CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 90 **July 3, 2019**

Theory III: Dynamics and Generalization in Deep Networks[1]

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

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

Abstract

Classical generalization bounds for classification in the setting of separable data can be optimized by maximizing the margin of a deep network under the constraint of unit p-norm of the weight matrix at each layer. A possible approach for solving numerically this problem uses gradient algorithms on exponential-type loss functions, enforcing a unit constraint in the p-norm. In the limiting case of continuous gradient flow, we analyze the dynamical systems associated with three algorithms of this kind and their close relation for $p = 2$ with existing weight normalization and batch normalization algorithms. An interesting observation is that unconstrained gradient descent has a similar dynamics with the same critical points and thus also optimizes the generalization bounds. Our approach extends some of the results of Srebro from linear networks to deep networks and provides a new perspective on the implicit bias of gradient descent. This elusive complexity control is likely to be responsible for generalization despite overparametrization in deep networks.

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

[1]This replaces previous versions of Theory IIIa and TheoryIIIb.

Theory III: Dynamics and Generalization in Deep Networks*

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

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

Abstract

Classical generalization bounds for classification in the setting of separable data can be optimized by maximizing the margin of a deep network under the constraint of unit p-norm of the weight matrix at each layer. A possible approach for solving numerically this problem uses gradient algorithms on exponential-type loss functions, enforcing a unit constraint in the p-norm. In the limiting case of continuous gradient flow, we analyze the dynamical systems associated with three algorithms of this kind and their close relation for $p = 2$ with existing weight normalization and batch normalization algorithms. We prove that unconstrained gradient descent has a similar dynamics with the same critical points and thus also maximizes margin wrt L_2 norm (but not other p-norms). Our approach extends some of the results of Srebro from linear networks to deep networks and provides a new perspective on the implicit bias of gradient descent. The elusive complexity control we describe is responsible, at least in part, for the puzzling empirical finding of good generalization despite overparametrization by deep networks.

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

*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 normalizaton" of standard gradient descent has been modified.

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. Remarkable results by [6] illuminate the apparent absence of "overfitting" in the special case of linear networks for binary classification. They prove that minimization of loss functions such as the logistic, the cross-entropy and the exponential loss yields asymptotic convergence to the maximum margin solution for linearly separable datasets, independently of the initial conditions and without explicit regularization. Here we discuss the case of nonlinear multilayer DNNs in the setting of separable data, under exponential-type losses and square loss, for several variations of the basic gradient descent algorithm.

Our main results are:

- in the separable setting, gradient descent algorithms that maximize the margin while enforcing a unit p-norm constraint on the weight of the normalized network, optimize certain margin-based generalization bounds;
- weight normalization and batch normalization are algorithms of that type for the 2-norm;
- standard gradient descent in the weights followed by L_2 normalization (performed once after convergence) has similar dynamics and similar properties to margin maximization under an explicit 2-norm constraint.

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.

2 Deep networks: definitions and properties

Definitions We define a deep network with K layers with the usual coordinate-wise scalar activation functions $\sigma(z): \quad \mathbf{R} \to \mathbf{R}$ as the set of functions $f(W;x) = \sigma(W^K \sigma(W^{K-1} \cdots \sigma(W^1 x)))$, where the input is $x \in \mathbf{R}^d$, the weights are given by the matrices W^k, one per layer, with matching dimensions. We use the symbol W as a shorthand for the set of W^k matrices $k = 1, \cdots, K$. For simplicity we consider here the case of binary classification in which f takes scalar values, implying that the last layer matrix W^K is $W^K \in \mathbf{R}^{1,K_l}$. The labels are $y_n \in \{-1, 1\}$. The weights of hidden layer l are collected in a matrix of size $h_l \times h_{l-1}$. There are no biases apart form the input layer where the bias is instantiated by one of the input dimensions being a constant. The activation function in this paper is the ReLU activation. The norm we use is the L_2 unless we say otherwise.

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 $f(W;x) = \prod_{k=1}^{K} \rho_k f(V_1, \cdots, V_K; x_n)$, where $W_k = \rho_k V_k$ with the matrix norm $||V_k||_p = 1$. This implies the following property of ReLU

networks w.r.t. their Rademacher complexity:

$$\mathbb{R}_N(\mathbb{F}) = \rho \mathbb{R}_N(\tilde{\mathbb{F}}), \tag{1}$$

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 (we call $f(V;x) = \tilde{f}(x)$ with the understanding that $f(x) = f(W;x)$). 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 norms of the K weight matrices of f. Thus $f = \rho \tilde{f}$. Note that

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

and that the definitions of ρ_k, V_k and $\tilde{f}$ all depend on the choice of the norm used in normalization.

Structural property The following structural property of the gradient of deep ReLU networks is sometime useful (Lemma 2.1 of [7]):

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

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

$$(W_k, \frac{\partial f(x)}{\partial W_k^{i,j}} = f(x) \tag{4}$$

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

Lemma 3 implies the following

Corollary 1 *The condition* $\left(\frac{\partial f(x)}{\partial W_k}\right) = 0$ *implies* $f(x) = 0$.

In the case of square loss, this condition restricts the non-fitting critical points of the gradient to be linear combinations $\sum_{n=1}^{N} (f(x_n) - y_n) \frac{\partial f}{\partial W_k} = 0$, with $\frac{\partial f}{\partial W_k} \neq 0, \forall k$ or $f(x) = 0$. A similar restriction also holds for the exponential loss (see Equation 28).

Gradient flow and continuous approximation We will speak of the gradient flow of the empirical risk L referring to

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

where $\gamma(t)$ is the learning rate (which we will often neglect in order to make the notation less cumbersome). 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.

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 fact here we mostly assume the special setting of *separable data*.

3 The optimization landscape of Deep RELU Networks under exponential-type loss

The *first part* of the argument of this section relis on the fact, derived in in section 9.2, that RELU networks under the hypothesis of an exponential-type loss function, do not have local minima that separate the data.

Lemma 2 *Assume an exponential-type loss, separable data and homogeneity of a network – such as kernel machines and deep RELU networks. Then the the only critical points of the gradient that separate the data are the global minima.*

Notice that the global minima are at $\rho = \infty$ when the exponential is zero. As a consequence, the Hessian is identically zero with all eigenvalues being zero. On the other hand any point of the loss at a finite ρ has nonzero Hessian: for instance in the linear case the Hessian is proportional to $\sum_n^N x_n x_n^T$. The local minima which are not global minima must missclassify. How degenerate are they?

Consider a linear network in the exponential loss case. Assume there is a finite w for which the gradient is zero in some of its components. One question is whether this is similar to the regularization case or not, that is whether *misclassification regularizes*.

Let us look at a linear example:

$$\dot{w} = F(w) = -\nabla_w L(w) = \sum_{n=1}^{n} x_n^T e^{-x_n^T w} \tag{6}$$

in which we assume that there is one classification ' error (say for $n = 1$), meaning that the term $e^{-x_1^T w}$ grows exponentially with w. Let us also assume that gradient descent converges to w^*. This implies that $\sum_{n=2}^{n} x_n^T e^{-x_n^T w^*} = -x_1^T e^{-x_1^T w^*}$: for w^* the gradient is zero and $\dot{w} = 0$. Is this a hyperbolic equilibrium? Let us look at a very simple $1D$, $n = 2$ case:

$$\dot{w} = -x_1 e^{x_1 w^*} + x_2 e^{-x_2 w^*} \tag{7}$$

If $x_2 > x_1$ then $\dot{w} = 0$ for $e^{(x_1+x_2)w^*} = \frac{x_2}{x_1}$ which implies $w^* = \frac{\log(\frac{x_2}{x_1})}{x_1+x_2}$. This is clearly a hyperbolic equilibrium point, since we have

$$\nabla_w F(w) = -x_1^2 e^{x_1 w^*} - x_2^2 e^{-x_2 w^*} < 0, \tag{8}$$

so the single eigenvalue in this case has no zero real part.

In general, if there are only a small number of classification errors, one expects a similar situation for some of the components. *Differently from the regularization case, misclassification errors do not "regularize" all components of w but only the ones in the span of the misclassified examples.*

The more interesting case is with $D > N$. An example of this case is $D = 3$ and $N = 2$ in the above equation. The Hessian of the minimum will be degenerate with at least one zero eigenvalue ad one negative eigenvalue. In general there will be several more negative eigenvalues – in the order of N, for any point of the loss not at infinity, even in the absence of missclassifications. If $W > N$ they will not be hyperbolic minima (all W eigenvalues strictly negative) at least in the overparametrized linear networks case.

Clearly, it would be interesting to characterize better the degeneracy of the local minima. For the goals of this section however the fact that they cannot be completely degenerate is sufficient. We thus have the following

Theorem 3 *Under the exponential loss, global minima are completely degenerate with all eigenvalues of the Hessian (W of them with W being the number of parameters in the network) being zero. The other critical points of the gradient are less degenerate, with at least one – and typically N – nonzero eigenvalues.*

The *second part* of our argument (in [8]) is that SGD concentrates on the most degenerate minima. The argument is based on the 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 [9]. In addition, more informally, there is a certain similarity between SGD and SGDL suggesting that in practice the solution of SGD may be similar to the solution of SGDL. The Boltzman distribution is

$$p(W_k^{i,j}) = \frac{1}{Z} e^{-\frac{L(f)}{T}}, \tag{9}$$

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 [8]. 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 9.

Together [10] and [8] imply the following

Theorem *For overparametrized deep networks under an exponential-type loss, SGD selects with high probability global minimizers of the empirical loss, which are fully degenerate in the separable case.*

Remark For RELU networks there is a related property under the square loss. The critical points W^* of the gradient correspond to

$$\sum_{n=1}^{N}(y_n - f(x_n))\frac{\partial f(x_n; w)}{\partial W_k^{i,j}} = 0. \tag{10}$$

which are as many equations as weights. By multiplying both sides of the vector equations for any layer k by $(W_k^*)^T$ and using the structural property 3, we obtain

$$\sum_{n=1}^{N}(y_n - f(W^*; x_n)f(W^*; x_n) = 0. \tag{11}$$

The local minima which are not global minima must satisfy Equation 11.

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 [11, 12, 13, 14] 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 [15, 16, 17, 18, 19, 20]. Some local convergence results have also been proved [21, 22, 23]. The most interesting such approach is [20], which focuses on minimizing the training loss and proving that randomly initialized gradient descent can achieve zero training loss (see also [24, 25, 26]) 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 [27, 28, 29]. Closest to our approach – which is focused on the role of batch and weight normalization – is the paper [30]. 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 maximize margin while controlling the complexity of the classifier without the need to add a regularizer or to use weight decay. In fact, we will show that even standard gradient descent on the weights implicitly controls the complexity.

Very recently, after submitting this manuscript (and coinciding with the revision v3 of this work), a paper [31] appeared on arXiv claiming some of the same results described here and using an approach strikingly similar to our reparametrization of the gradient flow in the weights in terms of ρ and V components.

5 Optimizing Generalization Bounds

Classical *generalization bounds for regression* suggest that minimizing the empirical error subject to constrained *complexity of the minimizer* provides an optimal upper bound on the generalization

error. In general, one should optimize a *tradeoff between complexity and empirical loss*. In the special setting of interpolating minimizers, that is in the setting of zero empirical loss, one should select the interpolating solutions of smallest complexity in order to minimize the bound on the expected loss. 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 [32].

Certain well-known *margin bounds for classification* suggest a similar approach in the separable setting of zero classification errors in training, that is $y_n f(x_n) > 0, \forall n)$: maximization of the margin of the *normalized* network (the weights at each layer are normalized by the Frobenius norm of the weight matrix of the layer). The margin is the value of yf over the support vectors (the data with smallest margin $\min_n y_n f(x_n)$,. This approach is similar in spirit to the minimum norm solution for regression because maximizing the margin under unit norm constraint is equivalent to minimize the norm under the margin being larger than 1 (see Appendix 12).

5.1 Regression: (local) minimum norm empirical minimizers

We recall the typical form of upper bounds on generalization for regression [33]:

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

$$L(f) \leq \hat{L}(f) + c_1 \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{12}$$

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

The bound together with the property Equation 1 implies that among the minimizers with zero square loss– that is among interpolating minimizers – the optimization algorithm should select the minimum norm solution. In this case, *the "regularization" problem of trading off empirical error with complexity disappears*[1]. To select a minimum norm solution, one can use a slight modification of the standard gradient descent algorithms to provide a norm-minimizing GD update – NMGD in short – as

$$W_{n+1} - W_n = -(1-\lambda_n)\gamma_n \nabla_w L(f) - \lambda_n W_n, \tag{13}$$

where γ_n is the learning rate and $\lambda_n = \frac{1}{n^a}$ (this is one of several choices) is the vanishing regularization-like Halpern (see Appendix 15) term.

[1] As we already mentioned, the minimum norm interpolating solution is not always the best [34]: depending on factors such as noise in the data, the best solution may be a regularized one, represing a tradeoff between empirical error and complexity.

5.2 Classification: maximizing the margin of the normalized minimizer

Margin bounds suggest a constrained optimization problem. A typical margin bound for classification [35] is as follows

Proposition 5

$$L_{binary}(f) \leq L_{surr}(f) + b_1 \frac{\mathbb{R}_N(\mathbb{F})}{\eta} + b_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{14}$$

where $\eta(f) = \min_n y_n f(x_n)$ is the margin of f, $L_{binary}(f)$ is the expected classification error, $L_{surr}(f)$ is the empirical loss of a surrogate loss such as the logistic or the exponential. We are as before in the *separable setting*: there are no classification errors in training. This also implies that a surrogate function such as the exponential loss can be minimized to reach zero empirical loss. In this case, *the "regularization" problem of trading off empirical error with complexity disappears.* Since $\mathbb{R}_N(\mathbb{F}) \sim \rho \mathbb{R}_N(\tilde{\mathbb{F}})$ because of homogeneity of deep RELU networks (see Equation 1), the bound on $L_{binary}(f)$ is optimized by minimizing the empirical loss while maximizing $\eta(\tilde{f}) = \min_n y_n \tilde{f}(x_n)$, that is the margin of $\tilde{f}$.

The generalization bounds suggest minimizing the empirical loss while maximizing the margin subject to the product of the layer p-norms being equal to one:

$$\min L(f) s.t. \underset{\prod_k ||V_k||_p = 1}{\arg\max} \quad \min_n y_n \rho \tilde{f}(x_n). \tag{15}$$

In words: *find the network weights that maximize the margin subject to a p-norm constraint while minimizing the loss.* [2].

