CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 073 **January 15, 2018**

Theory of Deep Learning III: explaining the non-overfitting puzzle

by

T. Poggio[†], K. Kawaguchi [†⊤], Q. Liao[†], B. Miranda[†], L. Rosasco[†]

with

X. Boix[†], J. Hidary[††], H. Mhaskar[◇],

[†]Center for Brains, Minds and Machines, MIT

[⊤]CSAIL, MIT

[††]Alphabet (Google) X

[◇]Claremont Graduate University

Abstract: A main puzzle of deep networks revolves around the absence of overfitting despite large overparametrization and despite the large capacity demonstrated by zero training error on randomly labeled data. In this note, we show that the dynamics associated to gradient descent minimization of nonlinear networks is topologically equivalent, near the asymptotically stable minima of the empirical error, to linear gradient system in a quadratic potential with a degenerate (for square loss) or almost degenerate (for logistic or crossentropy loss) Hessian. The proposition depends on the qualitative theory of dynamical systems and is supported by numerical results.

Our main propositions extend to deep nonlinear networks two properties of gradient descent for linear networks, that have been recently established (*1*) to be key to their generalization properties:

1. Gradient descent enforces a form of implicit regularization controlled by the number of iterations, and asymptotically converges to the minimum norm solution for appropriate initial conditions of gradient descent. This implies that there is usually an optimum early stopping that avoids overfitting of the loss. This property, valid for the square loss and many other loss functions, is relevant especially for regression.
2. For classification, the asymptotic convergence to the minimum norm solution implies convergence to the maximum margin solution which guarantees good classification error for "low noise" datasets. This property holds for loss functions such as the logistic and cross-entropy loss independently of the initial conditions.

The robustness to overparametrization has suggestive implications for the robustness of the architecture of deep convolutional networks with respect to the curse of dimensionality.

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

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 finally emerging. It covers the following questions: 1) *representation power* — what types of functions can deep neural networks (DNNs) represent and under which conditions can they be more powerful than shallow models; 2) *optimization* of the empirical loss — can we characterize the minima obtained by stochastic gradient descent (SGD) on the non-convex empirical loss encountered in deep learning? 3) *generalization* — why do the deep learning models, despite being highly over-parameterized, still predict well? Whereas there are satisfactory answers to the the first two questions (see for reviews (*2, 3*) and references therein), the third question is still triggering a number of papers (see among others (*4–8*)) with a disparate set of partial answers. In particular, a recent paper titled "Understanding deep learning requires rethinking generalization" (*9*) claims that the predictive properties of deep networks require a new approach to learning theory. This paper is motivated by observations and experiments by several authors, in part described in (*8*), where a more complex set of arguments is used to reach similar conclusions to this paper. It shows that the generalization properties of linear networks described in (*1*) and (*10*) can be extended to deep networks.

Using the classical theory of ordinary differential equations, our approach replaces a potentially fundamental puzzle about generalization in deep learning with elementary properties of gradient optimization techniques. This seems to explain away the generalization puzzle of today's deep networks and to imply that there is no fundamental problem with classical learning theory. The paper is at the level of formal rigor of a physicist (not a mathematician).

In this paper we focus on gradient descent (GD) rather than stochastic gradient descent (SGD). The main reason is simplicity of analysis, since we expect the relevant results to be valid in both cases. Furthermore, in "easy" problems, such as CIFAR, one can replace SGD with GD without affecting the empirical results, given enough computational resources. In more difficult problems, SGD, as explained in (*11*), not only converges faster but also is better at selecting global minima vs. local minima.

Notice that in all computer simulations reported in this paper, we turn off all the "tricks" used to improve performance such as data augmentation, weight decay etc. in order to study the basic properties of deep networks optimized with the SGD or GD algorithm (we keep however batch normalization in the CIFAR experiments). We also reduce in some of the experiments the size of the network or the size of the training stet. As a consequence, performance is not state of the art, but optimal performance is not the goal here (in fact we achieve very close to state-of-the-art performance using standard number of parameters and data and the usual tricks). In our experiments we used the cross entropy loss for training, which is better suited for classification problems (such as for CIFAR), as well as the square loss for regression (see SI 18).

Throughout the paper we use the fact that that a polynomial network obtained by replacing each ReLUs of a standard deep convolutional network with its univariate polynomial approximation shows all the properties of deep learning networks. As a side remark, this result shows that RELU activation functions can be replaced by smooth activations (which are not *positively homogeneous*) without affecting the main properties of deep networks.

2 Overfitting Puzzle

Classical learning theory characterizes generalization behavior of a learning system as a function of the number of training examples n. From this point of view deep learning networks behave as expected: the more training data, the smaller the test error, as shown in the left of Figure 1. Other aspects of their learning curves seem less intuitive but are also easy to explain. Often the test error decreases for increasing n even when the training error is zero (see Figure 1 left side). As noted in (*1*) and (*10*), this is because the classification error is reported, rather than the loss minimized for training, e.g. cross-entropy. It is also clear that deep networks do show *generalization*, technically defined as convergence for $n \to \infty$ of the training error to the expected error. The left and the right plots in Figure 1 indicates generalization for large n. This is expected from previous results such as Bartlett's (*12*) and especially from the stability results of Recht (*13*).

The property of generalization, though important, is however of academic importance only, since deep networks are typically used in the overparametrized case, that is, in a regime in which several classical generalization bounds are not valid. The real puzzle in this regime– and the focus of this paper – is the apparent lack of overfitting, despite usually large overparametrization, that is many more parameters than data. The same network which achieves zero training error for randomly labeled

data (right plot in Figure 1), clearly showing large capacity, does not have any significant overfitting in classifying normally labeled data (left plot in Figure 1).

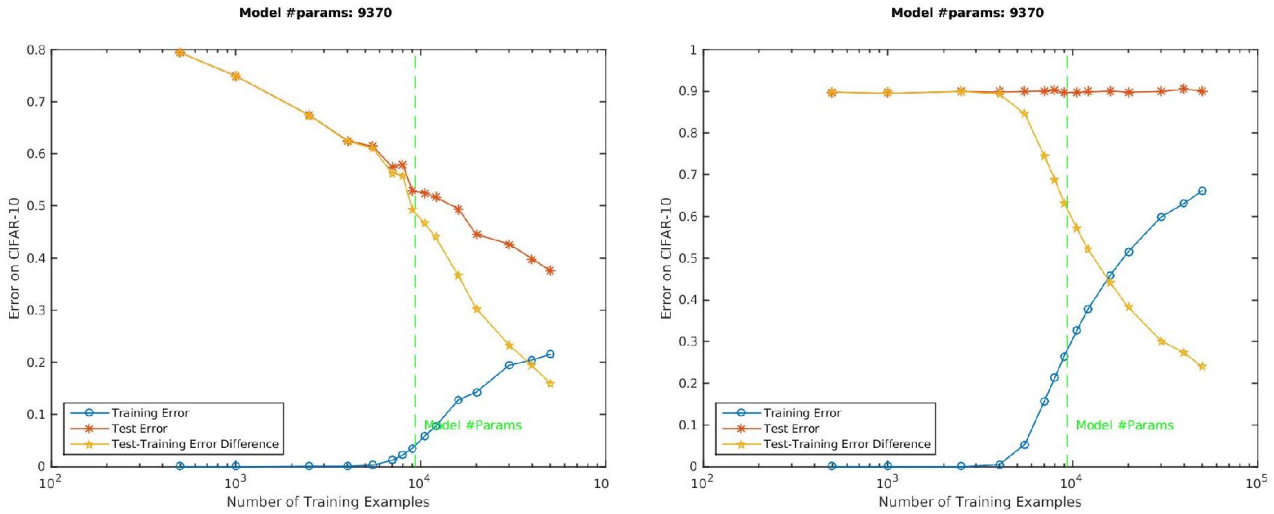


Figure 1: *The figure (left) shows the behavior the classification error for a deep network with RELU activations trained on subsets of the CIFAR database by minimizing the crossentropy loss. The figure on the right shows the same network trained on subsets of the CIFAR database in which the labels have been randomly scrambled. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer , resulting in only* $W \approx 10000$ *weights instead of the typical* $300,000$*. Neither data augmentation nor regularization is performed.*

As a general warning it should be clear that overparametrization cannot be decided simply in terms of the exact number of parameters vs number of data. As shown in SI 6.2.2, there is trivial and nontrivial parametrization. The number of parameters is just a rough guideline to overparametrization.

3 Linear Networks

In the special case of linear models, an explanation for the lack of overfitting has been recently proposed in (*1*) and (*10*). Two main properties are suggested to be important: the difference between classification error and loss, and the implicit regularization properties of gradient descent methods. Gradient descent iteratively controls the complexity of the model and the regularization parameter is the inverse of the number of iterations. As the number of iterations increases, less regularization is enforced, and in the limit the minimum norm solution is selected. The latter is the maximum margin solution ensuring good classification error for separable problems. Though this is valid for several different loss functions, the maximum margin solution has somewhat different properties in the case of square loss vs. logistic or cross-entropy loss.

3.1 Gradient descent methods yield implicit regularization

This section is based on (*10*).

In linear models with square loss, it is known that, for appropriate initial conditions, GD provides *implicit regularization* controlled by the number of iterations j (for a fixed gradient step) as $\lambda \propto \frac{1}{j}$ (see (*14*) and section 6.1 of SI for a summary).

As the number of iterations increases, the equivalent, implicit λ decreases and a larger set of possible solutions is explored (in analogy to Tikhonov regularization with a decreasing regularization parameter, see Figure 3). For overparametrized models one expects overfitting in the loss: the asymptotic solution for $t \to \infty$ – which corresponds to $\lambda \to 0$ – converges in the limit to the minimum norm solution (for square loss this is the pseudoinverse), which is not a regularized solution itself, but the limit of a regularized solution (see for instance section 5 in (*9*)). This overfitting in the loss can be avoided, even without explicit regularization, by appropriate early stopping, as shown in Figure 15.

3.2 Implicit regularization yields margin maximization

A direct consequence of the implicit regularization by gradient descent is that the solution obtained after sufficiently many iterations is the minimum norm solution. The proof holds in the case of linear networks for a variety of loss functions and in particular for the square loss ((*9*) and SI section 6.2.1 for a short summary).

Convergence to the minimum norm solution is suggested in (*1*) to be the main reason behind the lack of overfitting in classification. Indeed, the minimum norm solution is known to maximize classification margin, which in turn ensures good classification error for "low noise" data sets, which, informally, exactly the data sets where the classes are separated by a nicely behaving margin. The precise conditions on the data that imply good classification accuracy are related to Tsybakov conditions (see for instance (*15*)).

In the case of the logistic and crossentropy loss, properties of the convergence to the maximum margin solution which is well known for linear networks with the square loss, should be qualified further, since proofs such as (*9*) need to be modified. Lemma 1 in (*1*) shows that for loss functions such as cross-entropy, gradient descent on linear networks with separable data converges *asymptotically to the max-margin solution with any starting point* w_0*, while the norm* $||w||$ *diverges* in agreement with Lemma 6. Furthermore, (*1*) prove that the *convergence to the maximum margin solution is very slow and only logarithmic in the convergence of the loss itself*. This explains why optimization of the logistic loss helps decrease the classification error in testing, even after the training classification error is zero and the training loss is very small, as in Figure 1.

Remarks

- An alternative path to prove convergence to a maximum margin solution, connecting it directly to *flat minima* and assuming a classification setting with a loss such as the logistic loss, is described in (*8*) and in Lemma 6 of the SI: *SGD maximizes flatness of the minima, which is equivalent to robust optimization, which maximizes margin*.
- The crossentropy loss is an upper bound for the classification error.

4 Deep networks at global minima are topologically equivalent to linear networks

Qualitative properties of the dynamical system

To be able to use the linear results also for deep nonlinear networks, we introduce here classical properties of dynamical systems defined in terms of the gradient of a Lyapunov function such as the training loss.

The gradient dynamical system corresponding to training a deep network with gradient descent with a loss function L is

$$\dot{W} = -\nabla_W L(W) = -F(W). \tag{1}$$

The simplest example is $L(w) = \sum_1^n (f_w(x_i) - y_i)^2$ where f is the neural network parametrized by the weights and n is the number of example pairs x_i, y_i used for training.

We are interested in the qualitative behavior of the dynamical system near stable equilibrium points W^* where $F(W^*) = 0$. One of the key ideas in stability theory is that the qualitative behavior of an orbit under perturbations can be analyzed using

the linearization of the system near the orbit. Thus the first step is to linearize the system, which means considering the Jacobian of F or equivalently the Hessian of L at W^*, that is

$$(\mathbf{H}L)_{ij} = \frac{\partial^2 L}{\partial w_i \partial w_j} \tag{2}$$

.

We obtain

$$\dot{W} = -HW, \tag{3}$$

where the matrix H, which has only real eigenvalues (since it is symmetric), defines in our case (by hypothesis we do not consider unstable critical points) two main subspaces:

- the stable subspace spanned by eigenvectors corresponding to negative eigenvalues
- the center subspace corresponding to zero eigenvalues.

The center manifold existence theorem (*16*) states that if F has r derivatives (as in the case of deep polynomial networks) then at every equilibrium W^* there is a $\mathbf{C}^r$ stable manifold and a $\mathbf{C}^{r-1}$ center manifold which is sometimes called *slow manifold*. The center manifold emergence theorem says that there is a neighborhood of W^* such that all solutions from the neighborhood tend exponentially fast to a solution in the center manifold. In general properties of the solutions in the center manifold depends on the nonlinear parts of F. We assume that the center manifold is not unstable in our case, reflecting empirical results in training networks.

The Hessian of deep networks

The following two separate results imply that the Hessian of the loss function of deep networks is indeed degenerate at zero-loss minima (when they exist):

1. Polynomial deep networks can approximate arbitrarily well a standard Deep Network with ReLU activations, as shown theoretically in section 6.4 of SI and empirically in Figures 10 and 11. Bezout theorem about the number of solutions of sets of polynomial equations suggests many *degenerate* zero-minimizers of the square loss for polynomial activations (*3*). Note that the energy function L determining the gradient dynamics is a polynomial.
2. Section 6.2 in SI proves theorem 3 of which an informal statement is:

   Lemma 1. *Assume that gradient descent of a multilayer overparametrized network with nonlinear activation (under the square loss) converges to a minimum with zero error. Then the Hessian at the minimum has one or more zero eigenvalues.*

The fact that the Hessian is degenerate at minima (many zero eigenvalues) is in fact empirically well-established (for a recent account see (*17*)).

The following Proposition follows, where we use the term "stable limit point" or stable equilibrium, to denote an asymptotic limit for $t \to \infty$ of Equation 1, which is stable in the sense of Liapunov (e.g. there is a δ such that every W within an δ neighborhood of the limit $W*$ will remain under the gradient dynamics at distance smaller than ϵ of it).

Proposition 1. *If*

1. *W^* is a stable equilibrium of the gradient dynamics of a deep network trained with the square loss*
2. *W^* is a zero minimizer*

then the "potential function" L is locally quadratic and degenerate.

Proof sketch

- The first hypothesis corresponds to zero gradient at W^*, that is

$$(\nabla_w L)(W^*) = 0; \tag{4}$$

- Lemma 1 shows that if $L(W^*) = 0$ the Hessian has one or more zero eigenvalues, that is

  – let $\{v_1, \ldots, v_d\}$ be an orthonormal basis of eigenvectors of the Hessian at W^*, $\mathbf{H}L(W^*)$, and let $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ be the corresponding eigenvalues (the Hessian is a real symmetric and positive semidefinite matrix). Then, any vector $w \in \mathbb{R}^d$ can be written as $w = \sum_k^d \alpha_k v_k$ and

$$L(W^* + w) = w^\top \left(\mathbf{H}L(W^*)\right) w = \sum_k^j \lambda_k^2 \alpha_k^2, \tag{5}$$

  which is convex in $\alpha_1, \cdots, \alpha_j$ and degenerate in the directions $\alpha_{j+1}, \cdots, \alpha_d$, for which $\lambda_{j+1} = \lambda_d = 0$.

  Notice that the term "locally quadratic" in the statement of the Proposition means that $L(W^* + w)$ is quadratic in w.

  If $L(w)$ is a polynomial in w, the hypothesis implies that all linear terms disappear at $w*$ and that the quadratic terms have dimensionality $d_e < d$ with $d_e > 0$. Notice that L, as a polynomial, cannot have a flat minimum in a finite neighborhood around W^* without being identically zero (thus the "isotropically flat" minima claimed in (*8*) do not exist if L is a polynomial; furthermore the claim (*18*) that flat valleys are not directly important for generalization is then correct).

