Abstract

Classical generalization bounds for classification suggest maximization of the margin of a deep network under the constraint of unit Frobenius norm of the weight matrix at each layer. We show that this goal can be achieved by gradient algorithms enforcing a unit norm constraint. We describe three algorithms of this kind and their relation with existing weight normalization and batch normalization algorithms, thus explaining their effectiveness. We also show that continuous standard gradient descent with normalization at the end is equivalent to gradient descent with norm constraint. We conjecture that this surprising property corresponds to the elusive implicit regularization of gradient descent in deep networks responsible for generalization despite overparametrization.
Theory III: Dynamics and Generalization in Deep Networks

Andrzej Banburski 1, Qianli Liao1, Brando Miranda1, Tomaso Poggio1, Lorenzo Rosasco1, Bob Liang1, and Jack Hidary2

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

Abstract

Classical generalization bounds for classification suggest maximization of the margin of a deep network under the constraint of unit Frobenius norm of the weight matrix at each layer. We show that this goal can be achieved by gradient algorithms enforcing a unit norm constraint. We describe three algorithms of this kind and their relation with existing weight normalization and batch normalization algorithms, thus explaining their effectiveness. We also show that continuous standard gradient descent with normalization at the end is equivalent to gradient descent with norm constraint. We conjecture that this surprising property corresponds to the elusive implicit regularization of gradient descent in deep networks responsible for generalization despite overparametrization.

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 [1, 2, 3, 4, 5]). This paper addresses the third question.

We start by reviewing 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 [6] illuminate the apparent absence of "overfitting" in the special

*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. The observation on intrinsic normalization of standard gradient descent has been modified.
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 enforcing a weight normalization constraint achieve generalization;
- the fundamental reason for the effectiveness of existing batch normalization techniques is that they approximately implement maximization of the margin under unit norm constraint;
- even without implicit unit norm constraints, generalization can still be obtained for not-too-deep networks because the dynamics of standard gradient descent in the weights is similar to the dynamics with a unit norm constraint. In addition, the balance of the weights across different layers, if present at initialization, is maintained by the gradient flow\textsuperscript{[7]}.

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, in particular, by algorithms such as batch normalization. The control of the norm of the weights is related to Halpern iterations (see\textsuperscript{[14]} for minimum norm solutions which are equivalent to regularization with vanishing $\lambda(t)$.

2 Deep networks: definitions and properties

Definitions We define a deep network with $K$ layers with the usual coordinate-wise scalar activation functions $\sigma(z) : \mathbb{R} \rightarrow \mathbb{R}$ as the set of functions $f(W; x) = \sigma(W^K \sigma(W^{K-1} \cdots \sigma(W^1 x)))$, where the input is $x \in \mathbb{R}^d$, 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 $W^K \in \mathbb{R}^{1, K_l}$. 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 from 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:

$$\mathbb{R}_N(F) = \rho \mathbb{R}_N(\tilde{F}),$$

(1)
where $\rho = \rho_1 \cdots \rho_K$, $\mathbb{F}$ is the class of neural networks described above and accordingly $\tilde{\mathbb{F}}$ is 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

$$\frac{\partial f}{\partial \rho_k} = \frac{\rho}{\rho_k} \tilde{f}. \quad (2)$$

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

$$\sum_{i,j} \frac{\partial f(x)}{\partial W_{i,j}} W_{i,j} = f(x); \quad (3)$$

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

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

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

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

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 $\rho$ 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” $\rho$ 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 $\rho$ corresponds to margin maximization

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

A more formal argument may be developed extending theorems of [9] to the nonlinear case as described in [7].

**Separable data** When $y_n f(x_n) > 0 \forall n = 1, \cdots , N$ we say that the data are separable that is they can all be correctly classified by the network $f$. Notice that this is a strong condition if $f$ is linear but it can be often satisfied by overparametrized deep networks. In the following sections we assume separable data.
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 usual 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 $i = 1, \ldots, N$;
- critical points which correspond to saddles and to local minima for which the loss is not zero but $\nabla W \sum_{i=1}^N L(f(x_i), y_i) = 0$.

Suppose that the polynomial network does not have any of the symmetries characteristics of the RELU network, such as one-homogeneity. 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 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 its zeros 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: $\sum_{i=1}^N (f(x_i) - y_i) \nabla W f(x_i) = 0$. If this were a generic system of polynomial equations, we would expect a set of isolated critical points. A more careful analysis, as suggested by J. Bloom, follows the Preimage theorem and gives the result that while the degeneracy of the global zeros is generically $D - N$, the degeneracy of the critical points is 0. Thus the global minima are more degenerate than the critical points.

**Theorem 1** (The Preimage Theorem [? ]). Let $F : M \mapsto N$ be a smooth map between manifolds, and let $c \in N$ such that at each point $a \in F^{-1}(c)$, the derivative $DF_a$ is surjective. Then $F^{-1}(c)$ is a smooth manifold of dimension $\dim M - \dim N$.

The argument can be extended to the case in which there are degeneracies due to intrinsic symmetries of the network, corresponding to invariances under a continuous group (discrete groups, such as the permutation groups, do not induce infinite degeneracy). Suppose that the effective dimensionality of the continuous symmetries is $S < D$. Assume for simplicity that the symmetries are the same for $f$ and $\nabla_w f$. The critical points will be degenerate on a set of
dimension $S$; the global minima will be degenerate on a set of dimension which is at least $D - N$. Thus if $S \ll N$, which should be the case with the number of data commonly used, the global minima will be degenerate on an algebraic variety of higher dimension than the local minima, that is on a much larger volume in parameter space.

Thus, we have

**Theorem 2** *(informal statement):* For appropriate overparametrization, there are a large number of global zero-error minimizers which are degenerate; the other critical points – saddles and local minima – are generically (that is with probability one) degenerate on a set of lower dimensionality.

The proof is sketched in Appendix 10.

The second part of our argument (in [11]) is that SGD concentrates on the most 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

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

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 3** *(informal statement):* For appropriate overparametrization of the deep network, SGD selects with high probability the global minimizer of the empirical loss, which are highly 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 labels 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 \cite{29, 30, 31}. Closest to our approach – which is focused on the role of batch and weight normalization – is the paper \cite{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 Generalization Bounds and Implications

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 \cite{33}; it is also independently suggested by other techniques such as Lagrange multipliers, normalization and margin maximization theorems \cite{9} that we will discuss later.

Well-known margin bounds for classification suggest an approach that we prove is equivalent (see Appendix 11): 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 $y_f$ 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 \cite{10}. A similar approach to the linear case leads to minimum norm solutions, independently of initial conditions.

5.1 Regression: (local) minimum norm empirical minimizers

We recall the typical form of generalization bounds for regression \cite{34}:

\textbf{Proposition 4} The following generalization bounds apply to $\forall f \in \mathbb{F}$ with probability at least $(1 - \delta)$:

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

(8)

where $L(f) = \mathbb{E}[\ell(f(x), y)]$ is the expected loss, $R_N(\mathbb{F})$ 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 $\rho$. 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 8). We propose a slight modification of the standard gradient descent algorithms to provide a norm-minimizing GD update – NMGD in short – as

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

where $\gamma_n$ is the learning rate and $\lambda_n = \frac{1}{n}$ (this is one of several choices) is the vanishing regularization-like Halpern (see Appendix 14) term.

5.2 Classification: maximizing the margin of the normalized minimizer

A typical margin bound for classification [35] is as follows

**Proposition 5**

$$|L_{\text{binary}}(f) - L_{\text{surr}}(f)| \leq b_1 \frac{\mathbb{R}_N(\mathbb{F})}{\eta} + b_2 \sqrt{\frac{\ln(\frac{4}{\delta})}{2N}},$$

where $\eta$ is the margin, $L_{\text{binary}}(f)$ is the expected classification error, $L_{\text{surr}}(f)$ 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 $\mathbb{R}_N(\mathbb{F}) \sim \rho \mathbb{R}_N(\tilde{\mathbb{F}})$, 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 in Appendix 14 that for separable data, the following property holds true:

**Theorem 6** Maximizing the margin subject to unit norm constraint is equivalent to minimize the norm of $f$ subject to the constraint that the margin is greater than a positive constant.