Using exponential loss to implement constrained optimization.

A standard way to maximize margin of the normalized classifier while minimizing the surrogate loss is to minimize an exponential-type loss function (in the following we focus specifically on the exponential loss but our analysis should hold for similar loss functions such as the logistic and the cross-entropy). The reason is that minimization of the exponential loss is margin maximizing. The intuition is as follows: with $\tilde{f}$ being the normalized network (weights at each layer are normalized by the *p-norm* of the corresponding weight matrix) and ρ being the product of the norms, the exponential loss $L(f) = \sum_n e^{-y_n f(x_n)} = \sum_n e^{-\rho y_n \tilde{f}(x_n)}$ approximates for "large" ρ a max operation, selecting among all the data points x_n the ones, labeled here with $m \in [1, \cdots, N]$ with the smallest margin $\tilde{f}(x_m)$. Thus minimization of $L(f)$ for large ρ corresponds to margin maximization of $\tilde{f}$:

$$\arg\min L(f) \approx \underset{V_k=1}{\arg\max} \quad \min_n y_n \tilde{f}(x_n). \tag{16}$$

[2] Existing generalization bounds such as Equation 6 in [36], 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.

To state this property more formally, we adapt to our setting results due to [37, 6] among others. First we recall the definition of the empirical exponential loss

$$L(f)(\rho_k, V_K) = \sum_n \ell(y_n \rho f(V; x_n)) \tag{17}$$

with $\ell(yf) = e^{-yf}$. We call V the set of weight matrices across all k that is all the $W_k^{i,j}$ normalized by a matrix p-norm of the associated layer, $V_k = \frac{W_k}{\rho_k}$, where $\rho_k = ||W_k||_p$ and $\rho = \prod_k \rho_k$.

The margin maximation theorem is

Theorem 6 *In the separable setting consider*

$$\hat{V} = \arg\min_{V,\rho} L(f), \tag{18}$$

with L defined by Equation 17. Then the exponential loss ℓ is margin maximizing in the sense that $\hat{V}$ (corresponding to $\rho \to \infty$) is the maximum margin in the p-norm of the normalized network $f(V) = \tilde{f}$.

Proof Outline The proof of Theorem 2.1, part (a) in [37] extends easily from linear networks ((In the linear network case $\tilde{f}$ is replaced by $\frac{\beta}{||\beta||_p} x$) to homogeneous deep networks (see also [38] Theorem 2.1). Though the full statement of Theorem 2.1 [37] involves a vanishing regularizer, part (a) of the proof there does not require it. Part (a) is the $T = \infty$ case, which is the one relevant for the exponential loss.

Theorem 6 can be supplemented with the following lemma, proved in Appendix 12:

Lemma 7 *Margin maximization of the network with normalized weights is equivalent to norm minimization under strict separability ($y_n f(x_n) \geq 1$).*

Implementing constrained optimization.

Given theorem 6, the approach is then to minimize the exponential loss function $L(f(w)) = \sum_{n=1}^N e^{-f(W;x_n)y_n} = \sum_{n=1}^N e^{-\rho f(V_k;x_n)y_n}$, with $\rho = \prod \rho_k$, subject to $||V_k||_p^p \forall k$, that is under a unit norm constraint for the weight matrix at each layer (if $p = 2$ then $\sum_{i,j}(V_k)_{i,j}^2 = 1$ is the Frobenius norm). Clearly these constraints imply the constraint on the norm of the product of weight matrices in (15) for any p norm (because any induced operator norm is a sub-multiplicative matrix norm).

It may be tempting to assume that the above optimization problem (especially when written just for the L_2 norm) is really unconstrained and that $\min_W L(f(W))$ is equivalent to $\min_{\rho_k, V_k, \forall k} L(\rho f(V)) s.t. ||V_k|| = 1$. In the following, we explore gradient descent techniques to perform the constrained optimization problem. In the process, it will become clearer when the equivalence between the constrained and the unconstrained problem holds and why.

6 Gradient techniques for norm control

There are several ways to implement the minimization in the tangent space of $||V||^2 = 1$. In fact, a review of gradient-based algorithms with unit-norm constraints [39] 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 [39]. The four techniques are closely related and have the same goal: performing gradient descent optimization with a unit norm constraint.

Remarks

- Stability issues for numerical implementations are discussed in [39].
- Interestingly, there is a close relationship between the Fisher-Rao norm and the natural gradient [7]. In particular, the natural gradient descent is the steepest descent direction induced by the Fisher-Rao geometry.
- Constraints in optimization such as $||v||_p = 1$ imposes a geometric structure to the parameter space. If $p = 2$ the weight vectors satisfying the unit norm constraint form a n-dimensional hyperspere of radius = 1. If $p = \infty$ they form an hypercube. If $p = 1$ they form a hyperpolyhedron.

6.1 Lagrange multiplier method

We start with one of the techniques, the Lagrange multiplier method, because it enforces the unit constraint in an especially transparent way. We incorporate the constraint in the exponential loss by defining a new loss as

$$L = \sum_{n=1}^{N} e^{-\rho \tilde{f}((x_n)y_n} + \sum_{k=1}^{K} \lambda(||V_k||_p^p - 1) \tag{19}$$

where the Lagrange multipliers λ_k are chosen to satisfy $||V_k||_p = 1$ at convergence or when the algorithm is stopped.

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

$$\dot{\rho_k} = \sum_n \frac{\rho}{\rho_k} e^{-\rho(t)\tilde{f}(x_n)y_n} \tilde{f}(x_n), \tag{20}$$

and for each layer k

$$\dot{V}_k = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}(t) + \lambda_k(t) p V_k^{p-1}(t). \tag{21}$$

The sequences $\lambda_k(t)$ must satisfy $\lim_{t\to\infty} ||V_k||_p = 1 \quad \forall k$.
Remarks

1. In the case of $p = 2$, with the conditions $||V_k|| = 1$ at each t, $\lambda_k(t)$ must satisfy

$$||V_k(t) + \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k} - 2\lambda_k(t) V_k(t)|| = 1. \tag{22}$$

Thus defining $g(t) = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}(t)$ we obtain

$$||V_k(t) + g(t) + 2\lambda_k(t) V_k(t)|| = 1, \tag{23}$$

that is

$$||\alpha(t) V_k(t) + g(t)|| = 1, \tag{24}$$

with $\alpha(t) = 1 + 2\lambda_k(t)$. The solution for α is

$$\alpha(t) = \sqrt{1 - ||g(t)||^2 + (V_k^T g(t))^2} - V^T(t) g(t). \tag{25}$$

Thus λ goes to zero at infinity because $g(t)$ does and $\alpha \to 1$.

2. Since the first term in the right hand side of Equation (21) goes to zero with $t \to \infty$ and the Lagrange multipliers λ_k also go to zero, the normalized weight vectors converge at infinity to $\dot{V}_k = 0$. On the other hand, $\rho(t)$ grows to infinity. As shown in section 8, the norm square ρ_k^2 (when $p = 2$) of each layer grows at the same rate.

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

4. As in the case of the of the vanishing regularizer assumed in the original theorem of [37], the Lagrange multipliers λ_k here go to zero.

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

$$u(k+1) = u(k) + g(k) \tag{26}$$

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

6.3 Tangent gradient method

Theorem 8 *([39]) Let $||u||_p$ denote a vector norm that is differentiable with respect to the elements of u and let $g(t)$ be any vector function with finite L_2 norm. Then, calling $v(t) = \frac{\partial ||u||_p}{\partial u}_{u=u(t)}$, the equation*

$$\dot{u} = h_g(t) = Sg(t) = (I - \frac{vv^T}{||v||_2^2})g(t) \tag{27}$$

with $||u(0)|| = 1$, describes the flow of a vector u that satisfies $||u(t)||_p = 1$ 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 [39].

In the case of $p = 2$ we replace v in Equation 27 with u because $v(t) = \frac{\partial ||u||_2}{\partial u} = u$. This gives $S = (I - \frac{uu^T}{||u||_2^2}$ and $\dot{u} = Sg(t)$.

Remarks

- For $p = 2$ $v = \frac{\partial ||u||_p}{\partial u}_u$ is $v = \frac{u}{||u||_2}$
- For $p = 1$, $\frac{\partial ||u||_1}{\partial u_j} = \frac{u_j}{|u_j|}$.
- For $p = \infty$, $\frac{\partial ||u||_\infty}{\partial u_j} = \text{sign}(u_k)\delta_{k,j}$, if maximum is attained in coordinate k.

7 Standard dynamics, Weight Normalization and Batch Normalization

We now discuss the relation of some existing techniques for training deep networks with the gradient descent techniques under unit norm constraint of the previous section.

7.1 Standard unconstrained dynamics

The standard gradient dynamics is given by

$$\dot{W}_k^{i,j} = -\frac{\partial L}{\partial W_k^{i,j}} = \sum_{n=1}^{N} y_n \frac{\partial f(x_n; w)}{\partial W_k^{i,j}} e^{-y_n f(x_n; W)} \tag{28}$$

where W_k is the weight matrix of layer k. As we observed, this dynamics has global minima at infinity with zero loss, if the data are separable. The other critical points have loss greater than zero.

Empirical observations suggest that unconstrained gradient dynamics on deep networks converges to solutions that generalize, especially when SGD is used instead of GD. In our experiments, normalization at each iteration, corresponding to the coefficient normalization method, improves generalization but not as much as Weight Normalization does.

7.2 Weight Normalization

For each layer (for simplicity of notation and consistency with the original weight normalization paper), weight normalization [40] defines v and g in terms of $w = g\frac{v}{|v|}$. The dynamics on g and v is induced by the gradient dynamics of w as follows:

$$\dot{g} = \frac{v}{||v||}\frac{\partial L}{\partial w} \tag{29}$$

and

$$\dot{v} = \frac{g}{||v||}S\frac{\partial L}{\partial w} \tag{30}$$

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

This is the same dynamics obtained from tangent gradient for $p = 2$. In fact, compute the flows in ρ, v from $w = \rho v$ as

$$\dot{\rho} = \frac{\partial w}{\partial \rho}\frac{\partial L}{\partial w} = v^T\frac{\partial L}{\partial w} \tag{31}$$

and $\dot{v} = \frac{\partial w}{\partial v||}\frac{\partial L}{\partial w} = \rho\frac{\partial L}{\partial w}$.

We then have to impose the unit norm constraint on v in the latter equation using the tangent gradient transform that gives

$$\dot{v} = S\rho\frac{\partial L}{\partial w}. \tag{32}$$

The dynamics is the same as for weight normalization. It is also easy to see that the dynamics above is not equivalent to the standard dynamics on the w (see also Figure 1).

7.3 Batch Normalization

Batch normalization [41] for unit i in the network normalizes the input vector of activities to unit i – that is it normalizes $X^j = \sum_j W^{i,j}x_j$, where x_j are the activities of the previous layer. Then it sets the activity to be

$$Y^j = \gamma \cdot \hat{X}^j + \beta = \gamma\frac{X^j - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta,$$

where γ, β are learned subsequently in the optimization and

$$\mu_B = \frac{1}{N}\sum_{n=1}^{N} X_n \qquad \sigma_B^2 = \frac{1}{N}\sum_{n=1}^{N}(X_n - \mu_B)^2.$$

Note that both μ_B and σ_B^2 are vectors, so the division by $\sqrt{\sigma_B^2 + \epsilon}$ has to be understood as a point-wise Hadamard product $\odot(\sigma_B^2 + \epsilon)^{-1/2}$. The gradient is taken wrt the new activations defined by the transformation above.

Unlike Weight Normalization, the Batch Normalization equations do not include an explicit computation of the partial derivatives of L with respect to the new variables in terms of the standard gradient $\frac{\partial L}{\partial w}$. The reason is that Batch Normalization works on an augmented network: a BN module is added to the network and partial derivatives of L with respect to the new variables are directly computed on its output. Thus the BN algorithm uses only the derivative of L wrt the old variables as a function of the derivatives of L wrt new variables in order to update the parameters below the BN module by applying the chain rule. Thus we have to estimate what BN *implies* about the partial derivatives of L with the respect to the new variables as a function of the standard gradient $\frac{\partial L}{\partial w}$.

To see the nature of the dynamics implied by batch normalization we simplify the original Equations (in the Algorithm 1 box in [41]). Neglecting μ_B and β and γ, we consider the core transformation as $\hat{X} = \frac{X}{\sigma_B}$ which, assuming fixed inputs, becomes $\hat{X} = \frac{X}{|X|}$ which is mathematically identical with the transfomation of section 7.1 $v = \frac{w}{|w|}$. In a similar way the dynamics of $w = \frac{\partial L}{\partial w}$ induces the following dynamics on $\hat{X}$:

$$\dot{\hat{X}} = \frac{\partial \hat{X}}{\partial X} \dot{X} \tag{33}$$

where $\dot{x} = \nabla_x L$. We consider $X \in \mathbb{R}^{N \times D}$. In the $D = 1$ case, we get

$$\frac{\partial \hat{X}}{\partial X} = (\sigma_B^2 + \epsilon)^{-1/2} \left[-\frac{1}{N} \hat{X} \hat{X}^T + I \right].$$

In the general D-dimensional vector case, this generalizes to

$$\frac{\partial \hat{X}}{\partial X} = (\sigma_B^2 + \epsilon)^{-1/2} \left[-\frac{1}{N} \hat{X}^T \odot \hat{X} + I \right].$$

Notice that $I - \hat{X}\hat{X}^T = S$. Since $x = W x_{input}$ this shows that batch normalization is closely related to *gradient descent algorithms with unit L_2 norm constraint of the tangent gradient type*. Because of the simplifications we made, there are other differences between BN and weight normalization, some of which are described in the remarks below.

Remarks

1. Batch normalization (see Appendix 14.5.6), does not control directly the norms of $W_1, W_2, \cdots, W_K$ as WN does. Instead it controls the norms

   $$||x||, ||\sigma(W_1 x||, ||\sigma(W_2 \sigma(W_1 x))||, \cdots \tag{34}$$

   In this sense it implements a somewhat weaker version of the generalization bound.

2. In the multilayer case, BN controls separately the norms $||V_i||$ of the weights into unit i, instead of controlling the overall Frobenius norm of the matrix of weights as WN does. Of course control of the $||V_i||$ implies control of $||V||$ since $||V||^2 = \sum_i ||V_i||_i^2$.