The informal statement of this proposition is

Each of the zero minima found by GD or SGD is locally well approximated by a quadratic degenerate minimum – the multidimensional equivalent of the two-dimensional minimum of Figure 2. The dynamics of gradient descent for a deep network near such a minimum is topologically equivalent to the dynamics of the corresponding linear network.

4.1 Testing the local structure of global minima

For a test of our proposition or, more precisely, of its assumptions, consider the dynamics of a perturbation δW of the solution

$$\delta \dot{W} = -[F(W^* + \delta W) - F(W^*)] = -(\mathbf{D}L(W^*))\delta W. \tag{6}$$

where asymptotic stability in the sense of Liapunov is guaranteed if the sum of the eigenvalues of $-\mathbf{D}L$ – where DL is the Hessian of L – is negative.

Consider now the following experiment. After convergence apply a small random perturbation with unit norm to the parameter vector, then run gradient descent until the training error is again zero; this sequence is repeated m times. Proposition 1 makes then the following predictions:

- The training error will go back to zero after each sequence of GD.
- Any small perturbation of the optimum W^* will be corrected by the GD dynamics to push back the non-degenerate weight directions to the original values. Since however the components of the weights in the degenerate directions are in the null space of the gradient, running GD after each perturbation will not change the weights in those directions. Overall, the weights will change in the experiment.

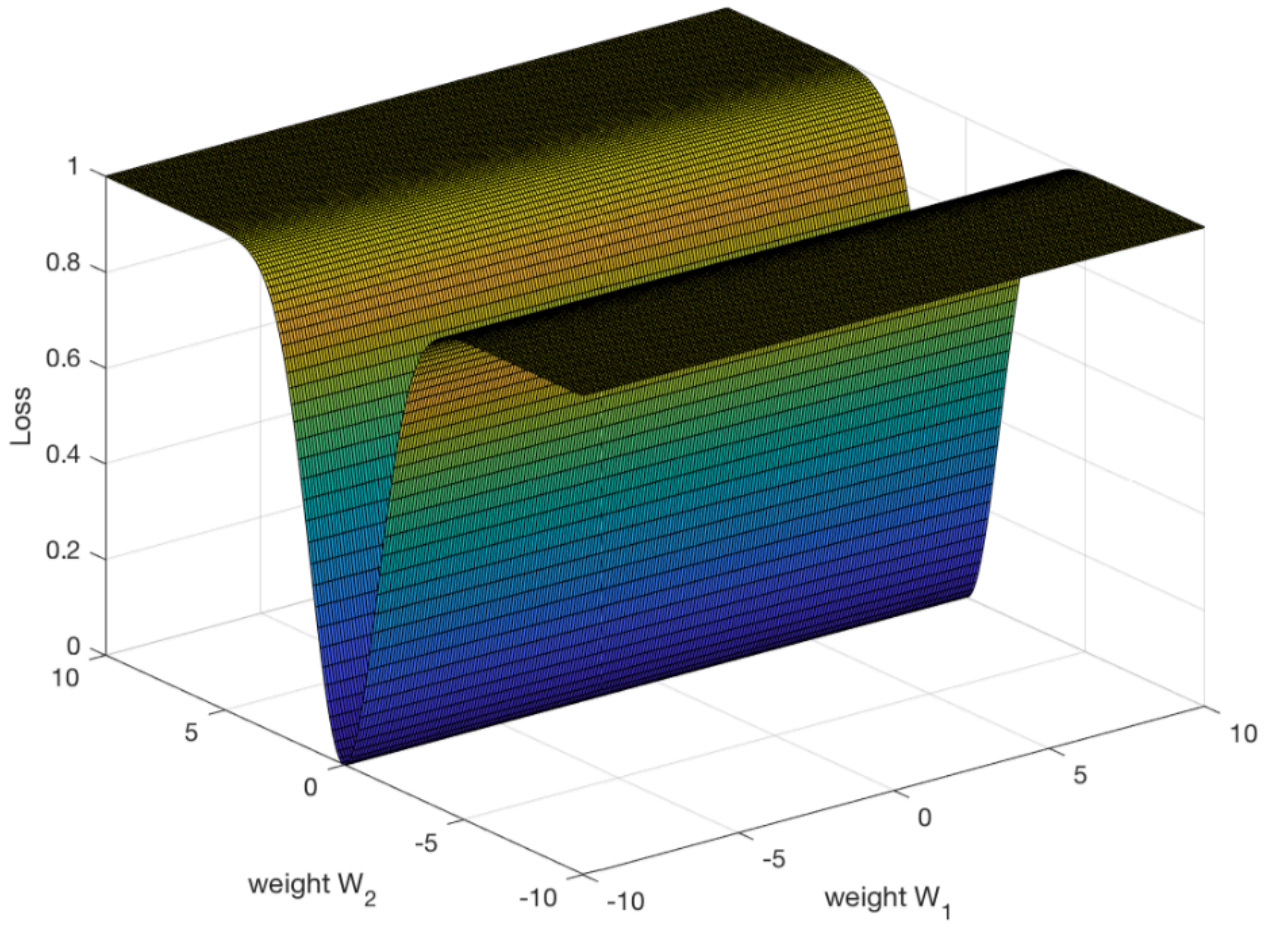


Figure 2: *An illustration of a quadratic loss function which is locally quadratic near the minimum in the two parameters w_1 and w_2. The minimum has a degenerate Hessian with a zero eigenvalue. In the proposition described in the text, this figure represents the "generic" situation in a small neighborhood of each of the 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 hypersurface with many degenerate global minima, each locally similar to a multidimensional version of the degenerate minimum shown here. For the crossentropy loss, the degenerate valley, instead of being flat, is slightly sloped downwards for $||w|| \to \infty$.*

- 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 *and* increase some norm of the parameters and increase the associated test error. In the linear case (see Figure 2) the L_2 norm of the projections of the weights in the null space undergoes a random walk: the increase in the norm of the degenerate components should then be proportional to $\approx \sqrt{m}$ with a constant of proportionality that depends on $\approx \sqrt{\frac{1}{N}}$, where N is the dimensionality of the null space.

Previous experiments (*3*) showed changes in the parameters and in the test – but not in the training – loss, consistently with our predictions above, which are further supported by the numerical experiments of Figures 14 and 3. Notice that the slight increase of the test loss without perturbation in Figure 14 is due to the use of the crossentropy risk. In the case of crossentropy the almost zero error valleys of the empirical loss function are slightly sloped downwards towards infinity, becoming flat only asymptotically (they also become "narrower and narrower multidimensional holes" in terms of sensitivity to perturbations).

The numerical experiments show, as predicted, that the behavior under small perturbations around a global minimum of the empirical loss for a deep networks is similar to that of linear degenerate regression. Figure 4 shows for comparison the latter case. Notice that in the case of linear degenerate regression (Figure 15) the test loss indicates a small overfitting, unlike

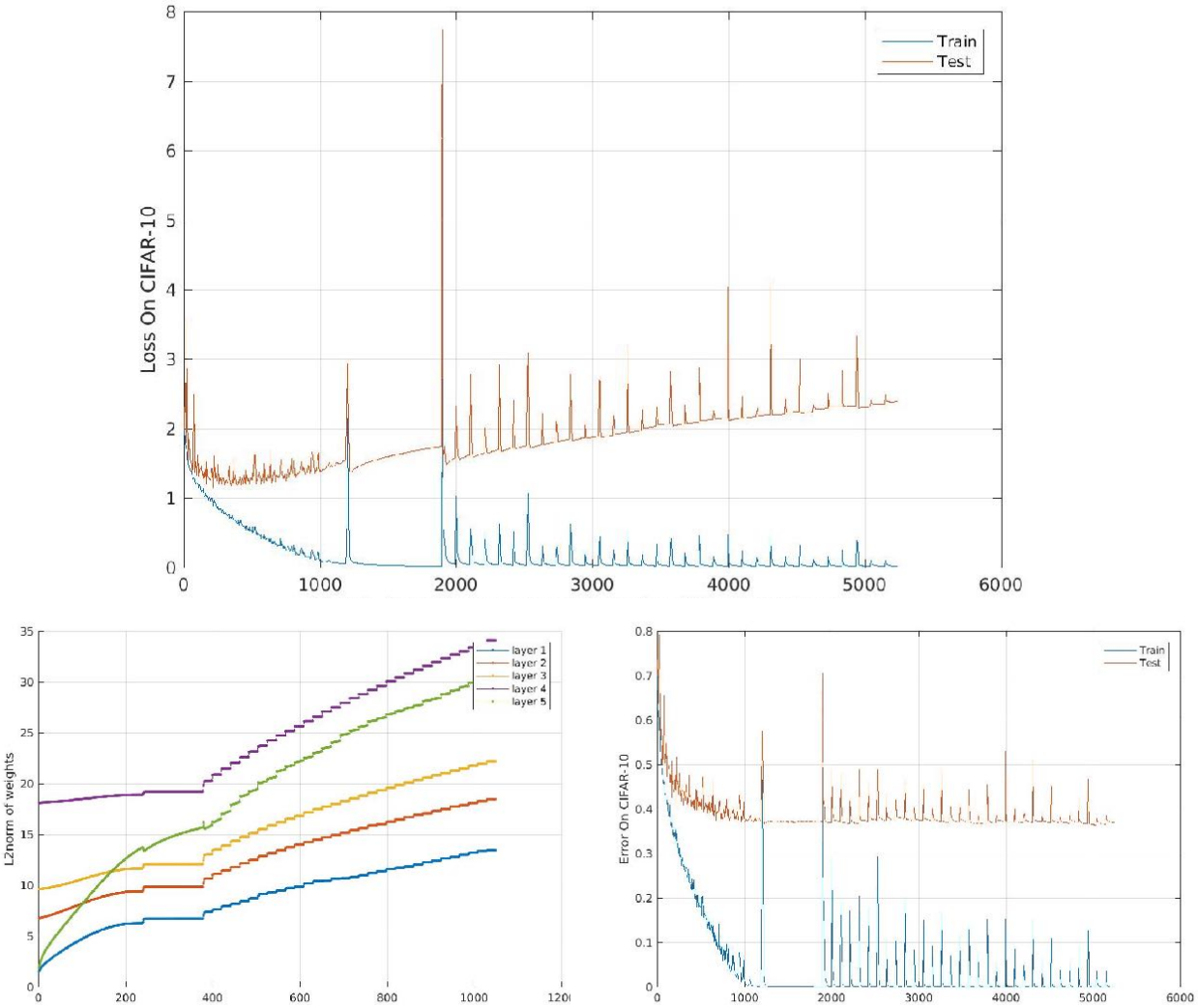


Figure 3: *We train a 5-layer convolutional neural networks on CIFAR-10 with Gradient Descent (GD) on crossentropy loss. The top plot shows the crossentropy loss on CIFAR during perturbations (see text). The bottom left is the corresponding square norm of the weights; the bottom right plot shows the classification error (see text). The network has 4 convolutional layers (filter size 33, 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. 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 the such standard deviations are added to every layer after every 100 training epochs. Training loss goes back to the original level after added perturbations, but test loss grows increasingly higher. As expected, the L_2-norm of the weights increases after each step of perturbation followed by gradient descent which reset the taining rror to zero. Compare with Figure 14 and see text.*

the nondegenerate case (Figure 16. In other words, the minimum of the test loss occurs at a finite number of iterations. This corresponds to an equivalent optimum non-zero regularization parameter λ as discussed earlier. Thus a specific "early stopping" is better than no stopping. The same phenomenon appears for the nonlinear, multilayer case (see Figure 14) with crossentropy loss. Note that the corresponding classification loss, however, does not show overfitting here.

4.2 Resistance to overfitting in deep networks

We discuss here the implications for deep nonlinear networks of the topological equivalence to linear gradient systems with a degenerate Hessian as in Figure 2.

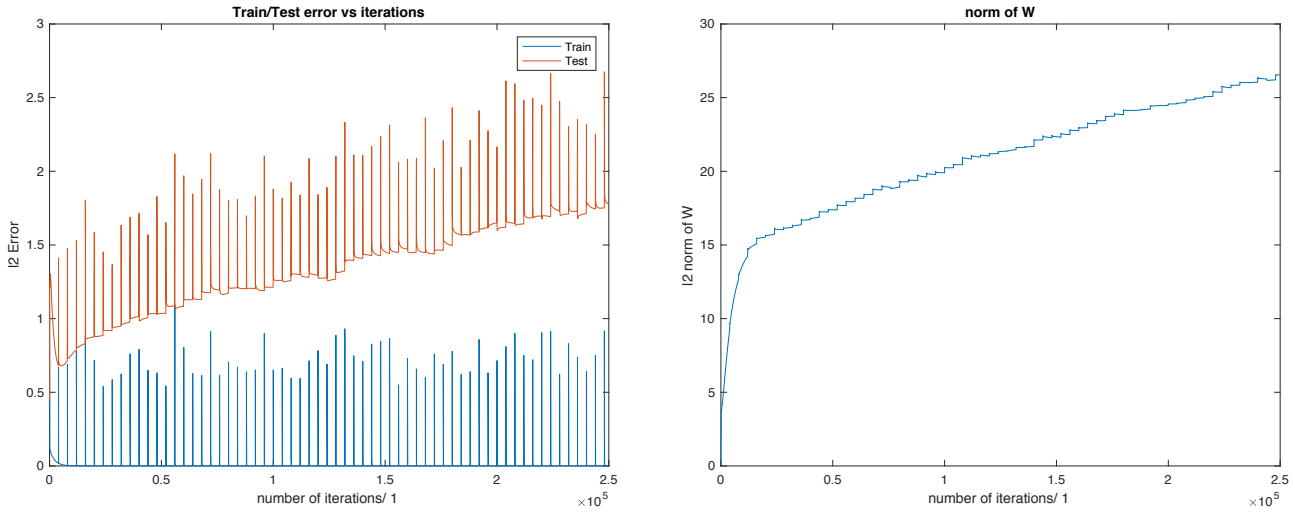


Figure 4: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 2. The feature matrix* $\phi(X)$ *is wrt a polynomial with degree 30. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points are* 9 *while the number of test points are* 100*. The training was done with full gradient descent with step size* 0.2 *for* $250,000$ *iterations. 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 addition of Gaussian noise with mean* 0 *and standard deviation* 0.6*. The* L_2 *norm of the weights is shown on the right. Note that training was repeated 30 times. The figure reports the average train and test error as well as average norm of the weights over the 30 repetitions. Figure 15 shows the same situation without perturbations.*

For the *square loss*, the existence of a non-unstable center manifold corresponding to the flat valley in Figure 2, implies that in the worst case the degenerate weight components under gradient descent will not change once a global minimum is reached, under conditions in which the quadratic approximation of Equation 5 is locally valid. The weights will be relatively small before convergence *if* the number of iterations up to convergence is small (consistently with the stability results in (*19*)). In this case 1) the solution may not be too different from a minimum norm solution 2) overfitting is expected but can be eliminated by early stopping.

It is interesting that the degeneracy of the Hessian can be eliminated by an arbitrarily small weight decay which corresponds to transforming the flat valley into a gentle convex bowl centered in W^*. Consider the dynamics associated with the regularized loss function $L_\gamma = L + \gamma||W||^2$ with γ small and positive. Then the stable points of the corresponding dynamical system will all be hyperbolic (the eigenvalues of the associated negative Hessian will all be negative). In this case the Hartman-Grobman theorem (*20*) holds. It says that the behavior of a dynamical system in a domain near a hyperbolic equilibrium point is qualitatively the same as the behavior of its linearization near this equilibrium point. Here is a version of the theorem adapted to our case.

Hartman-Grobman Theorem Consider a system evolving in time as $\dot{W} = -F(W)$ with $F = \nabla_W L(W)$ a smotth map $F : \mathbb{R}^d \to \mathbb{R}^d$. If F has a hyperbolic equilibrium state W^* and the Jacobian of F at W^* has no zero eigenavlues, then there exist a neighborhood N of W^* and a homeomorphism $h : N \to \mathbb{R}^d$, s.t. $h(W^*) = 0$ and in N the flow of $\dot{W} = -F(W)$ is topologically conjugate by the continous map $U = h(w)$ to the flow of the linearized system $\dot{U} = -HU$ where H is the Hessian of L.

