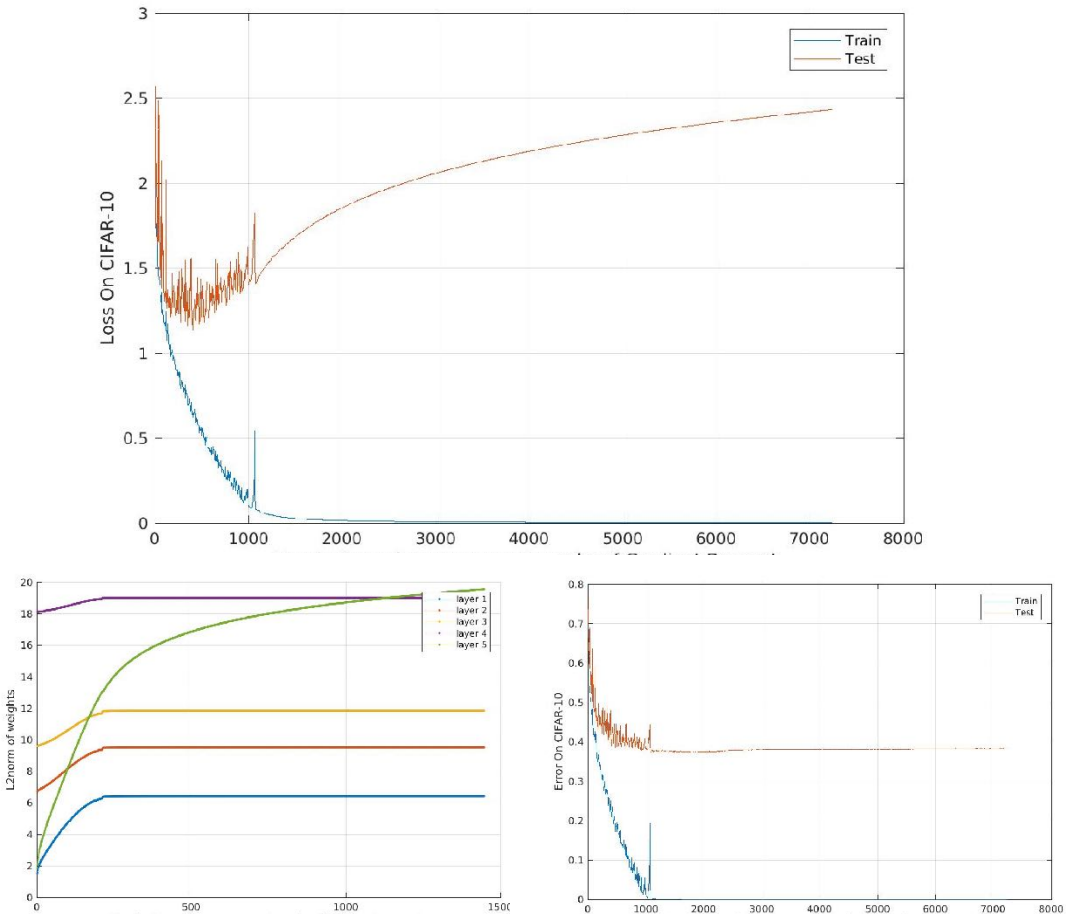


Figure 9: *Same as Figure 4 but without perturbations of weights. Notice that there is some overfitting in terms of the testing loss. Classification however is robust to this overfitting (see text).*

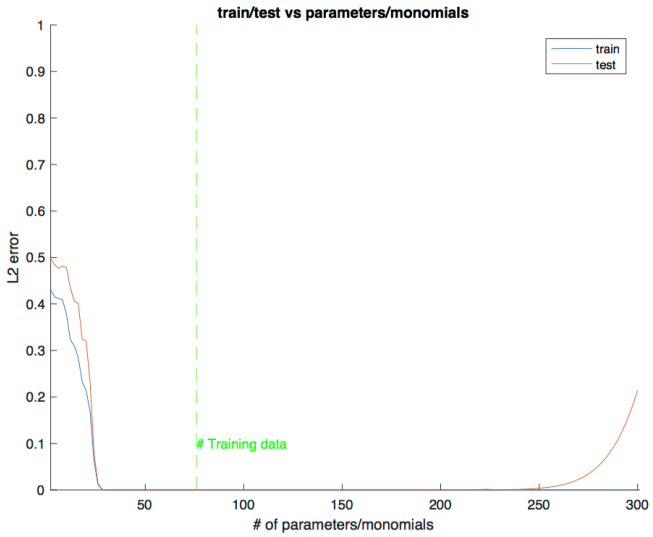


Figure 10: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\phi(X)$*) with a degenerate Hessian of the type of Figure 4. The feature matrix is a polynomial with increasing degree, from 1 to 300. The square loss is plotted vs the number of monomials, that is the number of parameters. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points where* 76 *and the number of test points were* 600*. The solution to the over-parametrized system was the minimum norm solution. More points where sampled at the edges of the interval* $[-1, 1]$ *(i.e. using Chebyshev nodes) to avoid exaggerated numerical errors. The figure shows how eventually the minimum norm solution overfits.*

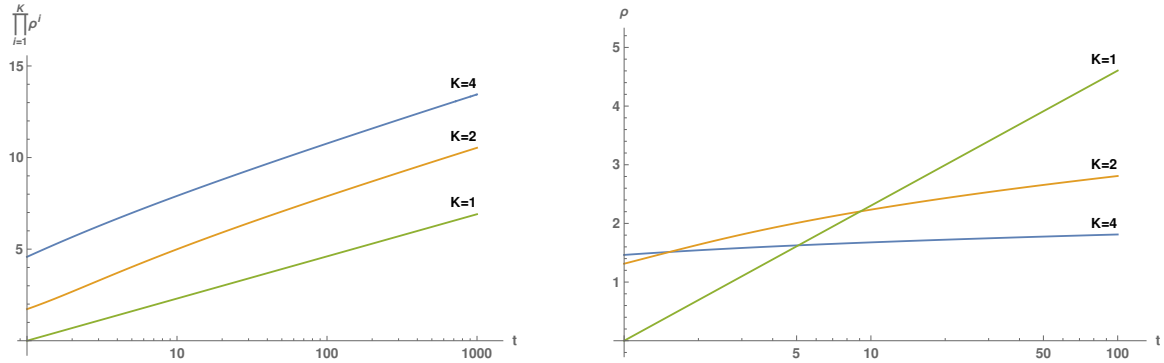


Figure 11: *The left graph shows how the product of weights* $\prod_{i=1}^{K}$ *scales as the number of layers grows when running gradient descent with an exponential loss. In the 1-layer case we have* $\rho = ||w|| \sim \log t$*, whereas for deeper networks the product of norms grows faster than logarithmically. As we increase the number of layers, the individual weights at each layer diverge slower than in the 1-layer case, as seen on the right graph.*