7.4 Weight Normalization and Batch Normalization enforce an explicit unit 2-norm constraint

Consider the *tangent gradient transformation* to a gradient increment $g(t) = \mu(k)\nabla_u L$ defined as $h_g = Sg(t)$ with $S = I - \frac{uu^T}{||u||_2^2}$. Theorem 8 says that the dynamical system $\dot{u} = h_g$ with $||u(0)||_2 = 1$ describes the flow of a vector u that satisfies $||u(t)|_2 = 1$ for all $t \geq 0$. It is obvious then that

Theorem 9 *(informal) The dynamical system describing weight normalization (Equations 29 and 30) as well as the dynamical system in the weights implied by batch normalization are not changed by the tangent gradient transformation.*

The proof follows easily for WN by using the fact that $S^2 = S$. The same argument can be applied to BN. The property is consistent with the statement that they enforce an L_2 unit norm constraint.

Thus all these techniques implement margin maximization of $\tilde{f}$ under unnit norm constraint of the weight matrices of $\tilde{f}$. 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: $\sum_n e^{-\rho(t)\tilde{f}(x_n)} \sim C \max_n e^{-\rho(t)\tilde{f}(x_n)}$, where C is the multiplicity of the x_n with minimal value of the margin of $\tilde{f}$. The data points with the corresponding minimum value of the margin $y_n\tilde{f}(x_n)$ are called support vectors (the x_i, y_i s.t $\arg\min_n y_n\tilde{f}(x_n)$). They are a subset of cardinality C of the N datapoints, all with the same margin η. In particular, the term $g(t) = \rho(t)\sum_n e^{-\rho(t)\tilde{f}(x_n)}\frac{\partial\tilde{f}(x_n)}{\partial V_k}$ becomes $g(t) \approx \rho(t)e^{-\rho(t)\eta}\sum_i^C \frac{\partial\tilde{f}(x_n)}{\partial V_k}$.

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

8 Generalization without explicit unit norm constraints: hidden complexity control/implicit bias

Empirically it appears that GD and SGD converge to solutions that can generalize even without batch or weight normalization. Convergence may be difficult for quite deep networks and generalization is not as good as with batch normalization but it still occurs. How is this possible? The answer is provided by the fact – trivial or surprising – that the unit vector $\frac{w(T)}{||w(T)||_2}$ computed

from the solution $w(T)$ of gradient descent $\dot{w} = -\nabla_w L$ at time T is the same, irrespectively of whether the constraint $||v||_2 = 1$ with $v = \frac{w}{||w||_2}$ is enforced during gradient descent. This confirm Srebro results for linear networks, extending some of them – though not his detailed analysis – to the deep network case. It also throws some light on the nature of the implicit bias or hidden complexity control.

We show this result next.

8.1 Reparametrization of standard gradient descent

We study the new dynamical system induced by the dynamical system in $\dot{W_k}^{i,j}$ under the reparametrization $W_k^{i,j} = \rho_k V_k^{i,j}$ with $||V_k||_2 = 1$. This is equivalent to changing coordinates from W_k to V_k and $\rho_k = ||W_k||_2$. For simplicity of notation we consider here for each weight matrix V_k the corresponding "vectorized" representation in terms of vectors $W_k^{i,j} = W_k$.

We use the following definitions and properties (for a vector w):

- The norm $||\cdot||$ is assumed in this section to be the L_2 norm.
- Define $\frac{w}{\rho} = v$; thus $w = \rho v$ with $||v||_2 = 1$ and $\rho = ||w||_2$.
- The following relations are easy to check:
  1. $\frac{\partial ||w||_2}{\partial w} = v$
  2. Define $S = I - vv^T = I - \frac{ww^T}{||w||_2^2}$. S has at most one zero eigenvalue since vv^T is rank 1 with a single eigenvalue $\lambda = 1$.
  3. $\frac{\partial v}{\partial w} = \frac{S}{\rho}$.
  4. $Sw = Sv = 0$
  5. $S^2 = S$
  6. In the multilayer case, $\frac{\partial f(x_n;W)}{\partial W_k} = \frac{\rho}{\rho_k} \frac{\partial \tilde{f}(V;x_n)}{\partial V_k}$

The unconstrained gradient descent dynamic system used in training deep networks for the exponential loss of Equation 88 is given by

$$\dot{W_k} = -\frac{\partial L}{\partial W k} = \sum_{n=1}^{N} y_n \frac{\partial f(W;x_n)}{\partial W_k} e^{-y_n f(W;x_n)}. \tag{35}$$

Following the chain rule *for the time derivatives*, the dynamics for W_k induces the following dynamics for $||W_k|| = \rho_k$ and V_k:

$$\dot{\rho_k} = \frac{\partial ||W_k||}{\partial W_k} \frac{\partial W_k}{\partial t} = V_k^T \dot{W_k} \tag{36}$$

and

$$\dot{V_k} = \frac{\partial V_k}{\partial W_k} \frac{\partial W_k}{\partial t} = \frac{S_k}{\rho_k} \dot{W_k} \tag{37}$$

where $S_k = I - V_k V_k^T$. We now obtain the time derivatives of V_k and ρ_k from the time derivative of W_k; the latter is computed from the gradients of L with respect to W_k that is from the gradient dynamics of W_k. Thus

$$\dot{\rho_k} = V_k^T \dot{W_k} = \sum_{n=1}^{N} V_k^T \frac{\partial f(x_n; W)}{\partial W_k} e^{-f(x_n;W)} = \frac{\rho}{\rho_k} \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \tilde{f}(x_n) \tag{38}$$

and

$$\dot{V_k} = \frac{\rho}{\rho_k^2} \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} (\frac{\partial \tilde{f}(x_n)}{\partial V_k} - V_k V_k^T \frac{\partial \tilde{f}(x_n)}{\partial V_k}). \tag{39}$$

In Equation 38, assuming that W_k is the vector corresponding to the weight matrix of layer k, we use Lemma 1 in [7]. We assume that f separates all the data, that is the margin is positive $\forall n$, that is $y_n f(x_n) > 0 \ \ \forall n$. Thus $\frac{d}{dt}\rho > 0$ and $\lim_{t\to\infty} \dot{\rho} = 0$. In the 1-layer network case the dynamics yields $\rho \approx \log t$ asymptotically. For deeper networks, this is different. In Section 14.5.1 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. Separately we show that for each layer $\frac{\partial \rho_k^2}{\partial t}$ is the same irrespectively of k.

An obvious sanity check is whether the normalized reparametrization of gradient descent on the W_ks in terms of ρ_k and V_k indeed gives the same dynamics. The answer is positive: the continuous unconstrained gradient descent dynamics $\dot{W_k}$ is equivalent to the dynamics yield by our reparametrization in terms of $\dot{\rho_k}$ and $\dot{V_k}$ with $W_k^{i,j} = \rho_k V_k^{i,j}$ and $||V_k|| = 1$. More precisely the following theorem holds

Theorem 10 *The continuous dynamical system* $\dot{W_k^{i,j}} = -\frac{\partial L}{\partial W_k^{i,j}}$ *describes the same dynamics as* $\dot{\rho_k}$ *and* $\dot{V_k^{i,j}}$ *with* $W_k^{i,j} = \rho_k V_k^{i,j}$ *and* $||V_k|| = 1$. *The latter dynamical system is*

$$\dot{\rho_k} = V_k^T \dot{W_k} \tag{40}$$

and

$$\dot{V_k} = \frac{S}{\rho_k} \dot{W_k} \tag{41}$$

The following identity holds: $\dot{W_k^{i,j}} = V_k^{i,j} \dot{\rho_k} + \rho_k \dot{V_k^{i,j}}$.

Since Theorem 10 shows that the standard dynamical system in the W_ks defined by $\dot{W_k} = -\frac{\partial L}{\partial W_k}$ is equivalent to the dynamical system of 40 and 41 (see Figure 1), and the latter contains the projector S, one would guess that the standard gradient flow is not changed by a tangent transformation (using the same argument used for weight normalization and batch normalization)[3].

[3]This is because S is a projector, so $S^2 = S$. Obviously, this should not be confused with saying that weight normalization or batch normalization would not lead to changes in gradient descent.

The better way to prove that unconstrained normalization is equivalent to constrained normalization for $p = 2$ is the following. Consider the dynamical system obtained by minimizing L in ρ and v with $w = \rho v$ under the constraint $||v||_2 = 1$. This the dynamical system for weight normalization in our notation that is

$$\dot{\rho} = v^T \dot{w} \quad \dot{v} = S\rho\dot{w} \tag{42}$$

which has to be compared with the standard gradient equations

$$\dot{\rho} = v^T \dot{w} \quad \dot{v} = \frac{S}{\rho}\dot{w}. \tag{43}$$

The two dynamical systems are not identical; however, they are very similar differing by a ρ^2 factor in the $\dot{v}$ equations. It is clear that the critical points of the gradient for the v vectors – that is the values for which $\dot{v} = 0$ – are the same in both cases since for any $t > 0$ $\rho(t) > 0$ and thus in both cases $\dot{v} = 0$ requires $S\dot{w} = 0$.

We have thus proven

Theorem 11 *Under exponential-type losses, if all the weights W_k converges to $W_k(T)$ for $T \to \infty$, the unit vector $\frac{W_k(T)}{\rho_k(T)}$ represents a maximum margin solution under L_2unit norm constraint.*

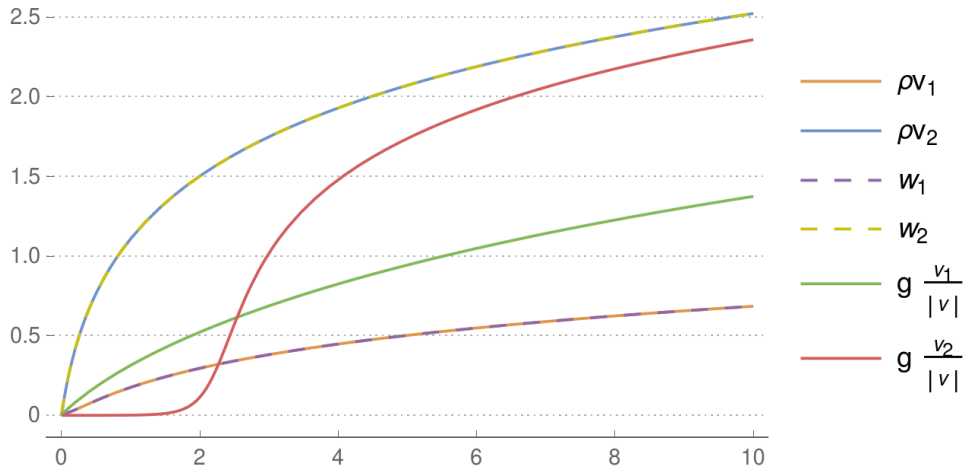


Figure 1: *Example comparison of dynamics $\dot{W}_k = -\frac{\partial L}{\partial W_k}$ (dashed lines) and their equivalence to 40 and 41 (orange and blue) as a function of time. Standard weight normalization leads to different trajectories of gradient descent (in green and red). The example is that of a linear network with only two parameters and two training examples, using an exponential loss.*

Another property of the dynamics of W_k, shared with the dynamics of V_k under explicit unit norm constraint, is suggested by recent work [42]: 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 approximatevely equal Frobenius norms across different layers, which is *a consequence of the norm constraint.*. This property is expected in a minimum norm situation, which is itself equivalent to maximum margin under unit norm (see Appendix 12). The observation of [42] is easy to prove in our framework. Consider the gradient descent equations

$$\dot{W}_k^{i,j} = \sum_{n=1}^{N} y_n [\frac{\partial f(W; x_n)}{\partial W_k^{i,j}}] e^{-y_n f(x_n;W)}. \tag{44}$$

The above dynamics induces the following dynamics on $||W_k||$ using the relation $||\dot{W}_k|| = \frac{\partial ||W_k||}{\partial W_k} \frac{\partial W_k}{\partial t} = \frac{W_k}{||W_k||} \dot{W}_k$:

$$||\dot{W}_k|| = \frac{1}{||W_k||} \sum_{n=1}^{N} y_n f(W; x_n) e^{-y_n f(x_n;W)}. \tag{45}$$

because of lemma 3. It follows that

$$||\dot{W}_k||^2 = 2 \sum_{n=1}^{N} y_n f(W; x_n) e^{-y_n f(x_n;W)}, \tag{46}$$

which shows 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 $\frac{\mathbb{R}_N(\mathbb{F})}{\min_x \rho^K \tilde{f}(x)}$. The balance of norms property allows us to cancel the dependence on ρ for all layers.

Theorem 11 for exponential-type losses is the *non-linear multi-layer extension of the main result of [6] for linear networks.* 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, 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 [43, 34]. Though this is still an open question, it seems possible that similar results may be valid for deep networks. In addition, commonly used weight decay with appropriate parameters can induce generalization since it is equivalent to regularization. Furthermore, typical implementations of data augmentation may also effectively avoid overparametrization: at each iteration of SGD only "new" data are used and depending on the number of iterations it is quite possible that the size of the training data exceeds the number of parameters. In any case, within this online framework, one expects convergence to the minimum of the expected risk (see Appendix 13) without the need to invoke generalization bounds.

9 Dynamics of Gradient Descent

In the appendices we discuss the dynamics of gradient descent in the continuous framework. 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 coupled equations – *as many as the number of weights* – have to be satisfied asymptotically

$$V_k = \frac{g(t)}{2\lambda}. \tag{47}$$

where $g(t) = \rho(t) \sum_n e^{-\rho(t)\tilde{f}(x_n)} \frac{\partial \tilde{f}(x_n)}{\partial V_k}$ and $\lambda(t)$ goes to zero at infinity at the same rate as $g(t)$. This suggests that weight matrices should be in relations of the type $W_K = (W_{K-1}...W_1 x)^T$ for linear K-layers nets; 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.* An instrucive case is the linear two-layers autoencoder and the solution to the training of it in terms of transposed weights for the two layers. The characterization of the conditions for convergence of Equations 47 is an open problem that we sketch in the Appendix.

The solutions for the critical points Equations 47 can be also written as (using non-zero α_n associated with the support vectors)

$$0 = \sum \alpha_n \left(\frac{\partial \tilde{f}(x_n)}{\partial V_k^{i,j}} - V_k^{i,j} V_k^{a,b} \frac{\partial \tilde{f}(x_n)}{\partial V_k^{a,b}} \right). \tag{48}$$

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

$$V_k^{i,j} = \frac{1}{\hat{f}} \frac{\partial \hat{f}}{\partial V_k^{i,j}}. \tag{49}$$

These equations – that are consistent – correspond to the property of normalization of the weights; normalization is essential for a non-trivial maximization of the margin. In standard gradient descent the normalization constraint is implicit, enforced by dividing the W_k obtained from the gradient descent iterations by the norm $||W_k||$ and using the resulting $\tilde{f}$ for classification.