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.

5.3 Optimization under norm constraint

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

$$\arg \max_{\prod_k ||V_k|| = 1} \min_n y_n \rho \tilde{f}(x_n).$$

7
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 $\frac{\mathbb{R}_N(F)}{\eta}$. In fact, existing generalization bounds such as Equation 6 in [37], see also [4] are given in terms of products of upper bounds on the norm of each layer: the bounds require that each layer is bounded, rather than just the product is bounded.

This constraint is implied by a unit constraint on the norm of each layer, which defines an equivalence class of networks $f$ because of Eq. (1).

A possible approach is to minimize the exponential loss function $L(f(w)) = \sum_{n=1}^{N} e^{-f(W;x_n)y_n} = \sum_{n=1}^{N} e^{-\rho f(V;x_n)y_n}$, over $\rho$ and $V_k$, 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 $||V||^2 = 1$. As we will discuss later they are related to to gradient descent techniques widely used for training deep networks, such as weight normalization (WN) [37] and batch normalization (BN) [38].

6 Gradient techniques for norm control

A review of gradient-based algorithms with unit-norm constraints [36] lists

1. the Lagrange multiplier method
2. the coefficient normalization method
3. the tangent gradient method
4. the true gradient method using natural gradient.

For small values of the step size, the first three techniques are equivalent to each other and are also good approximations of the true gradient method [36]. We assume, as usual, separable data. Stability issues for numerical implementations are discussed in [36]. The four techniques are closely related and have the same goal: performing gradient descent optimization with a unit norm constraint. In the case of GD (a single minibatch, including all data) the four techniques should behave in a very similar way. 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.

6.1 Lagrange multiplier method

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. We define the loss

$$L = \sum_{n=1}^{N} e^{-\rho f((x_n)y_n) + \sum_{k=1}^{K} \lambda_k (||V_k||^2 - 1)$$ (12)
where the Lagrange multipliers $\lambda_k$ 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 $\rho, V_k$. We obtain for $k = 1, \ldots, K$

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

and for each layer $k$

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

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 $\lambda_k$ 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 8, the norm square of each layer grows at the same rate.

**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 $\lambda$ the quantity $2\lambda$. Thus

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

(15)

This 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 $\rho_{\text{max}}$, controlled by a fixed regularization parameter $\lambda_\rho$: in this case the dynamics of $\rho$ converges to a (very large) $\rho_{\text{max}}$ set by a (very small) value of $\lambda_\rho$.

6.2 Coefficient normalization method

If $u(k)$ is unconstrained the gradient maximization of $L(u)$ with respect to $u$ can be performed using the algorithm

$$u(k + 1) = u(k) + g(k) \quad (16)$$

where $g(k) = \mu(k)\nabla_u L$. Such an update, however, does not generally guarantee that $||u^T(k+1)|| = 1$. In the coefficient normalization method, we employ the two step update $\hat{u}(k + 1) = u(k) + g(k)$ and $u(k + 1) = \frac{\hat{u}(k + 1)}{||\hat{u}(k + 1)||}$. 

9
6.3 Tangent gradient method

**Theorem 7** ([36]) Let $||u||$ denote any vector norm that is differentiable with respect to the elements of $w$ and let $g(t)$ be any vector function with finite $L_2$ norm. Then

$$
\dot{u} = h_g(t) = Sg(t) = (I - uu^T/||u||^2)g(t)
$$

with $||u(0)|| = 1$ describes the flow of a vector $u$ that satisfies $||u(t)|| = 1$ for all $t \geq 0$.

In particular, a form for $g$ is $g(t) = \mu(t) \nabla_u L$, the gradient update in a gradient descent algorithm. We call $Sg(t)$ the tangent gradient transformation of $g$. For more details see [36].

6.4 Margin maximization

In our case the methods described above implement margin maximization. Consider for instance the Lagrange multiplier method. 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:

$$
\sum_n e^{-\rho(t)\tilde{f}(x_n)} \approx C \max_n e^{-\rho(t)\tilde{f}(x_n)}
$$

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 $\eta$. 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 $g(t) \approx \rho(t)e^{-\rho(t)\eta} \sum_i H \frac{\partial f(x_n)}{\partial V_k}$.

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 (it is easy to check that the proofs in [9] hold for homogeneous networks and not only for linear ones). In fact, while updating the present version of this paper we noticed that [39] has theorems including such an extension.

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 $\rho$ grows: each set of SVs may correspond to a max margin, minimum norm solution without being the global minimum norm solution. Because of Bezuot-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 8** *(informal statement)*

Under the assumption of separable data and convergence, the gradient methods with norm constraint converge to maximum margin solutions with unit norm.
7 Standard unconstrained dynamics, Weight normalization and Batch normalization

7.1 Standard unconstrained dynamics and its reparametrization

The standard gradient dynamics is given by

\[ \dot{W}^{i,j}_k = -\frac{\partial L}{\partial W^{i,j}_k} = \sum_{n=1}^{N} y_n \frac{\partial f(x_n; w)}{\partial W^{i,j}_k} e^{-y_n f(x_n; W)} \]  

(18)

where \( W_k \) is the weight matrix of layer \( k \). We now consider a reparametrization in terms of \( W^{i,j}_k = \rho_k V^{i,j}_k \). This is equivalent to changing coordinates from \( W_k \) to \( V_k \) and \( \rho = ||W_k|| \). For simplicity of notation we consider here for each weight matrix \( V_k \) 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:

- The norm \( ||\cdot|| \) is assumed to be the \( L_2 \) norm.
- 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. Define \( S = I - \tilde{w}\tilde{w}^T = I - \frac{ww^T}{||w||^2} \). \( S \) has at most one zero eigenvalue since \( \tilde{w}\tilde{w}^T \) is rank 1 with a single eigenvalue \( \lambda = 1 \).
  3. \( \frac{\partial \tilde{w}}{\partial w} = \frac{S}{||w||} \).
  4. \( Sw = S\tilde{w} = 0 \)
  5. \( S^2 = S \)
- We assume \( f(w) = f(||w||, \tilde{w}) = ||w||f(1, \tilde{w}) = ||w||\tilde{f} \).

The gradient descent dynamic system used in training deep networks for the exponential loss of Equation 114 is given by

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

(19)

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

\[ \dot{||w||} = \frac{\partial ||w||}{\partial w} \dot{w} = \tilde{w} \dot{\tilde{w}} \]  

(20)
\[ \dot{\tilde{w}} = \frac{\partial \tilde{w}}{\partial w} = \frac{S}{|w|} \dot{w} \]  

(21)

Thus

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

(22)

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 \( \lim_{t \to \infty} |w| = 0 \). In the 1-layer network case the dynamics yields \( |w| \approx \log t \) asymptotically. For deeper networks, this is different. In Section 13 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.

