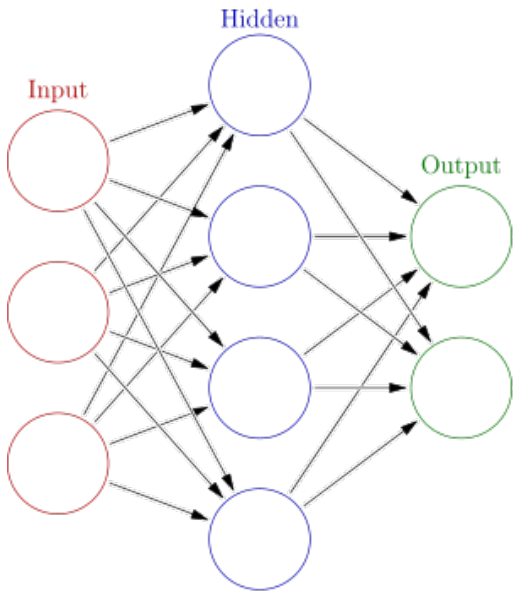
An empirical study of the loss surface of neural networks

Anastasia Borovykh

CWI Amsterdam

CWI Scientific Meeting

Quick reminder: Neural networks



- Neural network computes $y_j = \phi\left(\sum_{i=1}^{M} w_{j,i}x_i + \theta\right)$, where $w_{i,j}$ is the weight, θ the bias and $\phi(\cdot)$ the activation function.

Quick reminder: Neural networks

- Role of the layers: in each layer the network transforms the data, creating a new representation. Each layer stretches and squishes space but preserves topological properties. Number of layers needed depends on how 'entangled' the data is.
- Role of the weights: a linear transformation of the input by weight matrix W
- Role of the bias: provides each node with a trainable constant, allows to shift the activation function to the left or right to make prediction fit better
- Role of the activation function: pointwise application of non-linearity, e.g. sigmoid activation $\phi(x) = 1/(1 + e^{-x})$ squashes output into range $[0, 1]$, ReLU $\phi(x) = \max(0, x)$ caps output at zero

The loss surface

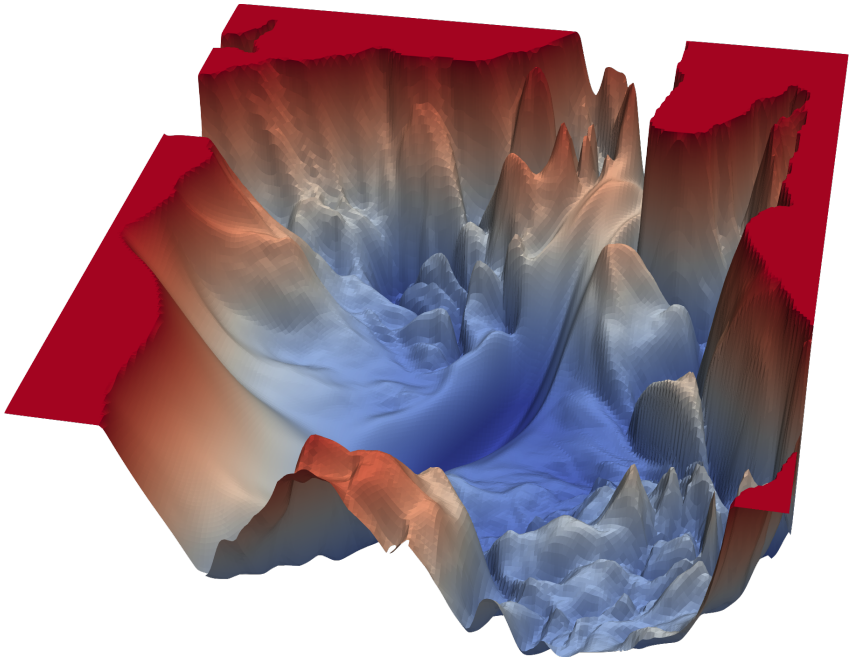
- Given a training set of input vectors x_i, $i = 1, ..., N$ and targets y_i. Find $\mathbf{w}$ to minimize a loss function e.g.