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 for the Lagrange multiplier case is

$$\sum_n \left[- \left(\prod_{i=1}^K \rho_i^2 \right) \frac{\partial \tilde{f}(V; x_n)}{\partial V_k} \frac{\partial \tilde{f}(V; x_n)}{\partial V_{k'}}^T + \left(\prod_{i=1}^K \rho_i \right) \frac{\partial^2 \tilde{f}(V; x_n)}{\partial V_k \partial V_{k'}} \right] e^{-\prod_{i=1}^K \rho_i \tilde{f}(V; x_n)} - 2\lambda(t)\mathbf{I}. \tag{50}$$

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).

An examination of the gradient descent dynamics on exponential losses, discussed above, suggests experimenting with *a new family of algorithms* in which $\rho(t)$ is designed and controlled independently of the gradient descent on the V_k: we need ρ to grow in order to drive the exponential loss towards the ideal classification loss but our real interest is in maximizing the margin of $\tilde{f}$. The ideal time course of $\rho(t)$ could be made adaptive to the data. It seems possible to test dynamics that may be quite different from $\rho \propto \log t$ and may instead be faster in later stages of the iterations.

9.1 Linear networks: rates of convergence

We consider here the linear case. We rederive some of the results by [6] in our gradient flow framework. In the separable case of a linear network ($f(x) = \rho V^T x$) the dynamics is

$$\dot{\rho} = \frac{1}{\rho} \sum_{n=1}^{N} e^{-\rho V^T x_n} V^T x_n \tag{51}$$

and

$$\dot{V} = \frac{1}{\rho} \sum_{n=1}^{N} e^{-\rho V^T x_n} (x_n - V V^T x_n). \tag{52}$$

As discussed earlier there are K support vectors with the same smallest margin. For t increasing, since $\rho \approx \log t$,

$$\dot{V} \propto \frac{1}{\rho} \sum_{j=1}^{K} \alpha_j (I - V V^T) x_j. \tag{53}$$

with $\alpha_j = e^{-\rho V^T x_j}$. If gradient descent converges, the solution V satisfyies $V V^T x = x$, where $x = \sum_{j=1}^{K} \alpha_j x_j$. It is easy to see that $V = ||x||^2 x^\dagger$ – where $x^\dagger$ is the pseudoinverse of x – is a solution since the pseudoinverse of a non-zero vector z is $z^\dagger = \frac{z^T}{||z||^2}$. On the other hand, in the linear case the operator T in $V(t+1) = TV(t)$ associated with equation 54 is not expanding because V has unit norm. Thus [44] there is a fixed point $V = x$ which is *independent of initial conditions.*

The rates of convergence of the solutions $\rho(t)$ and $v(t)$, derived in different way in [6], may be read out from the equations for ρ and v. It is easy to check that a general solution for ρ is of the form $\rho \propto C \log t$. A similar estimate for the exponential term, gives $e^{-\rho v^T x_n} \propto \frac{1}{t}$. We claim that a solution for the error $\epsilon = v - x$, since v converges to x, behaves as $\frac{1}{\log t}$. In fact we write $v = x + \epsilon$ and plug it in the equation for v in 54 (we also assume for notation convenience a single data point x). We obtain

$$\dot{\epsilon} = \frac{1}{\rho} e^{-\rho v^T x} (x - (x+\epsilon)(x+\epsilon)^T x) = \frac{1}{\rho} e^{-\rho v^T x} (x - x - x\epsilon^T - \epsilon x^T) \tag{54}$$

which has the form $\dot{\epsilon} = -\frac{1}{t \log t}(2x\epsilon^T)$. A solution of the form $\epsilon \propto \frac{1}{\log t}$ satisfies the equation: $-\frac{1}{t \log^2 t} = -B\frac{1}{t \log^2 t}$. Thus the error indeed converges as $\epsilon \propto \frac{1}{\log t}$.

A similar analysis for the weight normalization equations 42 considers the same dynamical system with a change in the equation for v which becomes

$$\dot{v} \propto e^{-\rho}\rho(I - vv^T)x. \tag{55}$$

This equation differs by a factor ρ^2 from equation 54. As a consequence equation 55 is of the form $\dot{\epsilon} = -\frac{\log t}{t}\epsilon$, with a general solution of the form $\epsilon \propto t^{-1/2 \log t}$. In summary, *GD with weight normalization converges faster to the same equilibrium than standard gradient descent: the rate for $\epsilon = v - x$ is $\frac{1}{t^{1/2 \log t}}$ vs $\frac{1}{\log t}$.*

Our simplified analysis implies that batch normalization has the same rate as weight normalization (in the linear one layer case BN is identical to WN). As we discussed, it is however different wrt WN in the multilayer case: it has for instance separate normalization for each unit, that is for each row of the weight matrix at each layer.

9.2 Critical points of the gradient under exponential loss for RELU networks

The critical points of the gradient of f in the nonlinear multilayer separable case under exponential loss are characterized by

$$\sum_{n=1}^{N} y_n \frac{\partial f(x_n; w)}{\partial W_k^{i,j}} e^{-y_n f(x_n;W)} = 0. \tag{56}$$

Lemma 12 *Under the exponential loss, multilayer RELU networks have only global zeros of the gradient that are data separating.*

The global zeros of the loss are at infinity, that is for $\rho \to \infty$ in the exponential. If other critical points were to exist for a value W^* of the weights, they would be given by zero-valued linear combinations with positive coefficients of $\frac{\partial f(x_n;w)}{\partial W_k^{i,j}}$. Use of the structural Lemma shows that $\frac{\partial f(x;w)}{\partial W_k^{i,j}} = 0, \forall i, j, k$ implies $f(W^*; x) = 0$. So nonglobal critical points that are data-separating do not exist.

10 Discussion

Our main results are *for classification* in the setting of *separable data* for *continous gradient flow* under *the exponential loss*:

- gradient descent methods with a p-norm constraint converge, when they converge, to a maximum margin solution and therefore generalize because they minimize classical margin upper bounds;

- examples of such techniques commonly used to train over-parametrized multilayer, RELU deep networks are weight normalization and batch normalization enforcing an explicit unit constraint in the 2-norm.

- surprisingly, *for classification, under an exponential loss, there is an implicit 2-norm constraint 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 additional explicit regularization or explicit norm constraints.

- the convergence rate of standard GD ($\frac{1}{\log t}$) is slower than the convergence rate ($\frac{1}{t^{\log(\sqrt{(t)}}}$) of weight normalization.

- the dynamics of W_k does not have any local minima but only global minima – for $||W_k|| \to \infty$. The dynamics of V_k has nontrivial global minima.

- several of these results *directly apply to kernel machines* for the exponential loss under the separability/interpolation assumption because they kernel machines are homogeneous functions like deep RELU networks.

Of course the gradient flows corresponding to normalization with different p-norms are expected to be different and to converge to different solutions, as in the classical case of support vector machines with L_2 vs L_1 regularizers. It is useful to emphasize that despite the similarities between some of the methods enforcing unit constraints in the 2-norm, they usually correspond to different dynamical flows but with the same qualitative dynamics and the same stationary solutions. In particular, batch normalization and weight normalization are not the same and in turn they are slightly different from standard gradient descent. Furthermore, our analysis has been restricted to the continous case; the discrete case may yield greater differences.

The fact that the solution corresponds to a maximum margin solution under a fixed norm constraint may explain the puzzling behavior of Figure 5, 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 *weight normalization and batch normalization are enforcing a fundamental constraint for generalization*, deeper than reducing covariate shifts (the properties described in [45] are fully consistent with our characterization in terms of a norm constraint). Controlling the norm of the weights is exactly what the generalization bounds prescribe. 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.

Our results assume densely connected deep networks. We did not explore the constraints implied by convolutional layers. There are many other open problems. Does the empirical landscape have multiple global minima with different minimum norms (see Figure 3), as we suspect? Or is the landscape "nicer" for large overparametrization – as hinted in several recent

papers (see for instance [46] and [47])? Can one ensure convergence to the global empirical minimizer with global minimum norm? What is the theoretical explanation of the empirical observation that batch normalization is better than weight normalization? This requires some explanation because all of them enforce implicitely or explicitely a unit constraint in the L_2 norm.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[19] Simon Du, Jason Lee, Yuandong Tian, Aarti Singh, and Barnabas Poczos. Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1339–1348, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

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

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

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

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

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

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

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

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

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

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

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

[31] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *CoRR*, abs/1906.05890, 2019.

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

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

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

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

[36] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-Independent Sample Complexity of Neural Networks. *arXiv e-prints*, page arXiv:1712.06541, Dec 2017.

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

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

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

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

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

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

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

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

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

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

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

[48] Nigel Hitchin. Differentiable manifolds. *www.maths.ox.ac.uk/ hitchin/*, 2012.

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

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

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

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

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

Appendices

11 The optimization landscape of Deep Nets with Generic Activations, loss functions and no separability: Bezout theorem

In [10, 8] 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 $f(x_i) - y_i = 0 \;\; for \; i = 1, \cdots, N$;
- critical points which correspond to saddles and to local minima for which the loss is not zero but $\nabla_W \sum_{i=1}^N L(f(x_i), y_i) = 0$,

Suppose that the polynomial network does not have any symmetry (not even the one-homogeneity symmetry characteristic for the RELU networks). 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.

On the other hand, 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*. Thus these simple observations suggest that the global minima are more degenerate than the local minima.

A more careful analysis, that we are developing with J. Bloom, is based on Sard's theorem (see Appendix 11.1) and gives the result that while the degeneracy of the global zeros is generically $D-N$, the degeneracy of the non-fitting critical points is typically 0 if none of the weights are zero. This follows from rewriting the equation for critical points as $\nabla_W (f(X) - Y) \cdot (f(X) - Y) = 0$, where now $\nabla_W (f(X) - Y)$ is an $N \times D$ matrix. The question reduces to studying the dimensionality of the kernel. For critical points that are not interpolating, $E \equiv (f(X) - Y) \neq 0$, we have to find a gradient orthogonal to E. We are trying to prove

Conjecture:*For appropriate overparametrization, the optimization landscape of deep networks under the square loss shows a large number of global zero-error minimizers which are degenerate*

with probability one on a set of dimension $D - N$; the other critical points – saddles and local minima – are degenerate on a set of lower dimensionality.

In particular, we would like to show that the most degenerate non-fitting critical points are those that arise from the weights in a single layer being all 0 – for deep networks this automatically makes the gradients of all other layers vanish. This is an extreme situation because the output of the network in this case would be zero irrespectivey of the input. Denoting by D_l the number of parameters in l-th layer, if all of the weights in this layer vanish, the only equation left to solve is $\nabla_{W_l}(E) \cdot E = 0$. This is an equation in D_l unknowns that depends on all the weights in the previous layers. This means that we expect the highest degeneracy when all the D_K weights in the last layer K disappear, with the number of degenerate directions we can calculate in the same way as above for the global minima, and which comes out to $(D - D_K) - D_K = D - 2D_K$. This conjecture is supported by numerical checks on a few architectures. Thus, provided that the last layers have a large number of parameters (a situation fairly typical when the last layers are fully connected), the global minima are more degenerate than the critical points. 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).

11.1 The Preimage theorem

We use the following theorem:

Theorem 13 *(The Preimage Theorem [48]). 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 $dimM - dimN$.*

The proof goes as follows. Consider the N maps $F_i : \mathbf{R}^D \mapsto \mathbf{R}$, that is $F_i : (w^1_{1,1}, \cdots w^K_{n,n}) \mapsto P_i$, where P_i is a polynomial in the D ws with coefficients provided by the training example x_i and $P_i = y_i$. As an example assume that $P_i = (w_1 x_1 + w_2 x_2)^2$. Then for, say, $x_1 = 1, x_2 = 2$, $P_i = (w_1 + 2w_2)^2$. Assume $y_i \neq 0$. Consider the set $H_{y_i} = F^{-1}(y_i)$. We need to check that DF_v is surjective for any $v = (v_1, v_2)$. DF_v is a linear transformation from $\mathbf{R}^2$ to $\mathbf{R}$ 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 $DF_v(v_1, v_2) = 2v_1^2 + 8v_2 v_1 + 8v_2^2 = 2y_i \neq 0$. Therefore DF_v is surjective $\forall v \in F^{-1}(y_i)$ and the preimage of y_i is a manifold of dimensionality $D - 1$.

11.2 Exponential loss

Under the exponential loss, no local minima exist that are not global and separate the data. Therefore the only local minima must missclassify. How degenerate are they?

Consider a linear network in the exponential loss case. Assume there is a finite w for which the gradient is zero in some of its components. One question is whether this is similar to the regularization case or not, that is whether *misclassification regularizes*.

Let us look at a linear example:

$$\dot{w} = F(w) = -\nabla_w L(w) = \sum_{n=1}^{n} x_n^T e^{-x_n^T w} \tag{57}$$

in which we assume that there is one classification ' error (say for $n = 1$), meaning that the term $e^{-x_1^T w}$ grows exponentially with w. Let us also assume that gradient descent converges to w^*. This implies that $\sum_{n=2}^{n} x_n^T e^{-x_n^T w^*} = -x_1^T e^{-x_1^T w^*}$: for w^* the gradient is zero and $\dot{w} = 0$. Is this a hyperbolic equilibrium? Let us look at a very simple $1D$, $n = 2$ case:

$$\dot{w} = -x_1 e^{x_1 w^*} + x_2 e^{-x_2 w^*} \tag{58}$$

If $x_2 > x_1$ then $\dot{w} = 0$ for $e^{(x_1+x_2)w^*} = \frac{x_2}{x_1}$ which implies $w^* = \frac{\log(\frac{x_2}{x_1})}{x_1+x_2}$. This is clearly a hyperbolic equilibrium point, since we have

$$\nabla_w F(w) = -x_1^2 e^{x_1 w^*} - x_2^2 e^{-x_2 w^*} < 0, \tag{59}$$

so the single eigenvalue in this case has no zero real part.

In general, if there are only a small number of classification errors, one expects a similar situation for some of the components. *Differently from the regularization case, misclassification errors do not "regularize" all components of w but only the ones in the span of the misclassified examples.*

The more interesting case is with $D > N$. An example of this case is $D = 3$ and $N = 2$ in the above equation. The Hessian of the minimum will be degenerate with at least one zero eigenvalue ad one negative eigenvalue. In general there will be several more negative eigenvalues – in the order of N, for any point of the loss not at infinity, even in the absence of missclassifications. If $W > N$ they will not be hyperbolic minima (all W eigenvalues strictly negative) at least in the overparametrized linear networks case.

