Random Matrix Theory in Deep Learning: An Introduction

Log-gases in Caeli Australi - Recent Developments in and around Random MATRIX Theory

Zhenyu Liao

School of Electronic Information and Communications
Huazhong University of Science and Technology

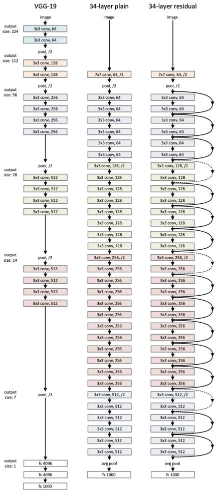
August 4, 2025

Outline

1. An Introduction Deep Learning for Mathematicians
2. Important Theoretical Questions for DL
3. Random (and Not-so Random) Matrix Theory in DL
   - Shallow and deep NN with random weights
   - NN with nonrandom weights
4. Conclusion

Outline

Question: what are deep neural networks?

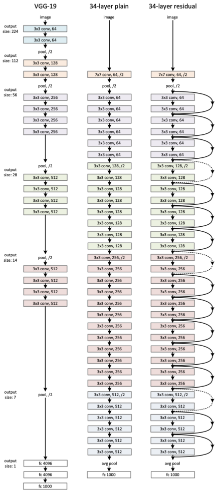


Deep Learning (DL) ≈ **multilayered neural network** (NN) is becoming the most popular machine learning (ML) model, but

- what is machine learning?
- what is a deep neural network (DNN)?
- how is such as network trained (i.e., the learning procedure)?
- is there any theory for DL, and if yes, how far is the theory from practice?

[1] Catherine F. Higham and Desmond J. Higham. "Deep Learning: An Introduction for Applied Mathematicians". In: *SIAM Review* 61.4 (Jan. 2019), pp. 860–891

Question: what are deep neural networks?



Deep Learning (DL) $\approx$ **multilayered neural network** (NN) is becoming the most popular machine learning (ML) model, but

- what is machine learning?
- what is a deep neural network (DNN)?
- how is such as network trained (i.e., the learning procedure)?
- is there any theory for DL, and if yes, how far is the theory from practice?

Credit: most materials in this part are borrowed from [HH19].

[1] Catherine F. Higham and Desmond J. Higham. "Deep Learning: An Introduction for Applied Mathematicians". In: *SIAM Review* 61.4 (Jan. 2019), pp. 860–891

Example: binary classification of points in $\mathbb{R}^2$

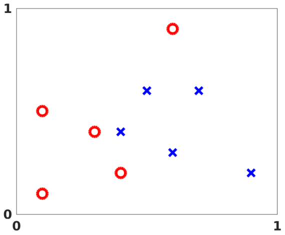


Figure: Labeled data points $\mathbf{x} \in \mathbb{R}^2$. Circles denote points in class $\mathcal{C}_1$. Crosses denote points in class $\mathcal{C}_2$.

- build a model/**function** f (from above historical data) that takes any points $\mathbf{x} \in \mathbb{R}^2$ and returns $\mathcal{C}_1$ or $\mathcal{C}_2$
- **logistic regression**: $\boxed{f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)}$ for $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ to be determined, and sigmoid function $\sigma(t) = \frac{1}{1+e^{-t}}$

Example: binary classification of points in $\mathbb{R}^2$

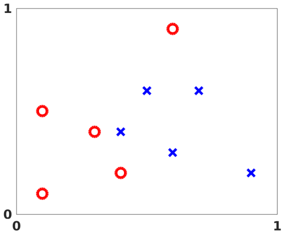


Figure: Labeled data points $\mathbf{x} \in \mathbb{R}^2$. Circles denote points in class $\mathcal{C}_1$. Crosses denote points in class $\mathcal{C}_2$.

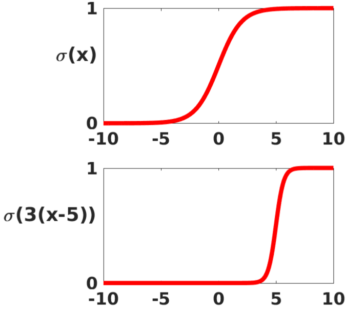


Figure: Sigmoid function.

- build a model/**function** f (from above historical data) that takes any points $\mathbf{x} \in \mathbb{R}^2$ and returns $\mathcal{C}_1$ or $\mathcal{C}_2$
- **logistic regression**: $\boxed{f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)}$ for $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ to be determined, and sigmoid function $\sigma(t) = \frac{1}{1+e^{-t}}$
- "learn" or estimate parameters $\mathbf{w}, b$ from data/samples, by minimizing some cost function (e.g., negative likelihood, MSE)
- predict $\mathbf{x} \in \mathcal{C}_1$ if $f(\mathbf{x}) < 1/2$ and $\mathbf{x} \in \mathcal{C}_2$ otherwise.

Neural networks are nothing but "cascaded" logistic regressors

- logistic regression $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ extends to

$$\boxed{f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^N} \quad \mathbf{W} \in \mathbb{R}^{N\times 2}, \mathbf{b} \in \mathbb{R}^N \tag{1}$$

  and $\sigma(\cdot)$ applied entry-wise: this is **one layer** of a DNN

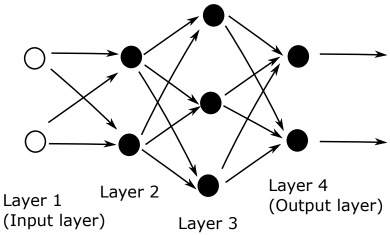


Figure: A network with four layers.

Neural networks are nothing but "cascaded" logistic regressors

- logistic regression $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ extends to

$$\boxed{f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^N} \quad \mathbf{W} \in \mathbb{R}^{N \times 2}, \mathbf{b} \in \mathbb{R}^N \quad (1)$$

  and $\sigma(\cdot)$ applied entry-wise: this is **one layer** of a DNN
- repeat this to make the network deep, with possibly different width in each layer

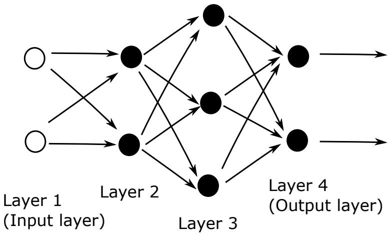


Figure: A network with four layers.

Neural networks are nothing but "cascaded" logistic regressors

- logistic regression $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ extends to

$$\boxed{f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^N} \quad \mathbf{W} \in \mathbb{R}^{N \times 2}, \mathbf{b} \in \mathbb{R}^N \quad (1)$$

  and $\sigma(\cdot)$ applied entry-wise: this is **one layer** of a DNN
- repeat this to make the network deep, with possibly different width in each layer
- $\sigma(\mathbf{W}_2 x + \mathbf{b}_2) \in \mathbb{R}^2, \sigma\left(\mathbf{W}_3 \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2) + \mathbf{b}_3\right) \in \mathbb{R}^3$

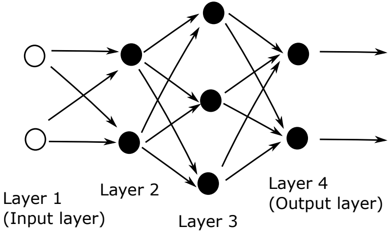


Figure: A network with four layers.

Neural networks are nothing but "cascaded" logistic regressors

- logistic regression $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ extends to

$$\boxed{f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^N} \quad \mathbf{W} \in \mathbb{R}^{N \times 2}, \mathbf{b} \in \mathbb{R}^N \tag{1}$$

  and $\sigma(\cdot)$ applied entry-wise: this is **one layer** of a DNN
- repeat this to make the network deep, with possibly different width in each layer

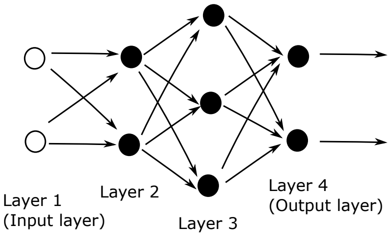


Figure: A network with four layers.

- $\sigma(\mathbf{W}_2 x + \mathbf{b}_2) \in \mathbb{R}^2$, $\sigma\left(\mathbf{W}_3 \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2) + \mathbf{b}_3\right) \in \mathbb{R}^3$
- $f_{4L-NN}(\mathbf{x}) = \sigma\left(\mathbf{W}_4 \sigma\left(\mathbf{W}_3 \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2) + \mathbf{b}_3\right) + \mathbf{b}_4\right) \in \mathbb{R}^2$

Neural networks are nothing but "cascaded" logistic regressors

- logistic regression $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$ extends to

$$\boxed{f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^N} \quad \mathbf{W} \in \mathbb{R}^{N \times 2}, \mathbf{b} \in \mathbb{R}^N \tag{1}$$

  and $\sigma(\cdot)$ applied entry-wise: this is **one layer** of a DNN
- repeat this to make the network deep, with possibly different width in each layer

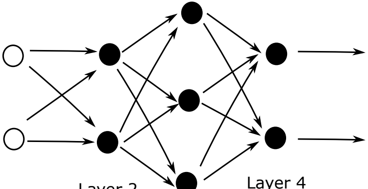


Figure: A network with four layers.

- $\sigma(\mathbf{W}_2 x + \mathbf{b}_2) \in \mathbb{R}^2$, $\sigma\left(\mathbf{W}_3 \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2) + \mathbf{b}_3\right) \in \mathbb{R}^3$
- $f_{4L-NN}(\mathbf{x}) = \sigma\left(\mathbf{W}_4 \sigma\left(\mathbf{W}_3 \sigma(\mathbf{W}_2 \mathbf{x} + \mathbf{b}_2) + \mathbf{b}_3\right) + \mathbf{b}_4\right) \in \mathbb{R}^2$

Define the label/target output as

$$\mathbf{y}(\mathbf{x}_i) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \mathbf{x}_i \in \mathcal{C}_1, \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \mathbf{x}_i \in \mathcal{C}_2. \end{cases} \tag{2}$$

the MSE cost function writes $\mathrm{Cost}\left(\mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4\right) = \frac{1}{10} \sum_{i=1}^{10} \|\mathbf{y}(\mathbf{x}_i) - f_{4L-NN}(\mathbf{x}_i)\|^2$

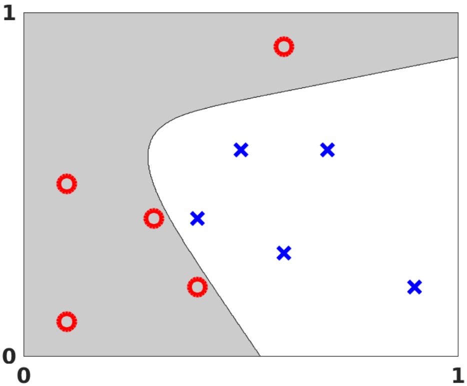


Figure: Visualization of output from a multilayered neural network applied to the data.

- from training to test!

General formulation and gradient decent training of DNN

We can define the network in a **layer-by-layer** fashion:

$$\mathbf{a}_0 = \mathbf{x} \in \mathbb{R}^{N_0}, \quad \boxed{\mathbf{a}_\ell = \sigma\left(\mathbf{W}_\ell \mathbf{a}_{\ell-1} + \mathbf{b}_\ell\right)} \in \mathbb{R}^{N_\ell}, \quad \ell = 1, \ldots, L,$$

with weights $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias $\mathbf{b} \in \mathbb{R}^{N_\ell}$ at layer ℓ.

