Machine Learning and Random Matrices

Jon Keating

Mathematical Institute
University of Oxford

keating@maths.ox.ac.uk

November 5$^{\text{th}}$ 2024

Joint work with Nick Baskerville (Sibylla AI), Francesco Mezzadri (Bristol), & Joseph Najnudel (Bristol):

The loss surfaces of neural networks with general activation functions, Journal of Statistical Mechanics: Theory and Experiment 2021 (6), 064001 (2021) [arXiv:2004.03959];

A spin glass model for the loss surfaces of Generative Adversarial Networks, Journal of Statistical Physics 186 (2), 1-45 (2022) [arXiv:2101.02524].

with Nick Baskerville (Sibylla AI) & Diego Grandiol (Oxford):

Appearance of Random Matrix Theory in deep learning, Physica A: Statistical Mechanics and its Applications 590, 126742 (2022) [arXiv:2102.06740].

with Nick Baskerville (Sibylla AI), Diego Grandiol (Oxford), Francesco Mezzadri (Bristol), & Joseph Najnudel (Bristol):

Universal characteristics of deep neural network loss surfaces from random matrix theory, Journal of Physics A, 55, 494002 (2022) [arXiv:2205.08601]

and with Connall Garrod (Oxford):

Unifying low dimensional observations in deep learning through the Deep Linear Unconstrained Feature Model [arXiv:2404.06106]

The Persistence of Neural Collapse Despite Low-Rank Bias: An Analytic Perspective Through Unconstrained Features [arXiv:2410.23169]

Machine Learning 101

Assume, for example, one is given data $\mathcal{D}$ consisting of tuples:

$$\mathcal{D} = \{(\vec{x}, y)\} \subset \mathbb{R}^d \times \mathcal{L}.$$

The $\vec{x}$ are vector data items and the y are the labels or target values associated with them. In *classification*, $\mathcal{L} = \{1, 2, \ldots, C\}$ where C is the number of classes, and in *regression* $\mathcal{L} = \mathbb{R}^t$.

- The data $\mathcal{D}$ define a function $\hat{f}_{\mathcal{D}}$ from a *finite subset* of $\mathbb{R}^d$ to $\mathcal{L}$. The task is to find a function

$$f : \mathbb{R}^d \to \mathcal{L}$$

such that

$$f|_{\mathcal{D}} \approx \hat{f}_{\mathcal{D}}.$$

- The data $\mathcal{D}$ define a function $\hat{f}_{\mathcal{D}}$ from a *finite subset* of $\mathbb{R}^d$ to $\mathcal{L}$. The task is to find a function

$$f : \mathbb{R}^d \to \mathcal{L}$$

such that

$$f|_{\mathcal{D}} \approx \hat{f}_{\mathcal{D}}.$$

- In reality this f is not much use because it does not necessarily *generalise*. One is really interested in data drawn from some distribution $P_{\text{data}} \in \mathbb{P}\left(\mathbb{R}^d \times \mathcal{L}\right)$. We will draw a particular *training set* $\mathcal{D}_{\text{train}}$ from P_{data} to construct f and will then want something like

$$corr\left(f|_{\mathcal{D}_{\text{train}}}, \hat{f}_{\mathcal{D}_{\text{train}}}\right)$$

to be as high as possible for $\mathcal{D}_{\text{train}} \sim P_{\text{data}}$.

Motivation and definition

One defines an *artificial neural network* (hereafter simply a neural network or NN) as follows

$$f(\vec{x}) = \sigma\left(W^{(H)}\sigma\left(W^{(H-1)}\sigma\left(\ldots\sigma\left(W^{(1)}\vec{x}\right)\right)\right)\right)$$

where the $W^{(i)}$ are *weight matrices* and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear *activation-function* and applied element-wise. Common choices for σ are

$$\mathrm{ReLU}(x) \equiv \max(0, x), \quad \tanh, \quad x \mapsto \frac{1}{1 + e^{-x}}.$$

Computation of f is straightforward and can be made efficient by optimising linear algebra primitives and the implementation of σ.

Training

The tune-able parameters are typically learned using a *loss function*:

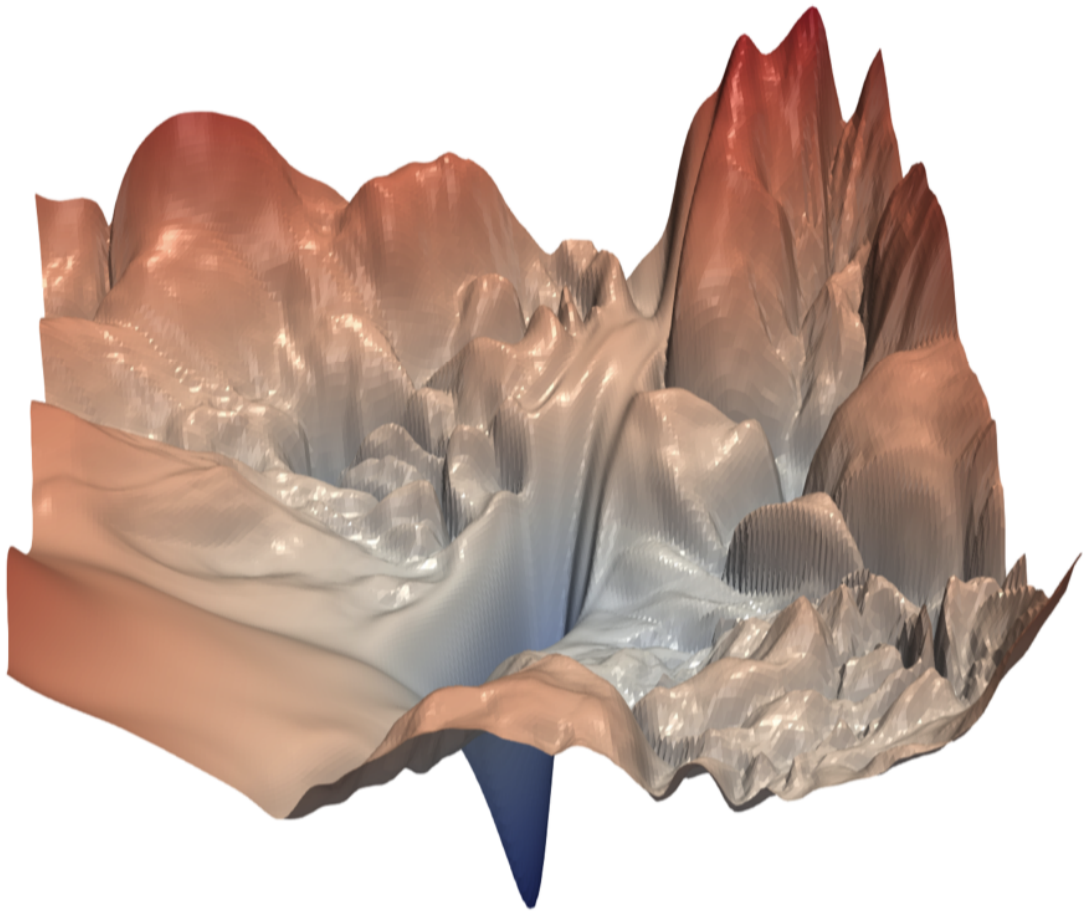
$$\ell\left(f, \mathcal{D}\right)$$

which assigns a real number to a function and a dataset measuring the performance of the function on the data. An obvious example is:

$$\ell\left(f, \mathcal{D}\right) = \sum_{(\vec{x}, y) \in \mathcal{D}} ||f(\vec{x}) - y||_2$$

Let W be shorthand for all parameters of a network f. We are then seeking to solve the following optimisation problem

$$\min_{W} \ell(f, \mathcal{D})$$



(From: Visualizing the Loss Landscape of Neural Nets, by Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, & Tom Goldstein)

(From: https://losslandscape.com)

Stochastic gradient descent

1. Too many parameters to optimise by brute force.

Stochastic gradient descent

1. Too many parameters to optimise by brute force.
2. Computing $\partial_W \ell(f, \mathcal{D})$ is conceptually simple but computationally expensive for large datasets.

Stochastic gradient descent

1. Too many parameters to optimise by brute force.
2. Computing $\partial_W \ell(f, \mathcal{D})$ is conceptually simple but computationally expensive for large datasets.
3. Solving $\partial_W \ell = 0$ is not tractable in general

Stochastic gradient descent

1. Too many parameters to optimise by brute force.
2. Computing $\partial_W \ell(f, \mathcal{D})$ is conceptually simple but computationally expensive for large datasets.
3. Solving $\partial_W \ell = 0$ is not tractable in general
4. therefore explore the surface choosing a downward direction randomly (i.e. stochastically) at each step

Does stochastic gradient descent work?

1. Stochastic gradient descent is guaranteed to converge to the global minimum given some strong assumptions about the *loss surface* (convexity, Lipschitz etc).

Does stochastic gradient descent work?

1. Stochastic gradient descent is guaranteed to converge to the global minimum given some strong assumptions about the *loss surface* (convexity, Lipschitz etc).
2. In practice, modern neural networks use millions of parameters and their loss surfaces are very complicated, peppered with local optima.

Does stochastic gradient descent work?

1. Stochastic gradient descent is guaranteed to converge to the global minimum given some strong assumptions about the *loss surface* (convexity, Lipschitz etc).
2. In practice, modern neural networks use millions of parameters and their loss surfaces are very complicated, peppered with local optima.
3. One has no right to expect SGD to do anything other than bounce around like a ping-pong ball in 10^7 dimensions before getting stuck in some unfortunate local optimum.

Does stochastic gradient descent work?

1. Stochastic gradient descent is guaranteed to converge to the global minimum given some strong assumptions about the *loss surface* (convexity, Lipschitz etc).
2. In practice, modern neural networks use millions of parameters and their loss surfaces are very complicated, peppered with local optima.
3. One has no right to expect SGD to do anything other than bounce around like a ping-pong ball in 10^7 dimensions before getting stuck in some unfortunate local optimum.
4. But it works!

Spin glasses

Spin glasses

The *Hamiltonian* of a spherical p-spin glass is defined as

$$f(\vec{w}) = \sum_{i_1,\ldots,i_p=1}^{N} X_{i_1\ldots i_p} \prod_{l=1}^{p} w_{i_l}$$

where the $X_{ij\ldots k}$ are i.i.d. standard Gaussians and $\vec{w} \in S^{N-1}$ are the spin variables.
We will think of N as being large, so f is a **random, high-dimensional function**.

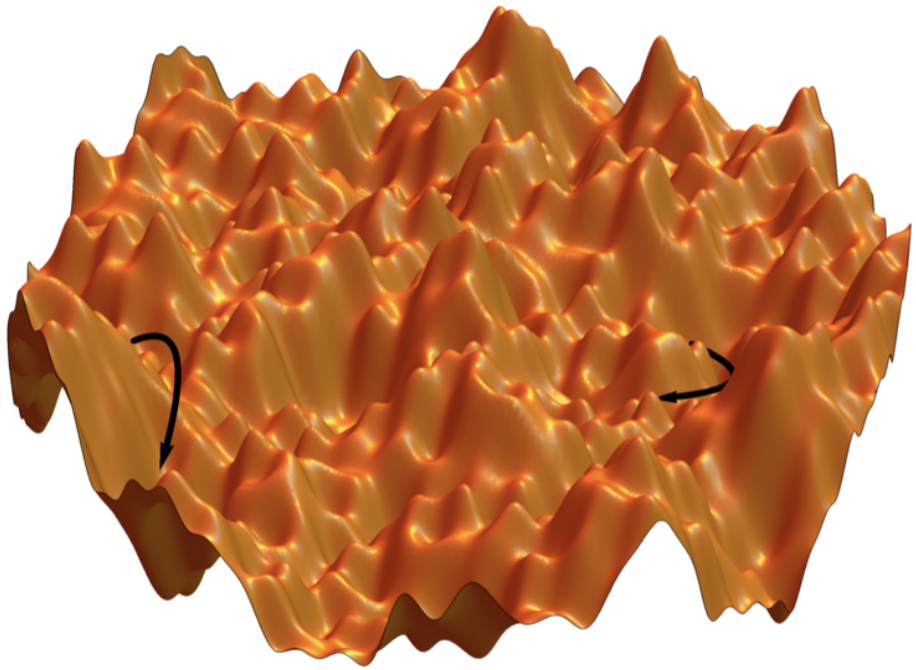


Figure (credit: Chiara Cammarota): A schematic rugged energy landscape with a multitude of energy minima, maxima, and saddles. Arrows denote some of the possible relaxation pathways.

