Chapter 12

Shape of the energy landscape of neural networks

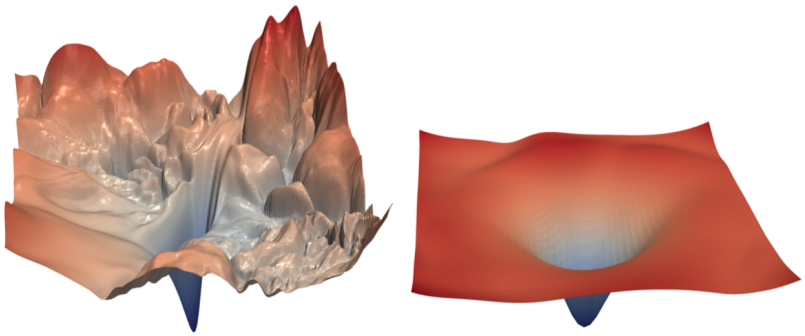
Reading

1. Goodfellow Chapter 13
2. "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima" by Baldi and Hornik (1989)
3. "Entropy-SGD: Biasing gradient descent into wide valleys" by Chaudhari et al. (2016)

In this chapter, we will try to understand the shape of the objective for training neural networks. We would like to characterize the difficulty of training neural networks. We know that the objective is not convex and training a network is difficult because of it. But how non-convex is the objective? The questions we want to answer here are of the following form.

1. How many global minima exist?
2. How many local minima and saddle points exist?
3. What is the loss at the local minima or saddle points? If we train with gradient descent or stochastic gradient descent what loss can we expect to obtain even if we don't reach the global minimum?
4. What is the local geometry of the loss function?
5. What is the global topology of the loss function?

This will help understand how SGD seems to train deep networks so efficiently and why we often get very good generalization error after training. As a pre-cursor to how the picture of the energy landscape of a neural network looks like, here's one picture from Li et al. (2018):



(a) without skip connections (b) with skip connections

Figure 12.1: A picture of the training loss. The picture on the left was created by sampling two directions randomly out of the millions of weights for a residual network without skip-connections and computing the training loss by discretization of this two-dimensional space. The picture on the right is a similar picture for the resnet with skip-connections intact. In this picture, we see that while the training loss is very complex on the left-hand side with lots of local minima and saddle points, the loss is much more benign on the right-hand side.

12.1 Introduction

Let us introduce a few quantities that will help characterize the energy landscape. We will consider the case when the function $\ell(w)$ is twice-differentiable.

Global minima are all points in the set

$$\{w : \ell(w) \leq \ell(w') \text{ for all } w'\}.$$

Note that there may exist many different locations all with the same loss $\ell(w)$, they would all be global minima in this case. Local minima are all points in the set

$$\left\{w : \nabla \ell(w) = 0, \nabla^2 \ell(w) \succeq 0\right\}.$$

i.e., all points w where the Hessian $\nabla^2 \ell(w)$ is positive semi-definite. Note that the two conditions (i) first-order stationarity $\nabla \ell(w) = 0$ and (ii) positive semi-definiteness of the Hessian $\nabla^2 \ell(w) \succeq 0$ also have to be satisfied for all global minima. Critical points are all locations which satisfy only first order stationarity

$$\{w : \nabla \ell(w) = 0\}.$$

Saddle points are critical points but which are neither local minima not local maxima

$$\left\{w : \nabla \ell(w) = 0, \nabla^2 \ell(w) \text{ is neither positive nor negative semi-definite}\right\}.$$

Non-convex functions, in general, can have all these different kinds of locations in the energy landscape and this makes minimizing the objective difficult. Our goal in this chapter is to learn theoretical and empirical results that help paint a mental picture of what the energy landscape looks like.

❓ Draw the Gibbs distribution of SGD if $\ell(w)$ has multiple global minima.

❓ Draw the Gibbs distribution of SGD if $\ell(w)$ has multiple global minima and multiple local minima.