In summary, the dynamics for \( w \) induces the following dynamics for \( |w| \) and \( \tilde{w} \):

\[ |\dot{w}| = \frac{\partial |w|}{\partial w} \dot{w} = \tilde{w} \dot{w} \]  

(23)

and

\[ \dot{\tilde{w}} = \frac{\partial \tilde{w}}{\partial w} \dot{w} = \frac{S}{|w|} \dot{w} \]  

(24)

The expression for \( \dot{\tilde{w}} \) gives (incorporating the labels \( y_n \))

\[ \dot{\tilde{w}} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \left( \frac{1}{\rho} \frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} - \tilde{w} \tilde{w}^T \frac{\partial \tilde{f}(x_n)}{\partial \tilde{w}} \right). \]  

(25)

An obvious question is whether the normalized reparametrization of gradient descent on the \( w \)s in terms of gradient descent on \( \rho \) and \( \tilde{w} \) provides an equivalent dynamics. The answer is positive: the continuous unconstrained gradient descent dynamics \( \dot{W} \) is equivalent to the dynamics yield by weight normalization in terms of \( \dot{\rho} \) and \( \dot{V} \) with \( W_{k}^{i,j} = \rho_k V_{k}^{i,j} \) and \( V_k \) \( \| \| = 1 \). More precisely the following theorem holds:

**Theorem 9 (informal statement)** The standard continuous dynamical system \( \dot{W}_{k}^{i,j} = -\frac{\partial L}{\partial W_{k}^{i,j}} \)

is equivalent to the dynamics of \( \dot{\rho} \) and \( \dot{V}_{k}^{i,j} \) with \( W_{k}^{i,j} = \rho_k V_{k}^{i,j} \) under the constraint \( \| V^{i,j} \| = 1 \). The latter dynamical system is

\[ \dot{v} = -\nabla_v L = \frac{S}{\rho} \tilde{w} = \sum_{n=1}^{N} e^{-\rho f(x_n; v)} \frac{x_n - vv^T x_n}{\rho} \]  

(26)

and

\[ \dot{\rho} = -\nabla_\rho L = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \tilde{f}(x_n). \]  

(27)
The following identity holds: $W_{i,j}^{i,j} = \dot{\rho}_k V_{i,j}^{i,j} + \dot{\rho}_k V_{i,j}^{i,j}$ in which the weights $V_{i,j}^{i,j}$ are normalized. Thus for both dynamics, normalization of the weights at the end of the iterations via $\frac{W_{i,j}^{i,j}}{\dot{\rho}_k} = V_{k}^{i,j}$ yields the normalized classifier $\tilde{f}$.

7.2 Classical Weight Normalization

Classical weight normalization defines $v$ and $g$ in terms of $w = \frac{g}{||v||}$. The dynamic is given by

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

(28)

and

$$\dot{v} = \frac{\partial L}{\partial v} = \frac{g}{||v||} S \frac{\partial L}{\partial w}$$

(29)

with $S = I - \frac{vv^T}{||v||^2}$.

The dynamics is not equivalent to the standard dynamics on the $w$. In fact, with $w = \frac{g}{||v||} v$ one obtains taking temporal derivatives

$$\dot{w} = \dot{g} \frac{v}{||v||} + g \frac{v}{||v||} \left(I - \frac{vv^T}{||v||^2}\right) \dot{v}$$

(30)

which yields

$$\dot{w} = \frac{vv^T}{||v||^2} \frac{\partial L}{\partial w} + \frac{g^2}{||v||^2} \left(I - \frac{vv^T}{||v||^2}\right)$$

(31)

which is different from $\dot{w} = \frac{\partial L}{\partial w}$.

7.3 Batch Normalization

Batch normalization for unit $i$ normalizes the input to unit $i$ – that is it normalizes $\sum_j W_{i,j} x_j$, where $x_j$ are the activities of the previous layer. Then it sets the activity to be $\gamma(t + 1) \frac{\sum_j W_{i,j} (t+1) x_j}{||W_{i,j}(t+1)||}$, where $\gamma$ is learned subsequently in the optimization. This is equivalent to splitting the weights $w$s into the product of $\rho$ – which can be the same for all units in one layer – and of normalized weights $v$. The normalization step is within gradient descent affecting gradient updates across layers.

In addition, note that batch normalization (see Appendix [18.3]), controls the norms

$$||x||, ||\sigma(W_1 x)||, ||\sigma(W_2 \sigma(W_1 x))||, \cdots$$

(32)

though it does not directly control the norm of each layer – as WN does. In this sense it implements a somewhat weaker version of the generalization bound. It is not too surprising that BN is more effective than WN, because of the explicit normalization carried out at each iteration (and perhaps also because it acts on $||W_1 x||, ||W_2 W_1 x||, \cdots$ instead of $||W_1||, ||W_2||, \cdots$).
Empirically it appears that GD and SGD converge to solutions that can generalize even without BN or other techniques that enforce explicit unit norm constraints. Without BN, convergence is difficult for quite deep networks and generalization is not as good as with BN but it still occurs. How is this possible?

The answer is provided by the following argument. Consider the tangent gradient transformation to a gradient increment \( g(k) = \mu(k) \nabla_u L \) is defined as \( h_g(k) = S g(k) \) with \( S = I - \frac{uu^T}{\|u\|^2} \).

Theorem 7 says that the dynamical system \( \dot{u} = h_g \) with \( \|u(0)\| = 1 \) describes the flow of a vector \( u \) that satisfies \( \|u(t)\| = 1 \) for all \( t \geq 0 \). It is easy to prove that

**Proposition 10** The dynamical system of Equations 26 and 27 is not changed by the tangent gradient transformation. The same is true for the system defined by Equations 28 and 29.

The proof follows easily by using the fact that \( S^2 = S \).

The proposition above suggests that both dynamical systems defined by the Equations 26 and 27 and 28 and 29 already enforce by themselves the norm constraint. In addition, since Theorem 9 shows that the standard dynamical system in the \( w \)s defined by \( \dot{w} = -\frac{\partial L}{\partial w} \) is equivalent to the dynamical system of 26 and 27, we have proven the following

**Theorem 11** The standard dynamical system used in deep learning, defined by \( \dot{w} = -\frac{\partial L}{\partial w} \), is not changed by the tangent gradient transformation, implying that it implicitly enforces a unit norm constraint. Thus if it converges to \( w(T) \), the unit vector \( \frac{w(T)}{\|w(T)\|} \) represents a maximum margin solution under unit norm constraint.

Another interesting property of the dynamics of \( W_k \), which is shared with the dynamics of \( V_k \) under unit norm constraint, is suggested by recent work [7]: the difference between the square of 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, which is part of constraint we enforce in an explicit way with the Lagrange multiplier or the WN technique. This property is expected in a minimum norm situation, which is itself equivalent to maximum margin under unit norm (see Appendix 11). The observation of [7] is easy to prove in our framework. 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

\[
\frac{d}{dt} \rho_k^2 = \frac{d}{dt} \rho_{k-1}^2, \forall k = 2, \cdots, K. \tag{33}
\]

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 \( \eta \) is small at initialization then the norm of the two layers will remain very similar under the gradient flow – a condition required by minimum norm solutions. A formal
proof can be sketched as follows using the standard gradient descent equations. Consider the
gradient descent equations

\[ W_{i,j}^k = \sum_{n=1}^{N} y_n \left[ \frac{\partial f(W; x_n)}{\partial W_{i,j}^k} \right] e^{-y_n f(x_n; W)}. \]  

(34)

The above dynamics induces the following dynamics on \( ||W_k|| \) using the relation \( \dot{||W_k||} = \frac{\partial ||W_k||}{\partial W_k} \frac{\partial W_k}{\partial t} = \frac{W_k}{||W_k||} \dot{W}_k \). Thus

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

(35)

because of lemma 3. It follows that

\[ ||\dot{W}_k||^2 = 2 \sum_{n=1}^{N} y_n f(W; x_n) e^{-y_n f(x_n; W)}, \]  

(36)

that implies that the rate of growth of \( ||W_k||^2 \) is independent of \( k \). If we assume that \( ||W_1|| = ||W_2|| = \cdots = ||W_K|| = \rho(t) \) initially, they will remain equal while growing throughout training. The norms of the layers are balanced, thus avoiding the situation in which one layer may contribute to decreasing loss by improving \( \tilde{f} \) but another may achieve the same result by simply increasing its norm. Following the discussion in section 5.2, generalization depends on bounding the ratio of Rademacher complexity to the margin \( R_N(F) \min \rho_K \tilde{f}(x) \). The balance of norms property allows us to cancel the dependence on \( \rho \) for all layers.