Counting critical points

One can use Kac-Rice formulae to compute for spin glasses the following *complexities* asymptotically in N (Fyodorov 2004, Auffinger *et al.* 2013)

$$C_{N,k}(u) = \left| \left\{ \vec{w} \in S^{N-1} \ : \ \nabla f(\vec{w}) = 0, f(\vec{w}) \leq \sqrt{N}u, \ \mathrm{i}(\nabla^2 f) = k \right\} \right|$$

$$C_N(u) = \left| \left\{ \vec{w} \in S^{N-1} \ : \ \nabla f(\vec{w}) = 0, f(\vec{w}) \leq \sqrt{N}u \right\} \right|$$

where

$$i(M) = index(M) = \#\{\text{negative eigenvalues of } M\}.$$

These are random quantities and we focus on their expectation, but note in passing that it can be shown that $\frac{\mathbb{E}C_N(u)}{C_N(u)} \xrightarrow{\mathbb{P}} 1$.

Banding

Banding

Both $C_{N,k}(u)$ and $C_N(u)$ grow exponentially quickly with N.

Banding

Both $C_{N,k}(u)$ and $C_N(u)$ grow exponentially quickly with N.

In high-dimensional random energy landscapes associated with spin glasses, the asymptotics of $C_{N,k}(u)$ and $C_N(u)$ reveal a potentially favourable banded structure of the low-index saddle points:

Banding

Both $C_{N,k}(u)$ and $C_N(u)$ grow exponentially quickly with N.

In high-dimensional random energy landscapes associated with spin glasses, the asymptotics of $C_{N,k}(u)$ and $C_N(u)$ reveal a potentially favourable banded structure of the low-index saddle points:

- saddle points with higher energies look more like maxima and those with lower energies typically look more like minima

Banding

Both $C_{N,k}(u)$ and $C_N(u)$ grow exponentially quickly with N.

In high-dimensional random energy landscapes associated with spin glasses, the asymptotics of $C_{N,k}(u)$ and $C_N(u)$ reveal a potentially favourable banded structure of the low-index saddle points:

- saddle points with higher energies look more like maxima and those with lower energies typically look more like minima
- most minima have low energies – close to the lowest

Back to machine learning

Let $f : \mathbb{R} \to \mathbb{R}$ be a suitably well-behaved (e.g. differentiable almost everywhere and with bounded gradient) non-linear *activation function* which is taken to applied entry-wise to vectors and matrices.

Back to machine learning

Let $f : \mathbb{R} \to \mathbb{R}$ be a suitably well-behaved (e.g. differentiable almost everywhere and with bounded gradient) non-linear *activation function* which is taken to applied entry-wise to vectors and matrices.
Consider multi-layer perceptron neural networks of the form

$$\vec{y}(\vec{x}) = f(W^{(H)} f(W^{(H-1)} f(\dots f(W^{(1)} \vec{x}) \dots)))$$

where the input data vectors $\vec{x}$ lie in $\mathbb{R}^d$ and the *weight matrices* $\{W^{(\ell)}\}_{\ell=1}^{H}$ have any shapes compatible with $\vec{x} \in \mathbb{R}^d$ and $\vec{y}(\vec{x}) \in \mathbb{R}^c$.

Back to machine learning

Let $f : \mathbb{R} \to \mathbb{R}$ be a suitably well-behaved (e.g. differentiable almost everywhere and with bounded gradient) non-linear *activation function* which is taken to applied entry-wise to vectors and matrices.
Consider multi-layer perceptron neural networks of the form

$$\vec{y}(\vec{x}) = f(W^{(H)} f(W^{(H-1)} f(\dots f(W^{(1)} \vec{x}) \dots)))$$

where the input data vectors $\vec{x}$ lie in $\mathbb{R}^d$ and the *weight matrices* $\{W^{(\ell)}\}_{\ell=1}^{H}$ have any shapes compatible with $\vec{x} \in \mathbb{R}^d$ and $\vec{y}(\vec{x}) \in \mathbb{R}^c$.

Ignore biases in the network.

Back to machine learning

Let $f : \mathbb{R} \to \mathbb{R}$ be a suitably well-behaved (e.g. differentiable almost everywhere and with bounded gradient) non-linear *activation function* which is taken to applied entry-wise to vectors and matrices.
Consider multi-layer perceptron neural networks of the form

$$\vec{y}(\vec{x}) = f(W^{(H)} f(W^{(H-1)} f(\dots f(W^{(1)} \vec{x}) \dots)))$$

where the input data vectors $\vec{x}$ lie in $\mathbb{R}^d$ and the *weight matrices* $\{W^{(\ell)}\}_{\ell=1}^{H}$ have any shapes compatible with $\vec{x} \in \mathbb{R}^d$ and $\vec{y}(\vec{x}) \in \mathbb{R}^c$.

Ignore biases in the network.

What lies behind the unreasonable efficacy of gradient descent on the high-dimensional and strongly non-convex loss surfaces of neural network models?

Modelling Assumptions

Modelling Assumptions

1. Components of the data vectors are i.i.d. standard Gaussians.
2. The neural network can be well approximated by a much sparser network that achieves very similar accuracy. [A network with N weights is sparse if it has s unique weight values and $s \ll N$.]
3. The unique weights of the sparse network are approximately uniformly distributed over the graph of weight connections.
4. The unique weights of the sparse neural network lie on a hyper-sphere of some radius.
5. The activation function is twice-differentiable almost everywhere in $\mathbb{R}$ and can be well approximated as a piece-wise linear function with finitely many linear pieces.
6. The action of the piece-wise linear approximation to the activation function on the network graph can be modelled as i.i.d. discrete random variables independent of the data at each node indicating which linear piece is active.

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Under the above assumptions

- The neural network reduces to a spin glass!

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Under the above assumptions

- The neural network reduces to a spin glass!
- One obtains expressions for $C_{k,H}$ and C_H as expectations w.r.t. the Gaussian Orthogonal Ensemble (GOE) of Random Matrix Theory using the Kac-Rice formula.