12 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. The norm assumed in this section is the L_2 norm though the proof can be extended. We show they directly extend beyond linearly parametrized functions as long as there is a one-homogeneity property, namely, for all $\alpha > 0$,

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

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

$$\min_{i=1,\ldots,N} y_i f(W; x_i). \tag{60}$$

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

$$\max_W \min_{i=1,\dots,N} y_i f(W; x_i), \qquad \text{subj. to} \quad \|W\| = 1. \tag{61}$$

The latter constraint is needed to avoid trivial solutions in light of the one-homogeneity property. We next show that Problem (61) is equivalent to

$$\min_W \frac{1}{2}\|W\|^2, \qquad \text{subj. to} \quad y_i f(W; x_i) \geq 1, \qquad i = 1, \dots, N. \tag{62}$$

To see this, we introduce a number of equivalent formulations. First, notice that functional margin (60) can be equivalently written as

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

Then, the max margin problem (61) can be written as

$$\max_{W,\gamma>0} \gamma, \qquad \text{subj. to} \quad \|W\| = 1, \quad y_i f(W; x_i) \geq \gamma, \qquad i = 1, \dots, N. \tag{63}$$

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

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

so that Problem (63) becomes

$$\max_{W,\gamma'>0} \frac{\gamma'}{\|W\|}, \qquad \text{subj. to} \qquad y_i f(W; x_i) \geq \gamma', \qquad i = 1, \dots, N. \tag{64}$$

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

$$\max_W \frac{1}{\|W\|}, \qquad \text{subj. to} \qquad y_i f(W; x_i) \geq 1, \qquad i = 1, \dots, N. \tag{65}$$

The result is then clear noting that

$$\max_W \frac{1}{\|W\|} \Leftrightarrow \min_W \|W\| \Leftrightarrow \min_W \frac{\|W\|^2}{2}.$$

13 Data augmentation and generalization with "infinite" data sets

In the case of batch learning, *generalization guarantees* on an algorithm are conditions under which the empirical error $I_{S_N}(f)$ on the training set converges to the expected error $I(f)$, ideally

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

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

14 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, mostly for the case of unnormalized weights.

1. *One-layer networks* The Hessian is in general degenerate. Regularization with arbitrarily small λ ensures independence from initial conditions for both the square and the exponential loss. In the absence of explicit regularization, GD converge to the minimum norm solution for zero initial conditions under the square loss. With NMGD-type iterations GD converges 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 there is convergence to the normalized solution $\tilde{f}$ that maximizes the margin (and that corresponds to the overall minimum norm solution),

2. *Deep networks, square loss* The Hessian is in general degenerate, even in the presence of regularization (with fixed λ). For $\lambda \to 0$, 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. *Implications of minimum norm for generalization in regression problems* NMGD-based minimization ensures minimum norm solutions.

Remarks

- NMGD can be seen as an interesting extension of regularization (that is weight decay) by requiring λ to decrease to zero. NMGD ensures minimum norm or equivalently maximum margin solutions.

- Notice that one of the definitions of the pseudoinverse of a linear operator corresponds to NMGD: it is the regularized solution to a degenerate minimization problem in the square loss for $\lambda \to 0$.
- The failure of regularization with a fixed λ to induce hyperbolic solutions in the multi-layer case was surprising to us. Technically this is due to contributions to non-diagonal parts of the Hessian from derivatives across layers and to the shift of the minimum.

14.1 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 λ ensures independence of initial conditions. Even in the absence of explicit regularization GD converge to the minimum norm solution for zero initial conditions. Convergence to the minimum norm holds also with NMGD-type iterations but now independently of initial conditions.

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

Consider

$$L(f(w)) = \sum_{n=1}^{N} (y_n - w^T x_n)^2 \tag{66}$$

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

1. *Dynamics* The dynamics is

$$\dot{w} = -F(w) = -\nabla_w L(w) = 2 \sum_{n=1}^{N} E_n x_n^T \tag{67}$$

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

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

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

$$J_F(w) = H = -\sum_{n=1}^{N} (x_n^i)(x_n^j) \tag{68}$$

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

$$\dot{\delta w} = J_F(w^0)\delta w. \tag{69}$$

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 $d > n$.

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

$$\dot{w} = -\nabla_w(L + \lambda|w|^2) = 2\sum_{n=1}^{N} E_n x_n^T - \lambda w \tag{70}$$

gives for $\dot{w} = 0$

$$w_0 = \frac{2}{\lambda}\sum_{n=1}^{N} E_n.x_n^T \tag{71}$$

The Hessian at w_0 is with

$$H(w) = -\sum_{n=1}^{N}(x_n^i)(x_n^j) - \lambda \tag{72}$$

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

$$\dot{\delta w} = H(w^0)\delta w. \tag{73}$$

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

14.2 One-layer, exponential loss

Linear networks under exponential loss and GD show growing Frobenius norm. On a compact domain ($\rho \leq R$) the exponential loss is L-smooth and corresponds to a non-expansive operator T. Regularization with arbitrarily small λ ensures convergence to a fixed point independent of initial conditions.

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

$$L(f(w)) = \sum_{n=1}^{N} e^{-w^T x_n y_n} \tag{74}$$

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

1. *Dynamics* The dynamics is

$$\dot{w} = F(w) = -\nabla_w L(w) = \sum_{n=1}^{N} x_n^T e^{-x_n^T w} \tag{75}$$

   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 Srebro result on convergence does not depend 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 14.5.3). The Hessian is

$$H = -\sum_{n=1}^{N} (x_n^i)(x_n^j) e^{-(w^T x_n)}. \tag{76}$$

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

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

$$\dot{\delta w} = H(w^0)\delta w. \tag{77}$$

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:

$$\dot{w} = -\nabla_w(L + \lambda|w|^2) = \sum_{n=1}^{N} y_n x_n e^{-y_n(w^T x_n)} - \lambda w. \tag{78}$$

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

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

$$-\sum_{n=1}^{N} x_n^T x_n e^{-y_n(x_n^T w^0)} - \lambda \tag{79}$$

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

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

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

14.2.1 Weight normalization, linear case

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

$$\dot{v} = \frac{\sum_n e^{-\rho v^T x_n}}{\rho}(x_n - v v^T x_n). \tag{80}$$

and

$$\dot{\rho} = \sum_n e^{-\rho v^T x_n} v^T x_n \tag{81}$$

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

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

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

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

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

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

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) $v^T x_n$ dominate. This is because for large $||\alpha||$, the term $\sum_n e^{-\alpha x_n} \approx e^{-\alpha x_*}, x_* = min_n x_n$. Thus $\dot{v} \approx \frac{e^{-||w||v^T x_*}}{||w||} \sum_*(x_* - vv^T x_*)$, where $\sum_*$ indicates a sum over the support vectors. This converges to

$$v = \sum_* \alpha_* x_*,$$

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 $\dot{v} = (x - vv^T x)$ is a Riccati-type ODE.

14.3 Degeneracy of the Hessian for deep networks

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

One might suspect that similar behavior might be exhibited by deep networks too. However, as seen above, away from the critical points the Hessian of deep nets can have eigenvalues of all signatures. Close to critical points obtained by GD however, numerical studies [49] 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 $\lambda\rho_F^2$ does not guarantee non-degenerate critical points.*

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

$$L(w) = e^{-w_1 w_2 - w_3 w_4} + \lambda(w_1^2 + w_2^2 + w_3^2 + w_4^2) \tag{82}$$

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

$$w_1^* = \frac{e^{-f_*}}{2\lambda} w_2^*, \qquad w_2^* = \frac{e^{-f_*}}{2\lambda} w_1^*, \tag{83}$$

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

$$\det H = e^{-4f_*} \left(4\lambda^2 e^{-2f_*} - 1\right) \left(4\lambda^2 e^{-2f_*} + 2\lambda e^{f_*}(w_1^2 + w_2^2 + w_3^2 + w_4^2) + 2f_* - 1\right). \tag{84}$$

At the origin we find a local minimum with a positive definite Hessian, but at $e^{-f_*} = 2\lambda$ the determinant is 0. Thus for arbitrary λ, the global minimum is degenerate. This degeneracy stems from a freedom of reparametrization provided by two circles in the w_1-w_2 and w_3-w_4 planes[4]. 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 [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 $\mathrm{GL}_{d_{L-1}}(\mathbb{R}) \times \mathrm{GL}_{d_{L-2}}(\mathbb{R}) \times \cdots \times \mathrm{GL}_{d_1}(\mathbb{R})$, that is invertible $d_k \times d_k$ matrices between the layers acting as $(W_k, W_{k+1}) \mapsto (GW_k, W_{k+1}G^{-1})$. The Hessian of an unregularized loss of this neural network will thus have number of zero eigenvalues equal to the dimensionality of this group.

What happens when we add a regularizer $\sum_k \lambda ||W_k||_F^2$? The regularized loss is no longer invariant under the action of this large group. Note, however, that the Frobenius norm of a matrix is left invariant under a multiplication by an orthogonal (rotation) matrix. Hence the

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

regularized loss is invariant under the action of the group $\mathrm{O}_{d_{L-1}}(\mathbb{R}) \times \mathrm{O}_{d_{L-2}}(\mathbb{R}) \times \cdots \times \mathrm{O}_{d_1}(\mathbb{R})$. While this is a smaller group, it still provides zero eigenvalues to the Hessian.

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

14.4 Deep networks, square loss

$$L(f(w)) = \sum_{n=1}^{N} (y_n - f(W; x_n))^2 \tag{85}$$

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

1. *Dynamics*

   The dynamics now is

   $$\dot{(W_k)}_{i,j} = -F_k(w) = -\nabla_{W_k} L(W) = 2 \sum_{n=1}^{N} E_n \frac{\partial f}{\partial (W_k)_{i,j}} \tag{86}$$

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

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

   $$\begin{aligned} J(W)_{kk'} &= 2 \sum_{n=1}^{N} (-(\nabla_{W_k} f(W; x_n))(\nabla_{W^{k'}} f(W; x_n)) + E_n \nabla^2_{W_k, W^{k'}} f(W; x_n)) \\ &= -2 \sum_{n=1}^{N} (\nabla_{W_k} f(W; x_n))(\nabla_{W^{k'}} f(W; x_n)), \end{aligned} \tag{87}$$

   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

Theorem 15 *(K. Takeuchi) Let H be a positive integer. Let $h_k = W_k\sigma(h_{k-1}) \in \mathbb{R}^{N_k,n}$ for $k \in \{2,\dots,H+1\}$ and $h_1 = W_1X$, where $N_{H+1} = d'$. Consider a set of H-hidden layer models of the form, $\hat{Y}_n(w) = h_{H+1}$, parameterized by $w = \mathrm{vec}(W_1,\dots,W_{H+1}) \in \mathbb{R}^{dN_1+N_1N_2+N_2N_3+\cdots+N_HN_{H+1}}$. Let $L(w) = \frac{1}{2}\|\hat{Y}_n(w) - Y\|_F^2$ be the objective function. Let w^* be any twice differentiable point of L such that $L(w^*) = \frac{1}{2}\|\hat{Y}_n(w^*) - Y\|_F^2 = 0$. Then, if there exists $k \in \{1,\dots,H+1\}$ such that $N_kN_{k-1} > n \cdot \min(N_k, N_{k+1},\dots,N_{H+1})$ where $N_0 = d$ and $N_{H+1} = d'$ (i.e., overparametrization), there exists a zero eigenvalue of Hessian $\nabla^2 L(w^*)$.*

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

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

14.5 Deep networks, exponential loss

Consider the exponential loss

$$L(f(W)) = \sum_{n=1}^{N} e^{-f(W;x_n)y_n} \tag{88}$$

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 ΔW_k.

1. *Dynamics*

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

$$\dot{W}_k = \sum_{n=1}^{N} y_n[\frac{\partial f(W;x_n)}{\partial W_k}]e^{-y_n f(x_n;W)} \tag{89}$$

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

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

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

$$J(W)_{kk'} = -\sum_{n=1}^{N} e^{-y_n f(W_0;x_n)} \left(\frac{\partial f(W;x_n)}{\partial W_k} \frac{\partial f(W;x_n)}{\partial W_{k'}} - y_n \frac{\partial^2 f(W;x_n)}{\partial W_k \partial W_{k'}} \right) \Bigg|_{W^0}. \tag{90}$$

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

3. *Regularization*

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

$$\dot{W}_k = -\nabla_w (L + \lambda |W_k|^2) = \sum_{n=1}^{n} y_n \frac{\partial f(W;x_n)}{\partial W_k} e^{-y_n f(x_n;W)} - \lambda_k W_k \tag{91}$$

   For compactness of notation, let us define

$$g_k^{(n)} = y_n \frac{\partial f}{\partial W_k} e^{-y_n f(W;x_n)}, \tag{92}$$

   with which we have a transcendental equation for the minimum.

$$\lambda_k (W_k)_{min} = \sum_n g_k^{(n)}. \tag{93}$$

   The negative Hessian of the loss is then

$$H_{kk'} = \sum_n \frac{\partial g_k^{(n)}}{\partial W_{k'}} - \lambda_k \delta_{kk'} \mathbb{I}. \tag{94}$$

14.5.1 Rate of growth of weights

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

$$\rho_k \dot{\rho}_k = \tilde{f}(x) \left(\prod_{i=1}^{k} \rho_i \right) e^{-\prod_{i=1}^{K} \rho_i \tilde{f}(x)} \tag{95}$$

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

$$\dot{\gamma} = \tilde{f}(x)^2 r^2, \qquad r\dot{r} = 2\frac{\log \gamma}{\gamma}, \tag{96}$$

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 (see Equation 46). It gives us the single differential equation

$$\dot{\rho} = \tilde{f}(x) K \rho^{K-1} e^{-\rho_k \tilde{f}(x)}. \tag{97}$$

This implies that for the exponentiated product of weights we have

$$\left(e^{\rho_k \tilde{f}(x)}\right)^{\cdot} = \tilde{f}(x)^2 K^2 \rho^{2K-2}. \tag{98}$$

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

$$\dot{R} = \tilde{f}(x)^{\frac{2}{K}} K^2 \left(\log R\right)^{2-\frac{2}{K}}. \tag{99}$$

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

Interestingly, while the product of weights scales faster than logarithmically, the weights at each layer diverge slower than in the linear network case, as can be seen in the right graph in Figure 2.