Overfitting is also expected for other loss functions such as the *logistic (and the cross entropy) loss* that have a global minimum $L \to 0$ for $||w(t) \to \infty$ reached with a non-zero but exponentially small slope. In the case of the cross entropy the local

topology is almost the same but the valley of Figure 2 is now gently sloped towards the zero minimum at infinity. The situation is thus more similar to the hyperbolic, regularized situation discussed above. Overfitting of the cross entropy is shown in Figure 5

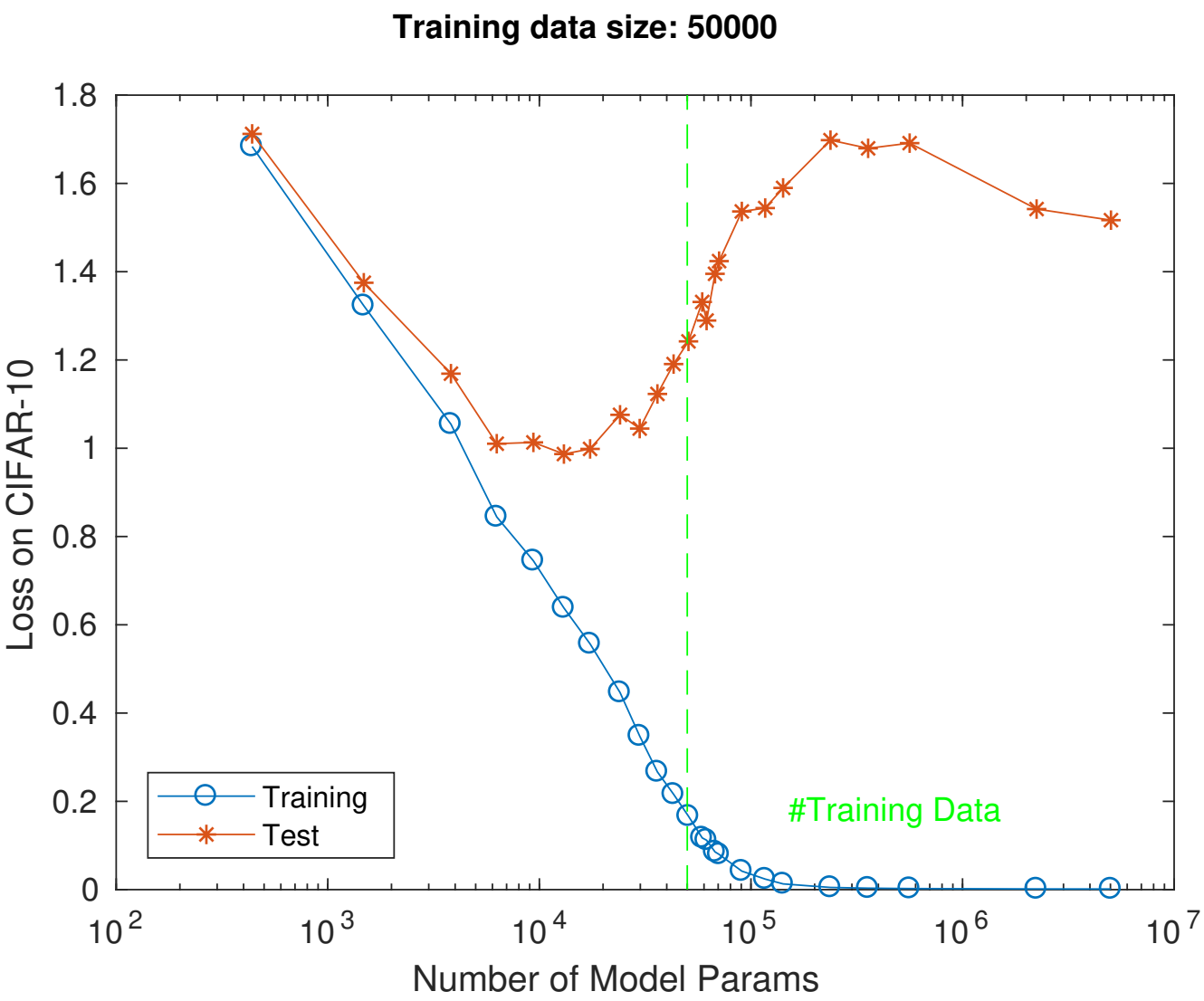


Figure 5: *The previous figures show dependence on n – number of training examples – for a fixed ReLU architecture with W parameters. This figure shows dependence on W of the cross entropy loss for a fixed training set of n examples. The network is again a 5-layer all convolutional network (i.e., no pooling) with ReLUs. All hidden layers have the same number of channels. Neither data augmentation nor regularization is performed. SGD was used with batch size $= 100$ for 70 epochs for each point. There is clear overfitting in the testing loss.*

As in the linear case, however, we can expect a better behavior for good data sets of the classification error associated with minimization of the cross entropy (*1*). Notice that, unlike the case of the degenerate square loss, gradient descent with separable data converges to the max-margin solution with any starting point w_0 (because of the non-zero slope). Thus, overfitting may not occur at all for the classification error, as shown in Figure 6, despite overfitting of the associated cross entropy loss. Notice that the classification error is not the actual loss which is minimized during training. The loss which is minimized by SGD is the crossentropy loss; the classification error itself is invisible to the optimizer. In the case of Figure 5 the crossentropy loss is clearly overfitting and increasing during the transition from under- to over-parametrization, while the classification error does not change. This is the expected behavior in the linear case.

In summary, Proposition 1 implies that multilayer, deep networks should behave similarly to linear models for regression and classification. The theory of dynamical system suggests a satisfactory approach to explain the central *puzzle of non overfitting* shown in Figure 6 (and Figure 5). Notice that Figures in the SI show the same behavior for a deep polynomial convolutional network.

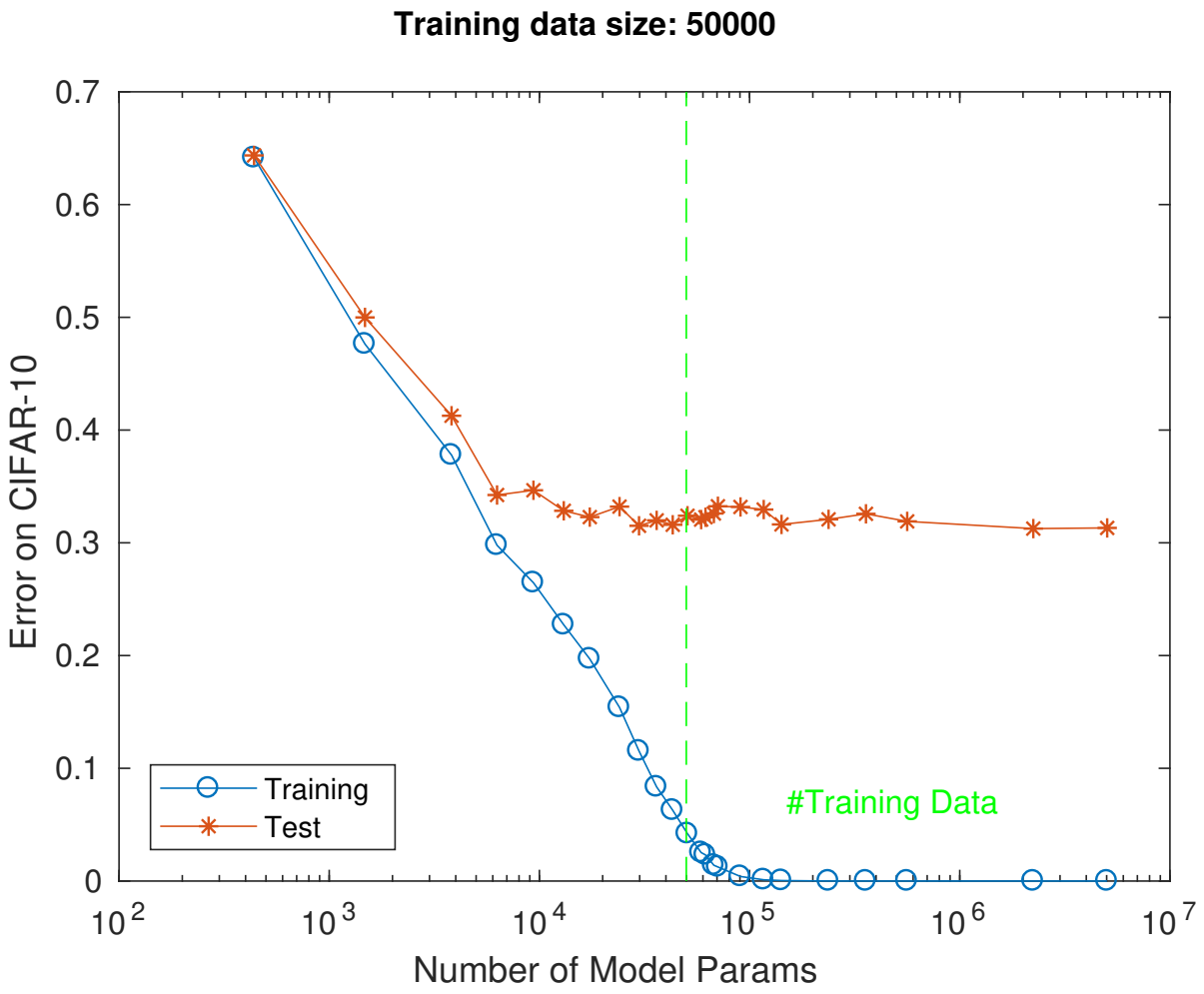


Figure 6: *The figure shows the classification error under the same conditions of Figure 5. The classical theory explains the generalization behavior on the left; the text explains the lack of overfitting for* $W > N$.

5 Discussion

Our main contribution is showing that near a global, flat minimum, gradient descent for deep networks is qualitatively equivalent to a linear system with a gradient dynamics, corresponding to a quadratic degenerate potential. Proposition 1 proves the conjecture in (*10*) that many properties of linear models should be inherited by deep networks. In particular, deep networks, similarly to linear models, are predicted to overfit the loss while not overfitting the classification error (for nice data sets). Through our reduction to linear from nonlinear dynamical system via Proposition 1, this follows from properties of gradient descent for linear network, namely *implicit regularization* of the loss and *margin maximization* for classification. In practical use of deep networks, explicit regularization (such as weight decay) together with other regularizing techniques (such as virtual examples) is usually added and it is often beneficial but not strictly necessary.

Thus there is nothing magic in deep learning that requires a theory different from the classical linear one with respect to generalization, intended as convergence of the empirical to the expected error, *and especially* with respect to the absence of overfitting in the presence of overparametrization. In our framework, the fact that deep learning networks optimized with gradient descent techniques *do generalize* follows from the implicit regularization by GD and from previous results such as Bartlett's (*12*) and Recht (*13*). More interestingly, Proposition 1 explains the puzzling property of deep networks, seen in several situations such as CIFAR, of overfitting the loss while not overfitting the classification error by showing that the properties of linear networks emphasized by (*1*) apply to deep networks under certain assumptions on the empirical minimizers and the datasets.

Of course, the problem of establishing quantitative and useful bounds on the performance of deep network, as well as the question of which type of norm is effectively optimized, remains an open and challenging problem (see (*21*)), as it is mostly

the case even for simpler one-hidden layer networks, such as SVMs. Our main claim is that the puzzling behavior of Figure 6 can be explained *qualitatively* in terms of the classical theory.

There are of course a number of open problems. Though we explained the absence of overfitting we did not explain in this paper why deep networks perform as well as they do. It is quite possible however that the answer to this question may be already contained in the following summary of the existing theoretical framework about deep learning, based on (*2*), (*3*), (*11*), (*21*) and this paper:

- unlike shallow networks deep networks can approximate the class of hierarchically local functions without incurring in the curse of dimensionality (*2, 22*);
- overparametrized deep networks yield many global degenerate – and thus flat – or almost degenerate minima (*3*) which are selected by SGD with high probability (*11*);
- overparametrization, which yields overfit of the loss, can avoid overfitting the classification error for nice datasets because of implicit regularization by gradient descent methods and the associated margin maximization.

According to this framework, the main difference between shallow and deep networks is in terms of approximation power. Unlike shallow networks, deep local (for instance convolutional) networks can avoid the curse of dimensionality (*2*) in approximating the class of hierarchically local compositional functions. SGD in overparametrized networks selects with very high probability degenerate minima which are often global. As shown in this note, overparametrization does not necessarily lead to overfitting of the classification error.

We conclude with the following, possibly quite relevant observation. The robustness of the expected error with respect to overparametrization suggests that, at least in some cases, the architecture of locally hierarchical networks may not need to be matched precisely to the underlying function graph: a relatively large amount of overparametrization (see Figure 11) may not significantly affect the predictive performance of the network, making it easier to design effective deep architectures.

Acknowledgment

We are grateful for comments on the paper by Sasha Rakhlin and for useful discussions on dynamical systems with Mike Shub. This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF 1231216. CBMM acknowledges the support of NVIDIA Corporation with the donation of the DGX-1 used in part for this research. HNM is supported in part by ARO Grant W911NF-15-1-0385

References and Notes

1. D. Soudry, E. Hoffer, and N. Srebro, "The Implicit Bias of Gradient Descent on Separable Data," *ArXiv e-prints*, Oct. 2017.
2. T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Theory I: Why and when can deep - but not shallow - networks avoid the curse of dimensionality," tech. rep., CBMM Memo No. 058, MIT Center for Brains, Minds and Machines, 2016.
3. T. Poggio and Q. Liao, "Theory II: Landscape of the empirical risk in deep learning," *arXiv:1703.09833, CBMM Memo No. 066*, 2017.
4. M. Hardt and T. Ma, "Identity matters in deep learning," *CoRR*, vol. abs/1611.04231, 2016.
5. B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," *arXiv:1706.08947*, 2017.
6. J. Sokolic, R. Giryes, G. Sapiro, and M. Rodrigues, "Robust large margin deep neural networks," *arXiv:1605.08254*, 2017.
7. P. Bartlett, D. J. Foster, and M. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *ArXiv e-prints*, June 2017.
8. 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.
9. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *CoRR*, vol. abs/1611.03530, 2016.
10. L. Rosasco and B. Recht, "Waiting for godot," *CBMM Memo 0XY*, 2017. Regression part only orally disclosed before writing this paper.
11. C. Zhang, Q. Liao, A. Rakhlin, K. Sridharan, B. Miranda, N.Golowich, and T. Poggio, "Theory of deep learning IIb: Optimization properties of SGD," *CBMM Memo 072*, 2017.

12. M. Anthony and P. Bartlett, *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.
13. M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *CoRR*, vol. abs/1509.01240, 2015.
14. L. Rosasco and S. Villa, "Learning with incremental iterative regularization," in *Advances in Neural Information Processing Systems*, pp. 1630–1638, 2015.
15. Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, pp. 289–315, Aug 2007.
16. J. Carr, *Applications of Centre Manifold Theory*. Springer Verlag, 1981.
17. L. Sagun, L. Bottou, and Y. LeCun, "Singularity of the hessian in deep learning," *CoRR*, vol. abs/1611.07476, 2016.
18. L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," *CoRR*, vol. abs/1703.04933, 2017.
19. M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *arXiv:1509.01240 [cs, math, stat]*, Sept. 2015. arXiv: 1509.01240.
20. T. Wanner, "The hartman-grobman theorem for caratheodory-type differential equations in banach spaces," 2000.
21. T. Liang, T. Poggio, A. Rakhlin, and J. Stokes, "Fisher-rao metric, geometry, and complexity of neural networks," *CoRR*, vol. abs/1711.01530, 2017.
22. H. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Analysis and Applications*, pp. 829–848, 2016.
23. J. Lin and L. Rosasco, "Optimal rates for multi-pass stochastic gradient methods," *Journal of Machine Learning Research*, vol. 18, no. 97, pp. 1–47, 2017.
24. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations (ICLR)*, 2017.
25. P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.
26. A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013.
27. K. Kawaguchi, "Deep learning without poor local minima," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.
28. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton Series in Applied Mathematics, Princeton University Press, October 2009.
29. H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *J. Mach. Learn. Res.*, vol. 10, pp. 1485–1510, Dec. 2009.
30. H. N. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network," *Advances in Computational Mathematics*, vol. 1, pp. 61–80, 1993.
31. J. Czipser and G. Freud, "Sur l'approximation d'une fonction periodique et de ses desivees successires par un polynome trigonometrique et par ses derivees successives," *Acta Math.,(Sweden)*, vol. 99, pp. 33–51, 1958.
32. R. A. DeVore and G. G. Lorentz, *Constructive approximation*, vol. 303. Springer Science & Business Media, 1993.
33. N. Hahm and I. H. BUM, "Simultaneous approximation algorithm using a feedforward neural network with a single hidden layer," *Journal of the Korean Physical Society*, vol. 54, no. 6, pp. 2219–2224, 2009.