- $\mathbf{W}_\ell$s and $\mathbf{b}_\ell$s obtained by minimizing cost function on a given training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of size n:

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \|\mathbf{y}_i - \mathbf{a}_L(\mathbf{x}_i)\|^2. \tag{3}$$

General formulation and gradient decent training of DNN

We can define the network in a **layer-by-layer** fashion:

$$\mathbf{a}_0 = \mathbf{x} \in \mathbb{R}^{N_0}, \quad \boxed{\mathbf{a}_\ell = \sigma\left(\mathbf{W}_\ell \mathbf{a}_{\ell-1} + \mathbf{b}_\ell\right)} \in \mathbb{R}^{N_\ell}, \quad \ell = 1, \ldots, L,$$

with weights $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias $\mathbf{b} \in \mathbb{R}^{N_\ell}$ at layer ℓ.

- $\mathbf{W}_\ell$s and $\mathbf{b}_\ell$s obtained by minimizing cost function on a given training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of size n:

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{y}_i - \mathbf{a}_L(\mathbf{x}_i)\|^2. \tag{3}$$

- update using (stochastic) gradient descent, for parameter P,

$$P(t+1) = P(t) - \eta \nabla_P \text{Cost}(P(t)). \tag{4}$$

Matlab code to train a simple NN

```
%%%%%% DATA %%%%%%%%%%%
x1 = [0.1,0.3,0.1,0.6,0.4,0.6,0.5,0.9,0.4,0.7]; x2 = [0.1,0.4,0.5,0.9,0.2,0.3,0.6,0.2,0.4,0.6]; y = [ones(1,5) zeros(1,5); zeros(1,5) ones(1,5)];

% Initialize weights and biases
W2 = 0.5*randn(2,2); W3 = 0.5*randn(3,2); W4 = 0.5*randn(2,3); b2 = 0.5*randn(2,1); b3 = 0.5*randn(3,1); b4 = 0.5*randn(2,1);

% Forward and Back propagate
eta = 0.05;                % learning rate
Niter = 1e6;               % number of SG iterations
savecost = zeros(Niter,1); % value of cost function at each iteration
for counter = 1:Niter
    k = randi(10);         % choose a training point at random
    x = [x1(k); x2(k)];
    % Forward pass
    a2 = activate(x,W2,b2); a3 = activate(a2,W3,b3); a4 = activate(a3,W4,b4);
    % Backward pass
    delta4 = a4.*(1-a4).*(a4-y(:,k)); delta3 = a3.*(1-a3).*(W4'*delta4); delta2 = a2.*(1-a2).*(W3'*delta3);
    % Gradient step
    W2 = W2 - eta*delta2*x'; W3 = W3 - eta*delta3*a2'; W4 = W4 - eta*delta4*a3'; b2 = b2 - eta*delta2; b3 = b3 - eta*delta3; b4 = b4 - eta*delta4;
    % Monitor progress
    newcost = cost(W2,W3,W4,b2,b3,b4)   % display cost to screen
    savecost(counter) = newcost;
end

% Show decay of cost function
semilogy([1:1e4:Niter],savecost(1:1e4:Niter))

  function costval = cost(W2,W3,W4,b2,b3,b4)
     costvec = zeros(10,1);
     for i = 1:10
         x =[x1(i);x2(i)];
         a2 = activate(x,W2,b2); a3 = activate(a2,W3,b3); a4 = activate(a3,W4,b4);
         costvec(i) = norm(y(:,i) - a4,2);
     end
     costval = norm(costvec,2)^2;
  end % of nested function
```

Matlab code to train a simple NN

```
function y = activate(x,W,b)
%ACTIVATE  Evaluates sigmoid function.
%
%  x is the input vector,  y is the output vector
%  W contains the weights, b contains the shifts
%
%  The ith component of y is activate((Wx+b)_i)
%  where activate(z) = 1/(1+exp(-z))

y = 1./(1+exp(-(W*x+b)));
```

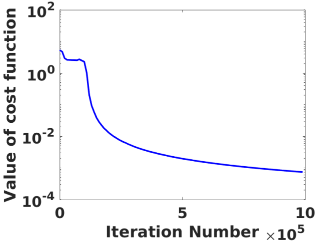


Figure: Vertical axis shows a scaled value of the cost function. Horizontal axis shows the iteration number. Here we used the stochastic gradient descent to train the aforementioned simple network.

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$
\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}
$$

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}$$

- different choice of activation and/or cost function:

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}$$

- different choice of activation and/or cost function:
  - rectified linear unit, or **ReLU**, activation: $\sigma(t) = \max(t, 0)$

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}$$

- different choice of activation and/or cost function:
  - rectified linear unit, or **ReLU**, activation: $\sigma(t) = \max(t, 0)$
  - **cross-entropy** cost function:

$$-\sum_{i=1}^{N} \log \left(\frac{e^{v_{l_i}^{\{i\}}}}{\sum_{j=1}^{K} e^{v_j^{\{i\}}}} \right). \tag{6}$$

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \text{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}$$

- different choice of activation and/or cost function:
  - rectified linear unit, or **ReLU**, activation: $\sigma(t) = \max(t, 0)$
  - **cross-entropy** cost function:

$$-\sum_{i=1}^{N} \log \left(\frac{e^{v_{l_i}^{\{i\}}}}{\sum_{j=1}^{K} e^{v_j^{\{i\}}}} \right). \tag{6}$$

- dropout, batch normalization, and other types of normalization, etc.

Some commonly used tricks in DNN

- **stochastic gradient descent**: sample (without replacement) a mini-batch for gradient $\frac{1}{B}\sum_{i=1}^{B} \nabla_P \mathrm{Cost}(\mathbf{x}_i)$
- **convolution neural network (CNN)**: repeatedly apply small linear kernel, or filter, across portions of input data, making weight matrices **sparse** and highly **structured**

$$\begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{5\times 6}. \tag{5}$$

- different choice of activation and/or cost function:
  - rectified linear unit, or **ReLU**, activation: $\sigma(t) = \max(t, 0)$
  - **cross-entropy** cost function:

$$-\sum_{i=1}^{N} \log \left(\frac{e^{v_{l_i}^{\{i\}}}}{\sum_{j=1}^{K} e^{v_j^{\{i\}}}} \right). \tag{6}$$

- dropout, batch normalization, and other types of normalization, etc.
- use of **tensors** instead of vectors or matrices for input data or intermediate representations

Outline

What do we care about DL, from a theoretical perspective?

What does **deep learning theory** care about and why?

- **theoretical guarantee**: explanation of **when** and **why** DL works in some cases, and not in others

What do we care about DL, from a theoretical perspective?

What does **deep learning theory** care about and why?

- **theoretical guarantee**: explanation of **when** and **why** DL works in some cases, and not in others
- theory-guided **design principles** for more efficient DNN (e.g., better performant, less computational demand, more novel ideas on how to make DL work better, etc.)

What do we care about DL, from a theoretical perspective?

What does **deep learning theory** care about and why?

- **theoretical guarantee**: explanation of **when** and **why** DL works in some cases, and not in others
- theory-guided **design principles** for more efficient DNN (e.g., better performant, less computational demand, more novel ideas on how to make DL work better, etc.)
- **too many** "tuning" hyperparameters in DNN design: number of layers, operator, width, and activation in each layer, different tricks, etc.

What do we care about DL, from a theoretical perspective?

What does **deep learning theory** care about and why?

- **theoretical guarantee**: explanation of **when** and **why** DL works in some cases, and not in others
- theory-guided **design principles** for more efficient DNN (e.g., better performant, less computational demand, more novel ideas on how to make DL work better, etc.)
- **too many** "tuning" hyperparameters in DNN design: number of layers, operator, width, and activation in each layer, different tricks, etc.
- for safety-related applications (e.g., self driving, healthcare), we need theory-supported DL

A (too) brief review of DL theory

From an approximation theoretical perspective:

- **universal approximation theorem**: for any (somewhat regular, e.g., Lebesgue p-integrable) function of interest $f \colon \mathbb{R}^{p \times K}$ and given $\varepsilon > 0$, there exists a fully-connected ReLU network F with width at least m such that $\int_{\mathbb{R}^p} \|f(\mathbf{x}) - F(\mathbf{x})\|^p d\mathbf{x} < \varepsilon$.
- different type of input space, e.g., $\mathbf{x} = [x_1, \ldots, x_p] \subset [0,1]^p$, function or data on graph?
- how activation, width, depth, etc. come into play, in particular, **depth versus width**?
- **LIMITATION**: do NOT provide recipe for construction, but that such a construction is possible

A (too) brief review of DL theory

From an approximation theoretical perspective:

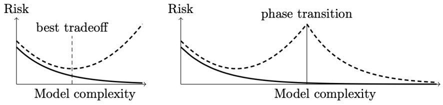
- **universal approximation theorem**: for any (somewhat regular, e.g., Lebesgue p-integrable) function of interest $f : \mathbb{R}^{p\times K}$ and given $\varepsilon > 0$, there exists a fully-connected ReLU network F with width at least m such that $\int_{\mathbb{R}^p} \|f(\mathbf{x}) - F(\mathbf{x})\|^p d\mathbf{x} < \varepsilon$.
- different type of input space, e.g., $\mathbf{x} = [x_1, \ldots, x_p] \subset [0,1]^p$, function or data on graph?
- how activation, width, depth, etc. come into play, in particular, **depth versus width**?
- **LIMITATION**: do NOT provide recipe for construction, but that such a construction is possible

From an optimization perspective:

- DNN training involves non-convex (and possibly non-smooth) optimization: challenging!
- **empirically** simple (stochastic) gradient descent seems to work well, WHY?
- GUESS: DL landscape has nice properties?
- e.g., how to converge better and faster?
- **IMPORTANT**: pure optimization deals only with training, and **NOT** test/**generalization**

From a statistical perspective:

- **generalization theory**: for which type of **data**, and by using which **ML model** (trained with which **algorithm**), can we get a high probability error bound of which **metric**



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

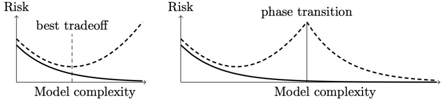
A Good DL theory should cover **both** optimization **and** generalization!

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580

[3] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep Learning: A Statistical Viewpoint". In: *Acta Numerica* 30 (May 2021), pp. 87–201

From a statistical perspective:

- **generalization theory**: for which type of **data**, and by using which **ML model** (trained with which **algorithm**), can we get a high probability error bound of which **metric**
- Rademacher complexity, PAC-Bayes bound, etc.



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

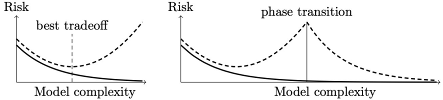
A Good DL theory should cover **both** optimization **and** generalization!

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580

[3] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep Learning: A Statistical Viewpoint". In: *Acta Numerica* 30 (May 2021), pp. 87–201

From a statistical perspective:

- **generalization theory**: for which type of **data**, and by using which **ML model** (trained with which **algorithm**), can we get a high probability error bound of which **metric**
- Rademacher complexity, PAC-Bayes bound, etc.
- **Question**: why DL models **generalize so well** despite high model complexity (i.e., **over-parameterized**)?