$$E(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^{N} (y(x_i, \mathbf{w}) - y_i)^2.$$

- When training a neural network we want to obtain minima that perform well on unseen data, i.e. that generalize well without overfitting on the noise in the train dataset.
- Financial data has lots of noise, heavy tails, non-linear dependencies; we do not want to overfit on the noise.
- Two definitions we will use:
  - Critical point: a point where the derivatives are zero; in our case also a point to which our optimisation algorithm has converged.
  - Index: the number of negative eigenvalues of the Hessian matrix; an index $k < N$ there are k directions pointing down and $N - k$ pointing up.

The loss surface

This loss function defines a multi-dimensional loss surface over the weights (a 3D plot):



Depends on: number of layers, number of nodes per layer, activation function, loss function.

Some properties of the loss surface

- Highly non-convex and depends on a large number of parameters
- Despite the non-convexity relatively easy to train: could mean that all local minima/saddle points are of good quality?
- As size increases, more saddle points than local minima
- Neural networks are extremely flexible; can fit almost any function (universal approximation theorem)
- Global minima typically overfit
- **Question 1**: How does the loss surface of neural networks look?
- **Question 2**: How do we find *good* minima that generalize well?

Relating the loss surface to a Gaussian random field

- Based on [Choromanska, 2015]
- The output of a neural network with non-linearity $f(\cdot)$ is given by

$$y(\mathbf{w}, x) = qf(w^{(L)} f(w^{(L-1)} \dots f(w^{(1)} x))) \dots),$$

with q some scaling factor.

- Let the non-linear activation function be the rectified linear unit $f(x) = \max(x, 0)$.
- Replace the activation function with the term $A_{i,j} \in \{0, 1\}$ denoting whether a path (i, j) is active.
- We then obtain,

$$y(\mathbf{w}, x) = q \sum_{i=1}^{n_0} \sum_{j=1}^{P} x_i A_{i,j} \prod_{k=1}^{L} w_{i,j}^{(k)},$$

with $P := n_0 n_1 \dots n_L$ the number of paths from a given input to networks output.

Relating the loss surface to a Gaussian random field

- We now make the first key assumption that each path is equally likely to be active and follows a Bernoulli distribution with probability ρ, independent of the input.
- We then obtain,

$$\mathbb{E}[y(\mathbf{w}, x)|x] = q \sum_{i=1}^{n_0} x_i \rho \prod_{k=1}^{L} w_{i,j}^{(k)}.$$

- Note: this expression is thus similar to a deep *linear* model, multiplied by the factor ρ.
- Second key assumption is to let the input be sampled independently as $x_i \sim \mathcal{N}(0, 1)$.
- Since we sum over i.i.d. normally distributed terms, the expected output is a Gaussian process with state space being the high-dimensional weight space $[W^{(1)}, ..., W^{(L)}] \in \mathbb{R}^{(n_0 \times n_1) \times \cdots \times (n_{(L-1)} \times n_{(L)})}$.

Gaussian random field on high dimensions

The authors of [Bray, Dean (2007)] give a result on the number of critical points for N-dimensional Gaussian process f.
As $N \to \infty$ there is:

- a certain structure of critical points,
- critical points whose error is much larger than the global minimum are exponentially likely to be high-index saddle points,
- all actual local minima are at an energy level close to that of the global minimum,
- there is thus a strong correlation between the error and the index: the larger the error the larger the index,
- furthermore, as we will see later on, there are typically large plateaus around critical points, complicating optimizibility of the network.

Are neural networks like Gaussian random fields?