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Under the above assumptions

- The neural network reduces to a spin glass!
- One obtains expressions for $C_{k,H}$ and C_H as expectations w.r.t. the Gaussian Orthogonal Ensemble (GOE) of Random Matrix Theory using the Kac-Rice formula. Both grow exponentially with the number of parameters.

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Under the above assumptions

- The neural network reduces to a spin glass!
- One obtains expressions for $C_{k,H}$ and C_H as expectations w.r.t. the Gaussian Orthogonal Ensemble (GOE) of Random Matrix Theory using the Kac-Rice formula. Both grow exponentially with the number of parameters.
- Essentially, the Hessian of the loss function at a random point behaves like a GOE random matrix plus a deterministic rank-2 perturbation (which is absent when the activation function is `ReLU`).

Consider $\vec{y}$ as a random function over a high-dimensional weight-space, the randomness coming from taking the input data to be random.

Define $C_H(u)$ to be the expected number of critical points taking values at most u, and $C_{k,H}(u)$ to be the expected number of critical points of index k taking values at most u.

Under the above assumptions

- The neural network reduces to a spin glass!
- One obtains expressions for $C_{k,H}$ and C_H as expectations w.r.t. the Gaussian Orthogonal Ensemble (GOE) of Random Matrix Theory using the Kac-Rice formula. Both grow exponentially with the number of parameters.
- Essentially, the Hessian of the loss function at a random point behaves like a GOE random matrix plus a deterministic rank-2 perturbation (which is absent when the activation function is `ReLU`).
- One again finds a 'banded structure': $\exists\ E_0 > E_1 > ... > E_\infty$ such that, with overwhelming probability, critical points taking (scaled) values in $(-E_k, -E_{k+1})$ have index at most $k + 2$.

RMT calculations

The special case (considered first by Choromanska *et al.* in 2015), where the activation function is `ReLU`, can be handled by standard probabilistic RMT techniques.

RMT calculations

The special case (considered first by Choromanska *et al.* in 2015), where the activation function is `ReLU`, can be handled by standard probabilistic RMT techniques.

In the general case, when one has a GOE matrix plus a deterministic rank-2 perturbation, one needs different techniques, e.g. using supersymmetric integrals as representations of ratios of determinants (Baskerville *et al.* in 2020).

Implications

Local optima of the the neural network loss surface are arranged so that, above a critical value $-\sqrt{N}E_\infty$, it is overwhelmingly likely that gradient descent will encounter high-index optima and so 'escape' and descend to lower loss. Below $-\sqrt{N}E_\infty$, the low-index optima are arranged in a 'banded' structure.

Implications

Local optima of the the neural network loss surface are arranged so that, above a critical value $-\sqrt{N}E_\infty$, it is overwhelmingly likely that gradient descent will encounter high-index optima and so 'escape' and descend to lower loss. Below $-\sqrt{N}E_\infty$, the low-index optima are arranged in a 'banded' structure.

band	**possible indices**
$(-\sqrt{N}E_0, -\sqrt{N}E_1)$	0,1,2
$(-\sqrt{N}E_1, -\sqrt{N}E_2)$	0,1,2,3
$(-\sqrt{N}E_2, -\sqrt{N}E_3)$	0,1,2,3,4
$(-\sqrt{N}E_3, -\sqrt{N}E_4)$	0,1,2,3,4,5

Piece-wise linear activation functions

- it is good enough to study piece-wise linear approximations to sensible (but general) activation functions.

Piece-wise linear activation functions

- it is good enough to study piece-wise linear approximations to sensible (but general) activation functions.
- A calculation shows that the spin glass is then replaced by the related object

$$g(\vec{w}) = \underbrace{\sum_{i_1,\ldots,i_H=1}^{N} X_{i_1,\ldots,i_H} \prod_{k=1}^{H} w_{i_k}}_{\text{Spin glass, stochastic}} + \underbrace{\sum_{\ell=1}^{H} \rho'_\ell \sum_{i_{\ell+1},\ldots,i_H=1}^{N} \prod_{k=\ell+1}^{H} w_{i_k}}_{\text{Deterministic}}$$

Piece-wise linear activation functions

- it is good enough to study piece-wise linear approximations to sensible (but general) activation functions.
- A calculation shows that the spin glass is then replaced by the related object

$$g(\vec{w}) = \underbrace{\sum_{i_1,\ldots,i_H=1}^{N} X_{i_1,\ldots,i_H} \prod_{k=1}^{H} w_{i_k}}_{\text{Spin glass, stochastic}} + \underbrace{\sum_{\ell=1}^{H} \rho'_\ell \sum_{i_{\ell+1},\ldots,i_H=1}^{N} \prod_{k=\ell+1}^{H} w_{i_k}}_{\text{Deterministic}}$$

- Details of the activation function are in ρ'_l.

Kac-Rice formula for complexity

We can compute $\mathbb{E}C_N(u)$ using a *Kac-Rice* formula:

$$\mathbb{E}C_N(u) = \int_{S^{N-1}} d\vec{x} \quad \underbrace{\varphi_{\nabla g(\vec{x})}(0)}_{\text{Density of } \nabla g(\vec{x})} \quad \times$$

$$\underbrace{\mathbb{E}\left[|\det \nabla^2 g(\vec{x})| \mathbb{1}\left\{g(\vec{x}) \leq \sqrt{N}u\right\} \mid \nabla g(\vec{x}) = 0\right]}_{\text{Average over } X}$$

1. Integrand is spherically symmetric so pick convenient coordinates around a the north pole to do calculations.
2. Compute the joint density of $(g, \partial_i g, \partial_{jk} g)$.

Conditional distributions

g and all its derivatives are Gaussian so it suffices to compute the means and covariances. Covariances can be computed by taking derivatives of

$$Cov(g(\vec{w}), g(\vec{w}')) = (\vec{w}^T \vec{w}')^H = \left(\hat{\vec{w}}^T \hat{\vec{w}}' + \sqrt{1 - ||\hat{\vec{w}}||_2} \sqrt{1 - ||\hat{\vec{w}}'||_2} \right)^H$$

$\partial_i g$ is independent of $(g, \partial_{jk} g)$, so $\mathbb{E} C_N^\ell(u)$ is equal to

$$\int_{S^{N-1}} d\vec{w} \varphi_{\nabla g(\vec{w})}(0) \int_{-\infty}^{\sqrt{N}u} dx \varphi_{g(\vec{w})}(x) \mathbb{E}\left[|\det \nabla^2 g(\vec{w}) \mid g(\vec{w}) = x \right].$$

Hence

$$\nabla^2 \ell \mid (\ell = x) \sim \underbrace{\sqrt{2H(H-1)(N-1)}\mathrm{GOE}^{N-1} - HxI}_{GOE} + \underbrace{S}_{\text{from non-random term}}$$

where S is a rank-2 matrix with entries of size $o(1)$.