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

A Good DL theory should cover **both** optimization **and** generalization!

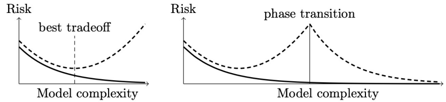
[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580

[3] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep Learning: A Statistical Viewpoint". In: *Acta Numerica* 30 (May 2021), pp. 87–201

A (too) brief review of DL theory

From a statistical perspective:

- **generalization theory**: for which type of **data**, and by using which **ML model** (trained with which **algorithm**), can we get a high probability error bound of which **metric**
- Rademacher complexity, PAC-Bayes bound, etc.
- **Question**: why DL models **generalize so well** despite high model complexity (i.e., **over-parameterized**)?
  1. nice property of the (over-parameterized) DL model: Neural Tangent Kernel [JGH18]



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

A Good DL theory should cover **both** optimization **and** generalization!

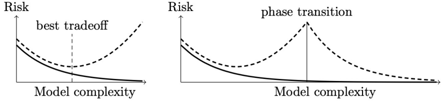
[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580

[3] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep Learning: A Statistical Viewpoint". In: *Acta Numerica* 30 (May 2021), pp. 87–201

A (too) brief review of DL theory

From a statistical perspective:

- **generalization theory**: for which type of **data**, and by using which **ML model** (trained with which **algorithm**), can we get a high probability error bound of which **metric**
- Rademacher complexity, PAC-Bayes bound, etc.
- **Question**: why DL models **generalize so well** despite high model complexity (i.e., **over-parameterized**)?
  1. nice property of the (over-parameterized) DL model: Neural Tangent Kernel [JGH18]
  2. inductive bias due to algorithm: Double Descent or Benign Overfitting [BMR21]



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

A Good DL theory should cover **both** optimization **and** generalization!

[2] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*. 2018, pp. 8571–8580

[3] Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep Learning: A Statistical Viewpoint". In: *Acta Numerica* 30 (May 2021), pp. 87–201

- kernel $K(\cdot,\cdot)\colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, **similarity** measure between input data points in $\mathbb{R}^p$

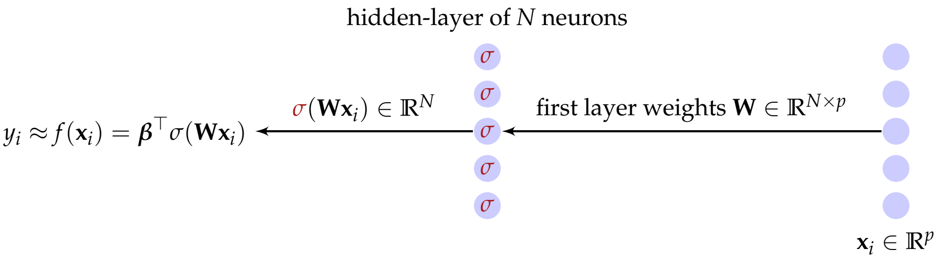
Neural Tangent Kernel

- kernel $K(\cdot,\cdot)\colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, **similarity** measure between input data points in $\mathbb{R}^p$
- examples include:
  - linear kernel $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, cosine kernel $= \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}$, Gaussian (RBF) kernel $= \exp(\|\mathbf{x}-\mathbf{y}\|^2/\gamma^2)$
  - kernel induced by NN: $K(\mathbf{x},\mathbf{y}) = \sigma(\mathbf{W}\mathbf{x})^\top \sigma(\mathbf{W}\mathbf{y})$, parameterized by the network (e.g., weights and activations)

Neural Tangent Kernel

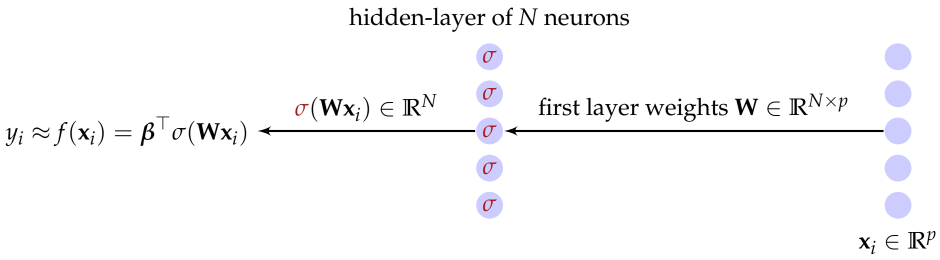
- kernel $K(\cdot,\cdot)\colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, **similarity** measure between input data points in $\mathbb{R}^p$
- examples include:
  - linear kernel $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, cosine kernel $= \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}$, Gaussian (RBF) kernel $= \exp(\|\mathbf{x}-\mathbf{y}\|^2/\gamma^2)$
  - kernel induced by NN: $K(\mathbf{x},\mathbf{y}) = \sigma(\mathbf{W}\mathbf{x})^\top \sigma(\mathbf{W}\mathbf{y})$, parameterized by the network (e.g., weights and activations)
- PS: kernels are widely studied in the ML literature, we know quite a lot (reproducing kernel Hilbert space, RKHS, etc.)

Example of a two-layer NN model



- Given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

$$f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}) = \sum_{\ell=1}^{n} \beta_\ell \sigma(\mathbf{w}_\ell^\top \mathbf{x}), \quad \boldsymbol{\theta} = [\beta_1, \ldots, \beta_N; \mathbf{w}_1, \ldots, \mathbf{w}_N]. \tag{7}$$



- Given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

$$f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}) = \sum_{\ell=1}^{n} \beta_\ell \sigma(\mathbf{w}_\ell^\top \mathbf{x}), \quad \boldsymbol{\theta} = [\beta_1, \ldots, \beta_N; \mathbf{w}_1, \ldots, \mathbf{w}_N]. \tag{7}$$

- linearization of the network at initialization, by Taylor expansion

$$f(\mathbf{x}; \boldsymbol{\theta}) \approx f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{x}; \boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boxed{\nabla_\theta f(\mathbf{x}; \boldsymbol{\theta}_0)}. \tag{8}$$

and

$$f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}_0 + \delta) = f(\mathbf{x}; \boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \phi_{\text{NTK}}(\mathbf{x}), \quad K_{e-NTK}(\mathbf{x}, \mathbf{y}) = \phi_{\text{NTK}}(\mathbf{x})^\top \phi_{\text{NTK}}(\mathbf{y}). \tag{9}$$

The big picture of NTK

- around initialization $\boldsymbol{\theta} \approx \boldsymbol{\theta}_0$, linearized network output

$$f(\mathbf{x}; \boldsymbol{\theta}) \approx f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}_0 + \delta) = f(\mathbf{x}; \boldsymbol{\theta}_0) + \delta^\top \phi_{\text{NTK}}(\mathbf{x}), \quad K_{e-NTK}(\mathbf{x}, \mathbf{y}) = \phi_{\text{NTK}}(\mathbf{x})^\top \phi_{\text{NTK}}(\mathbf{y}), \tag{10}$$

The big picture of NTK

- around initialization $\boldsymbol{\theta} \approx \boldsymbol{\theta}_0$, linearized network output

$$f(\mathbf{x};\boldsymbol{\theta}) \approx f_{\rm lin}(\mathbf{x};\boldsymbol{\theta}_0 + \delta) = f(\mathbf{x};\boldsymbol{\theta}_0) + \delta^\top \phi_{\rm NTK}(\mathbf{x}), \quad K_{e-NTK}(\mathbf{x},\mathbf{y}) = \phi_{\rm NTK}(\mathbf{x})^\top \phi_{\rm NTK}(\mathbf{y}), \tag{10}$$

Now, if there exists a neighborhood $B(\boldsymbol{\theta}_0)$ of $\boldsymbol{\theta}_0$ such that

1. for any $\boldsymbol{\theta} \in B(\boldsymbol{\theta}_0)$, we have $f(\mathbf{x};\boldsymbol{\theta}) \approx f_{\rm lin}(\mathbf{x};\boldsymbol{\theta})$, and closeness in cost function
2. it suffices to optimize in $B(\boldsymbol{\theta}_0)$ to reach an approx. global min, i.e., $f(\mathbf{x};\boldsymbol{\theta}_0) \approx f_{\rm lin}(\mathbf{x};\boldsymbol{\theta}_0) \approx 0$
3. from an optimization viewpoint, optimizing $f(\mathbf{x};\boldsymbol{\theta}) \approx$ optimizing $f_{\rm lin}$ and will not leave $B(\boldsymbol{\theta}_0)$

- around initialization $\boldsymbol{\theta} \approx \boldsymbol{\theta}_0$, linearized network output

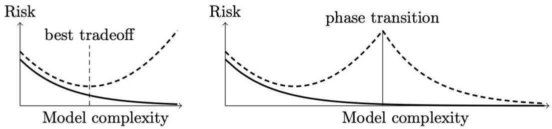
$$f(\mathbf{x}; \boldsymbol{\theta}) \approx f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}_0 + \boldsymbol{\delta}) = f(\mathbf{x}; \boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \phi_{\text{NTK}}(\mathbf{x}), \quad K_{e-NTK}(\mathbf{x}, \mathbf{y}) = \phi_{\text{NTK}}(\mathbf{x})^\top \phi_{\text{NTK}}(\mathbf{y}), \tag{10}$$

Now, if there exists a neighborhood $B(\boldsymbol{\theta}_0)$ of $\boldsymbol{\theta}_0$ such that

1. for any $\boldsymbol{\theta} \in B(\boldsymbol{\theta}_0)$, we have $f(\mathbf{x}; \boldsymbol{\theta}) \approx f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta})$, and closeness in cost function
2. it suffices to optimize in $B(\boldsymbol{\theta}_0)$ to reach an approx. global min, i.e., $f(\mathbf{x}; \boldsymbol{\theta}_0) \approx f_{\text{lin}}(\mathbf{x}; \boldsymbol{\theta}_0) \approx 0$
3. from an optimization viewpoint, optimizing $f(\mathbf{x}; \boldsymbol{\theta}) \approx$ optimizing f_{lin} and will not leave $B(\boldsymbol{\theta}_0)$

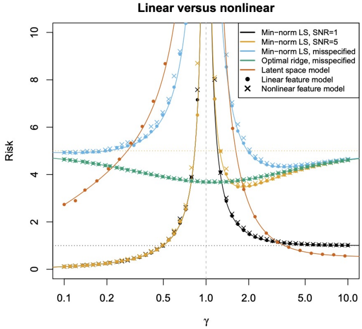
To reach the above is **over-parameterization** and/or **proper random initialization**, with small stochasticity (e.g., small learning rate or full batch GD)

- cost function (e.g., MSE) $\text{Cost}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) \approx \text{Cost}(f_{\text{lin}}(\mathbf{x}), \mathbf{y})$ linear (in the parameter $\boldsymbol{\theta}$) and convex!
- for MSE, $\text{Cost}(f_{\text{lin}}(\mathbf{X}), \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (f_{\text{lin}}(\mathbf{x}_i) - y_i)^2$, nothing but linear regression of type $\text{Cost} = \|\mathbf{y}' - \boldsymbol{\Phi}_{NTK}(\mathbf{X})^\top \delta\|^2$ with $y_i' = f(\mathbf{x}_i; \boldsymbol{\theta}_0) - y$



