# Synthetic Generation of Local Minima and Saddle Points for Neural Networks

**Luigi Malagò** [1] **Dimitri Marinelli** [1]

## Abstract

In this work-in-progress paper, we study the landscape of the empirical loss function of a feed-forward neural network, from the perspective of the existence and nature of its critical points, with respect to the weights of the network, for synthetically-generated datasets. Our analysis provides a simple method to achieve an exact construction of one or more critical points of the network, by controlling the dependent variables in the training set for a regression problem. Our approach can be easily extended to control the entries of the Hessian of the loss function allowing to determine the nature of the critical points. This implies that, by modifying the dependent variables for a sufficiently large number of points in the training set, an arbitrarily complex landscape for the empirical risk can be generated, by creating arbitrary critical points. The approach we describe in the paper not only can be adopted to generate synthetic datasets useful to evaluate the performance of different training algorithms in the presence of arbitrary landscapes, but more importantly it can help to achieve a better understanding of the mechanisms that allow the efficient optimization of highly non-convex functions during the training of deep neural networks. Our analysis, which is presented here in the context of deep learning, can be extended to the more general case of the study of the landscape associated to other quadratic loss functions for nonlinear regression problems.

## 1. Introduction

In statistics and machine learning, a common approach for the estimation of the parameters of a model, for instance for regression or classification problems, is based on the minimization of an average loss function, called empirical risk, over the points in the training set. The empirical risk, defined over the space of the parameters of the model, gives a measure of the fit of the model to describe the map between input and output variables with respect to the points in the dataset. A classical example is given by the use of the square loss for regression problems. During training, standard inference procedures tend to minimize the empirical loss over the training set, either with closed formula solutions, when they exist, such as for ordinary least squares, or by iterative algorithms, as in the case of stochastic gradient descent. The landscape of the average loss function represents how the measure of the fit of a given model to data changes for different assignments of the parameters of the model itself. Notice that the minimum of the empirical risk, especially for highly nonlinear models, may lead to overfitting, that is why in practice penalization terms are often added during training, changing the landscape of the function itself.

In the following we refer to the common distinction in machine learning between model fitting, i.e., the procedure for the estimation of the parameters of the model, which corresponds to a search in the parameter space with the purpose of minimizing the average loss function; and model selection, i.e., the procedure for the choice of a specific model, which in turns determine the space of parameters and the landscape of the empirical for given a dataset. The landscape of the average loss function depends from one side by the points in the training set, and from the other by model selection, for instance by the choice of the regression function. For deep neural networks, the landscape is determined by the choice of the topology of the network, such as the number of hidden layers, the number of their hidden units, and the activation functions, which produce different landscapes for the empirical loss function.

The characterization of the landscape of the average loss function reveals essential information about the coupling between the chosen model and the inference task (Mei et al., 2016), for instance about the fit of the model over the data, the robustness of the solution, some indicators which may suggest the presence of overfitting, possible symmetries in the space of parameters, and so on. Moreover, the properties of the landscape are strictly connected to the dif-

[1]Romanian Institute of Science and Technology (RIST), Cluj-Napoca, Romania. Correspondence to: Dimitri Marinelli <marinelli@rist.ro>, Luigi Malagò <malago@rist.ro>.

*Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning*, Sydney, Australia, 2017. 

ficulties that an algorithm has to face in the optimization of the empirical risk itself. Smooth landscapes, as well as the presence of few local minima and plateaux, imply, in general, that local search methods will be more effective during training (Dauphin et al., 2014).

The high dimensionality of the parameter space of a neural network makes more challenging the task of studying the landscape of the empirical risk. Tools to find, generate, and approximate critical points of the network, i.e., those points where the gradient of the empirical risk is vanishing, can be particularly useful in this type of analysis. In this paper, we propose a technique that can help in constructing toy models and counter examples for the study of the optimization process during training. Here we restrict our analysis to the case of feed-forward deep neural networks with fully connected layers for regression tasks. Our approach, however, can be easily generalized to other topologies of feed-forward networks, such as convolutional neural networks.