Averaging the determinant with supersymmetry

We use the supersymmetric representation:

$$|\det(M - xI + S)| = \lim_{\epsilon \searrow 0} \frac{\det(M - xI + S - i\epsilon)\det(M - xI + S + i\epsilon)}{\underbrace{\sqrt{\det(M - xI + S - i\epsilon)}\sqrt{\det(M - xI + S + i\epsilon)}}_{\equiv \Delta_\epsilon(M;x,S)}}$$

$$\Delta_\epsilon(M;x,S) \propto \int \underbrace{d\vec{x}_1 d\vec{x}_2}_{\text{Commuting}} \underbrace{d\zeta_1 d\zeta_1^\dagger d\zeta_2 d\zeta_2^\dagger}_{\text{Anti-commuting}}$$

$$\exp\left\{-i\mathrm{Tr}MA - i\mathrm{Tr}SA + i(x + i\epsilon)\vec{x}_1^T\vec{x}_1 + i(x - i\epsilon)\vec{x}_2^T\vec{x}_2\right\}$$

$$\exp\left\{-i(x + i\epsilon)\zeta_1^\dagger\zeta_1 - i(x - i\epsilon)\zeta_2^\dagger\zeta_2\right\}.$$

where $A = \vec{x}_1\vec{x}_1^T + \vec{x}_2\vec{x}_2^T + \zeta_1\zeta_1^\dagger + \zeta_2\zeta_2^\dagger$.

Averaging the determinant with supersymmetry

The GOE average can be performed using

$$\mathbb{E}_{M\sim GOE} e^{-i\mathrm{Tr}MA} = \exp\left\{-\frac{1}{8N}\mathrm{Tr}(A+A^T)^2\right\}.$$

Averaging the determinant with supersymmetry

The GOE average can be performed using

$$\mathbb{E}_{M\sim GOE} e^{-i\mathrm{Tr}MA} = \exp\left\{-\frac{1}{8N}\mathrm{Tr}(A+A^T)^2\right\}.$$

So we have reduced

$$N^2 \text{ integrals (GOE average)}$$
$$\downarrow$$
$$2N \text{ commuting and } 4N \text{ anti-commuting integrals.}$$

We then reduce further to a fixed number of integrals amenable to steepest descents analysis.

Averaging the determinant with supersymmetry

The GOE average can be performed using

$$\mathbb{E}_{M\sim GOE} e^{-i\mathrm{Tr}MA} = \exp\left\{-\frac{1}{8N}\mathrm{Tr}(A+A^T)^2\right\}.$$

So we have reduced

$$N^2 \text{ integrals (GOE average)}$$
$$\downarrow$$
$$2N \text{ commuting and } 4N \text{ anti-commuting integrals.}$$

We then reduce further to a fixed number of integrals amenable to steepest descents analysis.

The S term gets in the way, so we expand to leading order to cope with it.

Conditioning on index

We first calculate C_N and then use an LDP to get to $C_{N,k}$.

Conditioning on index

We first calculate C_N and then use an LDP to get to $C_{N,k}$.

Need to calculate $\mathbb{E}_{M\sim GOE}\left\{e^{-i\mathrm{Tr}MA}\mathbb{1}(i(M-xI)=k)\right\}$.

Conditioning on index

We first calculate C_N and then use an LDP to get to $C_{N,k}$.

Need to calculate $\mathbb{E}_{M\sim GOE}\left\{e^{-i\mathrm{Tr}MA}\mathbb{1}(i(M-xI)=k)\right\}$.

$$\int^{x} d\mu(\lambda_1,\ldots,\lambda_k)\int_{x} d\mu(\lambda_{k+1},\ldots,\lambda_N)$$
$$\prod_{i\leq k<j}|\lambda_i-\lambda_j|\int d\mu_{Haar}(O)e^{-i\mathrm{Tr}O\Lambda O^T A}$$

We show that

$$\mathbb{E}_{M\sim GOE}\left\{e^{-i\mathrm{Tr}MA}\mathbb{1}(i(M-xI)=k)\right\}$$
$$\approx \mathbb{E}_{M\sim GOE}\left\{e^{-i\mathrm{Tr}MA}\right\}\mathbb{P}(i(M-xI)=k).$$

Using the interlacing property of eigenvalues, we have that $e^{-(k-1)Nl_1(x)}e^{-\frac{1}{2N}\mathrm{Tr}A^2}$ is

$$\lesssim |\mathbb{E}_{M\sim GOE}\left\{e^{-i\mathrm{Tr}MA}\mathbb{1}(i(M+S-xI)\in\{k-1,k,k+1)\right\}|$$
$$\lesssim e^{-(k+1)Nl_1(x)}e^{-\frac{1}{2N}\mathrm{Tr}A^2}$$

The rest of the calculation can then proceed as for C_N.

Summary

Understanding the highly complex landscape functions/surfaces that arise in machine learning problems is a key challenge in the area.

Summary

Understanding the highly complex landscape functions/surfaces that arise in machine learning problems is a key challenge in the area.

The banding structure 'explains' the unreasonable efficacy of methods to find the overall minimum: high saddles look more like maxima, lower saddles look more like minima, and actual minima band together with similar heights.

Is this Emperor clothed?

Is this Emperor clothed?

- Some (all?) of the assumptions made are questionable from the ML perspective and are not compellingly supported by numerical experiments

Is this Emperor clothed?

- Some (all?) of the assumptions made are questionable from the ML perspective and are not compellingly supported by numerical experiments
- In particular, the mean density of states of the Hessian of the loss function at a random point looks nothing like that of a GOE random matrix!

Is this Emperor clothed?

- Some (all?) of the assumptions made are questionable from the ML perspective and are not compellingly supported by numerical experiments
- In particular, the mean density of states of the Hessian of the loss function at a random point looks nothing like that of a GOE random matrix!
- Moreover, there is considerable rank-degeneracy, and outliers in the spectrum play an important role.