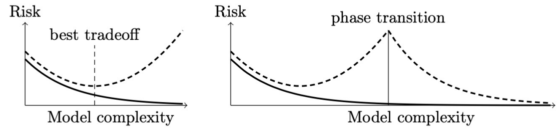
(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

- larger model, the better?! Maybe, due to double descent [Has+22] and implicit (norm-based?) bias



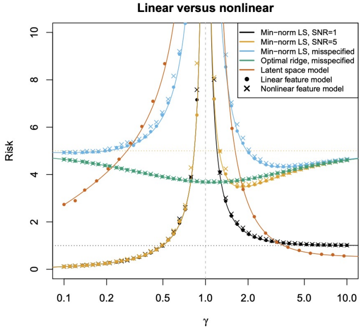
[4]Trevor Hastie et al. "Surprises in High-Dimensional Ridgeless Least Squares Interpolation". In: *The Annals of Statistics* 50.2 (Apr. 2022), pp. 949–986

Precise Characterization of Double Descent Curves



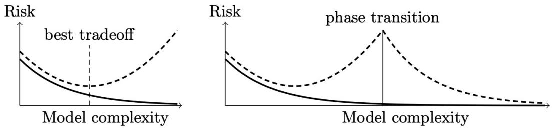
(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

- larger model, the better?! Maybe, due to double descent [Has+22] and implicit (norm-based?) bias
- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n-p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime



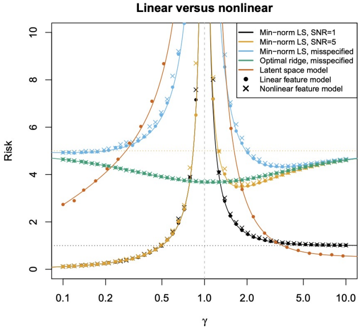
[4] Trevor Hastie et al. "Surprises in High-Dimensional Ridgeless Least Squares Interpolation". In: *The Annals of Statistics* 50.2 (Apr. 2022), pp. 949–986

Precise Characterization of Double Descent Curves



(a) Classical U-shaped curve (b) Modern double descent "UL"-shaped test curve

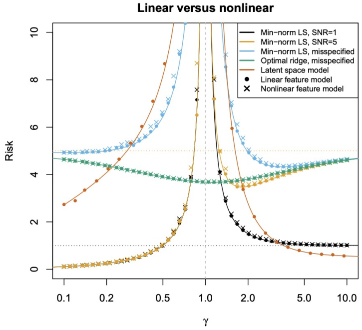
- larger model, the better?! Maybe, due to double descent [Has+22] and implicit (norm-based?) bias
- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$,
  depend on the sign $n - p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime
- **generalization** risk shows a double descent curve



[4] Trevor Hastie et al. "Surprises in High-Dimensional Ridgeless Least Squares Interpolation". In: *The Annals of Statistics* 50.2 (Apr. 2022), pp. 949–986

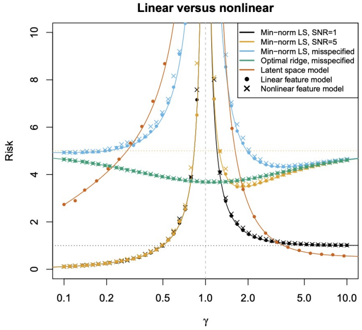
Precise Characterization of Double Descent Curves

- case of linear regression model
  Cost $= \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n - p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime



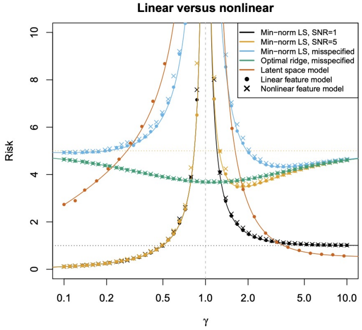
Precise Characterization of Double Descent Curves

- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n-p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime
- **generalization** risk shows a double descent curve [Has+22]



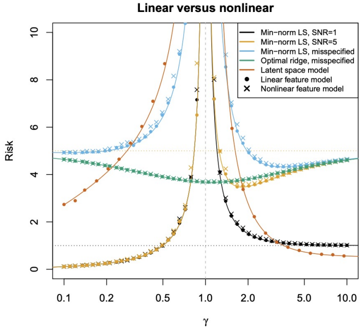
Precise Characterization of Double Descent Curves

- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n-p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime
- **generalization** risk shows a double descent curve [Has+22]
- very **understandable** for RMT experts:



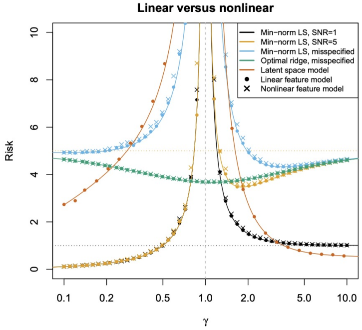
Precise Characterization of Double Descent Curves

- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n - p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime
- generalization risk shows a double descent curve [Has+22]
- very **understandable** for RMT experts:
- ridgeless least squares $\hat{\boldsymbol{\beta}} = (\mathbf{X}\mathbf{X}^\mathsf{T})^{-1}\mathbf{X}\mathbf{y}$ or $\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{y}$ and there is a **singular behavior** in the spectrum at $p = n$



Precise Characterization of Double Descent Curves

- case of linear regression model
  Cost $= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\beta}^\top \mathbf{x}_i - y_i)^2$, with $\boldsymbol{\beta}, \mathbf{x}_i \in \mathbb{R}^p$, depend on the sign $n-p$, either in the **over-parameterized** or **under-parameterized** (with min-norm solution) regime
- **generalization** risk shows a double descent curve [Has+22]
- very **understandable** for RMT experts:
- ridgeless least squares $\hat{\boldsymbol{\beta}} = (\mathbf{X}\mathbf{X}^\mathsf{T})^{-1}\mathbf{X}\mathbf{y}$ or $\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{y}$ and there is a **singular behavior** in the spectrum at $p = n$
- tons of extensions: relaxing assumption, (slightly) more involved models, etc., less progress in the sense of deep though



Outline

Technical challenges and opportunities for RMT in DL theory

- entry-wise non-linearity and depth: some **successful** efforts

Technical challenges and opportunities for RMT in DL theory

- entry-wise non-linearity and depth: some **successful** efforts
- gradient descent leads to involved correlation structure: even a **single** step makes things complicated

Technical challenges and opportunities for RMT in DL theory

- entry-wise non-linearity and depth: some **successful** efforts
- gradient descent leads to involved correlation structure: even a **single** step makes things complicated
- more involved DNN structure (convolution, recurrent, Attention core in Large Language Models, etc.)

Technical challenges and opportunities for RMT in DL theory

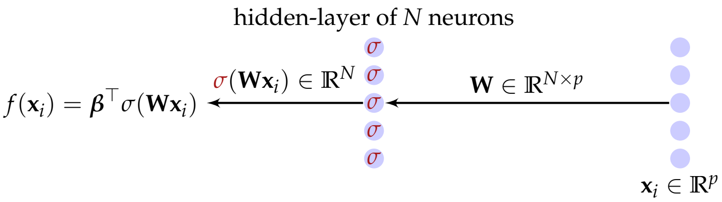
- entry-wise non-linearity and depth: some **successful** efforts
- gradient descent leads to involved correlation structure: even a **single** step makes things complicated
- more involved DNN structure (convolution, recurrent, Attention core in Large Language Models, etc.)
- statistical assumption on data to work with: largely open!

Technical challenges and opportunities for RMT in DL theory

- entry-wise non-linearity and depth: some **successful** efforts
- gradient descent leads to involved correlation structure: even a **single** step makes things complicated
- more involved DNN structure (convolution, recurrent, Attention core in Large Language Models, etc.)
- statistical assumption on data to work with: largely open!
- **beyond** the linear $n \sim p \sim N$ regime

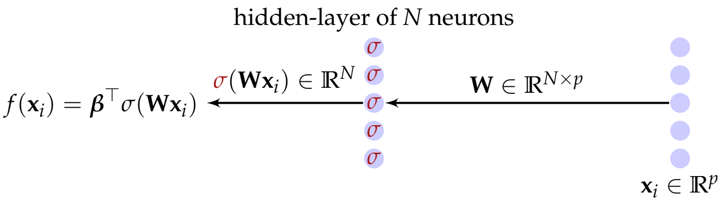
Outline

Two-layer network with random first layer



- for random (first-layer) weights $\mathbf{W} \in \mathbb{R}^{N \times p}$ having say i.i.d. standard Gaussian entries

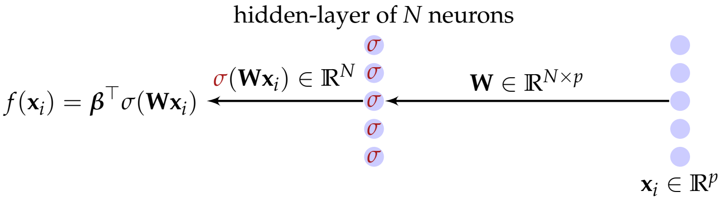
Two-layer network with random first layer



- for random (first-layer) weights $\mathbf{W} \in \mathbb{R}^{N \times p}$ having say i.i.d. standard Gaussian entries
- get second-layer $\boldsymbol{\beta}$ by minimizing Cost $= \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}_i))^2 + \gamma \|\boldsymbol{\beta}\|^2$ for some regularization parameter $\gamma > 0$, then

$$\boldsymbol{\beta} \equiv \frac{1}{n}\boldsymbol{\Sigma}\left(\frac{1}{n}\boldsymbol{\Sigma}^\top \boldsymbol{\Sigma} + \gamma \mathbf{I}_n\right)^{-1} \mathbf{y}, \quad (11)$$

Two-layer network with random first layer

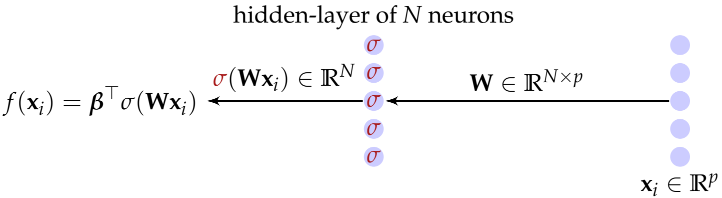


- for random (first-layer) weights $\mathbf{W} \in \mathbb{R}^{N\times p}$ having say i.i.d. standard Gaussian entries
- get second-layer $\boldsymbol{\beta}$ by minimizing Cost $= \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}_i))^2 + \gamma \|\boldsymbol{\beta}\|^2$ for some regularization parameter $\gamma > 0$, then

$$\boldsymbol{\beta} \equiv \frac{1}{n}\boldsymbol{\Sigma}\left(\frac{1}{n}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma} + \gamma \mathbf{I}_n\right)^{-1}\mathbf{y}, \tag{11}$$