12.2 Deep Linear Networks

Let us consider the simplest case of linear neural networks first. We will have a two-layer neural network which takes in inputs x^i and aims to predict targets y^i. For simplicity, we will consider the case when both

$$x^i, y^i \in \mathbb{R}^d.$$

and use the regression loss

$$\ell(A, B) = \frac{1}{2n} \sum_{i=1}^{n} \|y^i - AB\, x^i\|_2^2 \tag{12.1}$$

We use the standard trick of appending a 1 to the input x^i so that we don't have to carry around biases in our equations.

The matrices A, B are the weights of the neural network with

$$A \in \mathbb{R}^{d \times p}, B \in \mathbb{R}^{p \times d}.$$

We will consider the case when $p \leq d$. We are interested in finding A and B and will develop some results from Baldi & Hornik's paper.

Least squares solution Let us write the inputs and targets as matrices X, Y respectively, each row A simple calculation reveals that for a single-layer network the solution of the problem

$$L^* = \operatorname*{argmin}_L \frac{1}{2n} \sum_{i=1}^{n} \|y^i - Lx^i\|_2^2$$

is

$$L^* = \Sigma_{yx}\, \Sigma_{xx}^{-1} \tag{12.2}$$

where

$$\Sigma_{yx} = \sum_i y^i {x^i}^\top$$

$$\Sigma_{xx} = \sum_i x^i {x^i}^\top.$$

The matrices Σ_{yx} and Σ_{xx} are the data covaraiance matrices.

Projection of a vector onto a matrix It will be useful to define a projection matrix. Say we have a vector v that we want to project on the span of the columns of a full-rank matrix

$$M = \begin{bmatrix} m_1 & m_2 & \dots & m_n \end{bmatrix}.$$

If this projection is $\hat{v} \in \text{span}\,\{m_1, \dots, m_n\}$, we know that it has to satisfy

$$(v - \hat{v}) \perp m_k \text{ for all } k \leq n \quad \Rightarrow \quad m_k^\top (v - \hat{v}) = 0.$$

The vector $\hat{v}$ is also obtained by a combination of the columns of M, so there exists a vector c which allows us to write

$$\hat{v} = Mc.$$

These together imply

$$c = (M^\top M)^{-1} M^\top \hat{v}$$

and finally

$$\hat{v} = \underbrace{M(M^\top M)^{-1}M^\top}_{\text{projection matrix}} v$$
$$=: P_M\, v.$$

where the matrix P_M is called the projection matrix corresponding to the matrix M.

⚠ Note that $P_M^2 = P_M$, i.e., if we project the vector twice onto the column space of M, the second projection does nothing. Also, any projection matrix P is symmetric. To see this, consider two vectors v, w and the dot products

$$\langle Pv, w\rangle\,, \text{ and } \langle v, Pw\rangle\,.$$

In both cases, one of the vectors lies completely in the column space of M and therefore the dot product ignores any component that is orthogonal to the column space of M. This means

$$\langle Pv, w\rangle = \langle v, Pw\rangle = \langle Pv, Pw\rangle\,.$$

We can now rewrite the first equality to obtain

$$(Pv)^\top w = v^\top (Pw)$$
$$\Rightarrow v^\top P^\top w = v^\top P w$$

and since this is true for any two vectors v, w, we have that $P = P^\top$.

Back to deep linear networks We know from the homework problem that there is no unique solution to the problem

$$A^*, B^* = \operatorname*{argmin}_{A,B} \frac{1}{2n}\sum_{i=1}^{n} \|y^i - AB\, x^i\|_2^2.$$

If A^*, B^* are solutions, so are $A^*P, P^{-1}B^*$ for any invertible matrix P. We also showed in the homework that the objective is not convex. But if we fix either A or B and only optimize over the other, the loss is convex. Notice that the rank of AB is at most p.