6 Supplementary Information

6.1 Implicit/Iterative Regularization by GD and SGD for linear dynamical systems

We recall some recent results regarding the regularization properties of SGD (*23*). Here regularization does not arise from explicit penalizations or constraints. Rather it is implicit in the sense that it is induced by dynamic of the SGD iteration and controlled by the choice of either the step-size, the number of iterations, or both. More precisely, we recall recent results showing convergence and convergence rates to the minimum expected risk, but also convergence to the minimal norm minimizer of the expected risk.

All the the results are for the least square loss and assume linearly parameterized functions. With these choices, the learning problem is to solve

$$\min_{w\in\mathbb{R}^D} \mathcal{E}(w) \qquad \mathcal{E}(w) = \int (y - w^\top x)^2 d\rho$$

given only $(x_i, y_i)_{i=1}^n - \rho$ fixed, but unknown. Among all the minimizers $\mathcal{O}$ of the expected risk the minimal norm solution is $w^\dagger$ solving

$$\min_{w\in\mathcal{O}} \|w\|,$$

We add two remarks.

- The results we present next hold considering functions of the form

  $$f(x) = \langle w, \Phi(x)\rangle$$

  for some nonlinear feature map $\Phi : \mathbb{R}^D \to \mathbb{R}^p$.
- Further, all the results are dimension independent, hence allow to consider D (or p) to be infinite.

The SGD iteration is given by

$$\widehat{w}_{t+1} = \widehat{w}_t - \eta_t x_{i_t}(\widehat{w}_t^\top x_{i_t} - y_{i_t}), \quad t = 0, \dots T$$

where

- $(\eta_t)_t$ specifies the step-size,
- T stopping time, ($T > n$ multiple *"passes/epochs"*)
- $(i_t)_t$ specifies how the iteration visits the training points. Classically $(i_t)_t$ is chosen to be stochastic and induced by a uniform distribution over the training points.

The following result considers the regularization and learning properties of SGD in terms of the corresponding excess risk.
Theorem
Assume $\|x\| \le 1$ and $|y| \le 1$ for all $(\eta_t)_t = \eta$ and T, with high probability

$$\mathcal{E}(\widehat{w}_T) - \mathcal{E}(w^\dagger) \lesssim \frac{1}{\eta T} + \frac{1}{\sqrt{n}}\left(\frac{\eta T}{\sqrt{n}}\right)^2 + \eta\left(1 \vee \frac{\eta T}{\sqrt{n}}\right)$$

In particular, if $\eta = \frac{1}{\sqrt{n}}$ and $T = n$, then

$$\lim_{n\to\infty} \mathcal{E}(\widehat{w}_T) = \mathcal{E}(w^\dagger)$$

almost surely.
The next result establishes convergence to the minimal norm solution of the expected risk.
Theorem
Assume $\|x\| \le 1$ and $|y| \le 1$, if $\eta = \frac{1}{\sqrt{n}}$ and $T = n$, then

$$\lim_{n\to\infty} \|w_T - w^\dagger\| = 0.$$

almost surely.

6.2 Hessian of the loss for multilayer networks

6.2.1 No-hidden-layer linear systems

For a linear network without hidden layers with the training set (X, Y), a set of weights A can be found by minimizing

$$\min_A \|Y - AX\|_F^2, \tag{7}$$

where $Y \in \mathcal{R}^{d',n}$, $A \in \mathcal{R}^{d',d}$, $X \in \mathcal{R}^{d,n}$ and n is number of data points, yielding an equation for A that is degenerate when $d > n$. The general solution is in the form of

$$A = YX^\dagger + M,$$

where $X^\dagger$ is the pseudoinverse of the matrix X, and M is any matrix in the left null space of X. The minimum norm solution is

$$A = YX^\dagger. \tag{8}$$

Furthermore, as shown in the section before, the solution Equation 8 is found by GD and SGD and corresponds to iterative regularization where the role of the inverse of the regularization parameter λ is played by the product of step size and number of steps (*14*).

Let us look in more detail at the gradient descent solution (from (*8*)). Consider the following setup: $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n,d}$ are the data points, with $d > n$. We further assume that the data matrix is of full row rank: $\text{rank}(X) = n$. Let $y \in \mathbb{R}^n$ be the labels, and consider the following linear system:

$$Xw = y \tag{9}$$

where $w \in \mathbb{R}^d$ is the weights to find. This linear system has infinite many solutions because X is of full row rank and we have more parameters than the number of equations. Now suppose we solve the linear system via a least square formulation

$$L(w) = \frac{1}{2n}\|Xw - y\|^2 \tag{10}$$

by using gradient descent (GD) or stochastic gradient descent (SGD). In this section, we will show that both GD and SGD converges to the minimum norm solution, for the ℓ_2 norm. A similar analysis (with fewer details) was presented in (*24*).

Lemma 2. *The following formula defines a solution to* (9)

$$w_\dagger \triangleq X^\top (XX^\top)^{-1} y \tag{11}$$

and it is the minimum norm solution.

Proof Note since X is of full row rank, so $XX^\top$ is invertible. By definition,

$$Xw_\dagger = XX^\top (XX^\top)^{-1} y = y$$

Therefore, $w_\dagger$ is a solution. Now assume $\hat{w}$ is another solution to (9), we show that $\|\hat{w}\| \geq \|w_\dagger\|$. Consider the inner product

$$\begin{aligned}
\langle w_\dagger, \hat{w} - w_\dagger \rangle &= \langle X^\top (XX^\top)^{-1} y, \hat{w} - w_\dagger \rangle \\
&= \langle (XX^\top)^{-1} y, X\hat{w} - X_\dagger \rangle \\
&= \langle (XX^\top)^{-1} y, y - y \rangle \\
&= 0
\end{aligned}$$

Therefore, $w_\dagger$ is orthogonal to $\hat{w} - w_\dagger$. As a result, by Pythagorean theorem,

$$\|\hat{w}\|^2 = \|(\hat{w} - w_\dagger) + w_\dagger\|^2 = \|\hat{w} - w_\dagger\|^2 + \|w_\dagger\|^2 \geq \|w_\dagger\|^2$$

Lemma 3. *When initializing at zero, the solutions found by both GD and SGD for problem* (10) *live in the span of rows of* X*. In other words, the solutions are of the following parametric form*

$$w = X^\top \alpha \tag{12}$$

for some $\alpha \in \mathbb{R}^n$*.*

Proof

The gradient for (10) is

$$\nabla_w L(w) = \frac{1}{n} X^\top (Xw - y) = X^\top e$$

where we define $e = (1/n)(Xw - y)$ to be the error vector. GD use the following update rule:

$$w_{t+1} = w_t - \eta_t \nabla_w L(w_t) = w_t - \eta_t X^\top e_t$$

Expanding recursively, and assume $w_0 = 0$. we get

$$w_t = \sum_{\tau=0}^{t-1} -\eta_\tau X^\top e_\tau = X^\top \left(-\sum_{\tau=0}^{t-1} \eta_\tau e_\tau \right)$$

The same conclusion holds for SGD, where the update rule could be explicitly written as

$$w_{t+1} = w_t - \eta_t (x_{i_t}^\top w - y_{i_t}) x_{i_t}$$

where (x_{i_t}, y_{i_t}) is the pair of sample chosen at the t-th iteration. The same conclusion follows with $w_0 = 0$.

Q.E.D.

Theorem 1. *Let* w_t *be the solution of GD after* t*-th iteration, and* w_t *be the (random) solution of SGD after* t*-th iteration. Then* $\forall \varepsilon > 0$*,* $\exists T > 0$ *such that*

$$L(w_t) \leq \varepsilon, \quad \mathbb{E} L(\mathsf{w}_t) \leq \varepsilon$$

where the expectation is with respect to the randomness in SGD.

Corollary 1. *When initialized with zero, both GD and SGD converges to the minimum-norm solution.*

Proof Combining Lemma 3 and Theorem 1, GD is converging $w_t \to w_\star = X^\top \alpha_\star$ as $t \to \infty$ for some optimal $\alpha_\star$. Since $w_\star$ is a solution to (9), we get

$$y = X w_\star = X X^\top \alpha_\star$$

Since $XX^\top$ is invertible, we can solve for $\alpha_\star$ and get

$$w_\star = X^\top \alpha_\star = X^\top (X X^\top) y = w_\dagger$$

Similar argument can be made for SGD with expectation with respect to the randomness of the algorithm.

In the degenerate linear approximation of a nonlinear system near a global minimum the above situation holds but the null space is not guaranteed to remain unaffected by early steps in GD. Norms will remain bounded if the gradient is bounded (in fact the title of (*13*) *Train faster, generalize better...*" may apply here) and thus generalization is guaranteed. The solution is *not* strictly guaranteed to be of minimum norm in the case of quadratic loss. It is, however, *guaranted to be the maximum margin solution for separable data in the case of logistic and croos-entropy loss* (*1*).

6.2.2 One-hidden layer linear networks

g

Consider linear network of the same type analyzed by (*25, 26*) in the full rank case and shown in Figure 7. This is a network with an input layer, an output layer, a hidden layer of *linear* units and two sets of weights. Our results for the degenerate case are new.

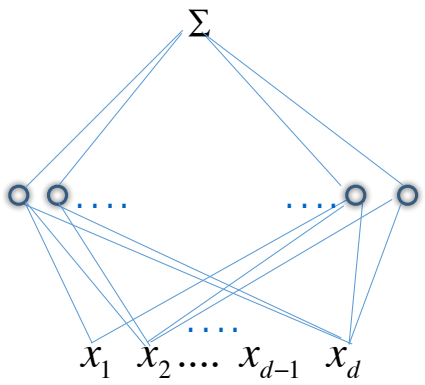


Figure 7: *A one-hidden layer network with weights represented by the matrix W_1 from the d inputs to the N hidden units and the matrix W_2 for the weights from the hidden units to the output.*

Suppose that gradient descent is used to minimize a quadratic loss on a set of n training examples. The activation function of the hidden units is linear, that is $h(z) = az$. Notice that though the mapping between the input x and the output $f(x)$ is linear, the dependence of f on the set of weights W_1, W_2 is nonlinear. Such deep linear networks were studied in a series of papers (*25–27*) in situations in which there are more data than parameters, $n \geq d$. In this section we show how the network will not overfit even in the overparametrized case, which corresponds to $n < d$.

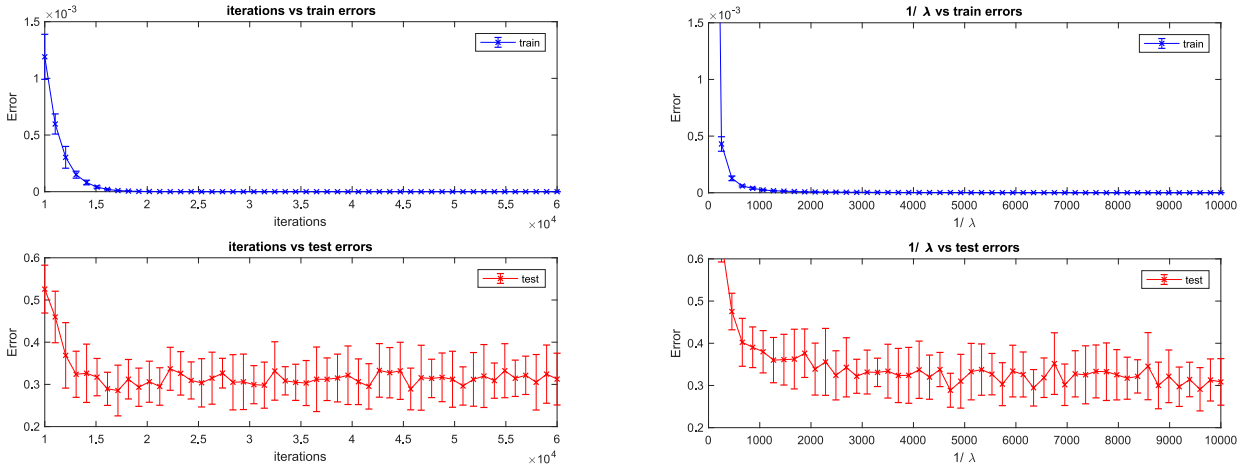


Figure 8: *In this figure we show how a linear network $W_2W_1X = y$ converges to the test error in two different training protocols: in the first the number of iterations grows, in the second $\frac{1}{\lambda}$ grows large (with optimization run to convergence for each λ. Optimization is performed in both case with SGD. The training on the right were done on by minimizing $\frac{1}{n}\sum_{i=1}^{n} V(f(x), y) + \lambda\|W_2W_1\|$ with various values of the regularization parameters λ. Similarly, the experiments on the right minimized the same cost function except that $\lambda = 0$. The number of data points were $n = 30$, the input dimension $d = 31$ (including the bias). The target function was generated by a two layer linear network with 1 hidden unit and thus a total of 32 weights. The approximating two linear network had 2 hidden units and thus 64 parameters. The test set had 32 data points. Note that there was no noise in the training or test set.*

Model Description and notation Consider a network, d inputs, N hidden *linear* units and d' outputs. We denote the loss with $L(w) = \frac{1}{2}||W_2W_1X - Y||^2$, where $X \in \mathbb{R}^{d,n}, Y \in \mathbb{R}^{d',n}$, $W_2 \in \mathbb{R}^{d',N}$ and $W_1 \in \mathbb{R}^{N,d}$. Let $E = W_2W_1X - Y \in \mathbb{R}^{d',n}$. Let $w = \text{vec}(W_1^\top, W_2^\top) \in \mathbb{R}^{Nd+d'N}$.

Gradient dynamics and Hessian We can write the dynamical system corresponding to its gradient as

$$\dot{W}_1 = -\nabla_{W_1} L(w) = -W_2^\top E X^\top = W_2^\top Y X^\top - W_2^\top W_2 W_1 X X^\top \tag{13}$$

and similarly

$$\dot{W}_2 = -Y X^\top W_1^\top + W_2 W_1 X X^\top W_1^\top \tag{14}$$

The linearization of this dynamical system (Equations 13 and 14) corresponds to the Hessian of the loss, which is

$$\nabla^2 L(w) = \begin{bmatrix} W_2^\top W_2 \otimes X X^\top & C \\ C^\top & I_{d'} \otimes X X^\top W_1 X X^\top W_1 \end{bmatrix} \in \mathbb{R}^{(Nd+d'N),(Nd+d'N)}$$

where

$$C = [W_2^\top \otimes X X^\top W_1^\top] + [I_N \otimes X (E^\top)_{\bullet 1}, \ldots, I_N \otimes X (E^\top)_{\bullet d'}].$$

Here, $(E^\top)_{\bullet i}$ denote the i-th column of $E^\top$.

Degeneracy of the dynamics Suppose that $d' < N$ (overparametrization). Then, $W_2^\top W_2$ has zero eigenvalue, which implies that $W_2^\top W_2 \otimes X X^\top$ has zero eigenvalue. In turn, this implies that $\nabla^2 L(w)$ has zero eigenvalue. Indeed, we can construct a degenerate direction as follows: consider a vector $w = ([w_1 \otimes w_2], 0)^\top$ such that w_1 is in the null space of W_2, with which we have that $w^\top \nabla^2 L(w) w = 0$.

However, along this degenerate direction, the output model does not change at all, because this direction corresponds to adding an element of null space of W_2 to W_1 (i.e., the product $W_2 W_1$ does not change in this direction) Also, (S)GD does not add element corresponding to this direction from Equation 13, since $\dot{W}_1$ is in the column space of $W_2^\top$ which is orthogonal to the null space of W_2.

Trivial and nontrivial degeneracy It is interesting that even in the overdetermined case $n >> d$, the solution is not unique in terms of W_1 and W_2. This corresponds to the "trivial" degeneracy, which is a terminology defined as follows: Let $f_w : \mathcal{X} \to \mathcal{Y}$ be a model parameterized by the parameter w. For example, in the case of the linear one-hidden layer model, $f_w(x) = W_2 W_1 x$ where $w = \mathrm{vec}(W_1^\top, W_2^\top)$ and $x \in \mathcal{X}$. Then, we call degeneracy "trivial" if and only if there exists $\epsilon > 0$ such that $f_w = f_{w+\epsilon \Delta w}$ where Δw is in the degenerate space. Here, we write $f_1 = f_2$ if and only if their graphs are equal: i.e., $\{(x, y) \in \mathcal{X} \times \mathcal{Y} : x \in \mathcal{X}, f_1(x) = y\} = \{(x, y) \in \mathcal{X} \times \mathcal{Y} : x \in \mathcal{X}, f_2(x) = y\}$. We call degeneracy "nontrivial" if and only if it is not trivial.