Theorem 11 for exponential-type losses is the non-linear multi-layer extension of Srebro result for linear networks and is also related to a well-known GD property for the linear case: GD starting from zero or from very small weights converges to the minimum \( L_2 \) norm. It is important to emphasize that in the multilayer, nonlinear case we expect several maximum margin solutions (unlike the linear case), depending on initial conditions and stochasticity of SGD.

Of course, other effects, in addition to the role of initialization and batch or weight normalization may be at work in practice, 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, commonly used weight decay with appropriate parameters can induce generalization. Typical implementations of data augmentation also obviate to the overparametrization problem: at each iteration of SGD only “new” data are used and depending on the number of iterations it is possible to obtain more training data than parameters. In any case, within this online framework, one expects convergence to the minimum of the expected risk (see Appendix 12) 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 initial conditions \[6\]. What matters is \(\frac{w}{||w||} = \frac{\tilde{w}}{\rho} = \tilde{w}\).

• The property \(\frac{d}{dt}\rho_k^2 = \frac{d}{dt}\rho_{k-1}^2\), \(\forall k = 2, \cdots, K\) also holds for the square loss.

• For exponential type losses and one-homogeneous networks in the case of classification the situation is similar since \(f\rho = \tilde{f}\rho = \tilde{f}\). With zero-norm initial conditions the norms of the \(K\) layers are approximately equal and \(\rho = \rho^K\). Notice that the degeneracy of the solutions of the gradient flow is reduced by the unit norm constraints on each of the layers. As we showed, the norm square of each layer grows – under unregularized gradient descent – at the same time-dependent rate. This means that the time derivative of the product of the norms squared does change with time. Thus a bounded product does not remain bounded even when divided by a common time-dependent scale, unless the norm of all layers are equal at initialization.

8.1 Dynamics of Gradient Descent

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

\[ V_k = \frac{g(t)}{2\lambda}, \quad (37) \]

where \(g(t) = \rho(t) \sum_n e^{-\rho(t)f(x_n)} \frac{\partial f(x_n)}{\partial V_k} \) 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 15)

\[ \sum_n \left[ - \left( \prod_{i=1}^{K} \rho_i^2 \right) \frac{\partial f(V; x_n)}{\partial V_k} \frac{\partial f(V; x_n)}{\partial V_k^T} + \left( \prod_{i=1}^{K} \rho_i \right) \frac{\partial^2 f(V; x_n)}{\partial V_k \partial V_{k'}} \right] e^{-\sum_{i=1}^{K} \rho_i f(V; x_n)} - 2\lambda(t)I. \quad (38) \]

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 \[16\]. 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.

9 Discussion

Our main results are

- multilayer, nonlinear, deep networks trained with gradient descent methods with norm constraint (GDNC) converge to maximum margin solutions and therefore generalize because of the classical classification bounds;

- popular gradient descent techniques such as weight normalization and batch normalization also generalize because they belong to the class of GDNC methods;

- surprisingly, there is an implicit norm constraint also in standard gradient descent for deep networks with RELUs. Therefore standard gradient descent on the weights, when it converges, provides a solution \( \tilde{f} \) that generalizes without the need of explicit regularization or constraints.

This is similar to the situation for linear networks\[1\] and in fact can be considered an extension of the result in \[6\].

It is useful to emphasize that despite the similarities between the various GDNC methods they correspond to different dynamical flows. In particular, tangent gradient and weight normalization are not the same and in turn they are different from standard gradient descent. Furthermore our analysis has been restricted to the continous case; the discrete case is expected to yield even greater differences.

The fact that the solution corresponds to a maximum margin solution under a fixed norm constraint explains the puzzling behavior of Figure \[4\] in which batch normalization was used. 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 weight normalization and especially batch normalization is based on fundamental reasons. In the case of batch normalization, the reason is deeper 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 the 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, in the same way that the Lagrange multiplier technique is related to the tangent gradient method.

\[1\]The prototypical (linear) example for over-parametrized deep networks is convergence of gradient descent to weights that represent the pseudoinverse of the input matrix.
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 2), as we suspect? Or is the landscape “nicer” for large overparametrization – as hinted in several 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 the coefficient normalization method and for tangent gradient – that ensure convergence to a maximum margin solution independently of initial conditions?

Acknowledgments

We thank Yuan Yao, Misha Belkin, Jason Lee 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


Appendices

10 Degeneracies

The proof goes as follows. Consider the \( N \) maps \( F_i : \mathbb{R}^D \rightarrow \mathbb{R} \), that is \( F_i : (w_{1,1}, \cdots, w_{n,n}) \rightarrow P_i \), where \( P_i \) is a polynomial in the \( D \) \( w \)s with coefficients provided by the training example \( x_i \) and \( P_i = y_i \). As an example assume that \( P_i = (w_1x_1 + w_2x_2)^2 \). Then for, say, \( x_1 = 1, x_2 = 2 \), \( P_i = (w_1 + 2w_2)^2 \). Assume \( y_i \neq 0 \). Consider the set \( H_{y_i} = F_i^{-1}(y_i) \). We need to check that \( DF_v \) is surjective for any \( v = (v_1, v_2) \). \( DF_v \) is a linear transformation from \( \mathbb{R}^2 \) to \( \mathbb{R} \) given by \( DF_v(w_1, w_2) = (2v_1 + 4v_2, 4v_1 + 8v_2) \circ (w_1, w_2) = (2v_1 + 4v_2)w_1 + (4v_1 + 8v_2)w_2 \). It is enough to find a single vector \( w_1, w_2 \) such that \( DF_v \neq 0 \). For instance choose \( (w_1, w_2) = (v_1, v_2) \). Then \( DF_v(v_1, v_2) = 2v_1^2 + 8v_2v_1 + 8v_2^2 = 2y_i \neq 0 \). Therefore \( DF_v \) is surjective \( \forall v \in F_i^{-1}(y_i) \) and the preimage of \( y_i \) is a manifold of dimensionality \( D - 1 \).

11 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 \( \pm 1 \), the functional margin is

\[
\min_{i=1, \ldots, N} y_i f(W; x_i).
\]  

(39)

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

\[
\max_W \min_{i=1, \ldots, N} y_i f(W; x_i), \quad \text{subj. to } \|W\| = 1.
\]  

(40)

The latter constraint is needed to avoid trivial solutions in light of the one-homogeneity property. We next show that Problem (40) is equivalent to

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

(41)

To see this, we introduce a number of equivalent formulations. First, notice that functional margin (39) can be equivalently written as

\[
\max_{\gamma > 0} \gamma, \quad \text{subj. to } y_i f(W; x_i) \geq \gamma, \quad i = 1, \ldots, N.
\]
Then, the max margin problem (40) can be written as
\[ \max_{W, \gamma > 0} \gamma, \quad \text{subj. to} \quad \|W\| = 1, \quad y_i f(W; x_i) \geq \gamma, \quad i = 1, \ldots, N. \] (42)

Next, we can incorporate the norm constraint noting that using one-homogeneity,
\[ y_i f(W; x_i) \geq \gamma \iff y_i f\left(\frac{W}{\|W\|}; x_i\right) \geq \gamma' \iff y_i f(W; x_i) \geq \|W\| \gamma = \gamma' \]
so that Problem (42) becomes
\[ \max_{W, \gamma' > 0} \gamma' \|W\|, \quad \text{subj. to} \quad y_i f(W; x_i) \geq \gamma', \quad i = 1, \ldots, N. \] (43)