Fact 1 (Critical points of B if A is fixed). For any A, the function $\ell(A, B)$ is convex in B and has a minimum at

$$\left(A^\top A\right) \widehat{B}(A)\Sigma_{xx} = A^\top \Sigma_{yx}.$$

If Σ_{xx} is invertible and A is full-rank, then we can write

$$\widehat{B}(A) = (A^\top A)^{-1} A^\top \Sigma_{yx}\Sigma_{xx}^{-1}. \tag{12.3}$$

Note that these are all locations when the gradient

$$\frac{\partial \ell}{\partial B} = 0.$$

Fact 2 (Critical points of A if B is fixed). We have an analogous version of the previous fact for A: if B is fixed, the loss is convex in A, for full-rank Σ_{xx} and B, then for $\frac{\partial \ell}{\partial A} = 0$, we should have

$$AB\Sigma_{xx}B^\top = \Sigma_{yx}\, B^\top. \tag{12.4}$$

or

$$\widehat{A}(B) = \Sigma_{yx}\, B^\top \left(B\Sigma_{xx}B^\top\right)^{-1}. \tag{12.5}$$

Fact 3 (Critical points of (A, B)). We now solve the equations (12.3) and (12.5) to get a critical point, i.e., the gradient of the objective in both A and B is zero. First

$$W = AB = P_A\, \Sigma_{yx}\Sigma_{xx}^{-1}. \tag{12.6}$$

from (12.3). Next, multiply on both sides of (12.4) by $A^\top$ and substitute the above value of W to get that the matrix A should satisfy

$$P_A\Sigma = \Sigma P_A = P_A\Sigma P_A.$$

where

$$\Sigma = \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}.$$

Fact 4 (If W is a critical point, then it can be written as a projection of the least squares solution $\Sigma_{yx}\Sigma_{xx}^{-1}$ on the subspace spanned by some p eigenvectors of Σ). This is an important fact. Let us say we have a full-rank Σ with distinct eigenvalues $\lambda_1 > \ldots > \lambda_d$. Let u_{i_k} be the eigenvector associated with the i_k^{th} eigenvalue of Σ. So given any set of p eigenvalues

$$\mathcal{I} = \{i_1, \ldots, i_p\} \text{ with } \quad 1 \leq i_k \leq d \text{ for all } k.$$

we can define a matrix of rank p

$$U_{\mathcal{I}} = \begin{bmatrix} u_{i_1} & u_{i_2} & \ldots & u_{i_p} \end{bmatrix}.$$

Then one can show that the matrices A and B are critical points if and only if there is a set $\mathcal{I}$ and an invertible matrix $C \in \mathbb{R}^{p \times p}$ such that

$$\begin{aligned} A &= U_{\mathcal{I}}\, C \\ B &= C^{-1} U_{\mathcal{I}}^{\top}\, \Sigma_{yx}\Sigma_{xx}^{-1}. \end{aligned} \tag{12.7}$$

You can find the proof in the Appendix of Baldi & Hornik's paper. Because $U_{\mathcal{I}}$ is a matrix of orthonormal vectors we also have

$$P_{U_{\mathcal{I}}} = U_{\mathcal{I}}\, U_{\mathcal{I}}^{\top}$$

and therefore

$$W = P_{U_{\mathcal{I}}}\, \Sigma_{yx}\Sigma_{xx}^{-1}$$

which is the same form for W as (12.6) in Fact 3 and L^* in (12.2). In other words, the solution $W = AB$ in a two-layer linear network is given by our original least squares regression matrix followed by an orthogonal projection onto the subspace spanned by p eigenvectors of Σ.

Fact 5 (If W is the global minimum for a two-layer network, then it is a projection of the solution for a single-layer network onto the subspace spanned by the top p eigenvectors of Σ). You can further show that the objective

$$\ell(A, B) = \text{trace}(\Sigma_{yy}) - \sum_{i_k \in \mathcal{I}} \lambda_{i_k}. \tag{12.8}$$

at a critical point (A, B). The first term is a constant with respect to the parameters of the network A, B. The second term is a sum of the eigenvalues of the matrix Σ at indices that we picked in our set $U_{\mathcal{I}}$. What is the index set that minimizes this loss? It is simply the largest p eigenvalues of Σ. This is also a unique value for the loss because we have assumed that all the eigenvalues are distinct. This also solidifies the connection of this model with Principal Component Analysis (PCA), the matrix W is projecting on the sub-space spanned by the top p eigenvectors in the auto-associative case.