Recently, (Swirszcz et al., 2017) have suggested that synthetic datasets, and the study of their landscapes, can provide interesting insights on the learning process during training. On the other side, the study of the effects of ad-hoc modifications of the points of a dataset, during training or testing, is central for the growing literature on adversarial examples in deep learning (Goodfellow et al., 2014a), and more in general in machine learning (Dalvi et al., 2004; Lowd & Meek, 2005). However, notice that differently from adversarial examples, which are constructed by perturbing the inputs, in this paper we study perturbations of the dependent variables in regression problems.

The paper is organized as follows. In Section 2, after introducing the standard notation for the computation of the output of a feed-forward network, and for the average loss function used for regression problems, we present our approach to the generation of a synthetic dataset which allows us to generate landscape with an arbitrary number of critical points. Next, we present a simple toy example for which we can choose new labels in the dataset for which two different assignments of the weights have vanishing gradient. For this example, we also explore the landscape of the empirical risk between these two critical points. Finally, in Section 3 we conclude the paper and present future directions of research.

## 2. Non-Linear Regression

We start this section by introducing standard notation in the literature of deep neural networks. Let us assume a training set of $m$ couples $\left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y},\ i = 1, \dots, m\right\}$ where $d_x$ and $d_y$ denote the dimensions of the input and the output

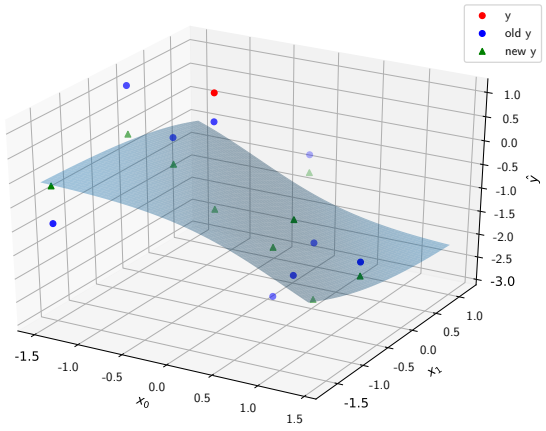


*Figure 1.* In this plot the dependent variables of a sub set of points of the dataset (`old y` blue circles) are changed (`new y` green triangles). The new dataset (green triangles plus the red circle) is built to make the neural network a critical network. The blue surface represents the output of the neural network for the particular assignment of weights. The old dataset (blue and red circles) were built adding noise to the output of the network. The `new y` are mostly closer to the surface.

layer, respectively. We can characterize the topology of a feed-forward network by fixing the number of nodes in each hidden layer by $(d_0, d_1, \dots, d_H, d_{H+1})$, with $d_0 = d_x$ and $d_{H+1} = d_y$, for a network with $H$ hidden layers. Moreover, let us denote with $a_l$ a real function $a_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ such that

$$a_l(\boldsymbol{z}) = \begin{bmatrix} \widetilde{a}_l(z_1) \\ \vdots \\ \widetilde{a}_l(z_{d_l}) \end{bmatrix},$$

i.e., $a_l$ acts element-wise on a vector. Let $W_l \in \mathbb{R}^{d_{l+1}, d_l}$, with $l = 0, \dots, H$, the matrix of weights associated to the fully connected network, we can write the output $\hat{\boldsymbol{y}}$ of a neural network as

$$\hat{\boldsymbol{y}}(\boldsymbol{x}; \boldsymbol{W}) = a_l(W^l a_{l-1}(\dots a_1(W^1 a_0(W^0 \boldsymbol{x})) \dots)) . \tag{1}$$

In the following, without any loss of generality, to maintain a more compact notation we omit for each layer $l$, with $l = 0, \dots, H$, the biases $b_l$.

In regression analysis, we look for minimization of the empirical risk which in the usual case of the square loss function reads

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{W}\right)\right)^2$$

The global minima of the empirical risk with respect to the weights of the network $\boldsymbol{W}$ represent the target of the

search during training. However, first-order optimization algorithms, which are usually used in training of neural networks, cannot discriminate between global minima and other types of critical points such as local minima or saddle points. Therefore, it becomes crucial to study the landscape of the average loss function. It has been argued that the power of neural network resides in the nature of the datasets that have been used to train the network (Lin et al., 2016), if this hypothesis would be true, than that would also be its weakness: neural networks can be easily fooled using synthetic datasets (Goodfellow et al., 2014a).