- Let us compute 50 minima of neural networks using stochastic gradient descent. We plot here the index vs. the loss level:

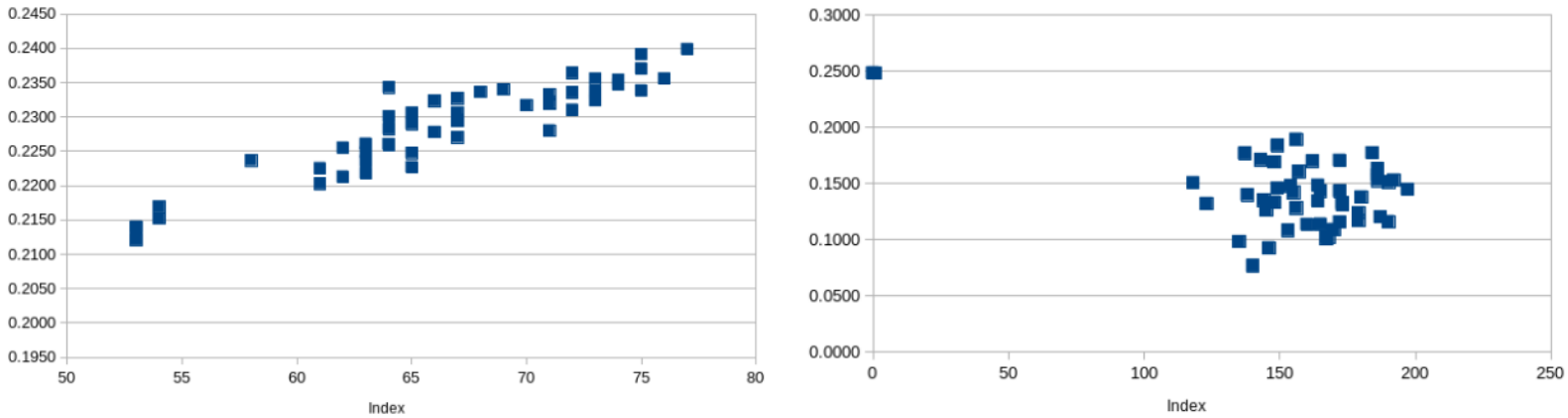


Figure: Right: 2 layers, 10 nodes per layer; left: 5 layers, 10 nodes per layer.

Width of the minima

- Generalisation: the ability of a network trained on the train dataset to perform good on the test dataset; i.e. its ability to perform well on unseen data.
- Measures how much a network has actually learned instead of simply fitted the noise.
- The authors of [Keskar et al., (2016)] showed that the width of the minima is a measure of the generalizibility.
- Intuitively, the wider a minimum the better its resistance to noisy transformations of the input data; still would give the same loss. In contrast, sharp minima are very sensitive to changes in the underlying data distribution.
- Other measures of generalizibility: the trace of the Hessian, indicates the flatness around a particular minimum. Low trace should give better generalizibility.

Width of the minima

- We compute the trace and width (computed as the number of steps we can take in random directions without increasing the loss) for different network architectures.

N_{depth}	N_{width}	Average width	$Tr(H)$ of best train	$Tr(H)$ of best test
2	10	40	39 (test loss 0.295)	2.2 (test loss 0.251)
2	50	29	87 (test loss 0.313)	10 (test loss 0.258)
5	10	9	1171 (test loss 0.371)	0.25 (test loss 0.246)
5	50	4	29 (test loss 0.491)	288 (test loss 0.430)

- We observe that
  - Increasing the number of parameters in general seems to decrease the width of the minima (more overfitting)
  - Minima that are able to generalize well have a much lower trace.
  - The trace seems to be a good indicator for out-of-sample performance.

Width of the minima

- We plot the eigenvalue spectrum of minima that perform best on train set (blue) and test set (orange).