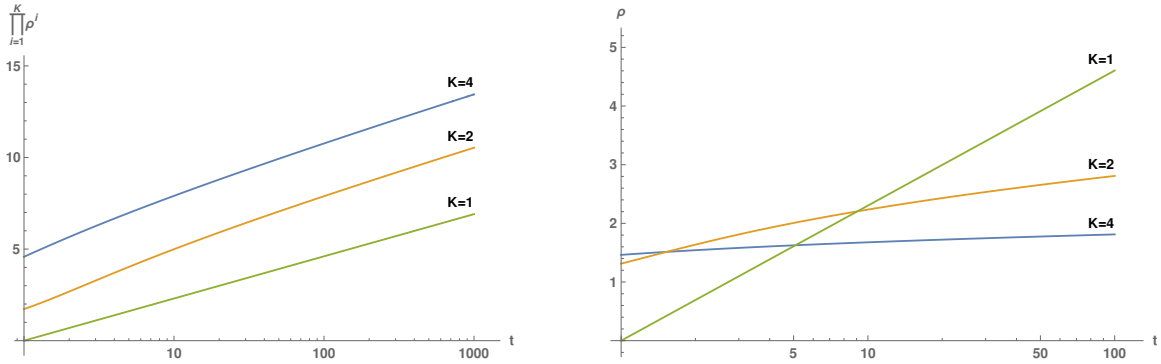


Figure 2: *The left graph shows how the product of weights $\prod_{i=1}^{K}$ scales as the number of layers grows when running gradient descent with an exponential loss. In the 1-layer case we have $\rho = \rho \sim \log t$, whereas for deeper networks the product of norms grows faster than logarithmically. 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.*

14.5.2 Deep Networks, weight normalization

From Equation the dynamics for the normalized weights is

$$\dot{v}_k^{i,j} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \frac{\rho}{\rho_k^2} \left(\frac{\partial \tilde{f}(x_n)}{\partial V_k^{i,j}} - V_k^{i,j} V_k^{a,b} \frac{\partial \tilde{f}(x_n)}{\partial V_k^{a,b}} \right). \tag{100}$$

In Equation (100) 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 $V_k^{i,j}$ which are not changed by gradient descent near the minimum will nevertheless change under this normalized dynamics.

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

$$0 = \sum \alpha_n \left(\frac{\partial \tilde{f}(x_n)}{\partial V_k^{i,j}} - V_k^{i,j} V_k^{a,b} \frac{\partial \tilde{f}(x_n)}{\partial V_k^{a,b}} \right). \tag{101}$$

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

$$V_k^{i,j} = \frac{1}{\hat{f}} \frac{\partial \hat{f}}{\partial V_k^{i,j}}. \tag{102}$$

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.

Consider briefly the multilayer linear and nonlinear cases:

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

$$f(x) = \sum_{i,jk} O_i V_2^{i,j} V_1^{j,k} x^k \tag{103}$$

gives

$$\frac{\partial \tilde{f}(x)}{\partial V_1^{q,h}} = \sum_{i,j,k} O^i V_2^{i,j} \delta^{j,q} \delta^{k,h} x^k = \sum_i O^i V_2^{i,q} x^h \tag{104}$$

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

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

Nonlinear case Consider a nonlinear network $f(x) = O\sigma(W_2\sigma(W_1x))$ where O is as before. Let us normalize weights as before. We use Lemma 3.1 in [7] to rewrite f as

$$f(x) = OD_2W_2D_1W_1x \tag{105}$$

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 105 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.

Remarks

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

14.5.3 Reparametrizations and normalized weights

- *We introduce a reparametrization for GD under the exponential loss which yields a continuous dynamics equivalent to the standard GD dynamics. The reparametrization is similar to weight normalization; the latter however is* **not** *tthe same dynamics.*
- *Our formulation converges to the normalized vector obtained by running GD on w and normalizing it at the end.*

For Deep Networks the results in section can be extended to weight matrices

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

Thus the generalization of Equation 39 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 $\partial \tilde{f} / \partial V_k$ we we gain additional ratios of norms:

$$S_k^{i,j,a,b} \frac{\partial f}{\partial W_k^{ab}} = \frac{\rho}{\rho_k} S_k^{i,j,a,b} \frac{\partial \tilde{f}}{\partial V_k^{a,b}} \tag{106}$$

14.5.4 Reparametrized dynamics is the same as the standard W dynamics

As described earlier

$$\dot{v} = S\dot{w} = \sum_{n=1}^{N} e^{-\rho v^T x_n} \frac{x_n - v v^T x_n}{\rho} \tag{107}$$

and

$$\dot{\rho} = \sum_{n=1}^{N} e^{-\rho v^T x_n} v^T x_n. \tag{108}$$

Let us now compute the dynamics of the weights w from $\dot{w} = \dot{\rho}v + \rho\dot{v}$. We obtain

$$\dot{\rho}v + \rho\dot{v} = \sum_{n=1}^{N} e^{-\rho v^T x_n)}(v^T x_n v + x_n - vv^T x_n) = \sum_{n=1}^{N} e^{-\rho v^T x_n} x_n. \tag{109}$$

It turn 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

$$\dot{v} = S\dot{w} = \sum_{n=1}^{N} e^{-\rho f(x_n;v)} \frac{x_n - vv^T x_n}{\rho} \tag{110}$$

and

$$\dot{\rho} = \sum_{n=1}^{N} e^{-\rho \tilde{f}(x_n)} \tilde{f}(x_n) \tag{111}$$

Integration by parts Consider the normalized solution at time T. We want to show that using $w(T) = \int_0^T \dot{w(t)}$ computed from the unconstrained gradient we prove $\frac{w(T)}{\rho(T)} = v(T)$. We use integration by parts to compute $\int_0^T \frac{d}{dt} v dt = \int_0^T S\dot{w} dt$. We obtain $Sw|_0^T - \int_0^T \frac{dS}{dt} w dt$ yielding (since $Sw = 0$, see later)

$$\int_o^T v \frac{\partial vv^T}{\partial t} = \int_0^T (\dot{v} - vv^T\dot{v}). \tag{112}$$

Since $\dot{v}$ is orthogonal to v, Equation 112 gives

$$\int_0^T \dot{v} = v(T), \tag{113}$$

as claimed.

Remarks

- For the square loss, if the network is linear, there is convergence to the minimum norm solution by GD with 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 $\frac{w}{||w||} = \frac{\rho v}{\rho} = v$.
- 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 $\frac{f}{\rho} = \frac{\rho \tilde{f}}{\rho} = \tilde{f}$. With zero-norm initial conditions the norms of the K layers are approximately equal and $\rho = \rho_1^K$. 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. 17) and would be happy to see you and

14.5.5 Weight normalization does not yield the same dynamics as standard gradient descent

Weight normalization [40] was claimed to be a reparametrization of the weight vectors in a neural network as follows

$$w = \frac{g}{||v||} v. \tag{114}$$

There are similarities between our normalized reparametrization of section 14.5.3 and classical weight normalization [40] as follow: $\tilde{w} = \frac{v}{||v||}$ and $\rho = \rho\, g$ (we call our v here $\tilde{w}$ to distinguish from the v of classical weight normalization). However the gradient descent equations are different in the two cases.

For the $\rho, \tilde{w}$ parametrization, the dynamics is given by

$$\frac{\partial L}{\partial \rho} = \tilde{w}^T \frac{\partial L}{\partial w} \tag{115}$$

and

$$\frac{\partial L}{\partial \tilde{w}} = S \frac{\partial L}{\partial w} \tag{116}$$

with

$$S = \frac{I - \tilde{w}\tilde{w}^T}{\rho} \tag{117}$$

The dynamics above in ρ and $\tilde{w}$ is equivalent to the dynamics of w using $\dot{w} = \rho\dot{\tilde{w}} + \dot{\rho}\tilde{w}$ – which follows from the definition $w = \rho\tilde{w}$.

The gradient descent equations for the g, v parametrization are

$$\frac{\partial L}{\partial g} = \frac{v}{||v||} \frac{\partial L}{\partial w} \tag{118}$$

and

$$\frac{\partial L}{\partial v} = \frac{g}{||v||} S' \frac{\partial L}{\partial w} \tag{119}$$

with

$$S' = I - \frac{vv^T}{||v||^2} = \frac{v^T v - vv^T}{v^T v} \tag{120}$$

It is easy to show that the implied dynamics for w is different from the standard dynamics.

14.5.6 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 $x_{new} = (x - E[x])[Cov(x)^{-1/2}]$.

Remarks

- Consider WN and BN in terms of coordinate transformations. Then, neglecting biases for simplicity of notation, WN uses a transformation A, defined as $w_{new} = A$ such that $||w_{new}|| = 1$. On the other hand the ideal BN (as described in the original paper) uses a transformation C, defined as $w_{new}x = Cwx$, such that $w_{new}w_{new}^T = \frac{1}{D}I$, where D is the dimensionality of w_{new}. Thus the trace of the matrix $w_{new}w_{new}^T$ is one implying that $||w_{new}|| = 1$. In practice, the current implementation of BN only enforces that the diagonal of $w_{new}w_{new}^T$ should be equal to $\frac{1}{D}I$. 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 $[Cov(x)]x_{new} = x$. Assume for simplicity of notation that $E[x] = 0$. One of several iterative techniques is the *Jacobi method*, which provides a solution x_{new} as

$$x_{new}^{(t+1)} = D^{-1}(x - Rx_{new}^{(t)}) \tag{121}$$

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

15 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 $||Tx - Ty|| \leq ||x - y||, \quad \forall x, y \in X$. Then the following result is classical ([44, 32])

Theorem 16 *[44] Let X be a strictly convex normed space. The set of fixed points of a non-expansive mapping $T : C \to C$ with C a closed convex subset of X is either empty or closed and convex. If it is not empty, it contains a unique element of smallest norm.*

In our case $T = (I - \gamma(t)\nabla_w L(f))$. To fix ideas, consider gradient descent on the square loss. As discussed later and in several papers, the Hessian of the loss function ($E = L(f(\cdot))$) of a deep networks with ReLUs has eigenvalues bounded from above (see for instance [51] and [52]) – 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 15.1. If we assume that the minimum is global and that there are effectively no local minima – in the sense that SGD does not "see" them – 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 [32]) of the form

$$x_{t+1} = (1 - s_t)Tx_t \tag{122}$$

where the sequence s_t satisfies conditions such as $\lim_{n\to\infty} s_n = 0$ and $\sum_{n=1}^{\infty} = \infty$ [5].

In particular, the following holds

Theorem 17 *[32] 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 $k_n = 1 - \frac{1}{n^a}$ and $0 < a < 1$ converges to the fixed point of T with minimum norm.*

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

$$w_{n+1} - w_n = -(1 - \lambda_n)\gamma_n \nabla_w L(f) - \lambda_n w_n \tag{123}$$

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

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

Remark

[5] Notice that these iterative procedures are often part of the numerical implementation (see [53] 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.

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. An even higher level view of several different optimization techniques including the case of regression, is to regard them as instances of Halpern iterations. 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 3). In the classification case, the minimum norm solution corresponds to a maximum margin solution for the normalized network. Globally optimal generalization is not guaranteed but generalization bounds such as Equation 14 are locally optimized.

15.1 $T = I - \nabla L(f)$ is a non-expanding operator

Definition 18 *A function $g(\cdot) : \mathbf{R}^n \to \mathbf{R}$ is L-smooth if its gradients are Lipschitz continuous that is $\forall x, y \in \mathbf{R}^n$*

$$||\nabla g(x) - \nabla g(y) \leq L||x - y|| \tag{124}$$

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

Lemma 19 *A sufficient condition for L-smoothness is that the eigenvalues of $H = \nabla^2 g$ are bounded from above and from below by L.*

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

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

Notice that we have

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

Applying the Cauchy-Schwarz theorem and the assumption we obtain

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

which completes the proof.

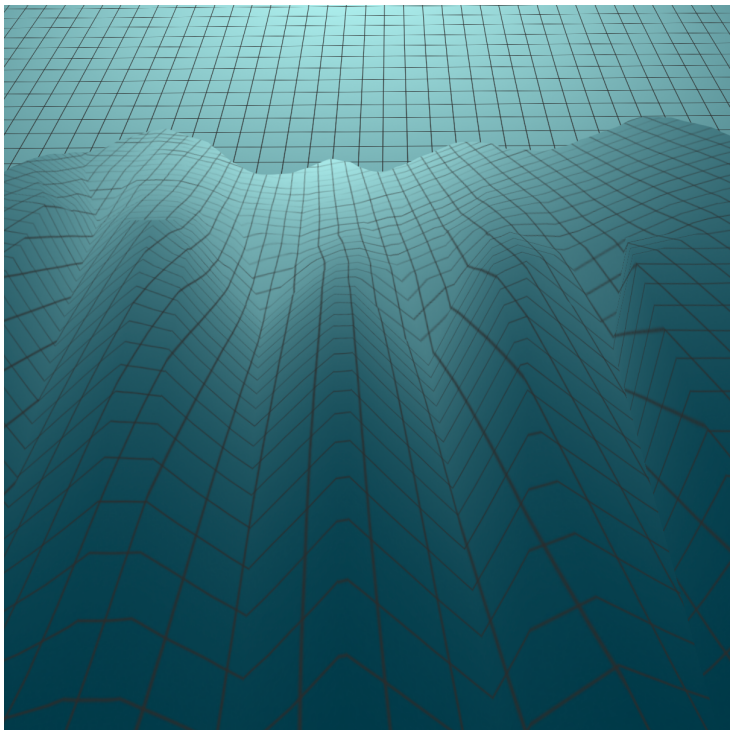


Figure 3: *Landscape of the empirical loss with unnormalized weights.* Suppose the empirical loss at the water level in the figure is $\leq \epsilon$. Then there are various global minima each with the same loss and different minimum norms. Because of the universality of deep networks from the point 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.