In this work-in-progress paper, we aim to study the problem of the landscape, from the point of view of the train set. We start from the computation of the the gradient of the square loss function for regression problems

$$\begin{aligned}&\nabla_{\boldsymbol{W}}\mathcal{L} = \\ &= -\frac{2}{m}\sum_{i=1}^{m}\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}(\boldsymbol{x}^{(i)};\boldsymbol{W})^T\right]\left(\boldsymbol{y}^{(i)}-\hat{\boldsymbol{y}}\left(\boldsymbol{x}^{(i)};\boldsymbol{W}\right)\right).\end{aligned} \tag{2}$$

A critical point is defined to be an assignment $\boldsymbol{W}$ such that $\nabla_{\boldsymbol{W}}\mathcal{L}=0$. This provides a system of $N=\sum_{l=0}^{H} d_l d_{l+1}$ equations that are in general non linear in $\boldsymbol{W}$, therefore the training process requires local search algorithms, such as stochastic gradient descend, able to converge only to local minima of the target function.

If we are building a synthetic dataset, we can look at the equations in $\nabla_{\boldsymbol{W}}\mathcal{L}=0$, provided an arbitrary set of weights, as a linear system where the unknown are the $\boldsymbol{y}^{(i)}$s, constructing in this way a dataset for which the network with weights $\boldsymbol{W}$ is indeed a critical network.

If we choose to have a dataset with the same number of points of the number of weights, namely $m\equiv N$, the solution of the system is particularly simple, $\boldsymbol{y}^{(i)}=\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(i)};\boldsymbol{W})$ where $\mathring{\boldsymbol{x}}^{(i)}$ are points arbitrarily fixed. That would represent the perfect overfit. When $m>N$ the system is under-determined and the solution is in general not unique. We can arbitrarily fix a subset of datapoints $\left(\mathring{\boldsymbol{x}}^{(f)},\mathring{\boldsymbol{y}}^{(f)}\right)$ with $f=1,\dots,k\equiv m-N$, and input data $\mathring{\boldsymbol{x}}^{(i)}$ $i=k+1,\dots,m$ and solve the system for the remaining $\boldsymbol{y}^{(i)}$. The system would read $A_{\boldsymbol{W}}\,Y+b=0$ where

$$Y=\begin{pmatrix} y^{(k+1)} \\ \vdots \\ y^{(m)}\end{pmatrix},$$

$$\begin{aligned}A_{\boldsymbol{W}} &= \left[\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(k+1)};\boldsymbol{W})^T\right],\dots,\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(m)};\boldsymbol{W})^T\right]\right], \\ b &= -\sum_{i=k+1}^{m}\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(i)};\boldsymbol{W})^T\right]\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(i)};\boldsymbol{W}) \\ &+\sum_{i=1}^{k}\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(i)};\boldsymbol{W})^T\right]\left(\mathring{\boldsymbol{y}}^{(i)}-\hat{\boldsymbol{y}}(\mathring{\boldsymbol{x}}^{(i)};\boldsymbol{W})\right)\end{aligned} \tag{3}$$

therefore the rank of the square matrix $A$ of dimension $N\times N\,d_y$ determines whether there exists $N$ different solutions. This dataset would lead to critical neural network without knowing if it would represent a minimum, a maximum or a saddle point of the loss function.