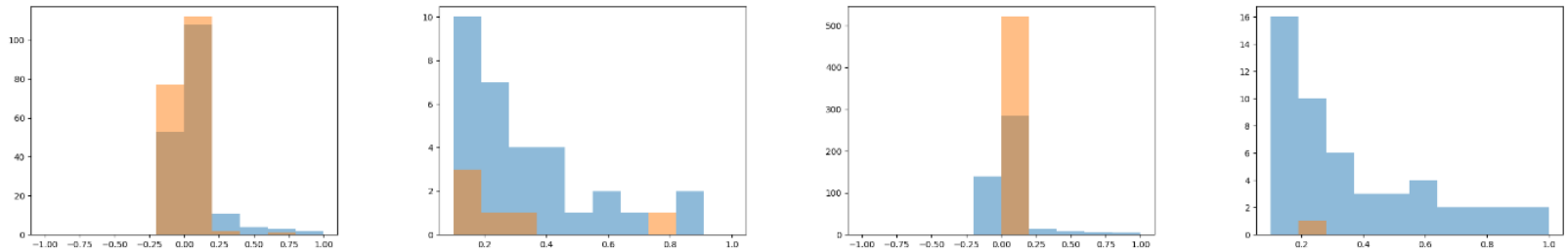


Figure: Right: two layers and 10 nodes per layer; left: five layers and 10 nodes per layer

- We observe a large mode around zero, i.e. the minima found typically have large plateaus around them. This explains why SGD gets stuck in them.
- Minima that overfit have many more large, positive outlier eigenvalues.

A trading example

- We have trained a neural network on the S&P500 index data to predict the next-day return $t+1$ (i.e. whether the price moves up or down) using $K+1$ days of data in the past $t-K, ..., t$
- Define a simple trading strategy; buy if price if predicted to go up, earning money if true price move is up; sell if price is predicted to go down, earning money if true price move is down.

A trading example

Train the network to find 50 minima, and construct the portfolio value of the resulting trading strategy on *unseen* data:

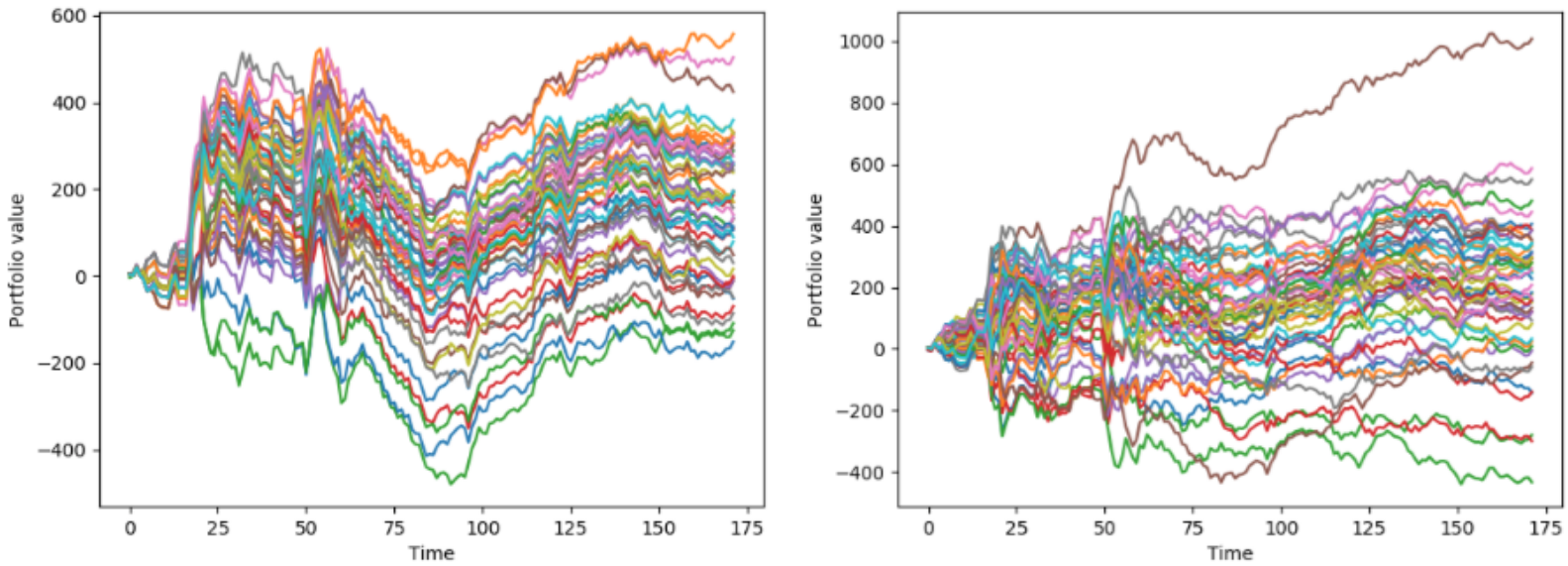


Figure: Right: two layers and 10 nodes per layer; left: five layers and 10 nodes per layer

A trading example: which minima are good?

- Use the trace of the Hessian at the critical points found by our optimization.
- The trace of the Hessian gives *some* indication on the performance; but still other (not yet determined) factors have an influence too.

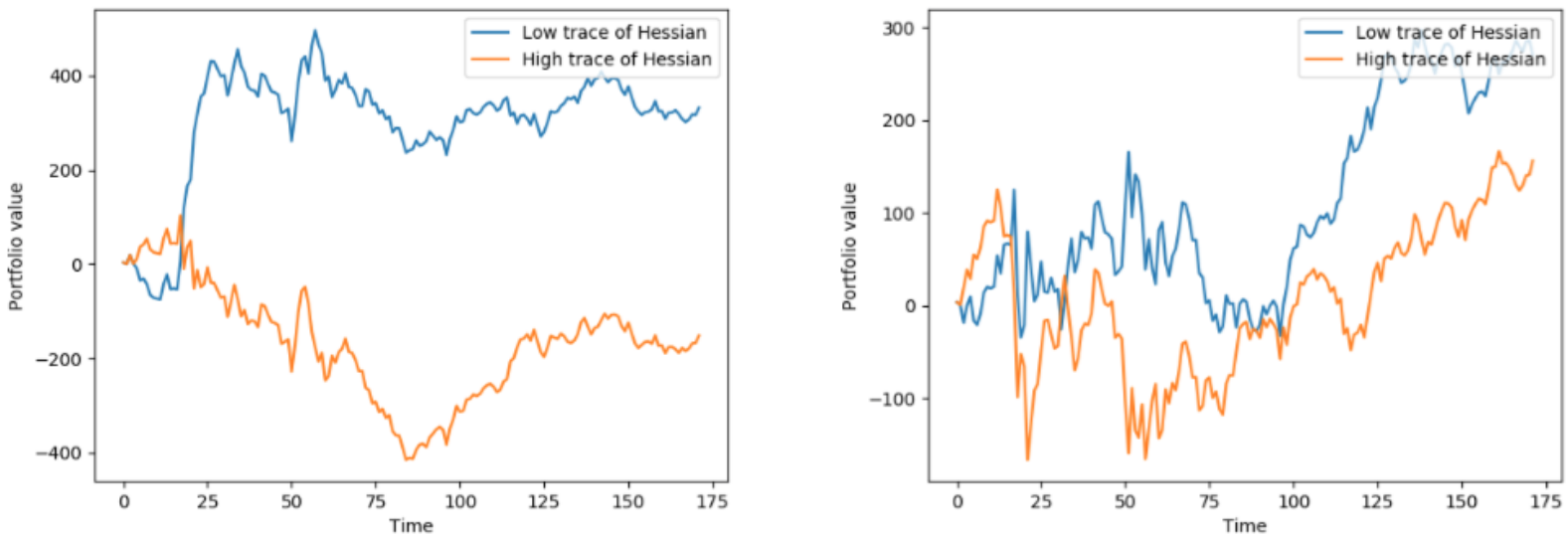


Figure: Right: two layers and 10 nodes per layer; left: five layers and 10 nodes per layer

THANK YOU!