Finally, using again one-homogeneity, without loss of generality, we can set \( \gamma' = 1 \) and obtain the equivalent problem
\[ \max_W \frac{1}{\|W\|}, \quad \text{subj. to} \quad y_i f(W; x_i) \geq 1, \quad i = 1, \ldots, N. \] (44)

The result is then clear noting that
\[ \max_W \frac{1}{\|W\|} \iff \min_W \|W\| \iff \min_W \frac{\|W\|^2}{2}. \]

12 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 \( S_\infty \). 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.
13 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

$$\rho_k \dot{\rho}_k = \tilde{f}(x) \left( \prod_{i=1}^k \rho_i \right) e^{-\prod_{i=1}^K \rho_i \tilde{f}(x)}$$  \hspace{1cm} (45)

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

$$\gamma = \tilde{f}(x)^2 r^2, \hspace{1cm} r \dot{r} = 2 \frac{\log \gamma}{\gamma},$$  \hspace{1cm} (46)

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 turns out to be true in general, as discussed elsewhere. It gives us the single differential equation

$$\dot{\rho} = \tilde{f}(x) K \rho^{K-1} e^{-\rho_k \tilde{f}(x)}.$$  \hspace{1cm} (47)

This implies that for the exponentiated product of weights we have

$$\left( e^{\rho_k \tilde{f}(x)} \right)' = \tilde{f}(x)^2 K^2 \rho^{2K-2}.$$  \hspace{1cm} (48)

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

$$\dot{R} = \tilde{f}(x)^2 K^2 (\log R)^{2 - \frac{2}{K}}.$$  \hspace{1cm} (49)

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 $R(t) = \text{li}^{-1}(\tilde{f}(x) K^2 t + C)$, where $\text{li}(z) = \int_0^z dt / \log t$ is the logarithmic integral function. We show a comparison of the 1-layer and 2-layer behavior in the left graph in Figure 1. 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 $\alpha_\infty = \lim_{K \to \infty} \tilde{f}(x)^2 K^2$ and $\mathcal{L}(z) = \text{li}(z) - \frac{z}{\log z}$.

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 1.
Figure 1: 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.

14 Halpern iterations: selecting minimum norm solution among degenerate minima

In this section we summarize a modification of gradient descent that can be applied – in a similar way as with Lagrange multipliers – to gradient descent optimization under the square and exponential loss for one-layer and nonlinear, deep networks.

We are interested in the convergence of solutions of such 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 $||Tx - Ty|| \leq ||x - y||, \forall x, y \in X$. Then the following result is classical (45, 33):

**Theorem 12** (45) Let $X$ be a strictly convex normed space. The set of fixed points of a non-expansive mapping $T : C \rightarrow C$ 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 17. 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

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

(50)

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

In particular, the following holds

**Theorem 13** [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 $w_{n+1} = k_{n+1}T(w_n$ with $k_n = 1 - \frac{1}{n^2}$ 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

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

(51)

where $\gamma_n$ is the learning rate and $\lambda_n = \frac{1}{n^2}$ (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.

**Remark**

For classification with exponential-type losses the Lagrange multiplier technique, Batch normalization and the Tangent gradient method try to achieve approximately the same result – maximize the margin while constraining the norm – assuming as usual separable data. 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 14 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 2). The minimum norm solution in the classification case corresponds to a maximum margin solution.

Notice that these iterative procedures are often part of the numerical implementation (see [38] 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.
Figure 2: *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.

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.

15 Dynamics of Lagrange multiplier approach

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

$$V_k = \frac{g(t)}{2\lambda}.$$  \hspace{1cm} (52)

We call the constraints Equations 52 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 $O = \frac{1}{q}(1, \ldots, 1)$ summing the outputs of $W_2$ into a single scalar. Let us consider normalized weights. Then

$$\frac{\partial f(x)}{\partial V_k} = O^T V_2^T x^T.$$  

In detail

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

(53)

gives

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

(54)

which via previous Equations yields $V_1^{ij,k} = \alpha_1 \frac{1}{q} \sum_k O^i V_2^{ij,k}$.

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 $\frac{\partial f(x)}{\partial V_2} = O^T x^T V_1^T$ which substituted in $f$ gives $f(x) = O^T x^T V_1^T V_1 x$. Using indeces, we have $f(x) = \alpha_1 \sum_{i,j,k,l} O_i V_2^{ij,k} O^k V_2^{j,l} x^l x^l$.

**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

$$f(x) = OD_2 W_2 D_1 W_1 x$$

(55)

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 55 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

$$\sum_n \left[ -\left(\prod_{i=1}^{K} \rho_i^2 \frac{\partial \tilde{f}(V; x_n)}{\partial \bar{V}_k} \frac{\partial \tilde{f}(V; x_n)}{\partial \bar{V}_k}^T \right) + \left(\prod_{i=1}^{K} \rho_i \frac{\partial^2 \tilde{f}(V; x_n)}{\partial \bar{V}_k \partial \bar{V}_k} \right) e^{-\prod_{i=1}^{K} \rho_i \tilde{f}(V; x_n)} - 2\lambda I \right]$$

(56)

16 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 $\lambda w^2$ 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 \cite{29} 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 apart in the limit of \( \lambda \to 0 \). In fact, the result below shows us it is not true.

**Lemma 14** 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 \( \lambda \). Consider the simplest case of a 2-layer network with 4 weights \( w_1, \ldots, w_4 \) and one training example \( x = (1,1) \). The loss is then

\[
L(w) = e^{-w_1w_2 - w_3w_4} + \lambda(w_1^2 + w_2^2 + w_3^2 + w_4^2) \tag{57}
\]

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

\[
w_1^* = \frac{e^{-f_*}}{2\lambda} w_{2*}, \quad w_2^* = \frac{e^{-f_*}}{2\lambda} w_{1*}, \tag{58}
\]

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

\[
det H = e^{-4f_*} \left( 4\lambda^2 e^{-2f_*} - 1 \right) \left( 4\lambda^2 e^{-2f_*} + 2\lambda e^{f_*} (w_1^2 + w_2^2 + w_3^2 + w_4^2) + 2f_* - 1 \right). \tag{59}
\]

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 \( \lambda \), 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.\footnote{This construction actually stems from reparametrizing a Gaussian in a quadratic potential well.}

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 \cite{50}, 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 \( \text{GL}_{d_{L-1}}(\mathbb{R}) \times \text{GL}_{d_{L-2}}(\mathbb{R}) \times \cdots \times \text{GL}_{d_k}(\mathbb{R}) \), 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 $O_{d_{L-1}}(\mathbb{R}) \times O_{d_{L-2}}(\mathbb{R}) \times \cdots \times O_{d_1}(\mathbb{R})$. 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.

\textbf{17} \quad T = I - \nabla L(f) \textit{ is a non-expanding operator}

\textbf{Definition 15} \quad A function $g(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is L-smooth if its gradients are Lipschitz continuous, that is

$$||\nabla g(x) - \nabla g(y)|| \leq L||x - y||$$ \hspace{1cm} (60)

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

\textbf{Lemma 16} \quad A sufficient condition for L-smoothness is that the eigenvalues of $H = \nabla^2 g$ are bounded from above and from below by $L$.

\textbf{Proof} \quad The 2-norm of the Hessian $||H||_2$ is equal to the absolute value of its highest eigenvalue $\lambda_{\text{max}} \leq L$. Let us consider the function $v^T \nabla g(z_\beta y)$, 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.

\textbf{Lemma 17} \quad $\nabla g(x) = 0$ if and only if $x \in \mathbb{R}^n$ is a fixed point of the operator $T^\gamma : \mathbb{R}^n \to \mathbb{R}^n$ with $T^\gamma x = x - \gamma \nabla g(x)$, that is $T^\gamma x = x$ for non-zero $\gamma$.