- training MSE (on the given training set $(\mathbf{X}, \mathbf{y})$) reads

$$E_{\text{train}} = \frac{1}{n}\|\mathbf{y} - \boldsymbol{\Sigma}^\top\boldsymbol{\beta}\|_F^2 = \frac{\gamma^2}{n}\mathbf{y}\mathbf{Q}^2(\gamma)\mathbf{y}, \quad \boxed{\mathbf{Q}(\gamma) \equiv \left(\frac{1}{n}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma} + \gamma\mathbf{I}_n\right)^{-1}} \tag{12}$$



- for random (first-layer) weights $\mathbf{W} \in \mathbb{R}^{N \times p}$ having say i.i.d. standard Gaussian entries
- get second-layer $\boldsymbol{\beta}$ by minimizing Cost $= \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x}_i))^2 + \gamma \|\boldsymbol{\beta}\|^2$ for some regularization parameter $\gamma > 0$, then

$$\boldsymbol{\beta} \equiv \frac{1}{n}\boldsymbol{\Sigma} \left(\frac{1}{n}\boldsymbol{\Sigma}^\top \boldsymbol{\Sigma} + \gamma \mathbf{I}_n \right)^{-1} \mathbf{y}, \tag{11}$$

- training MSE (on the given training set $(\mathbf{X}, \mathbf{y})$) reads

$$E_{\rm train} = \frac{1}{n}\|\mathbf{y} - \boldsymbol{\Sigma}^\top \boldsymbol{\beta}\|_F^2 = \frac{\gamma^2}{n}\mathbf{y}\mathbf{Q}^2(\gamma)\mathbf{y}, \quad \boxed{\mathbf{Q}(\gamma) \equiv \left(\frac{1}{n}\boldsymbol{\Sigma}^\top \boldsymbol{\Sigma} + \gamma \mathbf{I}_n \right)^{-1}} \tag{12}$$

- Similarly, the test MSE on a test set $(\hat{\mathbf{X}}, \hat{\mathbf{y}}) \in \mathbb{R}^{p \times \hat{n}} \times \mathbb{R}^{d \times \hat{n}}$ of size $\hat{n}$: $E_{\rm test} = \frac{1}{\hat{n}}\|\hat{\mathbf{y}} - \hat{\boldsymbol{\Sigma}}^\top \boldsymbol{\beta}\|_F^2, \quad \hat{\boldsymbol{\Sigma}} = \sigma(\mathbf{W}\hat{\mathbf{X}})$.

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\mathbf{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\top$ still has i.i.d. columns, but

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\mathbf{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\mathsf{T}$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\mathsf{T}\mathbf{W}^\mathsf{T}]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\mathbf{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\mathsf{T}$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\mathsf{T}\mathbf{W}^\mathsf{T}]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries
- **trace lemma** does not apply

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\boldsymbol{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\mathsf{T}$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\mathsf{T}\mathbf{W}^\mathsf{T}]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries
- **trace lemma** does not apply

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\boldsymbol{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\top$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\top \mathbf{W}^\top]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries
- **trace lemma** does not apply

Lemma (Concentration of nonlinear quadratic form, [LLC18, Lemma 1])

For $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$, 1-*Lipschitz* $\sigma(\cdot)$, *and* $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{X} \in \mathbb{R}^{p \times n}$ *such that* $\|\mathbf{A}\|, \|\mathbf{X}\|$ *bounded, then*

$$\mathbb{P}\left(\left| \frac{1}{n} \sigma(\mathbf{w}^\top \mathbf{X}) \mathbf{A} \sigma(\mathbf{X}^\top \mathbf{w}) - \frac{1}{n} \operatorname{tr} \mathbf{A}\mathbf{K} \right| > t \right) \leq C e^{-cn \min(t, t^2)}$$

for some $C, c > 0$, $p/n \in (0, \infty)$ *with* $\mathbf{K} \equiv \mathbf{K}_{\mathbf{X}\mathbf{X}} \equiv \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)}[\sigma(\mathbf{X}^\top \mathbf{w}) \sigma(\mathbf{w}^\top \mathbf{X})] \in \mathbb{R}^{n \times n}$.

- $\mathbf{K}$ is in fact the **conjugate kernel** (**CK**) matrix

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\boldsymbol{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\mathsf{T}$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\mathsf{T}\mathbf{W}^\mathsf{T}]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries
- **trace lemma** does not apply

Lemma (Concentration of nonlinear quadratic form, [LLC18, Lemma 1])

For $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$, 1-*Lipschitz* $\sigma(\cdot)$, *and* $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{X} \in \mathbb{R}^{p \times n}$ *such that* $\|\mathbf{A}\|, \|\mathbf{X}\|$ *bounded, then*

$$\mathbb{P}\left(\left| \frac{1}{n} \sigma(\mathbf{w}^\top \mathbf{X}) \mathbf{A} \sigma(\mathbf{X}^\top \mathbf{w}) - \frac{1}{n} \operatorname{tr} \mathbf{A}\mathbf{K} \right| > t \right) \leq C e^{-cn \min(t, t^2)}$$

for some $C, c > 0$, $p/n \in (0, \infty)$ *with* $\mathbf{K} \equiv \mathbf{K}_{\mathbf{X}\mathbf{X}} \equiv \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)}[\sigma(\mathbf{X}^\top \mathbf{w}) \sigma(\mathbf{w}^\top \mathbf{X})] \in \mathbb{R}^{n \times n}$.

- $\mathbf{K}$ is in fact the **conjugate kernel** (**CK**) matrix
- for well-behaved (e.g., Lipschitz) non-linearity, trace lemma holds in this nonlinear case

Nonlinear resolvent

$$\mathbf{Q}(\gamma) = \left(\frac{1}{n} \sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X}) + \gamma \mathbf{I}_n \right)^{-1} \tag{13}$$

- nonlinear $\mathbf{\Sigma}^\top = \sigma(\mathbf{W}\mathbf{X})^\mathsf{T}$ still has i.i.d. columns, but
- its i-th column $\sigma([\mathbf{X}^\mathsf{T}\mathbf{W}^\mathsf{T}]_{\cdot i})$ no longer has i.i.d. or linearly dependent entries
- **trace lemma** does not apply

Lemma (Concentration of nonlinear quadratic form, [LLC18, Lemma 1])

For $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$, 1-*Lipschitz* $\sigma(\cdot)$, *and* $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{X} \in \mathbb{R}^{p \times n}$ *such that* $\|\mathbf{A}\|, \|\mathbf{X}\|$ *bounded, then*

$$\mathbb{P}\left(\left| \frac{1}{n} \sigma(\mathbf{w}^\top \mathbf{X}) \mathbf{A} \sigma(\mathbf{X}^\top \mathbf{w}) - \frac{1}{n} \operatorname{tr} \mathbf{A}\mathbf{K} \right| > t \right) \leq C e^{-cn \min(t, t^2)}$$

for some $C, c > 0$, $p/n \in (0, \infty)$ *with* $\mathbf{K} \equiv \mathbf{K}_{\mathbf{X}\mathbf{X}} \equiv \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)}[\sigma(\mathbf{X}^\top \mathbf{w}) \sigma(\mathbf{w}^\top \mathbf{X})] \in \mathbb{R}^{n \times n}$.

- $\mathbf{K}$ is in fact the **conjugate kernel** (**CK**) matrix
- for well-behaved (e.g., Lipschitz) non-linearity, trace lemma holds in this nonlinear case
- get deterministic equivalent for $\mathbf{Q}$, establish (limiting) eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X})$, etc.

Theorem (Resolvent for nonlinear Gram matrix, [LLC18])

Let $\mathbf{W} \in \mathbb{R}^{N \times p}$ *be a random matrix with i.i.d. standard Gaussian entries,* $\sigma(\cdot)$ *be* 1*-Lipschitz, and* $\mathbf{X} \in \mathbb{R}^{p \times n}$ *be of bounded operator norm. Then, as* $n, p, N \to \infty$ *at the same pace, for* $\mathbf{Q} = (\sigma(\mathbf{X}^\top \mathbf{W}^\mathsf{T})\sigma(\mathbf{W}\mathbf{X})/n + \gamma \mathbf{I}_n)^{-1}$ *with* $\gamma > 0$,

$$\|\mathbb{E}[\mathbf{Q}] - \bar{\mathbf{Q}}\| \to 0, \quad \bar{\mathbf{Q}} \equiv \left(\frac{N}{n} \frac{\mathbf{K}}{1+\delta} + \gamma \mathbf{I}_n \right)^{-1}$$

for δ *the unique positive solution to* $\delta = \frac{1}{n} \operatorname{tr} \bar{\mathbf{Q}}\mathbf{K}$ *and* $\mathbf{K} = \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)}[\sigma(\mathbf{X}^\top \mathbf{w})\sigma(\mathbf{w}^\top \mathbf{X})] \in \mathbb{R}^{n \times n}$.

[5] Cosme Louart, Zhenyu Liao, and Romain Couillet. "A Random Matrix Approach to Neural Networks". In: *The Annals of Applied Probability* 28.2 (2018), pp. 1190–1248

Theorem (Resolvent for nonlinear Gram matrix, [LLC18])

Let $\mathbf{W} \in \mathbb{R}^{N \times p}$ *be a random matrix with i.i.d. standard Gaussian entries,* $\sigma(\cdot)$ *be* 1*-Lipschitz, and* $\mathbf{X} \in \mathbb{R}^{p \times n}$ *be of bounded operator norm. Then, as* $n, p, N \to \infty$ *at the same pace, for* $\mathbf{Q} = (\sigma(\mathbf{X}^\top \mathbf{W}^\mathsf{T})\sigma(\mathbf{W}\mathbf{X})/n + \gamma \mathbf{I}_n)^{-1}$ *with* $\gamma > 0$*,*

$$\|\mathbb{E}[\mathbf{Q}] - \bar{\mathbf{Q}}\| \to 0, \quad \bar{\mathbf{Q}} \equiv \left(\frac{N}{n} \frac{\mathbf{K}}{1+\delta} + \gamma \mathbf{I}_n \right)^{-1}$$

for δ *the unique positive solution to* $\delta = \frac{1}{n} \operatorname{tr} \bar{\mathbf{Q}} \mathbf{K}$ *and* $\mathbf{K} = \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)}[\sigma(\mathbf{X}^\top \mathbf{w})\sigma(\mathbf{w}^\top \mathbf{X})] \in \mathbb{R}^{n \times n}$*.*

Corollary (Asymptotic training and test MSEs)

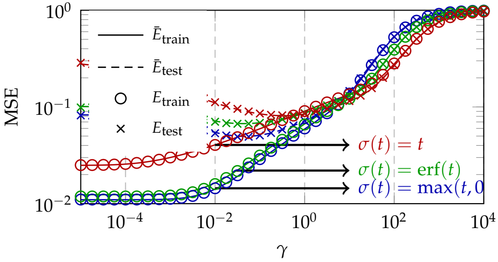
Under the setting and notations of Theorem 2, for bounded $\mathbf{X}, \hat{\mathbf{X}}, \mathbf{y}, \hat{\mathbf{y}}$*, then the training and test MSES, satisfy, as* $n, p, N \to \infty$*, we have* $E_{\text{train}} - \bar{E}_{\text{train}} \to 0$ *and* $E_{\text{test}} - \bar{E}_{\text{test}} \to 0$ *with*