If we are interested in creating a completely artificial dataset, a natural request would be to try to minimize $\mathcal{L}$. This is an optimization problem of a quadratic function subject to a linear constraint

$$\begin{aligned}&\underset{\left\{\boldsymbol{y}^{(i)}\right\}_{i=1}^{m}}{\arg\min}\ \mathcal{L} \\ &\text{s. t.}\quad \nabla_{\boldsymbol{W}}\mathcal{L}=0 \quad \text{for} \quad \left\{\boldsymbol{y}^{(i)}\right\}_{i=1}^{m}.\end{aligned}$$

However, we can use the calculation in the previous section to solve the constrained problem. If we split the dependent variables as in (3), namely $\mathring{Y}\equiv\left\{\mathring{\boldsymbol{y}}^{(f)}\right\}$ with $f=1,\dots,k\equiv m-N$ and $Y$, and $A_{\boldsymbol{W}}$ from the constraint in (3) is invertible, we can parametrize the submanifold of dimension $m-N$ defined by the constrains with

$$Y=-A_{\boldsymbol{W}}^{-1}\,b\left(\mathring{Y}\right). \tag{4}$$

In this subspace, the loss is at a critical point with respect to the dependent variable if

$$\nabla_{\mathring{Y}}\mathcal{L}=0. \tag{5}$$

This provide $m-N$ equations that can fix, together with Equation 4, all the dependent variables of the dataset.

A quite intriguing calculation is to evaluate two possible critical points for the same dataset. This phenomenon can occur because of the non-convexity of the loss function for the neural network. It could represent the case where a bad minimum and a global minimum appear in the same network. Or the case where one critical point is a saddle point and one is the minimum. This can help in studying the more efficient trajectories that a training algorithm should perform to avoid to be stacked for long time around non desired critical points. Let us assume we have two sets of weights that we want to be critical $\boldsymbol{W'}$ and $\boldsymbol{W''}$, the system becomes

$$A_{\boldsymbol{W'},\boldsymbol{W''}}Y+b=0 \quad \text{where} \quad A_{\boldsymbol{W'},\boldsymbol{W''}}\equiv\begin{bmatrix}A_{\boldsymbol{W'}} \\ A_{\boldsymbol{W''}}\end{bmatrix} \tag{6}$$

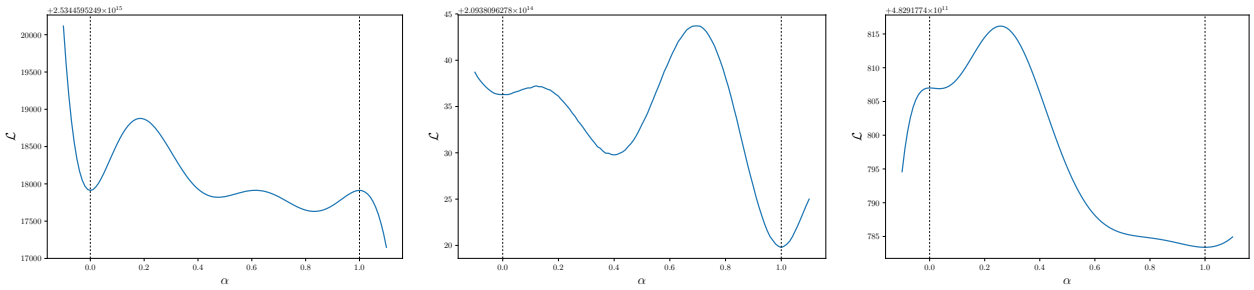

*Figure 2.* For $\alpha = 0$, the neural network has weights $\tilde{\boldsymbol{W}}$ (normally distributed) and, for the synthetic dataset, it is in a critical point. For $\alpha = 1$, the neural network has different weights $\boldsymbol{W}'$, which for the same dataset still represent a critical point. The plot shows how the value of the loss function changes along a linear combination of the two critical weight vectors, as in Equation 7 for different couples of critical weights $\tilde{\boldsymbol{W}}, \boldsymbol{W}'$. Notice the high values of the loss function, which are a consequence of our procedure which ensure the two critical points in the landscape of the loss function.

while $Y$ and $b$, are the same of (3). Here there are $2N$ equations, therefore the dimension of the dataset has to be $m \geq 2N$. Once the two critical networks are fixed, it is possible and very interesting to visualize the landscape as function

$$\boldsymbol{W} = (1 - \alpha)\, \boldsymbol{W'} + \alpha \boldsymbol{W''} \tag{7}$$

as proposed in (Goodfellow et al., 2014b).