Lemma 20 *$\nabla g(x) = 0$ if and only if $x \in \mathbf{R}^n$ is a fixed point of the operator $T^\gamma : \mathbf{R}^n \to \mathbf{R}^n$ with $T^\gamma x = x - \gamma \nabla g(x)$, that is $T^\gamma x = x$ for non-zero γ.*

Theorem 21 *For convex $g(x)$, the operator T^γ defined as $T^\gamma x = x - \gamma \nabla g(x)$ is non-expanding that is*

$$||T^\gamma x - T^\gamma y|| \leq ||x - y|| \tag{125}$$

Proof The standard proof uses the mean value theorem to write

$$T^\gamma x - T^\gamma y = (x - y)(I - \gamma \nabla^2 g(z)) \tag{126}$$

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

$$||T^\gamma x - T^\gamma y|| \leq ||x - y||\ ||(I - \gamma \nabla^2 g(z))||. \tag{127}$$

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

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

Theorem 22 *Assume that gradient descent starts from w_0 with g which is L-smooth*

$$w_{t+1} = w_t - \gamma \nabla g(w_t) \tag{128}$$

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

$$||w_{t+1} - w_*||^2 \leq (1 - \gamma L)^{2(t+1)} ||w_0 - w_*||^2 \tag{129}$$

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

$$||w_{t+1} - w_*||^2 \leq (1 - \gamma \mu)^{t+1} ||w_0 - w_*||^2 \tag{130}$$

Proof (a) L-smoothness gives us that

$$L||w_t - w_*||^2 \geq -(\nabla g(w_t) - \nabla g(w_*), w_t - w_*) \geq -L||w_t - w_*||^2. \tag{131}$$

We then have

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

where we have used the fact that $\nabla g(w_*) = 0$ in third equality, the result (131) in the first inequality and finally the definition of L-smoothness.

(b) μ-strong convexity gives us

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

It follows similarly to the previous case that

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

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

16 Summarizing Theorems

Most of the previous results can be summarized in the following

Theorem 23 *In the setting of separable data, deep RELU network f and exponential-type loss the following statements hold.*

1. *There are no nontrivial critical points of the gradient that are data-separating.*
2. *The margin generalization bound is optimized by maximizing the margin of $\tilde{f}$ under the constraint of unit p-norm of the weight matrices V_k of $\tilde{f}$ that is*

$$\arg\max_{||V_k||_p=1, \quad \forall k=1,\cdots,K} \min_n y_n \tilde{f}(x_n). \tag{132}$$

3. *Gradient descent methods with unit norm constraints on exponential type losses yield margin maximization with unit norm constraint, that is they yield*

$$\min_{\rho_k, V_k} \sum_i^N e^{-y_i \rho \tilde{f}(x_i)} \quad s.t. ||V_k||_p = 1 \tag{133}$$

   where $W_k = \rho_k V_k$, $||V_k||_p = 1$, $\rho = \prod_k \rho_k$.

4. *The two main families for gradient descent with unit norm constraint are*
   - *Lagrange multiplier methods with the sequence $\lambda_k(t)$ such that $\lim_{t\to\infty} ||V_k||_p = 1$:*
   - *Tangent gradient methods*
5. *Weight normalization and batch normalization are tangent gradient methods that performs margin maximization under explicit L_2 unit norm constraint.*
6. *Standard gradient descent followed by L_2 normalization, that is*

$$\dot{W}_k^{i,j} = -\frac{\partial L}{\partial W_k^{i,j}} = \sum_{n=1}^N y_n \frac{\partial f(x_n; w)}{\partial W_k^{i,j}} e^{-y_n f(x_n; W)} \tag{134}$$

   followed by $\tilde{f}(T) = \frac{f(T)}{\rho(T)}$, with $\rho = \prod_k \rho_k$ and $\rho_k = ||W_k||_2$ has a similar, but not identical, dynamics and the same critical points as the tangent gradient methods, that is it performs margin maximization under implicit unit L_2 norm constraint.

Proof Part (1) follows from using the structural lemma on the critical points of the gradient descent equations: 0 values of the gradient imply $f = 0$ apart from global minima at infinity, assuming $y_n f(x_n) > 0, \forall n$. Part (2) follows directly from the generalization bounds and the homogenity property of f.

For part (3) we need to prove that gradient descent on an exponenial loss is margin maximizing. For this we follow the proof of Part a of Theorem 2.1 in [38].

For Part (4) we refer to [39] .

Part (5) is obtained by looking at the weight normalization dynamics and realizing that the dynamics of the vector contains S.

Part (6) results from two steps. First we show that the dynamics of Equation 28 is the same as the dynamics of

$$\dot{\rho_k} = V_k^T \dot{W_k} \qquad \dot{V_k} = \frac{S_k}{\rho_k} \dot{W_k}. \tag{135}$$

The second step is to recognize that a tangent transformation on $\dot{V_k}$, written as $S\dot{V_k}$, does not change the dynamics since $S\dot{V_k} = \dot{V_k}$.

Remarks

- The results in the various parts of the theorem apply only to the gradient flow, that is the continuous version of the discrete algorithms used in practice.
- The approach described here is for classification and not for regression.
- Step 1 uses the generalization bound for classification. Obviously what matters for (binary) classification is $\tilde{f}$ and not f.
- Step 5 is a sanity check on the natural guess that $\tilde{f}(T) = \frac{f(T)}{\rho(T)}$ is the correct solution of gradient descent with unit norm constraint on $v = \frac{w}{||w||_2}$.

17 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 [6] 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.*

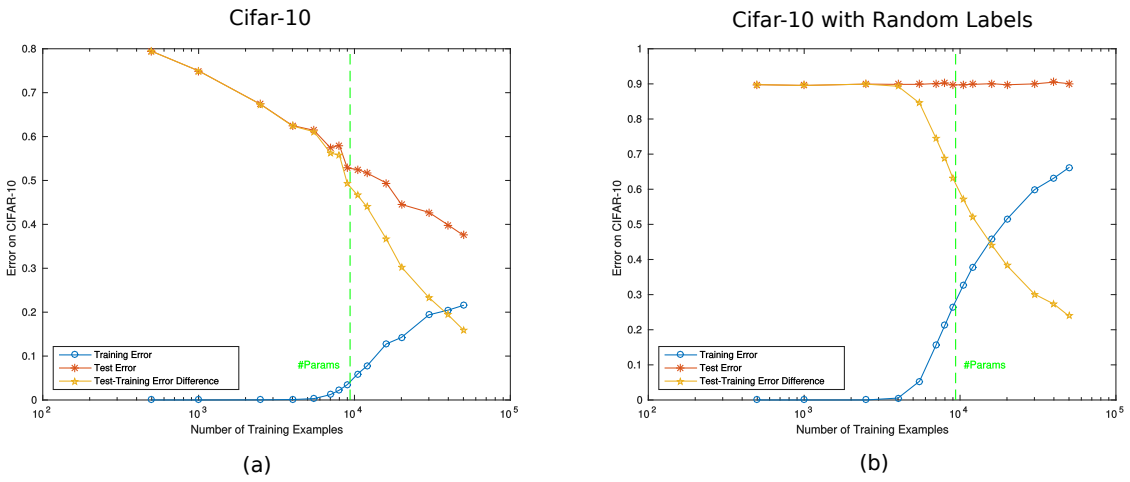


Figure 4: *Generalization for Different number of Training Examples.* (a) Generalization error in CIFAR and (b) generalization error in CIFAR with random labels. The DNN was trained by minimizing the cross-entropy loss and it is a 5-layer convolutional network (*i.e.*, no pooling) with 16 channels per 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 4 and 5.

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 6.

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

- The training error will go back to zero after each sequence of GD.
- Any small perturbation of the optimum W_0 will be corrected by the GD dynamics to push back the non-degenerate weight directions to the original values. Since the components of the weights in the degenerate directions are in the null space of the gradient, running

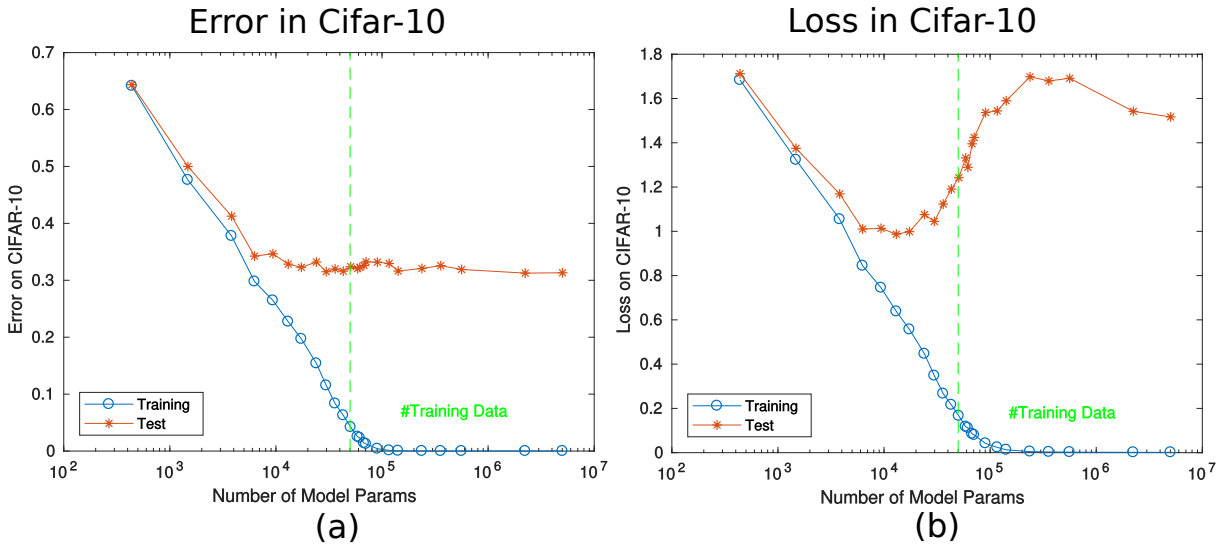


Figure 5: *Expected error in CIFAR-10 as a function of number of neurons.* The DNN is the same as in Figure 4. Batch normalization was also used. (a) Dependence of the expected error as the number of parameters increases. (b) Dependence of the cross-entropy risk as the number of parameters increases. 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 10. 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 10 with Figure 7). 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

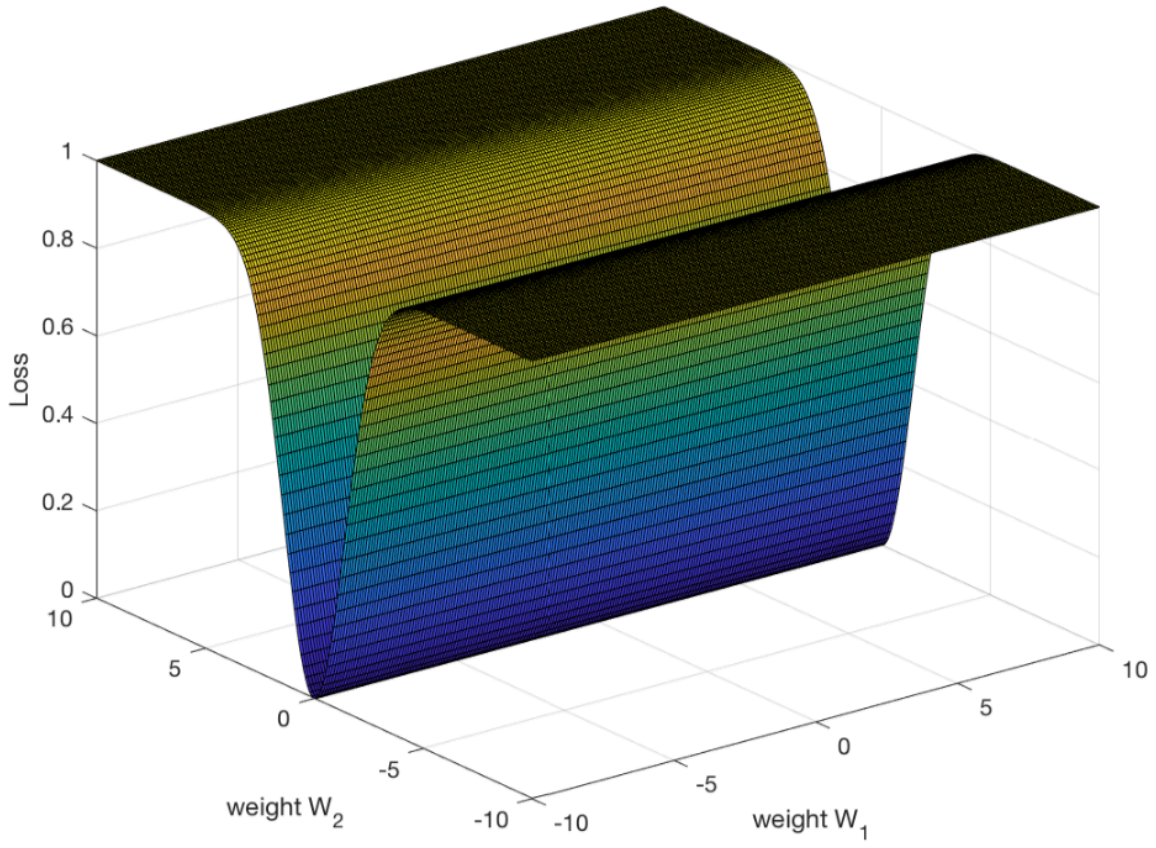
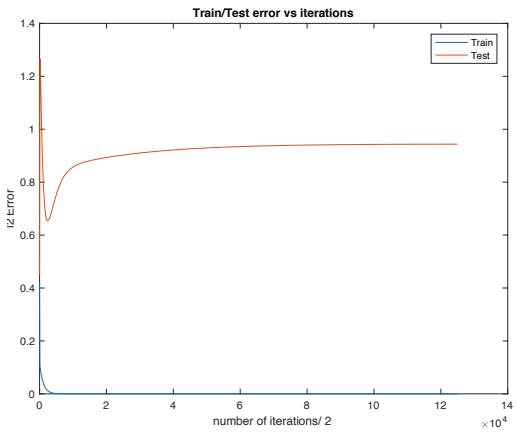


Figure 6: *A quadratic loss function in two parameters w_1 and w_2. The minimum has a degenerate Hessian with a zero eigenvalue. In the proposition described in the text, it represents the "generic" situation in a small neighborhood of zero minimizers with many zero eigenvalues – and a few positive eigenvalues – 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 λ. Thus a specific "early stopping" would be better than no stopping. The corresponding classification error, however, may not show overfitting.