\textbf{Theorem 18} \quad For convex $g(x)$, the operator $T^\gamma$ defined as $T^\gamma x = x - \gamma \nabla g(x)$ is non-expanding that is

$$||T^\gamma x - T^\gamma y|| \leq ||x - y||$$ \hspace{1cm} (61)
Proof The standard proof uses the mean value theorem to write
\[ T^\gamma x - T^\gamma y = (x - y)(I - \gamma \nabla^2 g(z)) \] (62)
with \( z = \beta x + (1 - \beta)y \) for a certain \( \beta \in [0, 1] \). Then submultiplicativity of norms yields
\[ ||T^\gamma x - T^\gamma y|| \leq ||x - y|| \|(I - \gamma \nabla^2 g(z))||. \] (63)

The last term on the right is the norm of the matrix \( I - H \) where \( H \) is the Hessian we consider for various versions 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 \( T \) 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 19** Assume that gradient descent starts from \( w_0 \) with \( g \) which is \( L \)-smooth
\[ w_{t+1} = w_t - \gamma \nabla g(w_t) \] (64)
and converges to a minimum \( w_* \). If \( \gamma L < 1 \), then
(a) \[ ||w_{t+1} - w_*||^2 \leq (1 - \gamma L)^2 ||w_0 - w_*||^2 \] (65)
(b) Additionally, if \( g \) is \( \mu \)-strongly convex, then this can be strengthened to
\[ ||w_{t+1} - w_*||^2 \leq (1 - \gamma \mu)^{t+1} ||w_0 - w_*||^2 \] (66)

Proof (a) \( L \)-smoothness gives us that
\[ L||w_t - w_*||^2 \geq -(\nabla g(w_t) - \nabla g(w_*), w_t - w_*) \geq -L||w_t - w_*||^2. \] (67)
We then have
\[ ||w_{t+1} - w_*||^2 = ||w_t - w_* - \gamma \nabla g(w_t)||^2 \]
\[ = ||w_t - w_*||^2 - 2\gamma (\nabla g(w_t), w_t - w_*) + \gamma^2 ||\nabla g(w_t)||^2 \]
\[ = ||w_t - w_*||^2 - 2\gamma (\nabla g(w_t) - \nabla g(w_*), w_t - w_*) + \gamma^2 ||\nabla g(w_t) - \nabla g(w_*)||^2 \]
\[ \leq (1 - 2\gamma L)||w_t - w_*||^2 + \gamma^2 ||\nabla g(w_t) - \nabla g(w_*)||^2 \]
\[ \leq (1 - 2\gamma L + \gamma^2 L^2)||w_t - w_*||^2 \]
where we have used the fact that \( \nabla g(w_*) = 0 \) in third equality, the result (67) in the first inequality and finally the definition of \( L \)-smoothness.
(b) $\mu$-strong convexity gives us

$$\langle \nabla g(w_t), w_s - w_t \rangle \leq g(w_s) - g(w_t) - \frac{\mu}{2}||w_t - w_s||^2$$

It follows similarly to the previous case that

$$||w_{t+1} - w_s||^2 = ||w_t - w_s - \gamma \nabla g(w_t)||^2$$

$$= ||w_t - w_s||^2 - 2\gamma \langle \nabla g(w_t), w_t - w_s \rangle + \gamma^2 ||\nabla g(w_t)||^2$$

$$\leq (1 - \gamma \mu) ||w_t - w_s||^2 - 2\gamma (g(w_t) - g(w_s)) + \gamma^2 ||\nabla g(w_t)||^2$$

$$\leq (1 - \gamma \mu) ||w_t - w_s||^2 - 2\gamma (1 - \gamma L)(g(w_t) - g(w_s))$$

$$\leq (1 - \gamma \mu) ||w_t - w_s||^2$$

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

18 Reparameterization with normalized weights

- We introduce a reparametrization for GD under the exponential loss which yields a continuous dynamics equivalent to the standard $W$ dynamics. The reparametrization is similar to the “classical” weight normalization; the latter however is not equivalent to the standard dynamics.

- Our formulation converges 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 $\tilde{w}$ (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. $\frac{\partial \tilde{w}}{\partial w} = \frac{I - \tilde{w}\tilde{w}^T}{||\tilde{w}||} = S$. $S$ has at most one zero eigenvalue since $\tilde{w}\tilde{w}^T$ is rank 1 with a single eigenvalue $\lambda_1 = 1$. Notice that this definition of $S$ is slightly different from the definition in the main text.

  3. $Sw = S\tilde{w} = 0$
4. \(||w||S^2 = S\)
5. \(\frac{\partial ||w||^2}{\partial w} = 0\)

- We assume \(f(w) = f(||w||, \bar{w}) = ||w||f(1, \bar{w}) = ||w||\tilde{f}\).
- Thus \(\frac{\partial f}{\partial w} = \bar{w} \tilde{f} + |w|S\frac{\partial \tilde{f}}{\partial w}\)

The gradient descent dynamic system used in training deep networks for the exponential loss of Equation 114 is given by

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

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

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

The dynamics above for \(w\) induces the following dynamics for \(||w||\) and \(\bar{w}\):

\[
||\dot{w}|| = \frac{\partial ||w||}{\partial w} \dot{w} = \bar{w} \tilde{w}
\]

and

\[
\dot{\bar{w}} = \frac{\partial \bar{w}}{\partial w} \dot{w} = S \dot{w}
\]

Thus

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

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 13 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.

In summary, the dynamics for \(w\) induces the following dynamics for \(||w||\) and \(\bar{w}\):

\[
||\dot{w}|| = \frac{\partial ||w||}{\partial w} \dot{w} = \bar{w} \tilde{w}
\]

and

\[
\dot{\bar{w}} = \frac{\partial \bar{w}}{\partial w} \dot{w} = S \dot{w}
\]
The expression for $\dot{\tilde{w}}$ gives (incorporating the labels $y_n$)

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

(75)

A version of the algorithm that can be used with NM iterations can also have a regularization term in the $\rho$ dynamics with a very small $\lambda_\rho$ of the form

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

(76)

to constrain $\rho$ to be large but finite.

For large $\rho$ 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

- $\frac{W^{i,j}}{||W_k||} = \tilde{W}^{i,j}$; thus $||W_k|| = 1$.
- $\frac{\partial ||W_k||}{\partial \tilde{W}^{i,j}} = \tilde{W}^{i,j}$
- $S_{k,k'}^{i,j,a,b} = \frac{\partial \tilde{W}^{i,j}}{\partial \tilde{W}^{a,b}} = \delta_{k,k'} \delta_{i,a} \delta_{j,b} - \tilde{W}^{i,j} \tilde{W}^{a,b} / ||W_k||$.

Thus the generalization of Equation 25 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:

$$S_{k,k'}^{i,j,a,b} \frac{\partial f}{\partial W^{a,b}_k} = \frac{\rho}{\rho_k} S_{k,k'}^{i,j,a,b} \frac{\partial \tilde{f}}{\partial \tilde{W}^{a,b}_k}$$

(77)

18.1 Equivalence of the $W$ dynamics under standard gradient descent and of the $\tilde{W}$ dynamics

The dynamics of gradient descent with weight normalization is the same as that of standard gradient descent. We define $w = \rho v$ and assuming $||v|| = 1$ we obtain as described earlier

$$\dot{v} = S \tilde{w} = \sum_{n=1}^{N} e^{-\rho v^T x_n} x_n - v v^T x_n / \rho$$

(78)

and

$$\dot{\rho} = \sum_{n=1}^{N} e^{-\rho v^T x_n} v^T x_n.$$ 

(79)
Let us now compute the dynamics of the weights \( w \) from \( \dot{w} = \dot{\rho} v + \rho \dot{v} \). We obtain
\[
\dot{\rho} v + \rho \dot{v} = \sum_{n=1}^{N} e^{-\rho v^T x_n} (v^T x_n v + x_n - v v^T x_n) = \sum_{n=1}^{N} e^{-\rho v^T x_n} x_n. \tag{80}
\]

It turns out that this is exactly the dynamic of standard gradient descent without normalization since \( \dot{w} = \sum_{n=1}^{N} e^{-w^T x_n} x_n \).