Is this Emperor clothed?

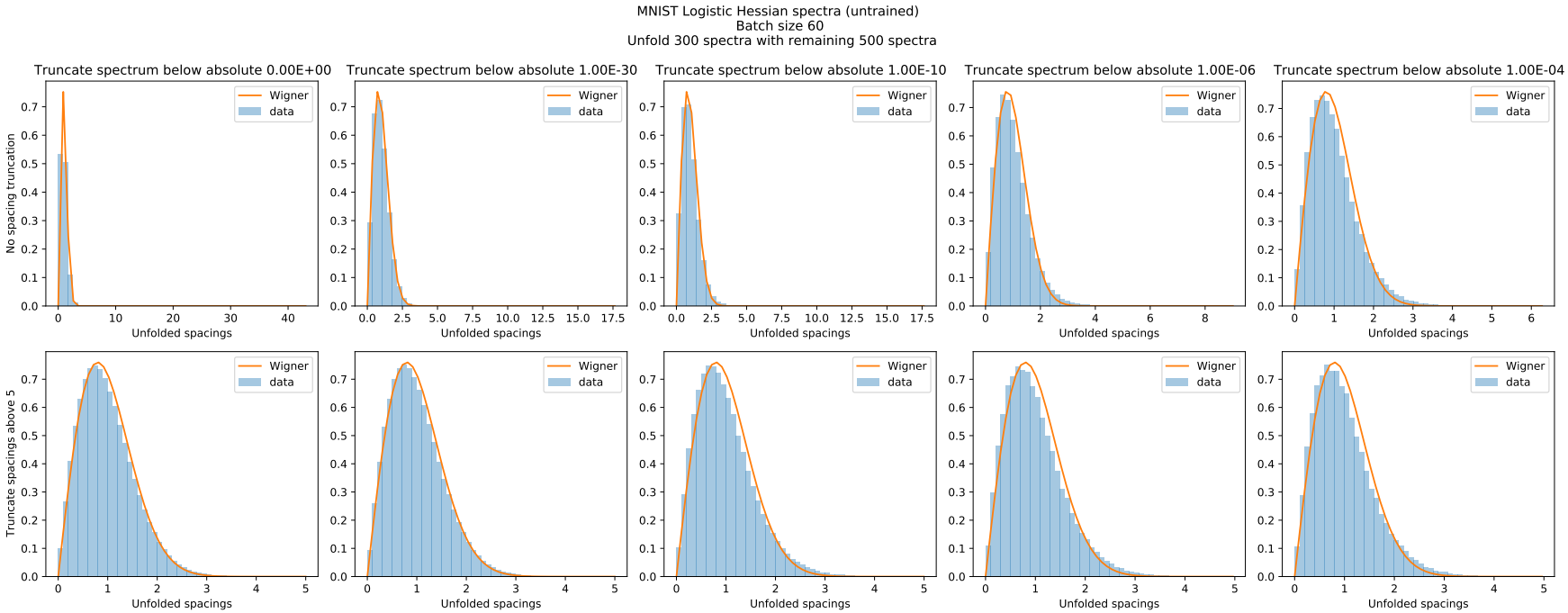
- Some (all?) of the assumptions made are questionable from the ML perspective and are not compellingly supported by numerical experiments
- In particular, the mean density of states of the Hessian of the loss function at a random point looks nothing like that of a GOE random matrix!
- Moreover, there is considerable rank-degeneracy, and outliers in the spectrum play an important role.

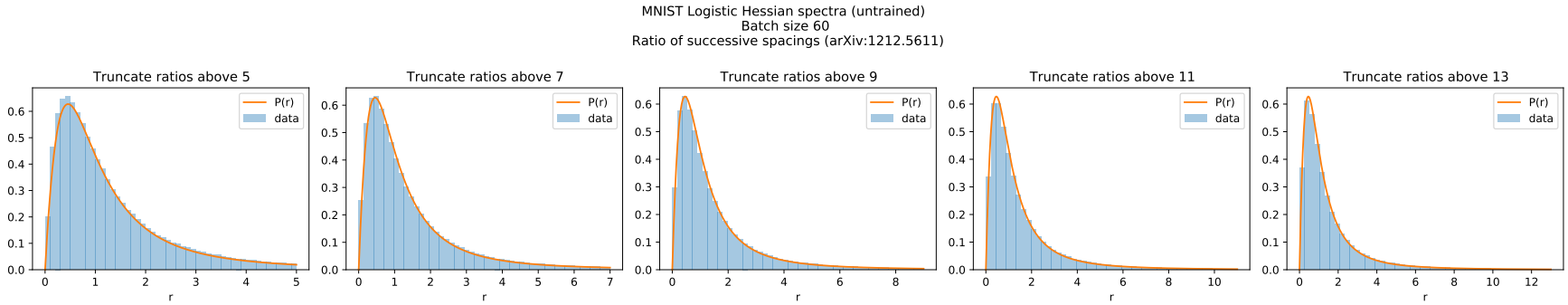
This raises the question as to what one should expect of a model.

On the other hand . . .

On the other hand . . .

Local statistics of the eigenvalues of the Hessian of the loss function (i.e. correlations on the scale of the mean eigenvalue separation) do match those of random GOE matrices closely! (Baskerville *et al.* 2022).




MNIST Logistic Hessian spectra (untrained)
Batch size 60
Ratio of successive spacings (arXiv:1212.5611)
Truncate ratios above 5
Truncate ratios above 7
Truncate ratios above 9
Truncate ratios above 11
Truncate ratios above 13
P(r)
data
r

Towards a realistic random matrix model

- a model can be developed based on general principles relating to local laws and eigenvector ergodicity

Towards a realistic random matrix model

- a model can be developed based on general principles relating to local laws and eigenvector ergodicity
- removes any need for GOE-type assumptions about the Hessian and instead leverages universal properties of the eigenvectors and eigenvalues of random matrices which are expected to hold for real networks

Towards a realistic random matrix model

- a model can be developed based on general principles relating to local laws and eigenvector ergodicity
- removes any need for GOE-type assumptions about the Hessian and instead leverages universal properties of the eigenvectors and eigenvalues of random matrices which are expected to hold for real networks
- seems to model both the bulk of the spectrum of the Hessian, and the outliers

Towards a realistic random matrix model

- a model can be developed based on general principles relating to local laws and eigenvector ergodicity
- removes any need for GOE-type assumptions about the Hessian and instead leverages universal properties of the eigenvectors and eigenvalues of random matrices which are expected to hold for real networks
- seems to model both the bulk of the spectrum of the Hessian, and the outliers
- appears to describe experimental data accurately, both qualitatively and quantitatively.