As we will see, trivial degeneracy occurs in multilayer networks with unit activation which is linear (section 6.2.2) or ReLU (section 6.2.4). The situation in terms of trivial and nontrivial degeneracy is quite different in the overdetermined case and underdetermined case:

- The solution in the overdetermined case yields a network that is unique despite the degeneracy. In other words, every possible degeneracy is *trivial degeneracy*. Furthermore, more data cannot change this situation.
- In the underdetermined case, we may have both types of degeneracy. More data would avoid *nontrivial degeneracy* which yields different networks.

GD and SGD converge to the Minimum Norm Solution Assume that $N, d \geq n \geq d'$ (overparametrization). Let $A = W_2 W_1$. For any matrix M, let $\mathrm{Col}(M)$ and $\mathrm{Null}(M)$ be the column space and null space of M.

Due to the convexity of the loss L as a function of the entries of A, the global minimum A^* in terms of the entries of A is obtained as follows:

$$\frac{\partial L}{\partial A^*} = (A^* X - Y) X^\top = 0$$
$$\Leftrightarrow A^* \in \{A = Y X^\top (X X^\top)^\dagger + B_X : B_X X = 0\}$$

The minimum norm solution $A^*_{\min}$ is the global minimum A^* with its rows not in $\mathrm{Null}(X^\top)$, which is

$$A^*_{\min} = YX^\top(XX^\top)^\dagger.$$

Due to the over-parameterization with w, for any entries of A, there exist entries of W_1 and W_2 such that $A = W_2W_1$. Thus, the global minimum solutions in terms of A and w are the same.

Lemma 4. *For gradient descent and stochastic gradient descent with any mini-batch size,*

- *any number of the iterations adds no element in* $\mathrm{Null}(X^\top)$ *to the rows of* W_1*, and hence*
- *if the rows of* W_1 *has no element in* $\mathrm{Null}(X^\top)$ *at anytime (including the initialization), the sequence converges to a minimum norm solution if it converges to a solution.*

Proof. From $\frac{\partial L}{\partial \mathrm{vec}(W_1)} = [X \otimes W_2^\top]\mathrm{vec}(E) = \mathrm{vec}(W_2^\top EX^\top)$, we obtain that

$$\frac{\partial L}{\partial W_1} = W_2^\top EX^\top.$$

For SGD with any mini-batch size, let $\bar{X}_t$ be the input matrix corresponding to a mini-batch used at t-th iteration of SGD. Let $\bar{L}_t$ and $\bar{E}_t$ be the corresponding loss and the error matrix. Then, by the same token,

$$\frac{\partial \bar{L}_t}{\partial W_1} = W_2^\top \bar{E}_t\bar{X}_t^\top.$$

From these gradient formulas, the first statement follows by noticing that for any t, $\mathrm{Col}(\bar{X}_t) \subseteq \mathrm{Col}(X) \perp \mathrm{Null}(X^\top)$. The second statement follows the fact that if the rows of W_1 has no element in $\mathrm{Null}(X^\top)$ at anytime t, then for anytime after that time t, $\mathrm{Col}(W_1^\top W_2^\top) \subseteq \mathrm{Col}(W_1^\top) \subseteq \mathrm{Col}(X) \perp \mathrm{Null}(X^\top)$.

□

The above lemma shows that if (stochastic) gradient descent sequence converges to a solution, then we can easily ensure the solution to be a minimum norm solution. However, we have not shown that (stochastic) gradient descent converges to a global minimum solution. This result is provided by the following lemma.

Lemma 5. *If* $W_2 \neq 0$*, every stationary point w.r.t.* W_1 *is a global minimum.*

Proof. For any global minimum solution A^*, the transpose of the model output is

$$(A^*X)^\top = X^\top(XX^\top)^\dagger XY^\top$$

which is the projection of $Y^\top$ onto $\mathrm{Col}(X^\top)$. Let $D = [W_2 \otimes X^\top]$. Then, with the transpose of the model output, the loss can be rewritten as

$$L(w) = \frac{1}{2}\|X^\top W_1^\top W_2^\top - Y^\top\|^2 = \frac{1}{2}\|D\mathrm{vec}(W_1^\top) - Y^\top\|^2.$$

The condition for a stationary point yields

$$\begin{aligned} 0 = \frac{\partial L}{\partial \mathrm{vec}(W_1^\top)} &= D^T(D\mathrm{vec}(W_1^\top) - Y^\top) \\ \Rightarrow D\mathrm{vec}(W_1^\top) &= D(D^TD)^\dagger D^TY^\top = \text{Projection of } Y^\top \text{ onto } \mathrm{Col}(D). \end{aligned}$$

If $W_2 \neq 0$, we obtain $\mathrm{Col}(D) = \mathrm{Col}(X^\top)$. Hence any stationary point w.r.t. W_1 is a global minimum. □

It follows from Lemmas 4 and 5 that, if we initialize the rows of W_1 with no element in $\mathrm{Null}(X^\top)$, and if $W_2 \neq 0$, GD and SGD find a minimum norm solution.

The summary is provided by

Theorem 2. *If*

- *the rows of W_1 are initialized with no element in $\mathrm{Null}(X^\top)$,*
- *and if $W_2 \neq 0$,*

then gradient descent (and stochastic gradient descent) converges to a minimum norm solution.

As a final remark, the analysis above holds for a broad range of loss function and not just the square loss. Asymptotically $\dot{W} = -\nabla_W L(W^\top X)$, in which we makes explicit the dependence of the loss L on the linear function $W^\top X$. Since $\nabla_W L(W^\top X) = X^\top \nabla_Z L(Z)$ the update to W is in the span of the data X, that is, it does not change the projection of W in the null space of X. Thus if the norm of the components of W in the null space of X was small at the beginning of the iterations, it will remain small at the end. This means that among all the solutions W with zero error, gradient descent selects the minimum norm one.

6.2.3 One-hidden layer polynomial networks

Consider a polynomial activation for the hidden units. The case of interest here is $n > d$. Consider the loss $L(w) = \frac{1}{2}||P_m(X) - Y||_F^2$ where

$$P_m(X) = W_2(W_1 X)^m. \tag{15}$$

where the power m is elementwise.

We obtain, denoting $E = P_m(X) - Y$, with $E \in \mathbb{R}^{d',n}$, $W_2 \in \mathbb{R}^{d',N}$, $W_1 \in \mathbb{R}^{N,d}$, $E' \in \mathbb{R}^{d',d}$

$$\nabla_{W_1} L(w) = m(W_1 X)^{m-1} \circ (W_2^T E)]X^\top = E' X^\top \tag{16}$$

where the symbol $\circ$ denotes Hadamard (or entry-wise) product. In a similar way, we have

$$\nabla_{W_2} L(w) = E(((W_1 X)^m)^\top). \tag{17}$$

6.2.4 Nonlinear deep networks

We now discuss an extension of the above arguments to the nonlinear activation case with activation function $\sigma(z)$, such as the ReLU activation.

Gradient dynamics of polynomial multilayer networks We remark that if $\sigma(z) = az + bz^2$, the dynamical system induced by GD and the square loss is

$$\dot{W}_1 = -(aW_2^\top E + 2b[(W_1 X) \circ (W_2^T E)])X^\top \tag{18}$$

and

$$\dot{W}_2 = -[aEX^\top W_1^\top + bE(((W_1 X)^2)^\top)]. \tag{19}$$

Hessian and degeneracy of nonlinear one-hidden layer networks For a general activation function σ, some block of the Hessian of the loss can be written as

$$\nabla^2 L(w) = \begin{bmatrix} - & \\ - & I_{d'} \otimes \sigma(W_1 X)\sigma(W_1 X)^\top \end{bmatrix}.$$

We can derive other blocks of the Hessian as follows: $\nabla^2 L(w) = \nabla \hat{Y}(w) \nabla \hat{Y}(w)^\top + \sum_{i=1}^n \nabla^2 \hat{Y}_i(w) E_i$, where $\hat{Y}(w) = \mathrm{vec}(W_2 \sigma(W_1 X))$ is the model output vector. **If** $E = 0$ at the limit point, then $\nabla^2 L(w) = \nabla \hat{Y}(w) \nabla \hat{Y}(w)^\top$. So, let us consider the case where $E = 0$ at the limit point.

$$\begin{aligned} &\nabla \hat{Y}(w) \nabla \hat{Y}(w)^\top \\ &= \begin{bmatrix} \nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top & \nabla_1 \hat{Y}(w) \nabla_2 \hat{Y}(w)^\top \\ \nabla_2 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top & \nabla_2 \hat{Y}(w) \nabla_2 \hat{Y}(w)^\top \end{bmatrix}. \end{aligned}$$

where

$$\nabla_1 \hat{Y}(w)^\top = [I_n \otimes W_2] \dot{\sigma}(\mathrm{vec}[W_1 X])[X^\top \otimes I_N],$$

and

$$\nabla_2 \hat{Y}(w)^\top = I_{d'} \otimes \sigma(W_1 X)^\top$$

We now show that the first block of the Hessian,

$$\nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top = [X \otimes I_N] \dot{\sigma}(\mathrm{vec}[W_1 X])[I_n \otimes W_2^\top W_2] \dot{\sigma}(\mathrm{vec}[W_1 X])[X^\top \otimes I_N],$$

has the zero eigenvalues when the model is over-parameterized as $dN > n \cdot \min(N, d')$. This happens with over-parameterization. Then, the rank of $[I_n \otimes W_2^\top W_2]$ is at most $n \cdot \min(N, d')$, which implies that the first block in the Hessian ($[X \otimes I_N] \dot{\sigma}(\mathrm{vec}[W_1 X])[I_n \otimes W_2^\top W_2] \dot{\sigma}(\mathrm{vec}[W_1 X])[X^\top \otimes I_N]$) of size dN by dN has zero eigenvalue (the rank of the product of matrices is at most the minimum of the ranks of matrices), which implies that $\nabla \hat{Y}(w) \nabla \hat{Y}(w)^\top$ has zero eigenvalue.

Hessian and degeneracy of nonlinear two-hidden layer networks For networks with two-hidden layers, we have that

$$\nabla \hat{Y}(w) \nabla \hat{Y}(w)^\top = \begin{bmatrix} \nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top & \nabla_1 \hat{Y}(w) \nabla_2 \hat{Y}(w)^\top & \nabla_1 \hat{Y}(w) \nabla_3 \hat{Y}(w)^\top \\ \nabla_2 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top & \nabla_2 \hat{Y}(w) \nabla_2 \hat{Y}(w)^\top & \nabla_2 \hat{Y}(w) \nabla_3 \hat{Y}(w)^\top \\ \nabla_3 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top & \nabla_3 \hat{Y}(w) \nabla_2 \hat{Y}(w)^\top & \nabla_3 \hat{Y}(w) \nabla_3 \hat{Y}(w)^\top \end{bmatrix},$$

where,

$$\begin{aligned} \nabla_1 \hat{Y}(w)^\top &= [I_n \otimes W_3] \dot{\sigma}(h_2)[I_n \otimes W_2] \dot{\sigma}(h_1)[X^\top \otimes I_{N_1}], \\ \nabla_2 \hat{Y}(w)^\top &= [I_n \otimes W_3] \dot{\sigma}(h_2)[\sigma(W_1 X)^\top \otimes I_{N_2}], \end{aligned}$$

and

$$\nabla_3 \hat{Y}(w)^\top = [\sigma(W_2 \sigma(W_1 X))^\top \otimes I_{N_3}].$$

By the same token, we have zero eigenvalue in the block $\nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top$ with overparametrization.

Let $D^\top = [I_n \otimes W_3] \dot{\sigma}(h_2)[I_n \otimes W_2] \dot{\sigma}(h_1)[X^\top \otimes I_{N_1}]$. If the nonlinear activation is ReLU, then we can write $\hat{Y}(w) = D^\top \mathrm{vec}(W_1)$ and $\nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top = D D^\top$. Thus, the zero eigenvalues of the Hessian block $\nabla_1 \hat{Y}(w) \nabla_1 \hat{Y}(w)^\top$ are the exactly those defining the nulls space of $D^\top$. In other words, the zero eigenvalue directions do not change the loss values. Furthermore, similarly to the no-null-element proof for deep linear case,

$$\nabla_{\mathrm{vec}(W_1)} \hat{L}(w) = D(\hat{Y}(w) - Y)$$

which implies that the gradient descent does not add any elements from the null space of $D^\top$. However, D is now a function of w (and X), and D is changing during training when far from the critical point.

Derivation of the Hessian We describe below the derivation of the Hessian for nonlinear two-hidden layer case. By the same token, we derived the Hessian for one hidden layer with nonlinear and linear cases.

To avoid introducing different symbols for different domains of activation functions, let $\sigma(M)$ be the elementwise application of nonlinear activation for any size of matrix M. Accordingly, given a size t of vector v, let $\dot{\sigma}(v)$ be the diagonal matrix of size t by t, with the diagonal elements being the derivative of the nonlinear activation function for each element v_k for $k = 1 \ldots, t$.

Let $L(w) = \frac{1}{2}\|W_3\sigma(W_2\sigma(W_1X)) - Y\|_F^2$. Let $\hat{Y}_n(w) = W_3\sigma(W_2\sigma(W_1X))$ and $E = \hat{Y}_n(w) - Y$. Then, $L(w) = \mathrm{vec}(E)^\top \mathrm{vec}(E)$.

Then,

$$\begin{aligned}
\mathrm{vec}(\hat{Y}_n(w)) &= \mathrm{vec}(W_3\sigma(W_2\sigma(W_1X))) \\
&= [I_n \otimes W_3]\mathrm{vec}(\sigma(W_2\sigma(W_1X))) \\
&= [I_n \otimes W_3]\sigma(\mathrm{vec}(W_2\sigma(W_1X))) \\
&= [I_n \otimes W_3]\sigma([I_n \otimes W_2]\mathrm{vec}(\sigma(W_1X))) \\
&= [I_n \otimes W_3]\sigma([I_n \otimes W_2]\sigma(\mathrm{vec}(W_1X))) \\
&= [I_n \otimes W_3]\sigma([I_n \otimes W_2]\sigma([X^\top \otimes I_{N_1}]\mathrm{vec}(W_1)))
\end{aligned}$$

Similarly,

$$\mathrm{vec}(\hat{Y}_n(w)) = [I_n \otimes W_3]\sigma([\sigma(W_1X)^\top \otimes I_{N_2}]\mathrm{vec}(W_2)),$$

and

$$\mathrm{vec}(\hat{Y}_n(w)) = [\sigma(W_2\sigma(W_1X))^\top \otimes I_{N_3}]\mathrm{vec}(W_3).$$

Let $h_2 = \mathrm{vec}(W_2\sigma(W_1X))$ and $h_1 = \mathrm{vec}(W_1X))$. Let $\nabla_k \mathrm{vec}(\hat{Y}_n(w))^\top = \nabla_{\mathrm{vec}(W_k)}\mathrm{vec}(\hat{Y}_n(w))^\top$.

Then,

$$\nabla_1 \mathrm{vec}(\hat{Y}_n(w))^\top = [I_n \otimes W_3]\dot{\sigma}(h_2)[I_n \otimes W_2]\dot{\sigma}(h_1)[X^\top \otimes I_{N_1}],$$

$$\nabla_2 \mathrm{vec}(\hat{Y}_n(w))^\top = [I_n \otimes W_3]\dot{\sigma}(h_2)[\sigma(W_1X)^\top \otimes I_{N_2}],$$

and

$$\nabla_3 \mathrm{vec}(\hat{Y}_n(w))^\top = [\sigma(W_2\sigma(W_1X))^\top \otimes I_{N_3}].$$

Degeneracy for nonlinear k-layer networks The following statement generalizes the above results about degeneracy of nonlinear two-hidden layer networks to those of nonlinear multilayer networks with an arbitrarily depth. We use the same notation as above unless otherwise specified.