The same argument holds for multilayer deep nets in which case
\[
\dot{v} = S \dot{w} = \sum_{n=1}^{N} e^{-\rho f(x_n;v)} \frac{x_n - v v^T x_n}{\rho} \tag{81}
\]
and
\[
\dot{\rho} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \tilde{f}(x_n) \tag{82}
\]

Additional proof Another proof of the same claim is the following. The goal is to check that \( \frac{w(T)}{\rho(T)} \) with \( w(T) = \int_0^T \dot{w}(t) \) is the same as \( v(T) \), where \( \dot{w} = \sum_{n=1}^{N} e^{-w^T x_n} x_n, \rho = ||w|| \) and \( \dot{v} = \sum_{n=1}^{N} e^{-\rho v^T x_n} (x_n - v v^T x_n) \frac{1}{\rho} \). We use integration by parts to compute \( \int_0^T \frac{d}{dt} v \dot{w} dt \). Integrating by parts we obtain \( Sw(0) - \int_0^T \frac{dS}{dt} w dt \) yielding (since \( Sw = 0 \), see Appendix 18)
\[
\int_0^T v \frac{\partial v v^T}{\partial t} = \int_0^T (\dot{v} - v v^T \dot{v}). \tag{83}
\]

As mentioned earlier about Equation (25) \( \dot{v} \) is orthogonal to \( v \). Thus Equation 83 gives
\[
\int_0^T \dot{v} = v(T), \tag{84}
\]
as claimed.

18.2 “Classical” weight normalization is not equivalent to the standard weight dynamics

Standard or classical weight normalization [37] was claimed to be a reparametrization of the weight vectors in a neural network as follows
\[
w = \frac{g}{||v||} v. \tag{85}
\]

There is a correspondence between our normalized reparametrization of section 18 and classical weight normalization [37] as follow: \( \tilde{w} = \frac{w}{||w||} \) and \( \rho = ||w|| g \). However the gradient descent equations are quite different in the two cases.

For the \( \rho, \tilde{w} \) parametrization, the dynamics is given by
\[ \frac{\partial L}{\partial \rho} = \mathbf{w}^T \frac{\partial L}{\partial \mathbf{w}} \quad (86) \]

and

\[ \frac{\partial L}{\partial \mathbf{w}} = S \frac{\partial L}{\partial \mathbf{w}} \quad (87) \]

with

\[ S = \frac{I - \mathbf{\hat{w}} \mathbf{\hat{w}}^T}{\rho} \quad (88) \]

The dynamics above in \( \rho \) and \( \mathbf{\hat{w}} \) is equivalent to the dynamics of \( \mathbf{w} \) using \( \dot{\mathbf{w}} = \rho \dot{\mathbf{\hat{w}}} + \dot{\rho} \mathbf{\hat{w}} \) – which follows from the definition \( \mathbf{w} = \rho \mathbf{\hat{w}} \).

The gradient descent equations for the \( g, v \) parametrization are

\[ \frac{\partial L}{\partial g} = \frac{v}{||v||} \frac{\partial L}{\partial \mathbf{w}} \quad (89) \]

and

\[ \frac{\partial L}{\partial v} = \frac{g}{||v||} S' \frac{\partial L}{\partial \mathbf{w}} \quad (90) \]

with

\[ S' = I - \frac{vv^T}{||v||^2} = \frac{v^Tv - vv^T}{v^Tv} \quad (91) \]

It is easy to show that the implied dynamics for \( \mathbf{w} \) is different from the standard dynamics.

### 18.3 “Classical” 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 \( x_{new} = (x - E[x])(Cov(x)^{-1/2}) \).

Consider the reparametrization defined in WN. As discussed in the previous section, the weights \( \mathbf{w} \) are replaced by \( \mathbf{w} = g \mathbf{\|v\|} \). 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 \( i = \text{ReLU}(Wx + b) \), then the activations per channel \( c \) are transformed at each iteration into

\[ o_{c}^{BN} = \gamma_c \frac{i_c - \mathbb{E}_B[i_c]}{\sqrt{\text{Var}_B[i_c]}} + \beta_c, \quad (92) \]
where $B$ denotes the batch of data over which the expectations are evaluated. In the linear case, this replaces $Wx$ with $\gamma \frac{Wx}{\sigma} + \beta$, effectively normalizing the weight matrix at each iteration (that is dividing by the norm $\rho$ and treating $\gamma$ as the new $\rho$). Batch normalization is thus equivalent to the “coefficient normalization method” of section 6.

Remarks

- Consider WN and BN in terms of coordinate transformations. Then, neglecting biases for simplicity of notation, WN uses a transformation $A$, defined as $w_{\text{new}} = A$ such that $||w_{\text{new}}|| = 1$. On the other hand the ideal BN (as described in the original paper) uses a transformation $C$, defined as $w_{\text{new}}x = Cwx$, such that $w_{\text{new}}w_{\text{new}}^T = \frac{1}{D} I$, where $D$ is the dimensionality of $w_{\text{new}}$. Thus the trace of the matrix $w_{\text{new}}w_{\text{new}}^T$ is one implying that $||w_{\text{new}}|| = 1$. In practice, the current implementation of BN only enforces that the diagonal of $w_{\text{new}}w_{\text{new}}^T$ should be equal to $\frac{1}{D} I$. This enforces “unit” norm as a byproduct of a stronger constraint.

- Since BN normalizes the activation at each stage of the network, the norm used to normalize the last layer is the norm of the composition of all the layers – which is usually smaller than the product of the norms.

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

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

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

19 Network dynamics 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 $\lambda$ 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 [6] 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 [19.2.1]. 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 $\lambda$). NMGD-type iterations lead to convergence not only to the fixed points – as vanilla GD does – but to the (locally) minimum norm fixed point.

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

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

which depends on the margin $\eta$. $L_{\text{binary}}(f)$ is the expected classification error; $L_{\text{surr}}(f)$ 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 $\mathbb{R}_N(F) \sim \rho$, the margin bound is optimized by effectively maximizing $\tilde{f}$ on the “support vectors”. As shown in Appendix 11 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 $\lambda$ 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 $\lambda$ 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.

### 19.1 One-layer, 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 $\lambda$ 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 $w_k^{i,j} = w_1^{i,j}$ because there is only one layer. Thus $f(W; x) = w^T x$ with $w_1^{i,j} = w^T$.

Consider

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

(95)

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 $y_n = w^T x_n \ \forall n = 1, \cdots, N$.

1. **Dynamics** The dynamics is

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

(96)

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

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

(97)

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

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

(98)

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 $d > n$ [51].

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

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

(99)

gives for $\dot{w} = 0$

40
\[ w_0 = \frac{2}{\lambda} \sum_{n=1}^{N} E_n x_n^T \]  

(100)

The Hessian at \( w_0 \) is with

\[ H(w) = -\sum_{n=1}^{N} (x_n^i)(x_n^j) - \lambda \]  

(101)

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

\[ \dot{\delta w} = H(w^0)\delta w. \]  

(102)

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 [17]). There are fixed points (\( w \) satisfying \( E_n = 0 \)) that are degenerate. Minimization using the NMGD method converges to the minimum norm minimizer.

19.2 One-layer, 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 \( \lambda \) 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 [6] gives a rather complete characterization of the dynamics, here we describe a different approach.