Towards a realistic random matrix model

- a model can be developed based on general principles relating to local laws and eigenvector ergodicity
- removes any need for GOE-type assumptions about the Hessian and instead leverages universal properties of the eigenvectors and eigenvalues of random matrices which are expected to hold for real networks
- seems to model both the bulk of the spectrum of the Hessian, and the outliers
- appears to describe experimental data accurately, both qualitatively and quantitatively.
- in particular, when applied to the Deep Linear Unconstrained Feature Model, this type of model describes well (and in this context explains) the important phenomenon of *neural collapse*.

Neural Collapse: Setting the Scene

A DNN $f : \mathbf{R}^d \to \mathbf{R}^K$ is applied to a K-class classification problem. View f as having two parts:

1. a feature map $h_\theta : \mathbf{R}^d \to \mathbf{R}^p$.
2. A last layer linear classifier $W(\cdot) + b : \mathbf{R}^p \to \mathbf{R}^K$, where $W \in \mathbf{R}^{K \times p}$, $b \in \mathbf{R}^K$.

The parameters (θ, W, b) are trained on a dataset $\cup_{c=1}^{K} \{x_{ic}\}_{i=1}^{n}$, where $x_{ic} \in \mathbf{R}^d$, using some variant of SGD on a loss function L, potentially with regularisation

$$\min_{\theta, W, b} \left\{ \sum_{c=1}^{K} \sum_{i=1}^{n} L(W h_\theta(x_{ic}) + b, y_c) + R(\theta, W, b) \right\}.$$

Early Stages of NC: The Simplex ETF

Define the following quantities:

$$\mu_c = \frac{1}{n}\sum_{i=1}^{n} h(x_{ic}), \quad \mu_G = \frac{1}{K}\sum_{c=1}^{K} \mu_c, \quad \tilde{\mu}_c = \mu_c - \mu_G$$

NC1: The feature vectors collapse to their class means $h(x_{ic}) \to \mu_c$

NC2: The globally centred feature means converge to a simplex equiangular tight frame

$$|\tilde{\mu}_c| - |\tilde{\mu}_{c'}| \to 0, \quad \forall c, c'$$

$$\frac{\tilde{\mu}_c \cdot \tilde{\mu}_{c'}}{|\tilde{\mu}_c||\tilde{\mu}_{c'}|} \to -\frac{1}{K-1}, \quad \forall c \neq c'$$

Late Stages of NC: Self-Duality

NC3: The linear classifier converges to the centred class means up to rescaling

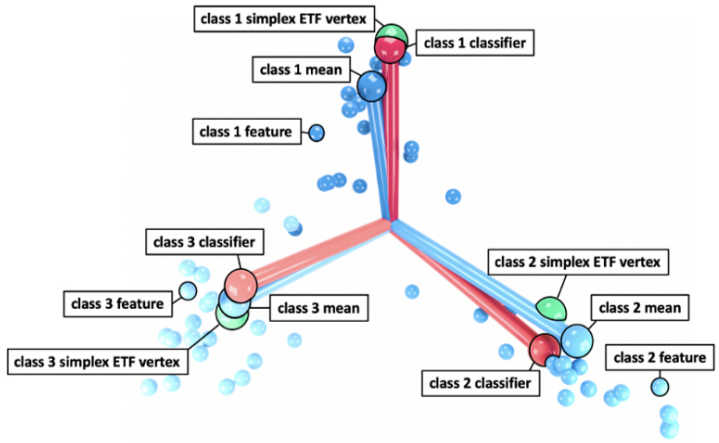
$$\left\| \frac{W^T}{\|W\|_F} - \frac{M}{\|M\|_F} \right\|_F \to 0$$

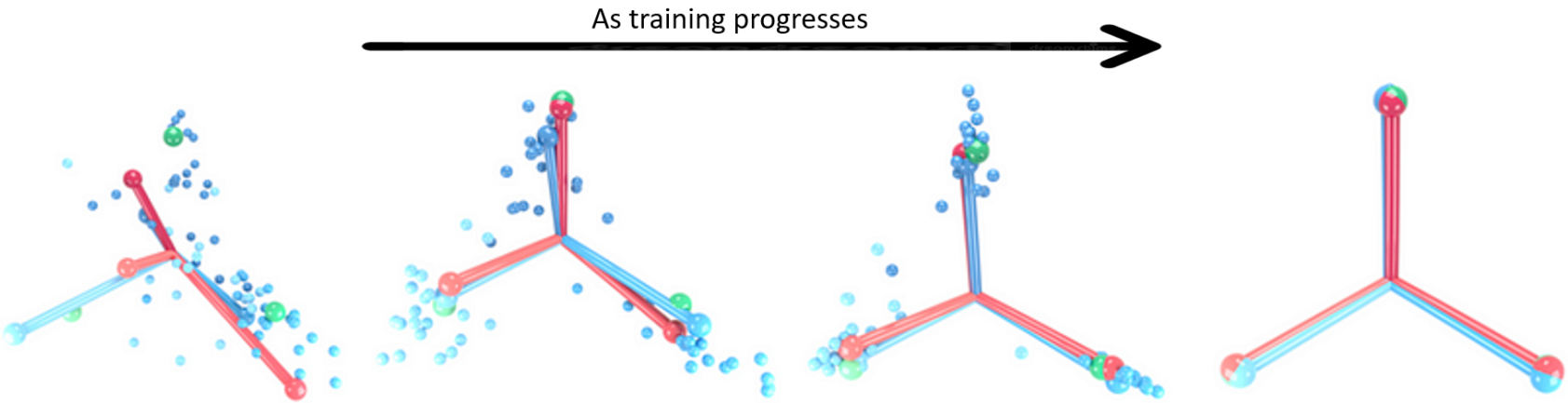
Where $M = [\tilde{\mu}_1, ..., \tilde{\mu}_K] \in \mathbf{R}^{p \times K}$ is the matrix whose columns are the centred class means.