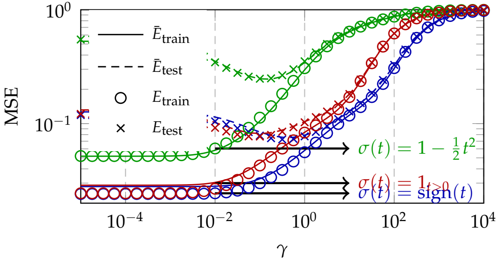
$$\bar{E}_{\text{train}} = \frac{\gamma^2}{n} \mathbf{y}^\top \bar{\mathbf{Q}} \left(\frac{\frac{1}{N} \operatorname{tr} \bar{\mathbf{Q}} \bar{\mathbf{K}} \bar{\mathbf{Q}}}{1 - \frac{1}{N} \operatorname{tr} \bar{\mathbf{K}} \bar{\mathbf{Q}} \bar{\mathbf{K}} \bar{\mathbf{Q}}} \bar{\mathbf{K}} + \mathbf{I}_n \right) \bar{\mathbf{Q}} \mathbf{y}$$

$$\bar{E}_{\text{test}} = \frac{1}{\hat{n}} \|\hat{\mathbf{y}} - \bar{\mathbf{K}}_{\mathbf{X}\hat{\mathbf{X}}}^\top \bar{\mathbf{Q}} \mathbf{y}\|_F^2 + \frac{\frac{1}{N} \mathbf{y}^\top \bar{\mathbf{Q}} \bar{\mathbf{K}} \bar{\mathbf{Q}} \mathbf{y}}{1 - \frac{1}{N} \operatorname{tr} \bar{\mathbf{K}} \bar{\mathbf{Q}} \bar{\mathbf{K}} \bar{\mathbf{Q}}} \left(\frac{1}{\hat{n}} \operatorname{tr} \bar{\mathbf{K}}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} - \frac{1}{\hat{n}} \operatorname{tr} (\mathbf{I}_n + \gamma \bar{\mathbf{Q}})(\bar{\mathbf{K}}_{\mathbf{X}\hat{\mathbf{X}}} \bar{\mathbf{K}}_{\mathbf{X}\hat{\mathbf{X}}}^\top \bar{\mathbf{Q}}) \right)$$

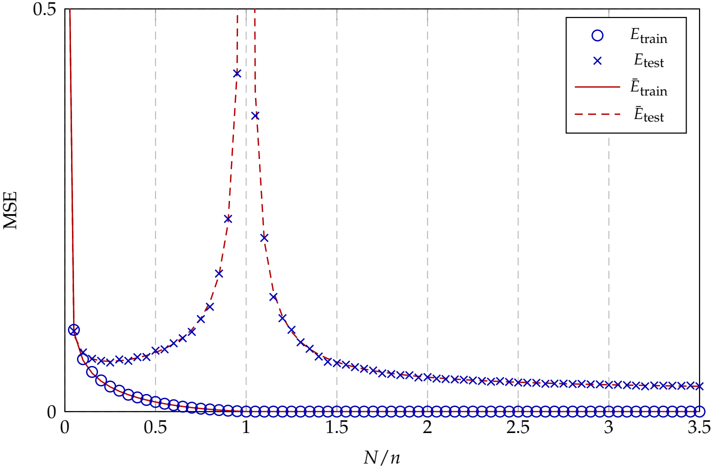
[5] Cosme Louart, Zhenyu Liao, and Romain Couillet. "A Random Matrix Approach to Neural Networks". In: *The Annals of Applied Probability* 28.2 (2018), pp. 1190–1248

Numerical results





Numerical results: double descent



Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top \sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top \sigma(\mathbf{WX})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top \sigma(\mathbf{W}\mathbf{X})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: 2201.04753 [cs, math]

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for $\mathbf{W},\mathbf{X}$ having sub-gaussian entries
  1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
  2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.
- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for sub-gaussian $\mathbf{W},\mathbf{X}$ and centered $\sigma(\cdot)$

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for sub-gaussian $\mathbf{W}, \mathbf{X}$ and centered $\sigma(\cdot)$
- despite being a **white model**, spikes may appear!

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$ for sub-gaussian $\mathbf{W}, \mathbf{X}$ and centered $\sigma(\cdot)$
- despite being a **white model**, spikes may appear!

1. if $\mathbb{E}[\xi^2\sigma(\xi)] = 0$, then no spike

---

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: 2201.04753 [cs, math]

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{W}\mathbf{X})^\top\sigma(\mathbf{W}\mathbf{X})$ for sub-gaussian $\mathbf{W}, \mathbf{X}$ and centered $\sigma(\cdot)$
- despite being a **white model**, spikes may appear!

1. if $\mathbb{E}[\xi^2\sigma(\xi)] = 0$, then no spike
2. otherwise, at most two spikes

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Some further RMT investigations on the two-layer model

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for sub-gaussian $\mathbf{W}, \mathbf{X}$ and centered $\sigma(\cdot)$
- despite being a **white model**, spikes may appear!

1. if $\mathbb{E}[\xi^2\sigma(\xi)] = 0$, then no spike
2. otherwise, at most two spikes

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: `2201.04753 [cs, math]`

Eigenspectra of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$:

- [PW17] first guess expression of the eigenvalue behavior
- [BP21]: eigenvalue distribution of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for $\mathbf{W}, \mathbf{X}$ having sub-gaussian entries

1. for "centered" $\sigma(\cdot)$ with respect to Gaussian measure: $\mathbb{E}[\sigma(\xi)] = 0$ for $\xi \sim \mathcal{N}(0,1)$
2. take a rather explicit form (3rd order poly ST equation) and depends on σ only via $\mathbb{E}[\sigma^2(\xi)]$ and $\mathbb{E}[\sigma(\xi)\xi]$.

- [BP22]: behavior of largest eigenvalue of $\frac{1}{n}\sigma(\mathbf{WX})^\top\sigma(\mathbf{WX})$ for sub-gaussian $\mathbf{W}, \mathbf{X}$ and centered $\sigma(\cdot)$
- despite being a **white model**, spikes may appear!

1. if $\mathbb{E}[\xi^2\sigma(\xi)] = 0$, then no spike
2. otherwise, at most two spikes

Question: what happen if either $\mathbf{W}$ or $\mathbf{X}$ has some structure? Any different phase transition behavior?

[6] Jeffrey Pennington and Pratik Worah. "Nonlinear random matrix theory for deep learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2634–2643

[7] Lucas Benigni and Sandrine Péché. "Eigenvalue Distribution of Some Nonlinear Models of Random Matrices". In: *Electronic Journal of Probability* 26.none (Jan. 2021), pp. 1–37

[8] Lucas Benigni and Sandrine Péché. *Largest Eigenvalues of the Conjugate Kernel of Single-Layered Neural Networks*. Jan. 2022. arXiv: 2201.04753 [cs, math]

Some further RMT investigations on random DNNs

- design of DNN to achieve dynamical isometry, **accelerate** training at the **beginning** stage of training

- Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. "Resurrecting the Sigmoid in Deep Learning through Dynamical Isometry: Theory and Practice". In: *Advances in Neural Information Processing Systems*. Vol. 30. NIPS'17. Curran Associates, Inc., 2017, pp. 4785–4795
- Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. "Dynamical Isometry and a Mean Field Theory of RNNs: Gating Enables Signal Propagation in Recurrent Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 873–882
- Lechao Xiao et al. "Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 5393–5402
- Dar Gilboa et al. "Dynamical Isometry and a Mean Field Theory of LSTMs and GRUs". In: *arXiv* (2019). eprint: `1901.08987`

Some further RMT investigations on random DNNs

- design of DNN to achieve dynamical isometry, **accelerate** training at the **beginning** stage of training
  - Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. "Resurrecting the Sigmoid in Deep Learning through Dynamical Isometry: Theory and Practice". In: *Advances in Neural Information Processing Systems*. Vol. 30. NIPS'17. Curran Associates, Inc., 2017, pp. 4785–4795
  - Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. "Dynamical Isometry and a Mean Field Theory of RNNs: Gating Enables Signal Propagation in Recurrent Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 873–882
  - Lechao Xiao et al. "Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 5393–5402
  - Dar Gilboa et al. "Dynamical Isometry and a Mean Field Theory of LSTMs and GRUs". In: *arXiv* (2019). eprint: `1901.08987`

- understand how weight distribution **interact** with activation in DNNs
  - Leonid Pastur. "On Random Matrices Arising in Deep Neural Networks. Gaussian Case". In: *arXiv* (2020). eprint: `2001.06188`
  - Leonid Pastur and Victor Slavin. "On Random Matrices Arising in Deep Neural Networks: General I.I.D. Case". In: *Random Matrices: Theory and Applications* 12.01 (Jan. 2023), p. 2250046
  - Leonid Pastur. "Eigenvalue Distribution of Large Random Matrices Arising in Deep Neural Networks: Orthogonal Case". In: *Journal of Mathematical Physics* 63.6 (2022), p. 063505
  - Zhou Fan and Zhichao Wang. "Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7710–7721

Outline

Gradient descent dynamics on linear regression model

- **gradient descent dynamics (GDDs)** of ridge regression learning (i.e., of a single-layer linear network)

Gradient descent dynamics on linear regression model

- **gradient descent dynamics (GDDs)** of ridge regression learning (i.e., of a single-layer linear network)
- given training data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ with associated labels/targets $\mathbf{y} = [y_1, \ldots, y_n] \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^p$ is learned via gradient descent by minimizing the (ridge-regularized) squared loss

$$L(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}^\top \mathbf{w}\|^2 + \frac{\gamma}{2} \|\mathbf{w}\|^2 \tag{14}$$

for some regularization penalty $\gamma \geq 0$.

Gradient descent dynamics on linear regression model

- **gradient descent dynamics (GDDs)** of ridge regression learning (i.e., of a single-layer linear network)
- given training data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ with associated labels/targets $\mathbf{y} = [y_1, \ldots, y_n] \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^p$ is learned via gradient descent by minimizing the (ridge-regularized) squared loss

$$L(\mathbf{w}) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}^\top \mathbf{w}\|^2 + \frac{\gamma}{2} \|\mathbf{w}\|^2 \tag{14}$$

  for some regularization penalty $\gamma \geq 0$.
- gradient given by $\nabla L(\mathbf{w}) = -\frac{1}{n}\mathbf{X}(\mathbf{y} - \mathbf{X}^\top \mathbf{w}) + \gamma \mathbf{w}$ so that, for small gradient descent steps (or learning rate) α, continuous-time approximation (in fact, **gradient flow**) of the time evolution $\mathbf{w}(t)$ of $\mathbf{w}$:

$$\frac{\partial \mathbf{w}(t)}{\partial t} = -\alpha \nabla L(\mathbf{w}) = \frac{\alpha}{n}\mathbf{X}\mathbf{y} - \alpha \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top + \gamma \mathbf{I}_p \right) \mathbf{w}$$

  solution explicitly given by

$$\boxed{\mathbf{w}(t) = e^{-\alpha t \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top + \gamma \mathbf{I}_p \right)} \mathbf{w}_0 + \left(\mathbf{I}_p - e^{-\alpha t \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top + \gamma \mathbf{I}_p \right)} \right) \mathbf{w}_\infty} \tag{15}$$

  with $\mathbf{w}_0 = \mathbf{w}(t=0)$ (the initialization of gradient descent) and

$$\mathbf{w}_\infty = \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top + \gamma \mathbf{I}_p \right)^{-1} \frac{1}{n}\mathbf{X}\mathbf{y} \tag{16}$$

  the ridge regression solution with regularization parameter γ.