Theorem 3. *Let H be a positive integer. Let $h_k = W_k\sigma(h_{k-1}) \in \mathbb{R}^{N_k,n}$ for $k \in \{2, \ldots, H+1\}$ and $h_1 = W_1X$, 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 $w = \mathrm{vec}(W_1, \ldots, W_{H+1}) \in \mathbb{R}^{dN_1+N_1N_2+N_2N_3+\cdots+N_HN_{H+1}}$. 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 $k \in \{1, \ldots, H+1\}$ such that $N_kN_{k-1} > n \cdot \min(N_k, N_{k+1}, \ldots, N_{H+1})$ where $N_0 = d$ and $N_{H+1} = d'$ (i.e., overparametrization), there exists a zero eigenvalue of Hessian $\nabla^2 L(w^*)$.*

Proof. Since $L(w) = \frac{1}{2}\|\hat{Y}_n(w) - Y\|_F^2 = \frac{1}{2}\mathrm{vec}(\hat{Y}_n(w) - Y)^\top \mathrm{vec}(\hat{Y}_n(w) - Y)$, we have that $\nabla^2 L(w) = \nabla \mathrm{vec}(\hat{Y}_n(w))\nabla \mathrm{vec}(\hat{Y}_n(w))^\top + \sum_{j=1}^{nd'} \nabla^2 \mathrm{vec}(\hat{Y}_n(w))_j(\mathrm{vec}(\hat{Y}_n(w))_j - \mathrm{vec}(Y)_j)$. Therefore,

$$\nabla^2 L(w^*) = \nabla \mathrm{vec}(\hat{Y}_n(w^*))\nabla \mathrm{vec}(\hat{Y}_n(w^*))^\top.$$

To obtain a formula of $\nabla \mathrm{vec}(\hat{Y}_n(w))$, we first write $\mathrm{vec}(\hat{Y}_n(w))$ recursively in terms of $\mathrm{vec}(h_k)$ as follows. For any $k \in \{2, \dots, H+1\}$,

$$\mathrm{vec}(h_k) = W_k \sigma(h_{k-1}) = [I_n \otimes W_k]\mathrm{vec}(\sigma(h_{k-1})) = [I_n \otimes W_k]\sigma(\mathrm{vec}(h_{k-1})),$$

where the last equality follows the fact that σ represents the elementwise application of nonlinear activation function (for an input of any finite size). Thus,

$$\mathrm{vec}(\hat{Y}_n(w)) = \mathrm{vec}(h_{H+1}) = [I_n \otimes W_{H+1}]\sigma(\mathrm{vec}(h_H)),$$

and we can expand $\mathrm{vec}(h_H)$ recursively by the above equation of $\mathrm{vec}(h_k)$ for any $k \in \{2, \dots, H+1\}$.

Moreover, for any $k \in \{2, \dots, H+1\}$,

$$\mathrm{vec}(h_k) = W_k \sigma(h_{k-1}) = [\sigma(h_{k-1})^\top \otimes I_{N_k}]\mathrm{vec}(W_k),$$

and

$$\mathrm{vec}(h_1) = W_1 X = [X^\top \otimes I_{N_1}]\mathrm{vec}(W_1).$$

From these, we obtain that

$$\nabla_k \mathrm{vec}(\hat{Y}_n(w))^\top = \tilde{h}_k[\sigma(h_{k-1})^\top \otimes I_{N_k}] \quad \text{for any } k \in \{2, \dots, H+1\},$$

and

$$\nabla_1 \mathrm{vec}(\hat{Y}_n(w))^\top = \tilde{h}_1[X^\top \otimes I_{N_1}],$$

where

$$\tilde{h}_k = \tilde{h}_{k+1}[I_n \otimes W_{k+1}]\dot{\sigma}(h_k) \quad \text{for any } k \in \{1, \dots, H\},$$

and

$$\tilde{h}_{H+1} = I_{nN_{H+1}}.$$

Therefore, for any $k \in \{2, \dots, H+1\}$,

$$\nabla_k \mathrm{vec}(\hat{Y}_n(w))\nabla_k \mathrm{vec}(\hat{Y}_n(w))^\top = [\sigma(h_{k-1}) \otimes I_{N_k}]\tilde{h}_k^\top \tilde{h}_k[\sigma(h_{k-1})^\top \otimes I_{N_k}],$$

which is a $N_k N_{k-1}$ by $N_k N_{k-1}$ matrix, and

$$\nabla_1 \mathrm{vec}(\hat{Y}_n(w))\nabla_1 \mathrm{vec}(\hat{Y}_n(w))^\top = [X \otimes I_{N_1}]\tilde{h}_1^\top \tilde{h}_1[X^\top \otimes I_{N_1}],$$

which is a $N_1 d$ by $N_1 d$ matrix. Here, the rank of $\nabla_k \mathrm{vec}(\hat{Y}_n(w))\nabla_k \mathrm{vec}(\hat{Y}_n(w))_k^\top$ is at most $n \cdot \min(N_k, N_{k+1}, \dots, N_{H+1})$ for any $k \in \{1, \dots, H+1\}$ where $N_{H+1} = d'$, because the rank of the product of matrices is at most the minimum of the ranks of matrices. This implies that for any $k \in \{1, \dots, H+1\}$, $\nabla_k \mathrm{vec}(\hat{Y}_n(w))\nabla_k \mathrm{vec}(\hat{Y}_n(w))^\top$ has a zero eigenvalue if $N_k N_{k-1} > n \cdot \min(N_k, N_{k+1}, \dots, N_{H+1})$ where $N_0 = d$ and $N_{H+1} = d'$. Therefore, the Hessian $\nabla^2 L(w^*) = \nabla \mathrm{vec}(\hat{Y}_n(w^*))\nabla \mathrm{vec}(\hat{Y}_n(w^*))^\top \succeq 0$ has (at least) a zero eigenvalue if there exists $k \in \{1, \dots, H+1\}$ such that $N_k N_{k-1} > n \cdot \min(N_k, N_{k+1}, \dots, N_{H+1})$ where $N_0 = d$ and $N_{H+1} = d'$.

□

If σ is additionally assumed to be ReLU, we can write that for any $k \in \{1, \dots, H+1\}$,

$$\mathrm{vec}(\hat{Y}_n(w^*)) = \nabla_k \mathrm{vec}(\hat{Y}_n(w^*))^\top \mathrm{vec}(W_k^*).$$

The directions $\Delta_k \mathrm{vec}(w) = \mathrm{vec}(0, \dots, 0, \Delta W_k, 0, \dots, 0)$ found in the proof of Theorem 3 such that

$$\Delta_k \mathrm{vec}(w)^\top \nabla^2 L(w^*) \Delta_k \mathrm{vec}(w) = 0$$

corresponds to

$$\nabla_k \mathrm{vec}(\hat{Y}_n(w^*))^\top \Delta W_k = 0.$$

Note that $\nabla_k \mathrm{vec}(\hat{Y}_n(w))$ depends on W_k only because of the dependence of the ReLU activations on W_k. Thus, at a differentiable point w^*, there exists a sufficiently small $\epsilon > 0$ such that the model outputs on the training dataset $\hat{Y}_n(w)$ does not change in these directions $\Delta_k \mathrm{vec}(w) = \mathrm{vec}(0, \dots, 0, \Delta W_k, 0, \dots, 0)$ as

$$\begin{aligned}
\mathrm{vec}(\hat{Y}_n(w^* + \epsilon \Delta_k \mathrm{vec}(w))) &= \nabla_k \mathrm{vec}(\hat{Y}_n(w^*))^\top (\mathrm{vec}(W_k^*) + \epsilon \Delta W_k) \\
&= \nabla_k \mathrm{vec}(\hat{Y}_n(w^*))^\top \mathrm{vec}(W_k^*) \\
&= \mathrm{vec}(\hat{Y}_n(w^*)).
\end{aligned}$$

6.3 For linear networks SGD selects flat minima which imply robust optimization which provides maximization of margin

In the next subsections we extend previous results (*28*) to relate minimizers that corresponds to flatness in the minimum to robustness: thus SGD performs robust optimization of a certain type. In particular, we will show that robustness to perturbations is related to regularization (see (*28, 29*)). We describe the separable and linear case.

The argument in (*11*) shows why SGDL (stochastic gradient descent Langevin) selects degenerate minimizers and, among those, the ones corresponding to larger flat regions of the loss. In fact SDGL shows concentration in probability – *because of the high dimensionality* – of its asymptotic distribution for minima that are the most robust to perturbations of the weights. The above fact suggests the following

Conjecture 1. *Under regularity assumptions on the landscape of the empirical minima, SGDL corresponds to the following robust optimization*

$$\min_{w} \max_{(\delta_1,\cdots,\delta_n)} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f_{w+\delta_i w}(\mathbf{x_i})). \tag{20}$$

It is especially important to note that *SGDL – and approximately SGD – maximize volume and "flatness" of the loss in weight space*.

6.3.1 SGD converges to a large-margin classifier for linear functions

Assume that the conjecture 1 holds and consider a binary classification problem. In linear classification we try to separate the two classes of a binary classification problem by a hyperplane $\mathcal{H} = \{x : w^\intercal x + b = 0\}$, where $w \in \mathbb{R}^d$. There is a corresponding decision rule of the form $y = sign(w^\intercal x + b)$ Let us now assume a *separability condition* on the data set: the decision rule makes no error on the data set. This corresponds to the following set of inequalities

$$y_i(w^\intercal x_i + b) > 0, \quad i = 1, \cdots, n \tag{21}$$

The separability conditions implies that the inequalities are feasible. Then (changing slightly the notation for the classification case) the robustness formulation of Equation 1 is equivalent to robustness of the classifier wrt perturbations δw in the weights which we assume here to be such that $||\delta w||_2 \leq \rho ||w||_2$ with $\rho \geq 0$. In other words, we require that $\forall \delta w$ such that $\|\delta w\|_2 \leq \rho \|w\|_2$:

$$y_i((w + \delta w)^\intercal x_i + b) \geq 0, \quad i = 1, \ldots, n \tag{22}$$

Further assume that $\|x_i\|_2 \approx 1$, then for $i = 1, \ldots, n$, if we let $\delta w = -\rho y_i x_i \|w\|_2 / \|x_i\|_2$,

$$y_i(w^\intercal x_i + b) \geq -y_i \delta w^\intercal x_i = \rho \|w\|_2 \|x_i\|_2 \approx \rho \|w\|_2$$

We obtain a *robust counterpart* of Equation (21)

$$y_i(w^\intercal x_i + b) \geq \rho ||w||_2, \quad i = 1, \cdots, n \tag{23}$$

Maximizing ρ – the margin – subject to the constraints Equation (23) leads to minimizing w, because we can always enforce $\rho ||w||_2 = 1$ and thus to

$$\min_{w,b} \{||w||_2 : y_i(w^\intercal x_i + b) \geq 1 \quad i = 1, \cdots, n\} \tag{24}$$

We have proven the following result (a variation of (*28*), page 302)

Lemma 6. *Maximizing flatness (as SGD does (11)) is equivalent to maximizing robustness and thus classification margin in binary classification of separable data.*

Remarks

1. Notice that the classification problem is similar to using the loss function $V(y, f(x)) = \log(1 + e^{-yf(x)})$ which penalizes errors but is otherwise very small in the zero classification error case. (*11*) implies that SGD maximizes flatness at the minimum, that is SGD maximizes δw.

2. The same Equation 24 follows if the maximization is with respect to spherical perturbations of radius ρ around each data point x_i. In either case the resulting optimization problems is equivalent to hard margin SVMs.

3. If we start from the less natural assumption that $||\delta w||_\infty \leq \rho||w||_2$ with $\rho \geq 0$ and assume that each of the d components $|(x_i)_j| \approx 1, \quad \forall i$, then the *robust counterpart* of Equation 21 is, since $(\delta w)^\intercal x \leq ||(\delta w)^\intercal||_{\ell_\infty}||x||_{\ell_1}$,

$$y_i(w^\intercal x_i + b) \geq (\delta w)^\intercal \sup \delta w = \rho||w||_2, \quad i = 1, \cdots, n \tag{25}$$

   .

4. In the non-separable case, the hinge loss $V(y, f(x)) = [1 - yf(x)]_+$, with y binary (used by (*29*)) – which is similar to the cross-entropy loss $V(y, f(x)) = \log(1 + e^{-yf(x)})$ – leads to the following robust minimization problem

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b) + \rho||w||_2]_+. \tag{26}$$

   Note that the robust version of this worst-case loss minimization is not the same as in classical SVM because the regularization term is inside the hinge loss. Note also that standard regularization is an upper bound for the robust minimum since

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b) + \rho||w||_2]_+ \leq \min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(w^\intercal x_i + b)]_+ + \rho||w||_2]. \tag{27}$$

   In the case of the square loss, robust optimization gives with $||\delta w||_2 \leq \rho||w||_2$

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [(y_i - w^\intercal x_i)^2 + \rho^2||w||_2]_+. \tag{28}$$

   The summary here is that depending on the loss function and on the uncertainity set allowed for the perturbations δw one obtains a variety of robust optimization problems. In general they are not identical to standard regularization but usually they contain a regularization term. Notice that a variety of regularization norms (for instance ℓ_1 and ℓ_2) guarantee CV_{loo} stability and therefore generalization. The conclusion is that *in the case of linear networks and linearly separable data, SGD provides a solution closely related to hinge-loss SVM.*

6.3.2 Robustness wrt Weights and Robustness wrt Data

A natural intuition is that several forms of stability are closely related. In particular, *perturbations of the weights follow from perturbations of the data points in the training set* because the function $f(x)$ resulting from the training is parametrized by the weights that depend on the data points S_n: $f(x) = f(x; w(z_1, \cdots, z_n))$, with $z = (x, y)$[1] . Thus *flat regions in weight space of the global minimum of the empirical loss indicate stability with respect to the weights; the latter in turn indicates stability with respect to training examples*.

[1] In a differentiable situation, one would write $df = \frac{df}{dw}\frac{dw}{dS}dS$ and $dw = \frac{dw}{dS}dS$. The latter equation would show that perturbations in the weights depend on perturbations of the training set S.

6.4 Polynomial Networks

There are various reasons why it is interesting to approximate nonlinear activation functions with univariate polynomials thus leading to deep polynomial networks. For this paper polynomial approximations of deep networks justify the use of Bezout theorem to characterize minima of the loss function. In addition, it is worthwhile to remark that the good empirical approximation by polynomial networks of the main properties of ReLU networks implies that specific properties of the ReLU activation function such as its discontinuous derivative, and the property of *non-negative homogeneity*, e.g. $\sigma(\alpha z) = \alpha\sigma(z)$ do not play any significant role. Furthermore, the characterization of certain properties of deep networks becomes easier by thinking of them as polynomial and thus analytic functions.

A generic polynomial $P_k(x)$ of degree k in d variables is in the linear space $\mathcal{P}_k = \cup_{s=0}^{k} H_s$ composed by the union of homogeneous polynomials $\mathcal{H}_s$, each of degree s. $\mathcal{H}_k$ is of dimensionality $r = dim\mathcal{H}_k = \binom{d-1+k}{k}$: the latter is the number of monomials and thus the number of coefficients of the polynomials in $\mathcal{H}_k$. A generic polynomial in $\mathcal{P}_k$ can always be written (see Proposition 3.5 in (*30*)) as

$$P_k(x) = \sum_{i=1}^{r} p_i(\langle w_i, x\rangle). \tag{29}$$

where p_i is a univariate polynomial of degree at most k and $\langle w_i, x\rangle)$ is a scalar product. In fact

$$P(x) = \sum_j a_j x^j = \sum_{k=1}^{N} c_k((w_k, x) + b_k)^n = \sum_{k=1}^{N} c_k \sum_{|j|=n} \binom{n}{j} u_k^j y^j \tag{30}$$

with $y = (x, 1)$ and $u_k = (w_k, b_k)$ so that $a_j = \binom{n}{j} \sum_{k=1}^{N} c_k u_k^j$.

Notice that the representation of a polynomial in d variables with total degree $\leq k$ (so dimension N) with monomial or any other fixed basis involves exactly N parameters to optimize. In the representation that is produced by a singe hidden layer in a network as $\sum_k a_k(w_k \cdot x + b_k)^k$, there are $(d + 2)N$ weights.

6.4.1 Polynomial approximation of deep networks

First, we discuss the univariate case. Let C^* be the class of all continuous, univariate, 2π-periodic functions on $\mathbb{R}$, equipped with the supremum norm $\|\cdot\|^*$, and $\mathbb{H}_n$ be the class of all trigonometric polynomials of order $< n$. For $f \in C^*$, let

$$E_n^*(f) = \min_{T\in\mathbb{H}_n} \|f - T\|^*.$$

The following simple theorem was proved by Czipser and Freud (*31, Theorem 1*):

Theorem 4. *Let $r \geq 1$ be an integer, $f \in C^*$ be r times continuously differentiable. For integer $n \geq 1$, and $\epsilon > 0$, if T is a trigonometric polynomial of order $< n$ such that*

$$\|f - T\|^* \leq \epsilon,$$

then

$$\|f^{(r)} - T^{(r)}\|^* \leq 3 \cdot 2^r n^r \epsilon + 4E_n^*(f^{(r)}).$$

where the minimum is taken over all trigonometric polynomials P of order $< n$.