Interestingly enough, we can also also move to the second-order analysis, and study the Hessian. We recall that if the Hessian is a positive defined matrix, then the critical point is a minimum. For this, we need to to evaluate a system of $N + N\,(N+1)\,/2$ equations. At a critical point, where the gradient of the empirical risk is vanishing, the Hessian matrix reads

$$\mathscr{H}\left[\mathcal{L}\right](\boldsymbol{W}) = \\ = -\frac{2}{m}\sum_{i=1}^{m}\left(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}\right)^{T} \nabla_{\boldsymbol{W}'}\left[\nabla_{\boldsymbol{W}}\left[\hat{\boldsymbol{y}}^{T}\right]^{T}\right] ,$$

where $\hat{\boldsymbol{y}}$ is a shorthand for $\hat{\boldsymbol{y}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{W}\right)$ clearly, at the perfect overfit, the Hessian is everywhere zero.

## 3. Conclusions and Future Works

This work-in-progress paper deals with the study of the landscape of the empirical risk for non-linear regression problems, in particular for feed-forward neural networks. We have presented a simple procedure which allows us to generate landscapes for the average loss function which admit among their critical points an arbitrary set of of points. Our method is general, in the sense that it can be applied not only in deep learning for any regression problem with a square loss function, independently from the topology of the network and the activation functions, but more in general for non-linear regression problems, whenever the gradient of the average loss function is linear in the dependent variables.

The method that we have proposed can be useful in different scenarios, for instance for the generation of fully-synthetic datasets, or for the perturbation of existing ones. From this perspective a natural extension to this work would be the study of the behavior of different training algorithms in presence of pathological landscapes, for which we could control the existence and nature of different critical points on the space of the weights, cf. (Shalev-Shwartz et al., 2017). Another analysis in which our method could be useful would be related to the study of the landscape of the empirical risk in presence of sharp and local minima (Hochreiter & Schmidhuber, 1997; Keskar et al., 2016; Dinh et al., 2017), which could be easily generated with our method, for fixed assignments of the weights. Finally, another scenario would be the study of impact of the present of saddle-points, which have been identified as one of the possible factors which slow down gradient descent methods in high-dimensional spaces (Dauphin et al., 2014).

As a concluding remark, we would like to extend this type of analysis to other types of loss functions, in non-linear regression, as well as to classification problems.

## Acknowledgements

The authors would like to thank one of the anonymous reviewers for his/her helpful and constructive comments.

## References

Dalvi, Nilesh, Domingos, Pedro, Mausam, Sanghai, Sumit, and Verma, Deepak. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 99–108, New York, NY, USA, 2004. ACM.

Dauphin, Yann, Pascanu, Razvan, Gülçehre, Çaglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv:1406.2572*, 2014.

Dinh, Laurent, Pascanu, Razvan, Bengio, Samy, and Bengio, Yoshua. Sharp minima can generalize for deep nets. *arXiv:1703.04933*, 2017.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014a.

Goodfellow, Ian J., Vinyals, Oriol, and Saxe, Andrew M. Qualitatively characterizing neural network optimization problems. *arXiv:1412.6544*, December 2014b.

Hochreiter, Sepp and Schmidhuber, Jürgen. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Keskar, Nitish Shirish, Mudigere, Dheevatsa, Nocedal, Jorge, Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv:1609.04836*, 2016.

Lin, H. W., Tegmark, M., and Rolnick, D. Why does deep and cheap learning work so well? *ArXiv e-prints*, 2016.

Lowd, Daniel and Meek, Christopher. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pp. 641–647, New York, NY, USA, 2005. ACM.

Mei, Song, Bai, Yu, and Montanari, Andrea. The Landscape of Empirical Risk for Non-convex Losses. *arXiv:1607.06534*, July 2016.

Shalev-Shwartz, Shai, Shamir, Ohad, and Shammah, Shaked. Failures of deep learning. *arXiv:1703.07950*, 2017.

Swirszcz, Grzegorz, Czarnecki, Wojciech Marian, and Pascanu, Razvan. Local minima in training of neural networks. *arXiv:1611.06310*, 2017.