Some RMT results on GDD in classification

- to study statistical evolution of $\mathbf{w}(t)$, consider binary Gaussian mixture model for input data

$$\mathcal{C}_1 : \mathbf{x}_i \sim \mathcal{N}(-\boldsymbol{\mu}, \mathbf{I}_p) \quad \mathcal{C}_2 : \mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_p)$$

  with associated labels $y_i = -1$ and $y_i = 1$, respectively.

Some RMT results on GDD in classification

- to study statistical evolution of $\mathbf{w}(t)$, consider binary Gaussian mixture model for input data

$$\mathcal{C}_1 : \mathbf{x}_i \sim \mathcal{N}(-\boldsymbol{\mu}, \mathbf{I}_p) \quad \mathcal{C}_2 : \mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_p)$$

  with associated labels $y_i = -1$ and $y_i = 1$, respectively.
- study training and test **misclassification error rates** as

$$\mathbb{P}(\mathbf{x}_i^\top \mathbf{w}(t) > 0 \mid y_i = -1), \quad \text{and} \quad \mathbb{P}(\hat{\mathbf{x}}^\top \mathbf{w}(t) > 0 \mid \hat{y} = -1),$$

  for $\hat{\mathbf{x}} \sim \mathcal{N}(-\boldsymbol{\mu}, \mathbf{I}_p)$ a new test datum (independent of the training set $(\mathbf{X}, \mathbf{y})$) of genuine label $\hat{y} = -1$.

- to study statistical evolution of $\mathbf{w}(t)$, consider binary Gaussian mixture model for input data

$$\mathcal{C}_1 : \mathbf{x}_i \sim \mathcal{N}(-\boldsymbol{\mu}, \mathbf{I}_p) \quad \mathcal{C}_2 : \mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_p)$$

  with associated labels $y_i = -1$ and $y_i = 1$, respectively.
- study training and test **misclassification error rates** as

$$\mathbb{P}(\mathbf{x}_i^\top \mathbf{w}(t) > 0 \mid y_i = -1), \quad \text{and} \quad \mathbb{P}(\hat{\mathbf{x}}^\top \mathbf{w}(t) > 0 \mid \hat{y} = -1),$$

  for $\hat{\mathbf{x}} \sim \mathcal{N}(-\boldsymbol{\mu}, \mathbf{I}_p)$ a new test datum (independent of the training set $(\mathbf{X}, \mathbf{y})$) of genuine label $\hat{y} = -1$.
- we can of course consider different statistical **model** and/or different **task** (e.g., regression)

Theorem (Training and test performance of GDD, [LC18])

For a random initialization $\mathbf{w}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_p/p)$ *independent of* $\mathbf{X}$, $\mathbf{x}$ *a column of* $\mathbf{X}$ *of mean* $\boldsymbol{\mu}$ *and* $\hat{\mathbf{x}}$ *an independent copy of* $\mathbf{x}$, *as* $n, p \to \infty$ *with* $p/n \to c \in (0, \infty)$, *we have*

$$\mathbb{P}(\hat{\mathbf{x}}^\top \mathbf{w}(t) > 0 \mid \hat{y} = -1) - Q\left(\frac{E_{\rm test}}{\sqrt{V_{\rm test}}}\right) \to 0, \quad \mathbb{P}(\mathbf{x}^\top \mathbf{w}(t) > 0 \mid y = -1) - Q\left(\frac{E_{\rm train}}{\sqrt{V_{\rm train}}}\right) \to 0,$$

almost surely, where

$$E_{\rm test} = -\frac{1}{2\pi\imath} \oint_\Gamma \frac{1 - f_t(z)}{z} \frac{\rho m(z)\, dz}{(\rho + c)\, m(z) + 1}, \quad V_{\rm test} = \frac{1}{2\pi\imath} \oint_\Gamma \left[\frac{\frac{1}{z^2} (1 - f_t(z))^2}{(\rho + c)\, m(z) + 1} - \sigma^2 f_t^2(z) m(z) \right] dz$$

$$E_{\rm train} = -\frac{1}{2\pi\imath} \oint_\Gamma \frac{1 - f_t(z)}{z} \frac{dz}{(\rho + c)\, m(z) + 1}, \quad V_{\rm train} = \frac{1}{2\pi\imath} \oint_\Gamma \left[\frac{\frac{1}{z} (1 - f_t(z))^2}{(\rho + c)\, m(z) + 1} - \sigma^2 f_t^2(z) z m(z) \right] dz - E_{\rm train}^2$$

with $\rho = \lim_{p\to\infty} \|\boldsymbol{\mu}\|^2$, Γ *a positive contour surrounding the support of the Marčenko–Pastur law (shifted by* $\gamma \geq 0$*) and the points* $(\gamma, 0)$ *and* $(\gamma + \lambda_s, 0)$ *with* $\lambda_s = c + 1 + \rho + c/\rho$, $f_t(z) \equiv \exp(-\alpha t z)$ *and* $m(z)$ *unique ST solution to* $c(z-\gamma)m^2(z) - (1 - c - z + \gamma)m(z) + 1 = 0$.

Some further simplifications

- choose the contour Γ as, e.g., rectangle circling around both **main bulk** and **isolated eigenvalue** (if any)

[9] Zhenyu Liao and Romain Couillet. "The Dynamics of Learning: A Random Matrix Approach". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, 2018, pp. 3072–3081

- choose the contour Γ as, e.g., rectangle circling around both **main bulk** and **isolated eigenvalue** (if any)

This leads to

$$E_{\rm test} = \int \frac{1-f_t(x+\gamma)}{x+\gamma}\omega(dx) \quad V_{\rm test} = \frac{\rho+c}{\rho}\int \frac{(1-f_t(x+\gamma))^2\omega(dx)}{(x+\gamma)^2} + \sigma^2 \int f_t^2(x+\gamma)\mu(dx)$$

$$E_{\rm train} = \frac{\rho+c}{\rho}\int \frac{1-f_t(x+\gamma)}{x+\gamma}\omega(dx), \quad V_{\rm train} = \frac{\rho+c}{\rho}\int \frac{x(1-f_t(x+\gamma))^2\omega(dx)}{(x+\gamma)^2} + \sigma^2 \int x f_t^2(x+\gamma)\mu(dx) - E_{\rm train}^2$$

where we recall $\rho = \lim \|\boldsymbol{\mu}\|^2$, $f_t(x) = \exp(-\alpha t x)$, $\mu(x)$ the MP law

$$\mu(dx) = \frac{\sqrt{(x-\lambda_-)^+(\lambda_+-x)^+}}{2\pi c x}dx + (1-c^{-1})^+\delta(x), \tag{17}$$

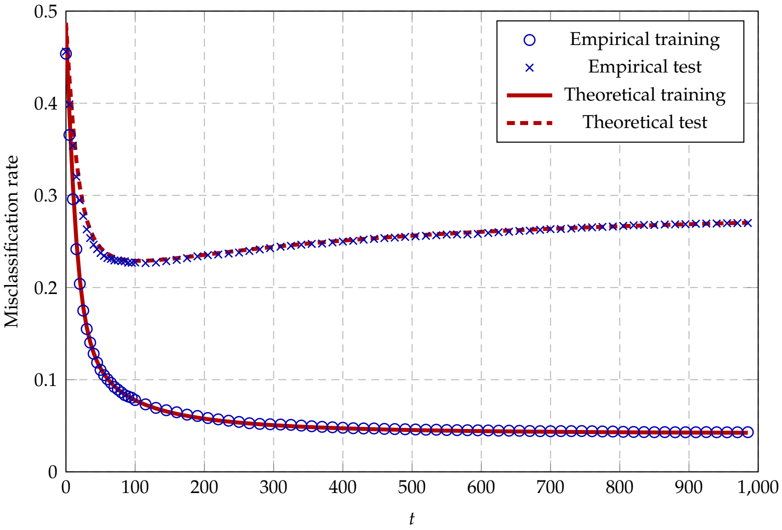
and

$$\omega(dx) \equiv \frac{\sqrt{(x-\lambda_-)^+(\lambda_+-x)^+}}{2\pi(\lambda_s - x)}dx + \frac{(\rho^2-c)^+}{\rho}\delta_{\lambda_s}(x) \tag{18}$$

for $\lambda_s = c + 1 + \rho + c/\rho$ the (possible) spike location.

[9] Zhenyu Liao and Romain Couillet. "The Dynamics of Learning: A Random Matrix Approach". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, 2018, pp. 3072–3081

Numerical results



Some further RMT efforts on high-dimensional dynamics

From the statistical physics community: reduces to low-dimensional ODE or SDE

- Sebastian Goldt et al. "Dynamics of Stochastic Gradient Descent for Two-Layer Neural Networks in the Teacher-Student Setup". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019
- Francesca Mignacco et al. "Dynamical Mean-Field Theory for Stochastic Gradient Descent in Gaussian Mixture Classification". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 9540–9550
- Rodrigo Veiga et al. "Phase Diagram of Stochastic Gradient Descent in High-Dimensional Two-Layer Neural Networks". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 23244–23255

Some further RMT efforts on high-dimensional dynamics

From the statistical physics community: reduces to low-dimensional ODE or SDE

- Sebastian Goldt et al. "Dynamics of Stochastic Gradient Descent for Two-Layer Neural Networks in the Teacher-Student Setup". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019
- Francesca Mignacco et al. "Dynamical Mean-Field Theory for Stochastic Gradient Descent in Gaussian Mixture Classification". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 9540–9550
- Rodrigo Veiga et al. "Phase Diagram of Stochastic Gradient Descent in High-Dimensional Two-Layer Neural Networks". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 23244–23255

From the optimization community: how RMT results apply to characterize average-case behavior in optimization

- Courtney Paquette et al. "SGD in the Large: Average-case Analysis, Asymptotics, and Stepsize Criticality". In: *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, July 2021, pp. 3548–3626
- Courtney Paquette et al. "Halting Time Is Predictable for Large Models: A Universality Property and Average-Case Analysis". In: *Foundations of Computational Mathematics* 23.2 (Apr. 2023), pp. 597–673

Some further RMT efforts on high-dimensional dynamics

From the statistical physics community: reduces to low-dimensional ODE or SDE

- Sebastian Goldt et al. "Dynamics of Stochastic Gradient Descent for Two-Layer Neural Networks in the Teacher-Student Setup". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019
- Francesca Mignacco et al. "Dynamical Mean-Field Theory for Stochastic Gradient Descent in Gaussian Mixture Classification". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 9540–9550
- Rodrigo Veiga et al. "Phase Diagram of Stochastic Gradient Descent in High-Dimensional Two-Layer Neural Networks". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 23244–23255

From the optimization community: how RMT results apply to characterize average-case behavior in optimization

- Courtney Paquette et al. "SGD in the Large: Average-case Analysis, Asymptotics, and Stepsize Criticality". In: *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, July 2021, pp. 3548–3626
- Courtney Paquette et al. "Halting Time Is Predictable for Large Models: A Universality Property and Average-Case Analysis". In: *Foundations of Computational Mathematics* 23.2 (Apr. 2023), pp. 597–673