To discuss the aperiodic case, let C be the class of all continuous functions on $[-1, 1]$, equipped with the supremum norm $\|\cdot\|$, and Π_n be the class of all algebraic polynomials of degree $< n$. For $f \in C$, we define

$$E_n(f) = \min_{P\in\Pi_n} \|f - P\|.$$

Writing $\phi(x) = \sqrt{1 - x^2}$, it is easy to deduce the following corollary of Theorem 4 using the standard substitution $x = \cos\theta$:

Corollary 2. *Let $f \in C$ be continuously differentiable, $n \geq 1$ and $P \in \Pi_n$. If*

$$\|f - P\| \leq \epsilon, \tag{31}$$

then

$$\|(f' - P')\phi\| \leq 3 \cdot 2^r n^r \epsilon + 4E_{n-1}(\phi f'). \tag{32}$$

Thus, if f is a smoothened version of the relu function, and P is an approximation to f satisfying (31), then $\nabla_{\mathbf{w},b} f(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ can be approximated by $\nabla_{\mathbf{w},b} P(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ in the weighted uniform norm as in (32).

The extension of Corollary 2 to higher derivatives is not so straightforward. For $\gamma > 0$, we define for $f \in C$,

$$\|f\|_\gamma = \|f\| + \sup_{j \geq 0} 2^{j\gamma} E_{2^j}(f),$$

and let W_γ be the space of all $f \in C$ for which $\|f\|_\gamma > 0$. Thus, if $r \geq 1$ is an integer, and f is r times continuously differentiable, then $f \in W_r$. However, W_r is the not the same as the space of all r times continuously differentiable functions on $[-1, 1]$. A complete characterization of the spaces W_γ is given in (*32, Chapter 8, Section 7*). The following theorem can be proved using the Bernstein inequality (*32, Chapter 8, Theorem 7.6*).

Theorem 5. *Let $r \geq 1$ be an integer, $\gamma > r$, $f \in W_\gamma$. Let $n \geq 1$ be an integer, $P \in \Pi_n$, and (31) hold for some $\epsilon > 0$. Then*

$$\|(f^{(r)} - P^{(r)})\phi^r\| \leq cn^r \{\epsilon + n^{-\gamma} \|f\|_\gamma\}, \tag{33}$$

where $c > 0$ is a constant depending only on r and γ.

Theorem 5 can be extended to the multi-variate setting using tensor products. Finally, we note that theorems analogous to Theorem 5 are given in (*33*). However, these theorems merely assert the existence of neural networks evaluating a sigmoidal $\tanh$ activation function so that the derivatives of these networks approximate the derivatives of the function.

6.4.2 Deep polynomial networks behave as deep networks

Nonlinear activations in a deep networks can be approximated up to an arbitrary accuracy over a bounded interval by a univariate polynomial of appropriate degree (see (*3*)) which approximates well also the derivative of the function (see SI 6.4.1). Since so much is known about polynomials, several interesting results follow from these approximation properties such as a characterization of when deep convolutional networks can be exponential better than shallow networks in terms of representational power ((*2*)). We show here empirical evidence that the puzzling generalization properties of deep convolutional networks hold for networks with the same architecture in which all the ReLU's have been replaced by a univariate polynomial approximation (of degree 10 in our simulations). The resulting 5-layers networks are obviously polynomial networks, that compute a polynomial in the input x – and *that are also polynomials in the weight parameters*. Figure 9 demonstrates that polynomial and ReLU networks with the same architecture have very similar performance in terms of training and testing error. The same puzzling lack of overfitting in the overparametrized case can be seen in Figure 11. Together Figures 10,11 include all the puzzles discussed in the main text and described by (*9*).

As we discussed in the main text, Figure 6 shows that the increase in the number of parameters does not induce overfitting, at least in terms of classification error. We show in Figure 11 that the same behavior is shown for polynomial networks. As shown in Figure 12, the norm of the weights increases during training until an asymptotic value is reached,

The same very high degree polynomial represented by a 5-layers network with univariate polynomial activation of its hidden units can be, in principle, parametrized in a standard way in terms of all the relevant monomials, with each parameter corresponding to the coefficient of one of the monomials6.4. Such a parametrization corresponds to a one-hidden layer network where the weights of the first layer are fixed (for instance equal to 1), each hidden unit computes one of the monomial from the d inputs and the learnable weights are from the hidden units to the output.

It is well known that the coefficients of the polynomial in the standard parametrization can be learned by using gradient descent or stochastic gradient descent. This corresponds to optimizing a linear network with square loss which is equivalent to minimizing a possibly degenerate quadratic loss function.

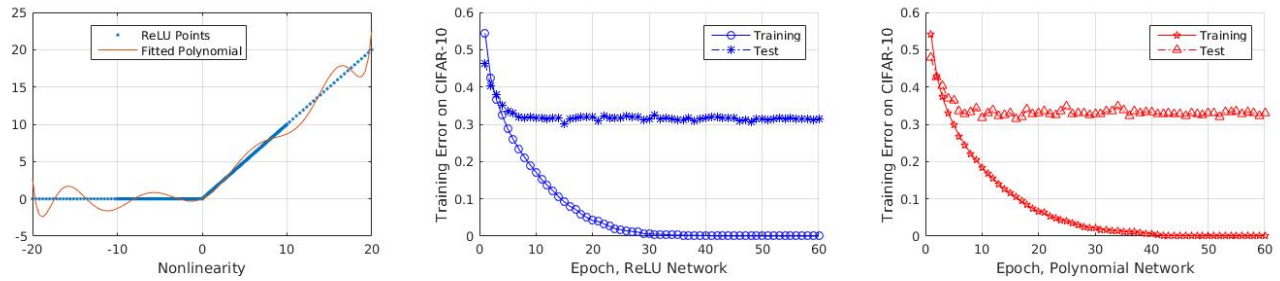


Figure 9: *A standard convolutional deep network is converted into a polynomial function by replacing each of the REL units with a univariate polynomial approximation of the ReLU function. As long as the nonlinearity approximates the ReLU function well (especially near 0), the "polynomial network" performs quantitatively similarly to the corresponding ReLU net. The polynomial shown in the inset on the left is of degree* 10.

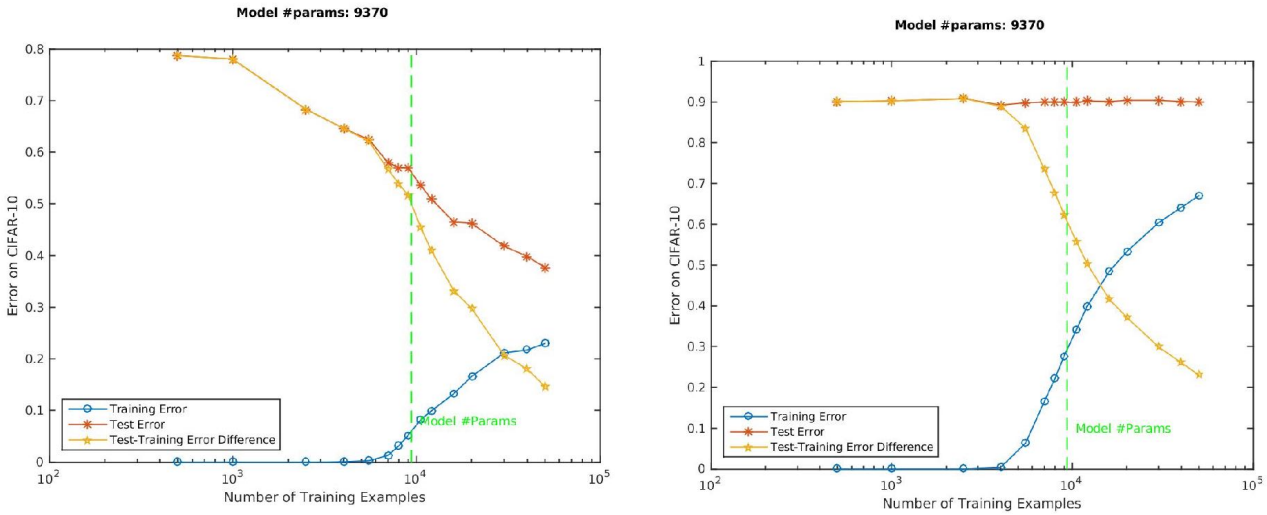


Figure 10: *The figure (left) shows the behavior of a polynomial deep network trained on subsets of the CIFAR database. The figure on the right shows the same network trained on subsets of the CIFAR database in which the labels have been randomly scrambled. The network is a 5-layer all convolutional network (i.e., no pooling) with 16 channels per hidden layer , resulting in only* $W \approx 10000$ *weights instead of the typical* $300,000$*. Neither data augmentation nor regularization is performed.*

6.5 Additional experiments

Figure 13 shows the testing error for an overparametrized linear network optimized under the square loss. The Figure is related to Figure 5 in the main text.

Figure 14 shows the behavior of the loss in CIFAR in the absense of perturbations. This should be compared with Figure 15 which shows the case of an overparametrized linear network under quadratic loss corresponding to the multidimensional equivalent of the degenerate situation of Figure 2. The nondegenerate, convex case is shown in Figure 16. Figure

As shown in Figure 17 and Figure 18, qualitative properties of deep learning networks under the crossentropy loss function seem to hold, as expected, under the quadratic loss.

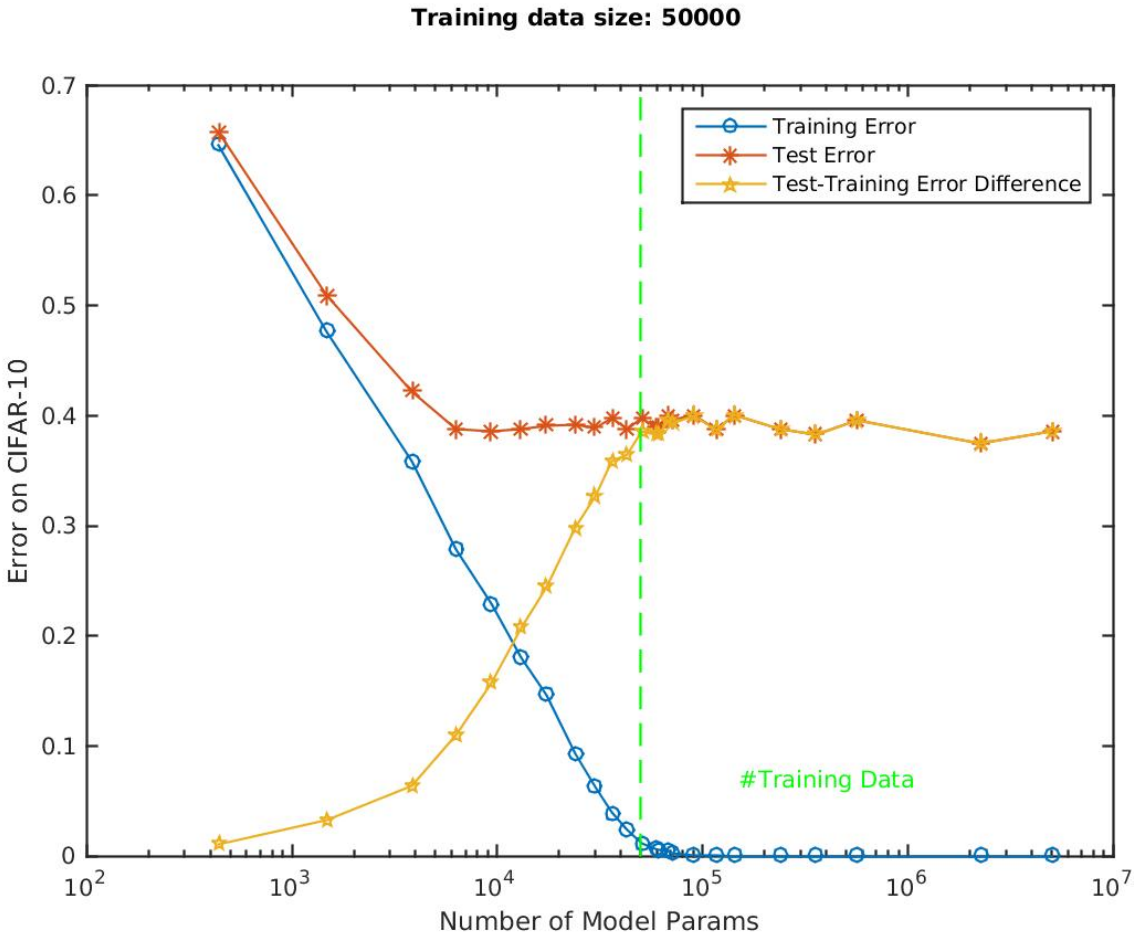


Figure 11: *The previous figures show dependence on N – number of training examples – for a fixed architecture with W parameters. This figure shows dependence on W for a fixed training set with N examples. The network is again a 5-layer all convolutional polynomial network. All hidden layers have the same number of channels. Neither data augmentation nor regularization is performed. The classical theory explains the generalization behavior on the left; the challenge is to explain the lack of overfitting for $W > n$. As shown here, there is zero error for $W \geq n$.*

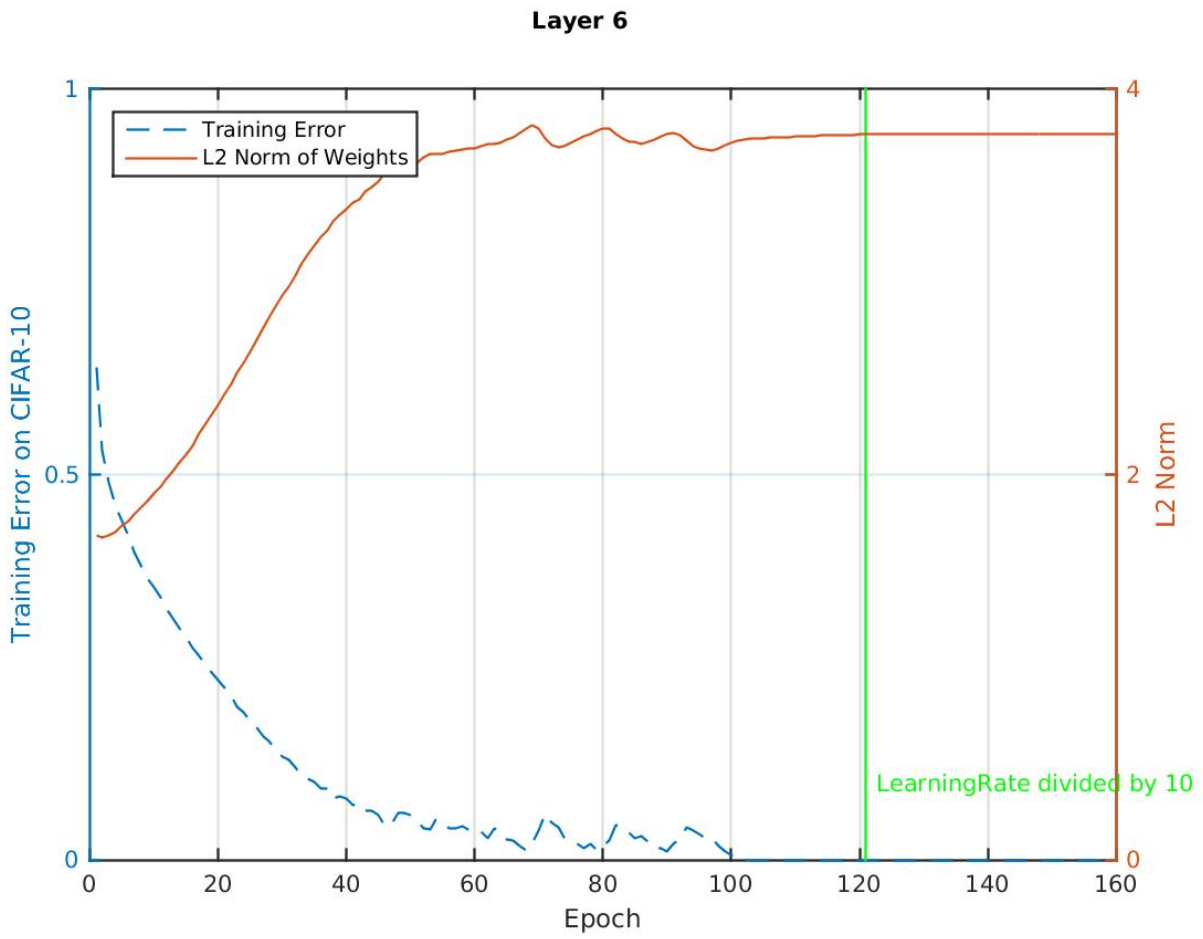


Figure 12: *The norm of the weights increases with training epochs of SGD until a certain level after which it does not change anymore. The behaviour shown here is typical for all layers for CIFAR-10.*