NC4: The network classifier simplifies to performing nearest class centre classification

$$\text{argmax}_{c'}(w_{c'}^T h + b_{c'}) \to \text{argmin}_{c'} \|h - \mu_{c'}\|_2$$

Neural Collapse in Action



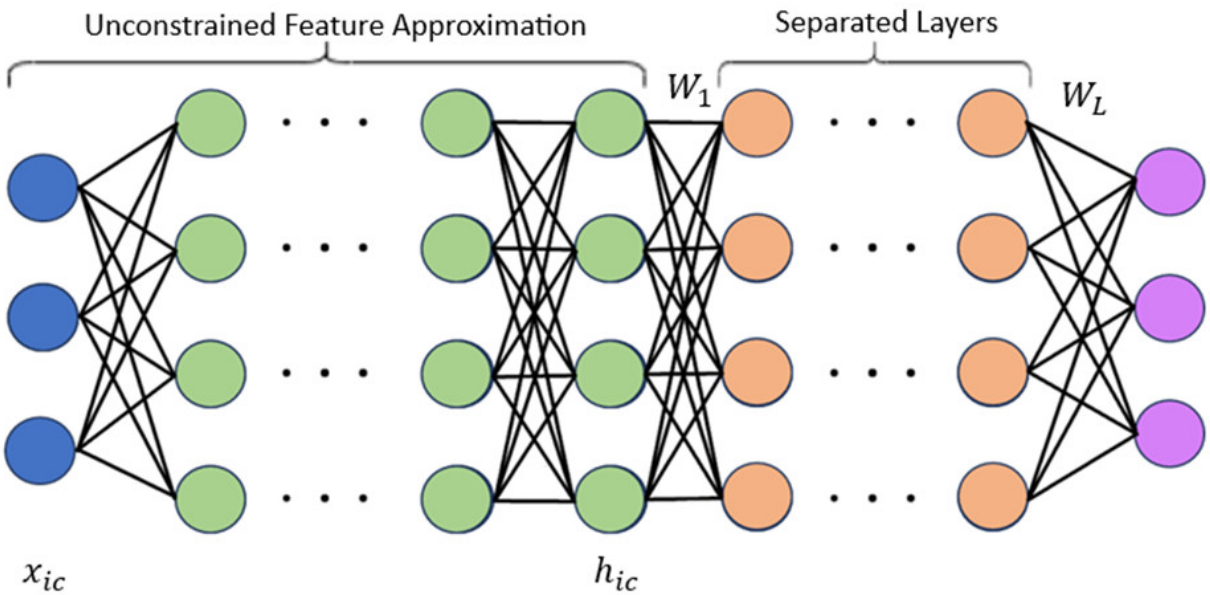


NC: A Connection to The Loss Surface

We can also choose to define our feature vectors as the output of an earlier layer than the penultimate one.

Unconstrained features: treat $h_{ic} = h(x_{ic})$ as freely optimised variables. We are now optimising over the parameters $(\{h_{ic}\}, W_1, ..., W_L)$.

Within this model neural collapse occurs on all of the separated layers.



NC: A Connection to The Loss Surface

Theorem (Keating and Garrod):
Consider the deep linear UFM. Let the width of the separated layers be greater than K, and regularisation be suitably small. Let $(W_L^*, ..., W_1^*, H_1^*)$ be a global optimum, then the rank of $Hess_l$ at this optimum is K^2. The (unnormalised) eigenvectors corresponding to non-zero eigenvalues are given by

$$\mu_c^{(l+1)} \otimes \mu_{c'}^{(l)}, \text{ for } c, c' \in \{1, ..., K\}.$$

In addition all the non-zero eigenvalues are equal, and have value $\alpha^{(l+1)2} \|\mu_c^{(l+1)}\|_2^2 \|\mu_{c'}^{(l)}\|_2^2 / K$, where $\alpha^{(l+1)}$ is a scaling constant expressible in terms of hyperparameters of the network.

Low Rank Bias: Is Neural Collapse Really Optimal?

outside of the linear MSE case, neural collapse is not globally optimal in deep unconstrained feature models. The culprit is a low-rank bias induced by weight decay.

$$\frac{1}{2}L\lambda\|H_L\|_{S_{2/L}}^{2/L} = \min_{W_l,H_1:H_L=W_{L-1}\ldots W_1H_1}\left\{\frac{1}{2}\lambda\sum_{l=1}^{L-1}\|W_l\|_F^2+\frac{1}{2}\lambda\|H_1\|_F^2\right\},$$

$$\|M\|_{S_{2/L}}^{2/L} = \sum_{i=1}^{\mathrm{rank}(M)} s_i^{\frac{2}{L}}.$$

Theorem (Keating & Garrod): *Consider the linear CE deep UFM, with $d \geq K$. If $K \geq 4, L \geq 3$, or $K \geq 6, L = 2$, then no solution with DNC structure can be a global minimum.*

Neural Collapse Persists Despite Low Rank Bias

Theorem (Keating & Garrod): *Consider the linear CE deep UFM with $d \geq K$. When the level of regularization $\lambda > 0$ is suitably small, we have the following:*

(i) There exists solutions with DNC structure at two different scales that are optimal points of the model. One of these scales induces a Hessian matrix that is positive semi-definite to leading order.

(ii) Let D_{DNC} denote the dimension of the space of network parameters that produce this DNC structure. Similarly, let D_{LR} denote the dimension of the space of parameters that produce some other optimal point of rank $r \in [2, K-1)$. Define the ratio of these dimensions $R(d) = D_{DNC}(d)/D_{LR}(d)$. This ratio is a monotonic increasing function on $d \geq K$, starting below 1 and tending towards $(K-1)/r > 1$ as $d \to \infty$.

Conclusions

Conclusions

- Ideas from random matrix theory and statistical mechanics appear naturally in the analysis of statistical features of machine learning.

Conclusions

- Ideas from random matrix theory and statistical mechanics appear naturally in the analysis of statistical features of machine learning.
- They model experimental data and explain key generic features that have been observed.

Conclusions

- Ideas from random matrix theory and statistical mechanics appear naturally in the analysis of statistical features of machine learning.
- They model experimental data and explain key generic features that have been observed.
- Much remains to be done in refining these connections and in analysing them asymptotically.