Figure 11 shows the behavior of the loss in CIFAR in the absense of perturbations. This should be compared with Figure 7 which shows the case of an overparametrized linear network under quadratic loss corresponding to the multidimensional equivalent of the degenerate situation



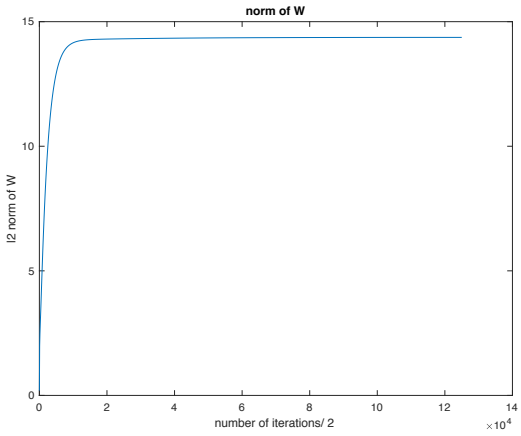


Figure 7: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 6. The feature matrix* $\phi(X)$ *is a polynomial with degree 30. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training point are* 9 *while the number of test points are* 100*. The training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. The weights were not perturbed in this experiment. The* L_2 *norm of the weights is shown on the right. Note that training was repeated 30 times and what is reported in the figure is the average train and test error as well as average norm of the weights over the 30 repetitions. There is overfitting in the test error.*

of Figure 6. The nondegenerate, convex case is shown in Figure 9.

Figure 12 shows the testing error for an overparametrized linear network optimized under the square loss.This is a special case in which the minimum norm solution is theoretically guaranteed by zero inital conditions without NMGD.

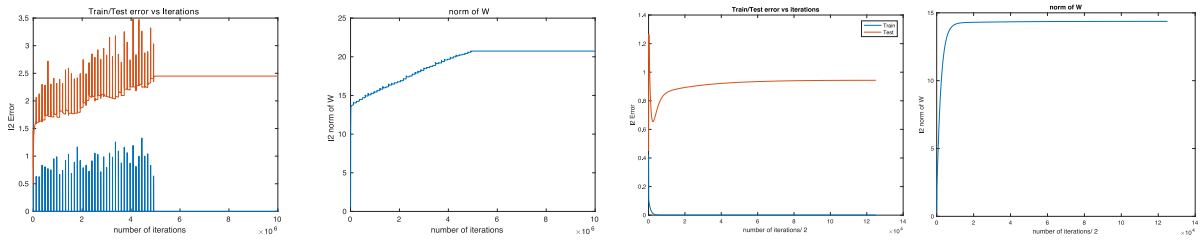


Figure 8: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 6. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points is* 9 *while the number of test points is* 100*. For the first pair of plots the feature matrix* $\phi(X)$ *is a polynomial with degree 39. For the first pair had points were sampled to according to the Chebyshev nodes scheme to speed up training to reach zero on the train error. Training was done with full Gradient Descent step size* 0.2 *for* $10,000,000$ *iterations. Weights were perturbed every* $120,000$ *iterations and Gradient Descent was allowed to converge to zero training error (up to machine precision) after each perturbation. The weights were perturbed by addition of Gaussian noise with mean* 0 *and standard deviation* 0.45*. The perturbation was stopped half way at iteration* $5,000,000$*. The* L_2 *norm of the weights is shown in the second plot. Note that training was repeated 29 times figures reports the average train and test error as well as average norm of the weights over the repetitions. For the second pair of plots the feature matrix* $\phi(X)$ *is a polynomial with degree 30. Training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. The* L_2 *norm of the weights is shown in the fourth plot. Note that training was repeated 30 times figures reports the average train and test error as well as average norm of the weights over the repetitions. The weights were not perturbed in this experiment.*

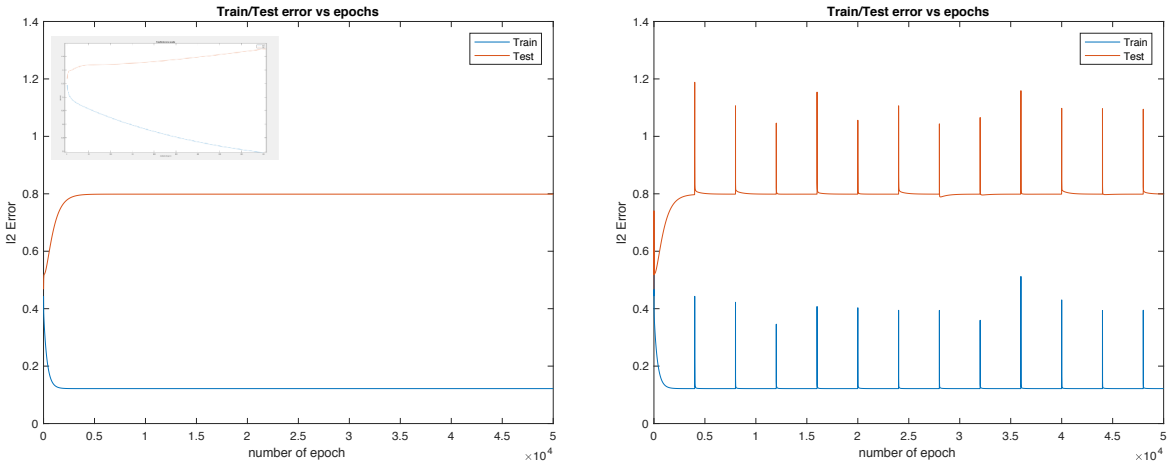


Figure 9: *The graph on the left shows training and testing loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) in the nondegenerate quadratic convex case. The feature matrix* $\phi(X)$ *is a polynomial with degree 4. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with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

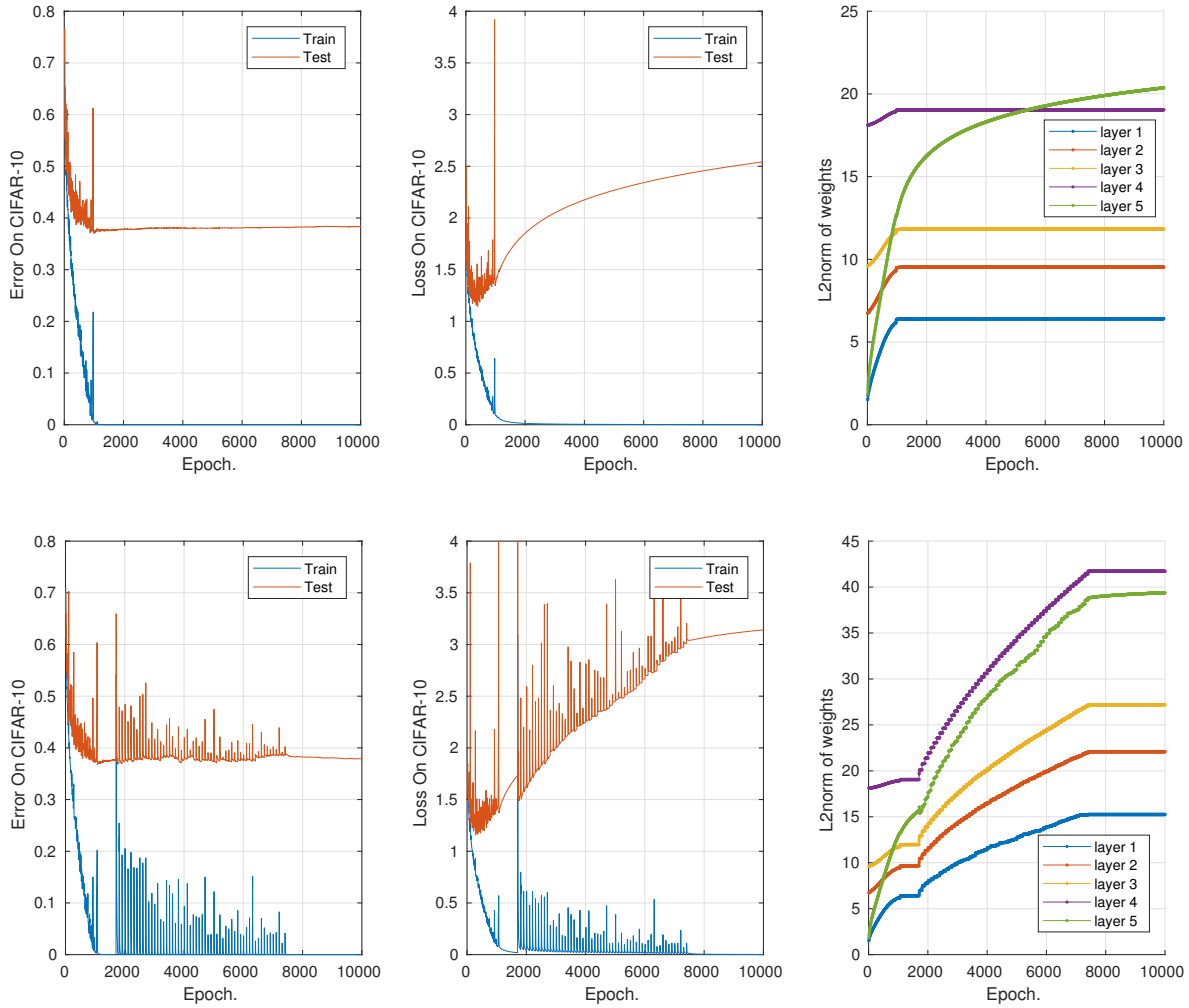


Figure 10: *We train a 5-layer convolutional neural networks on CIFAR-10 with Gradient Descent (GD) on cross-entropy loss with and without perturbations. The main results are shown in the 3 subfigures in the bottom row. Initially, the network was trained with GD as normal. After it reaches 0 training 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. It is an open question of why the norm of the layers, apart the last one, stop increasing. The network has 4 convolutional layers (filter size* 3×3*, stride 2) and a fully-connected layer. The number of feature maps (i.e., channels) in hidden layers are 16, 32, 64 and 128 respectively. Neither data augmentation nor regularization is performed.*

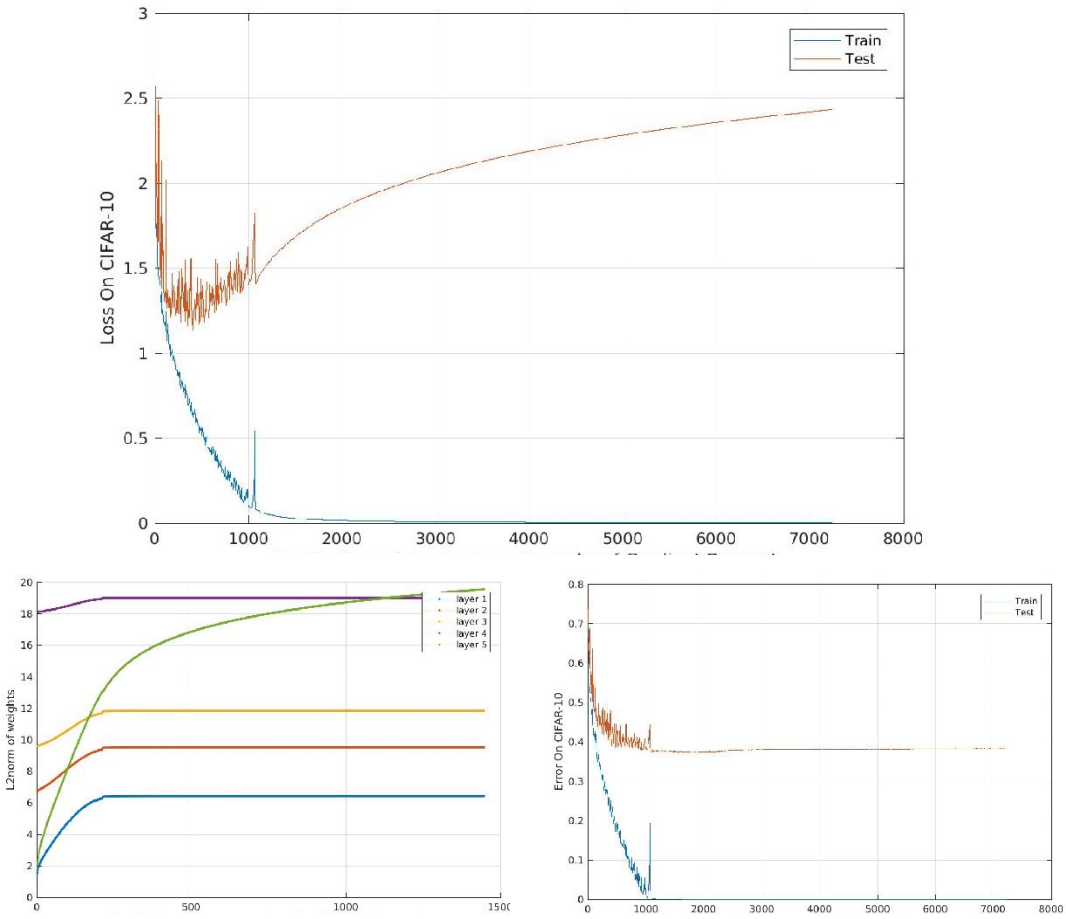


Figure 11: *Same as Figure 4 but without perturbations of weights. Notice that there is some overfitting in terms of the testing loss. Classification however is robust to this overfitting (see text).*

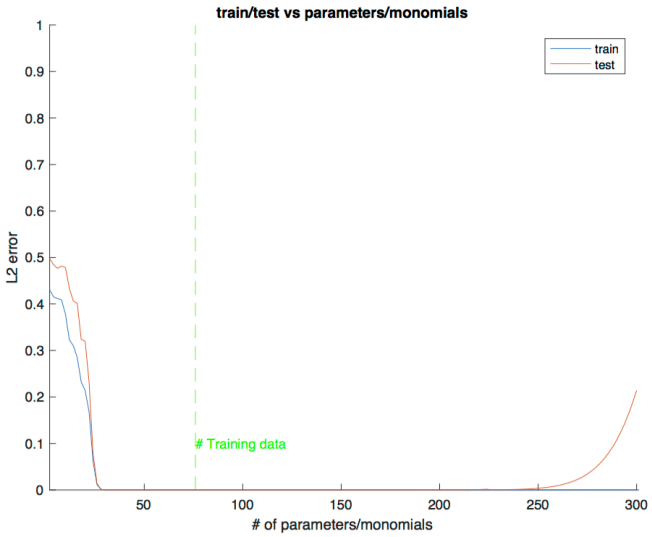


Figure 12: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\phi(X)$*) with a degenerate Hessian of the type of Figure 6. The feature matrix is a polynomial with increasing degree, from 1 to 300. The square loss is plotted vs the number of monomials, that is the number of parameters. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points where* 76 *and the number of test points were* 600*. The solution to the over-parametrized system was the minimum norm solution. More points where sampled at the edges of the interval* $[-1, 1]$ *(i.e. using Chebyshev nodes) to avoid exaggerated numerical errors. The figure shows how eventually the minimum norm solution overfits.*