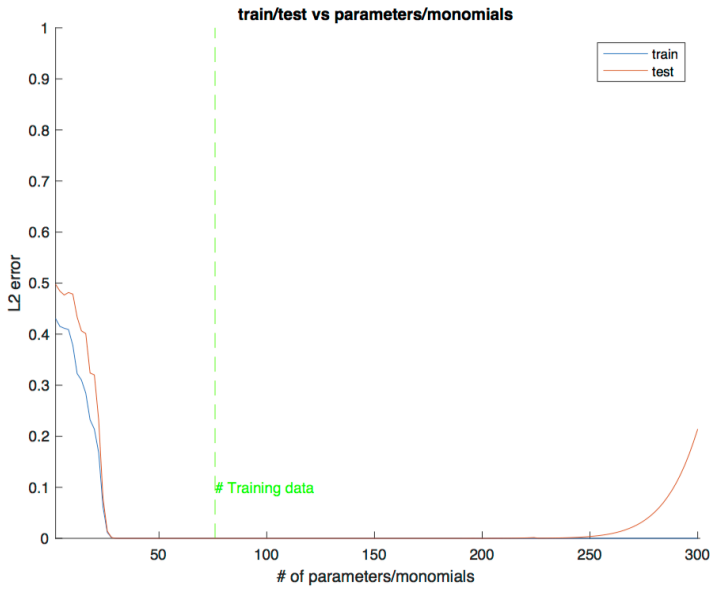


Figure 13: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\phi(X)$*) with a degenerate Hessian of the type of Figure 2. The feature matrix is a polynomial with increasing degree, from 1 to 300. The square loss is plotted vs the number of monomials, that is the number of parameters. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with frequency* $f = 4$ *on the interval* $[-1, 1]$*. The number of training points where* 76 *and the number of test points were* 600*. The solution to the over-parametrized system was the minimum norm solution. More points where sampled at the edges of the interval* $[-1, 1]$ *(i.e. using Chebyshev nodes) to avoid exaggerated numerical errors. The figure shows how eventually the minimum norm solution overfits.*

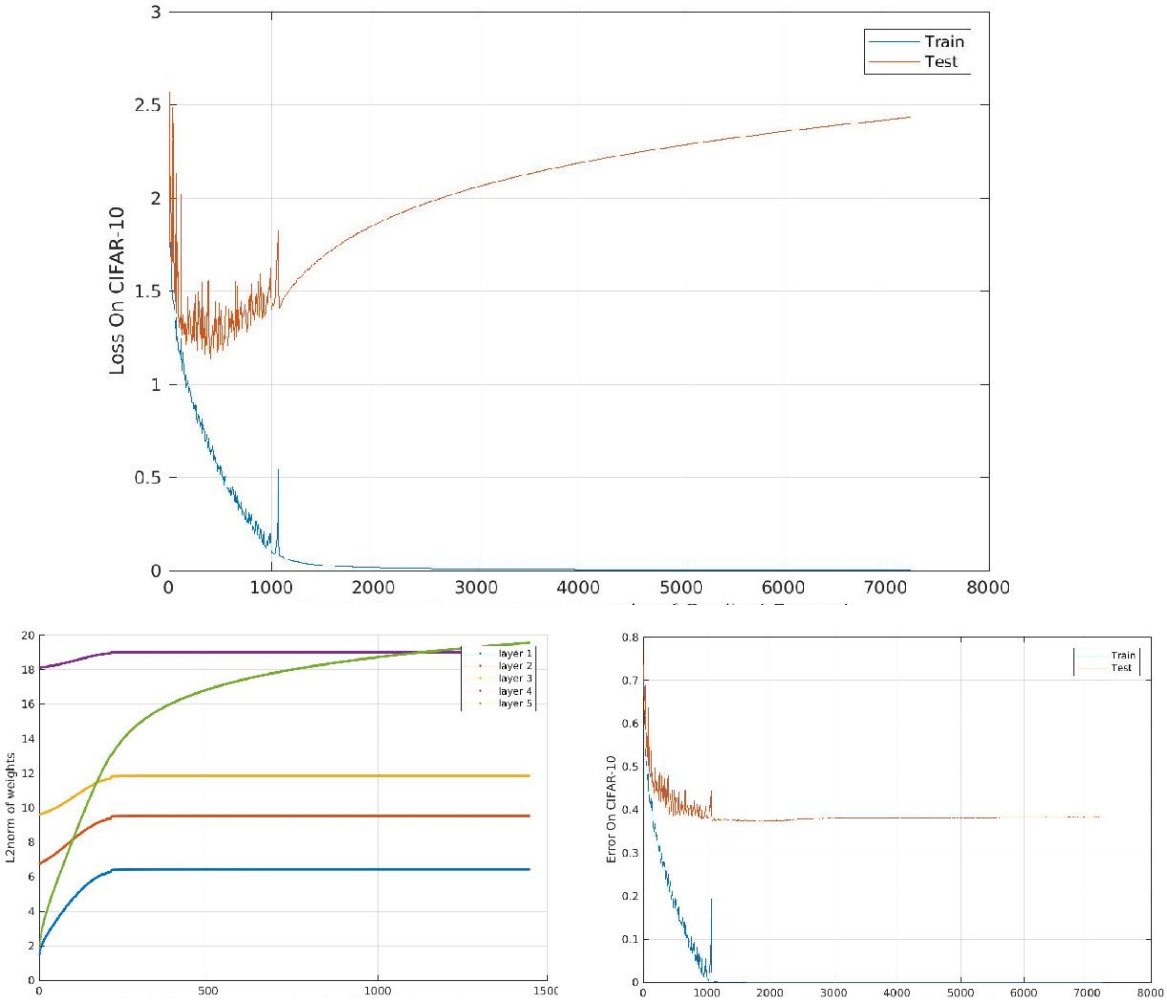
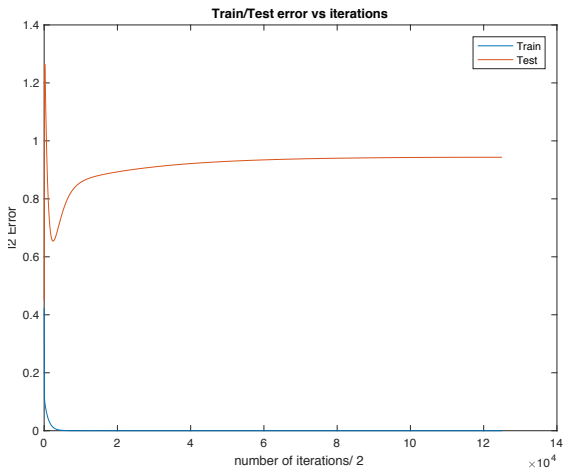


Figure 14: *Same as Figure 4 but without perturbations of weights. Notice that there is some overfitting in terms of the testing loss. Classification however is robust to this overfitting (see text).*



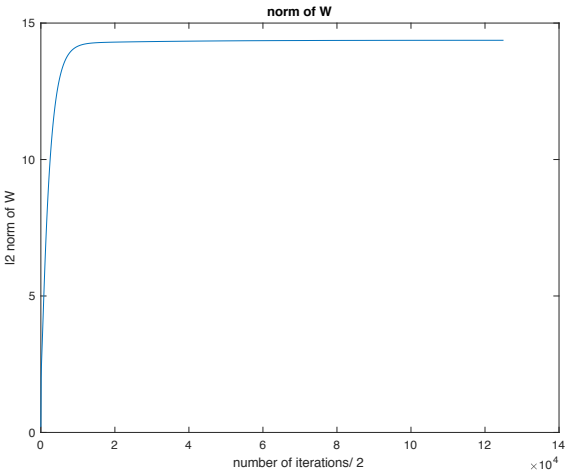


Figure 15: *Training and testing with the square loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) with a degenerate Hessian of the type of Figure 2. The feature matrix* $\phi(X)$ *is a polynomial with degree 30. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with 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.*

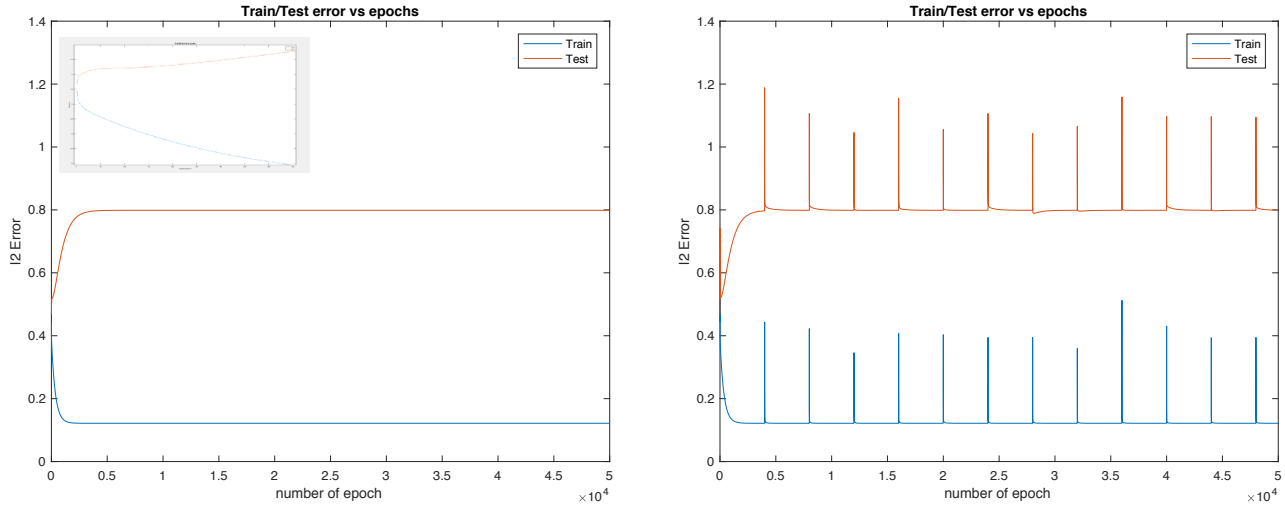


Figure 16: *The graph on the left shows training and testing loss for a linear network in the feature space (i.e.* $y = W\Phi(X)$*) in the nondegenerate quadratic convex case. The feature matrix* $\phi(X)$ *is a polynomial with degree 4. The target function is a sine function* $f(x) = sin(2\pi fx)$ *with 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.*

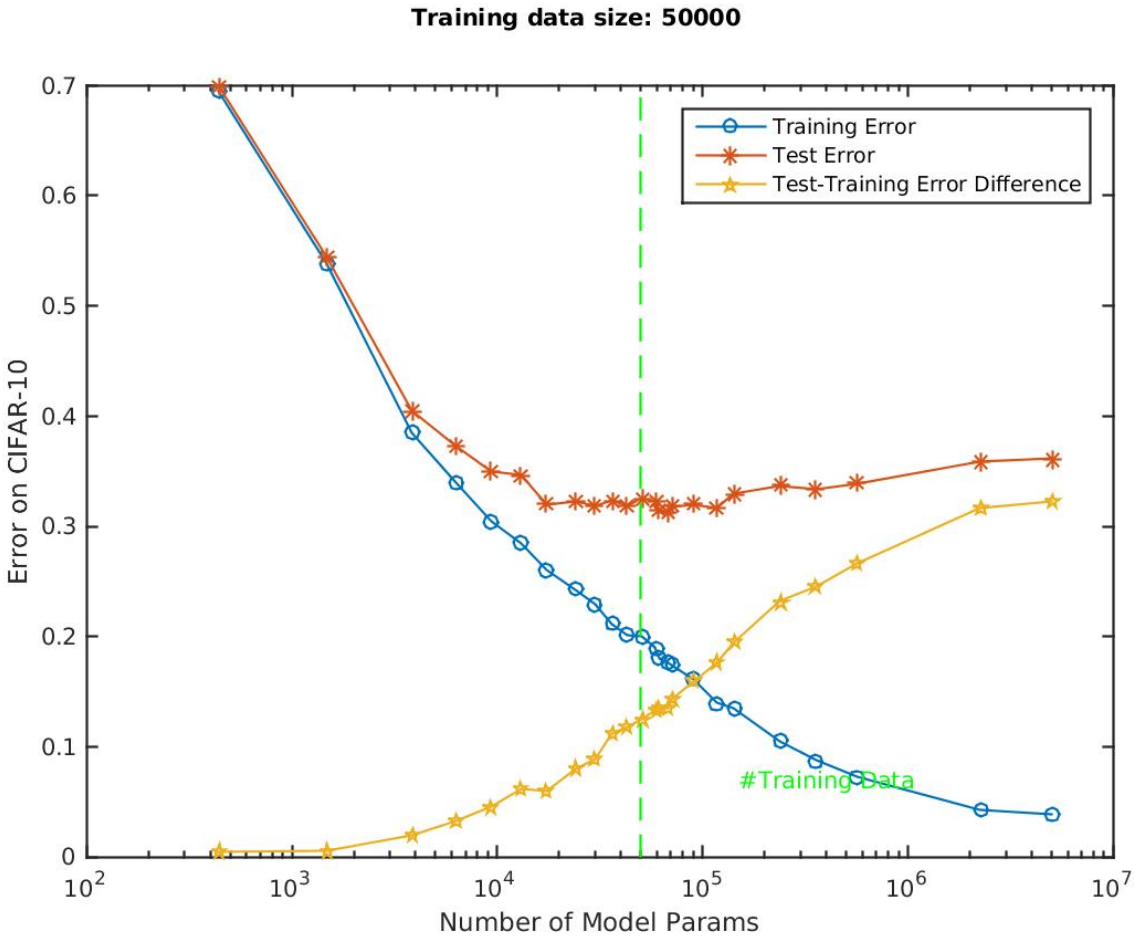


Figure 17: *The figure shows the behavior of a deep polynomial network trained on the CIFAR database, using the square loss. To be compared with Figure 6*

.

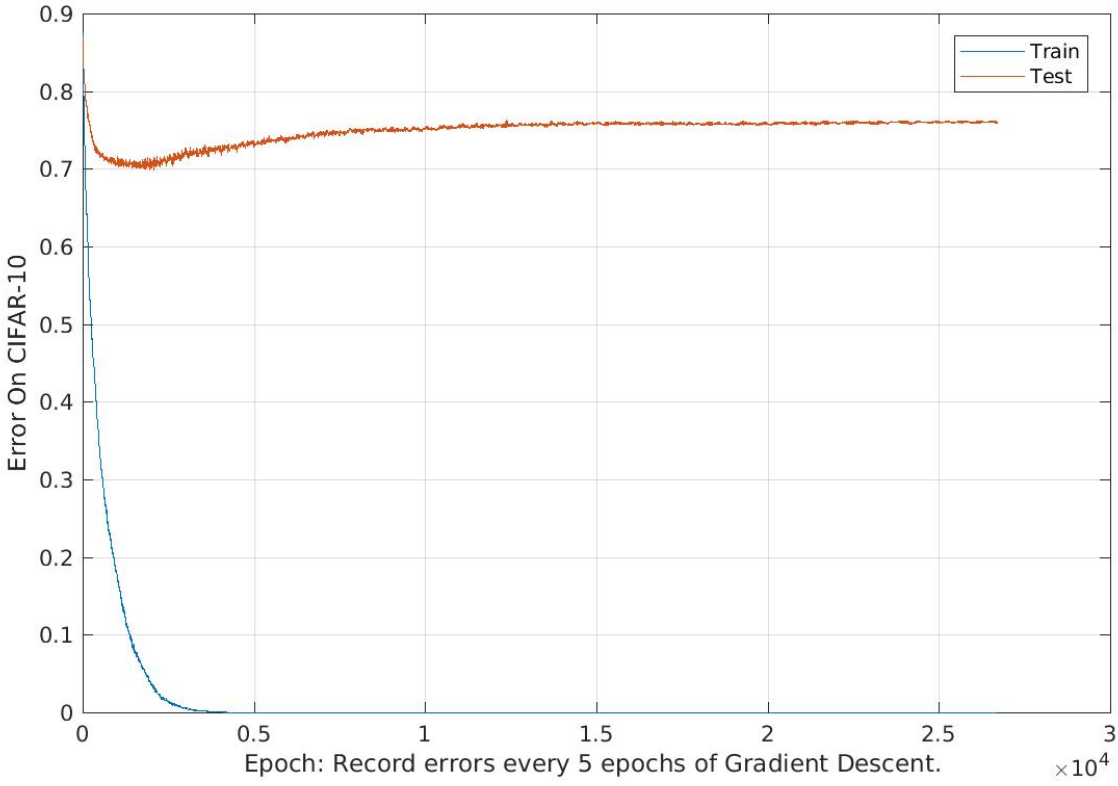


Figure 18: *Classification error on CIFAR obtained with GD optimizing the square loss risk. The training set has* 2000 *examples and the network has* 188810 *parameters. Overfitting appears here for the classification error.*