Fact 6 (There are exponentially many saddle points for a two-layer network). There are a total of $\binom{d}{p}$ possible index sets $\mathcal{I}$. One of them as we saw above corresponds to a global minimum. It can be shown that all the others are saddle points. Note that there are exponentially many saddle points. This is an important fact to remember: there are exponentially many saddle points in a hierarchical architecture. Smaller the number of neurons in the hiddle layer p (also the upper bound for the rank of the weight matrices), fewer are the

❓ Based on the previous two facts, what can you say about the solution W if $p \geq d$ and Σ is invertible? Since the two-layer network simply projects on the p eigenvalues of Σ, if $p \geq d$ and Σ is invertible, the solution already lies in the column-space of Σ and therefore $W = L^*$.

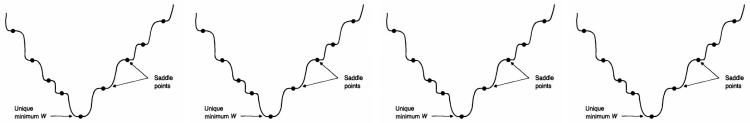
number of saddle points but this also creates a dimensionality bottleneck in the feature space. If p is too small as compared to d we lose large amounts of information necessary to classify the image and the network may not work well.

Fact 7 (No local minima in a deep linear network, all minima are global minima). The proof of the previous fact (see the Appendix of Baldi and Hornik (1989)) shows that any index set $\mathcal{I} \neq \{1, \ldots, p\}$ cannot be a local minimum. There are no local minima for a deep linear network, only global minima and saddle points. This is often said as "linear networks have no bad local minima".

Fact 8 (The global minimum is not unique). This is perhaps the most important point of this chapter. The loss at the global minimum is unique, not the global minimum itself. Any full-rank square matrix $C \in \mathbb{R}^{p \times p}$ of our choice gives a pair of solutions (A, B). How many such solutions are there? There are lots and lots of such solutions, in fact, given any solution with a particular C if we can perturb the C without losing rank (quite easy to do by, say, changing the eigenvalues slightly) we get another solution of a linear network.

Fact 9 (All the previous results are true for multi-layer linear networks). The same results are true for deep linear networks (Kawaguchi, 2016). These results also hold if $\dim(y_i) = 1$, i.e., for the regression case.

We used a simple two-layer linear network to obtain an essentially complete understanding of how the loss function looks like. A schematic looks as follows.



There are lots of locations where the global minimum of the function is achieved. There are lots of saddle points in the energy landscape. The Gibbs distribution for this energy landscape has a lot of modes, one each at the global minima.

How does weight-decay

$$\Omega(A, B) = \lambda \left(\|A\|_F^2 + \|B\|_F^2 \right)$$

change the energy landscape of deep linear networks? It changes the number of global minima, only the ones that have the smallest ℓ_2 norm remain in the energy landscape. It also reduces the number of saddle points because the Hessian at saddle points has an extra additive term that involves λ.

12.3 Extending the picture to deep networks

Let us think carefully about the non-uniqueness of the solution for a two-layer network. We know that all critical points are of the form

$$\begin{aligned} A &= U_{\mathcal{I}} C, \\ B &= C^{-1} U_{\mathcal{I}}^{\top} \Sigma_{yx} \Sigma_{xx}^{-1}. \end{aligned}$$

The gradient at these critical points is zero. Given a particular C, we can perturb it slightly and obtain a new critical point (a new saddle point, or a new global minimum) and this keeps the objective unchanged. Effectively, we have a connected set of global minima and saddle points for a deep linear networks.

If one were to try to visualize this energy landscape and extend the picture heuristically to deep networks with nonlinearities, we can think of the global minimum as looking like the basin of the Colorado river.

The important point to remember from this picture is that all the points at the basin of the river are solutions that obtain a good training loss. Although gradient-based algorithms (GD/SGD etc.) do not allow us to travel along the river (the gradient is zero along it), if the river basin snakes around in the entire domain, no matter where the network is initialized, we always have a global minimum close to the initialization. Essentially, the objective of deep networks is not convex, but current results indicate that it is quite benign. And this is perhaps the reason why it is so easy to train them.

Bibliography

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2016). Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv:1611.01838*.

Kawaguchi, K. (2016). Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399.