The exponential loss for a linear network is

\[ L(f(w)) = \sum_{n=1}^{N} e^{-w^T x_n y_n} \]  

(103)

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 \( y_i(\tilde{w})^T x_n \geq \epsilon, \forall n = 1, \cdots, n \ \epsilon > 0 \). In some cases below (it will be clear from context) we incorporate \( y_n \) into \( x_n \).
1. **Dynamics** The dynamics is

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

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 [18]). The Hessian is

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

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

\[ \dot{\delta w} = H(w^0)\delta w. \]

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:

\[ \dot{w} = -\nabla_w (L + \lambda|w|^2) = \sum_{n=1}^{N} y_n x_n e^{-y_n (w^T x_n) - \lambda w}. \]
The minimum is given by \( \sum_n x_n e^{-w^T x_n} = \lambda w \), which can be solved by \( w = \sum_n k_n x_n \) with \( e^{-k_n x_n} \sum_j x_j = k_n \lambda \) 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

\[
-\sum_{n=1}^N x_n^T x_n e^{-y_n(x_n^T w^0)} - \lambda
\]  

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 \( \infty \) for \( \lambda \to 0 \). In the limit of \( \lambda \to 0 \) the equilibrium converges to a maximum margin solution for \( \tilde{w} = \frac{w}{||w||} \).

### 19.2.1 Weight normalization, linear case

Assume that all \( x_n \) are normalized vectors in \( \mathbb{R}^{d+1} \) (that is they are on \( \mathbb{S}^d \)) 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 \( \tilde{w}^T x_n > 0 \) \( \forall n = 1, \ldots, N \); this is always true if \( N \leq d + 1 \)). The dynamical system is

\[
\dot{\tilde{w}} = \frac{\sum_n e^{-\rho \tilde{w}^T x_n}}{\rho} (x_n - \tilde{w} \tilde{w}^T x_n).
\]  

and

\[
\dot{\rho} = \sum_n e^{-\rho \tilde{w}^T x_n} \tilde{w}^T x_n
\]

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

The dynamics imply \( \tilde{w} \to 0 \) 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 (25) \( \dot{\tilde{w}} \) 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 to 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 [6] but our approach is different and relies on the use of GD with unit constraint.

Consider the full dynamics

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

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 \( \sum_n e^{-\alpha x_n} \approx e^{-\alpha x_*}, x_* = \text{min}_n x_n \). 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. Note that the vector differential equation \( \dot{\tilde{w}} = (x - \tilde{w}\tilde{w}^T x) \) is a Riccati-type ODE.

### 19.3 Deep networks, square loss

\[
L(f(w)) = \sum_{n=1}^N (y_n - f(W; x_n))^2 \quad (111)
\]

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

1. **Dynamics**

The dynamics now is

\[
(W_k)_{i,j} = -F_k(w) = -\nabla_{W_k} L(W) = 2 \sum_{n=1}^N E_n \frac{\partial f}{\partial(W_k)_{i,j}} \quad (112)
\]

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

2. The Jacobian of \(-F\) – and Hessian of \(-L\) – for \( W = W_0 \) is
\[ J(W)_{kk'} = 2 \sum_{n=1}^{N} (-(\nabla W_k f(W; x_n)) (\nabla W_{k'} f(W; x_n)) + E_n \nabla^2 W_{kk'} f(W; x_n)) \]
\[ = -2 \sum_{n=1}^{N} (\nabla W_k f(W; x_n)) (\nabla W_{k'} f(W; x_n)), \tag{113} \]

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 [51]:

**Theorem 20** (K. Takeuchi) Let \( H \) be a positive integer. Let \( h_k = W_k \sigma(h_{k-1}) \in \mathbb{R}^{N_k,n} \) for \( k \in \{2, \ldots, H+1\} \) and \( h_1 = W_1 X \), 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 \( w = \text{vec}(W_1, \ldots, W_{H+1}) \in \mathbb{R}^{dN_1 + N_1N_2 + N_2N_3 + \ldots + N_HN_{H+1}} \). 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 \( k \in \{1, \ldots, H+1\} \) such that \( N_k N_{k-1} > n \cdot \min(N_k, N_{k+1}, \ldots, N_{H+1}) \) 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.

### 19.4 Deep networks, exponential loss

Consider the exponential loss

\[ L(f(W)) = \sum_{n=1}^{N} e^{-f(W; x_n)y_n} \tag{114} \]

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 [6].

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 $\Delta W_k$.

1. **Dynamics**

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

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

   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 $\delta W_k = J(W)\delta W$, with

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

   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 $i = 1, \cdots, K$

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

   For compactness of notation, let us define

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

   with which we have a transcendental equation for the minimum.

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

   The negative Hessian of the loss is then

   $$H_{kk'} = \sum_n \frac{\partial g_k^{(n)}}{\partial W_{k'}} - \lambda_k \delta_{kk'} I.$$
19.5 Deep Networks, weight normalization

Using Appendix 18, we obtain the dynamics for the normalized weights

\[
\dot{\tilde{W}}_{i,j}^k = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \frac{\rho}{\rho_k^2} \left( \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_{i,j}^k} - \tilde{W}_{i,j}^a, b \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_{a,b}^k} \right).
\]  

(121)

In Equation (121) 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 \(\tilde{W}_{i,j}^k\) which are not changed by gradient descent near the minimum will nevertheless change under this normalized dynamics.

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

\[
0 = \sum \alpha_n \left( \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_{i,j}^k} - \tilde{W}_{i,j}^a, b \frac{\partial \tilde{f}(x_n)}{\partial \tilde{W}_{a,b}^k} \right).
\]  

(122)

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

\[
\tilde{W}_{i,j}^k = \frac{1}{\hat{f}} \frac{\partial \hat{f}}{\partial \tilde{W}_{i,j}^k}.
\]  

(123)

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 \(f(x) = \frac{1}{K+1} \sum_{k=0}^{K} (f'_k)^T f'_k\) because \(W_k = f'_k\).

20 Experiments

Summary:

- SGD easily finds global minima in CIFAR10 suggesting that under appropriate over-parametrization it does not “see” any local minima of the empirical loss landscape.

- Different initializations affect the 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 linear case).

- Similar to initialization, perturbations of the weights increase norm and increase test error.

- The training loss of the normalized network predicts well the test performance of the same networks relative to other similar networks.
Figure 3: 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 with batch normalization was used with batch size = 100 for 70 epochs for each point. Neither data augmentation nor regularization were used.

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 3 and 4.

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 5.

In this very simple case we test our theoretical analysis with the following experiment. After convergence of GD, we apply a small random perturbation $\delta W$ 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
Figure 4: *Expected error in CIFAR-10 as a function of number of neurons.* The DNN is the same as in Figure 3. Batch normalization was also used. (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).

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.

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 9. 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 9 with Figure 6). 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
Figure 5: 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.

an equivalent optimum (because of “noise”) non-zero and non-vanishing regularization parameter $\lambda$. Thus a specific “early stopping” would be better than no stopping. The corresponding classification error, however, may not show overfitting.

Figure 10 shows the behavior of the loss in CIFAR in the absence of perturbations. This should be compared with Figure 9 which shows the case of an overparametrized linear network under quadratic loss corresponding to the multidimensional equivalent of the degenerate situation.
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 5. The feature matrix $\phi(X)$ is a polynomial with degree 30. The target function is a sine function $f(x) = \sin(2\pi f x)$ with frequency $f = 4$ on the interval $[-1, 1]$. The number of training points 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.

of Figure 5. The nondegenerate, convex case is shown in Figure 8. Figure 11 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 initial conditions without NMGD.
Figure 7: 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 5. 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 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 $1,000,000$ 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 halfway 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.
Figure 8: 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 points 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 absence 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.
Figure 9: 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 $3 \times 3$, 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 10: 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 11: 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 5. 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 were 76 and the number of test points were 600. The solution to the over-parametrized system was the minimum norm solution. More points were 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.