And from the RMT community as well

- Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. "Online Stochastic Gradient Descent on Non-Convex Losses from High-Dimensional Inference". In: *Journal of Machine Learning Research* 22.106 (2021), pp. 1–51
- Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. "High-Dimensional Limit Theorems for SGD: Effective Dynamics and Critical Scaling". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 25349–25362
- Gerard Ben Arous et al. *High-Dimensional SGD Aligns with Emerging Outlier Eigenspaces*. Oct. 2023. arXiv: `2310.03010 [cs, math, stat]`

One step gradient beyond random network

- extends to wide DNN model via NTK, see, e.g., Y. Du, Z. Ling, R. C. Qiu, Z. Liao, "High-dimensional Learning Dynamics of Deep Neural Nets in the Neural Tangent Regime", High-dimensional Learning Dynamics Workshop, The Fortieth International Conference on Machine Learning (ICML'2023), 2023

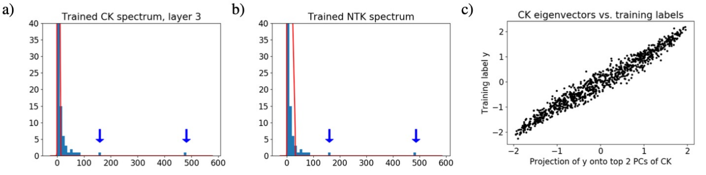


Figure 3: Eigenvalues of (a) K^{CK} and (b) K^{NTK} in a *trained* network, for training labels $y_\alpha = \sigma(\mathbf{x}_\alpha^\top \mathbf{v})$. The limit spectra at random initialization of weights are shown in red. Large outlier eigenvalues, indicated by blue arrows, emerge over training. (c) The projection of training labels onto the first 2 eigenvectors of the trained matrix K^{CK} accounts for 96% of the training label variance.

[10] Zhou Fan and Zhichao Wang. "Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7710–7721

One step gradient beyond random network

- extends to wide DNN model via NTK, see, e.g., Y. Du, Z. Ling, R. C. Qiu, Z. Liao, "High-dimensional Learning Dynamics of Deep Neural Nets in the Neural Tangent Regime", High-dimensional Learning Dynamics Workshop, The Fortieth International Conference on Machine Learning (ICML'2023), 2023
- however, limited in the NTK and linearized regime

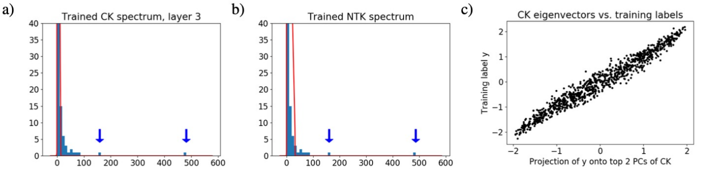


Figure 3: Eigenvalues of (a) K^{CK} and (b) K^{NTK} in a *trained* network, for training labels $y_\alpha = \sigma(\mathbf{x}_\alpha^\top \mathbf{v})$. The limit spectra at random initialization of weights are shown in red. Large outlier eigenvalues, indicated by blue arrows, emerge over training. (c) The projection of training labels onto the first 2 eigenvectors of the trained matrix K^{CK} accounts for 96% of the training label variance.

[10] Zhou Fan and Zhichao Wang. "Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7710–7721

One step gradient beyond random network

- extends to wide DNN model via NTK, see, e.g., Y. Du, Z. Ling, R. C. Qiu, Z. Liao, "High-dimensional Learning Dynamics of Deep Neural Nets in the Neural Tangent Regime", High-dimensional Learning Dynamics Workshop, The Fortieth International Conference on Machine Learning (ICML'2023), 2023
- however, limited in the NTK and linearized regime
- what about **nonlinear** **feature learning** during gradient descent (**different** from initialization)?

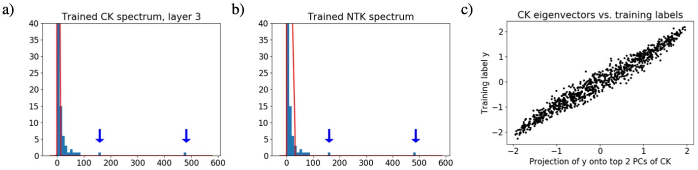


Figure 3: Eigenvalues of (a) K^{CK} and (b) K^{NTK} in a *trained* network, for training labels $y_\alpha = \sigma(\mathbf{x}_\alpha^\top \mathbf{v})$. The limit spectra at random initialization of weights are shown in red. Large outlier eigenvalues, indicated by blue arrows, emerge over training. (c) The projection of training labels onto the first 2 eigenvectors of the trained matrix K^{CK} accounts for 96% of the training label variance.

[10] Zhou Fan and Zhichao Wang. "Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7710–7721

One step gradient beyond random network

- extends to wide DNN model via NTK, see, e.g., Y. Du, Z. Ling, R. C. Qiu, Z. Liao, "High-dimensional Learning Dynamics of Deep Neural Nets in the Neural Tangent Regime", High-dimensional Learning Dynamics Workshop, The Fortieth International Conference on Machine Learning (ICML'2023), 2023
- however, limited in the NTK and linearized regime
- what about **nonlinear feature learning** during gradient descent (**different** from initialization)?
- **empirical observation**: spikes appear in the NTK spectra during gradient descent training [FW20]

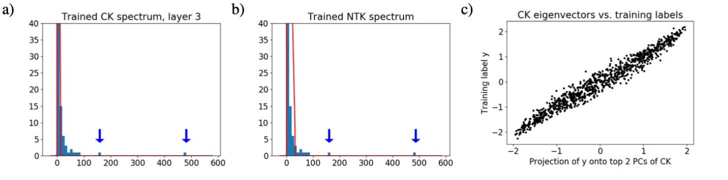
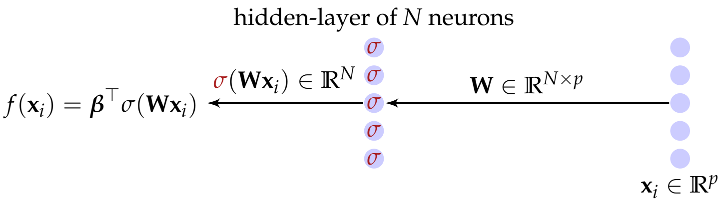


Figure 3: Eigenvalues of (a) $K^{\rm CK}$ and (b) $K^{\rm NTK}$ in a *trained* network, for training labels $y_\alpha = \sigma(\mathbf{x}_\alpha^\top \mathbf{v})$. The limit spectra at random initialization of weights are shown in red. Large outlier eigenvalues, indicated by blue arrows, emerge over training. (c) The projection of training labels onto the first 2 eigenvectors of the trained matrix $K^{\rm CK}$ accounts for 96% of the training label variance.

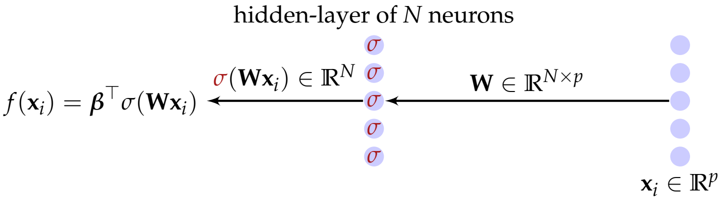
[10] Zhou Fan and Zhichao Wang. "Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 7710–7721

Two-layer random network after one step training



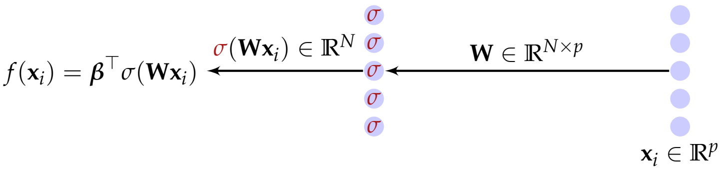
- two-layer NN having N neurons, with output $f(\mathbf{x}) = \frac{1}{\sqrt{N}} \boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x})$, for input $\mathbf{x} \in \mathbb{R}^p$, first-layer weight $\mathbf{W} \in \mathbb{R}^{N \times p}$, second-layer weight $\boldsymbol{\beta} \in \mathbb{R}^N$, and nonlinear σ

Two-layer random network after one step training



- two-layer NN having N neurons, with output $f(\mathbf{x}) = \frac{1}{\sqrt{N}}\boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x})$, for input $\mathbf{x} \in \mathbb{R}^p$, first-layer weight $\mathbf{W} \in \mathbb{R}^{N\times p}$, second-layer weight $\boldsymbol{\beta} \in \mathbb{R}^N$, and nonlinear σ
- model trained on $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of size n, by minimizing

$$\text{Cost} = \frac{1}{2n}\sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 . \tag{19}$$



- two-layer NN having N neurons, with output $f(\mathbf{x}) = \frac{1}{\sqrt{N}}\boldsymbol{\beta}^\top \sigma(\mathbf{W}\mathbf{x})$, for input $\mathbf{x} \in \mathbb{R}^p$, first-layer weight $\mathbf{W} \in \mathbb{R}^{N\times p}$, second-layer weight $\boldsymbol{\beta} \in \mathbb{R}^N$, and nonlinear σ
- model trained on $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of size n, by minimizing

$$\text{Cost} = \frac{1}{2n}\sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2. \tag{19}$$

- first-layer gradient explicitly given by

$$\frac{\partial \text{Cost}}{\partial \mathbf{W}} = -\frac{1}{n}\left(\left(\frac{1}{\sqrt{N}}\boldsymbol{\beta}\left(\mathbf{y}^\top - \frac{1}{\sqrt{N}}\boldsymbol{\beta}^\top\sigma(\mathbf{W}\mathbf{X})\right)\right)\odot\sigma'(\mathbf{W}\mathbf{X})\right)\mathbf{X}^\top \in \mathbb{R}^{N\times p}, \tag{20}$$

with $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$, and $\mathbf{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$.

Two-layer random network after one step GD training

- consider first step gradient update on $\mathbf{W}$ as $\mathbf{W}_1 = \mathbf{W}_0 + \sqrt{N}\eta_0 \mathbf{G}_0$, with $\mathbf{G}_0 = \frac{1}{n}\left(\left(\frac{1}{\sqrt{N}}\boldsymbol{\beta}_0\left(\mathbf{y}^\top - \frac{1}{\sqrt{N}}\boldsymbol{\beta}_0^\top \sigma(\mathbf{W}_0\mathbf{X})\right)\right) \odot \sigma'(\mathbf{W}_0\mathbf{X})\right)\mathbf{X}^\top$

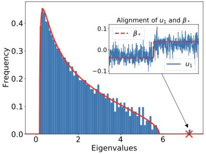


Figure 3: Main: empirical singular values of $\boldsymbol{W}_1$ (blue) vs. analytic prediction (red). Subfigure: overlap between $\boldsymbol{u}_1$ and the teacher vector $\boldsymbol{\beta}_* \propto [-\mathbf{1}_{d/2}; \mathbf{1}_{d/2}] \in \mathbb{R}^d$. We set $\sigma = \tanh$, $f^*(\boldsymbol{x}) = \mathrm{ReLU}(\langle \boldsymbol{x}, \boldsymbol{\beta}_* \rangle)$, $\eta = 2$, $\psi_1 = 4, \psi_2 = 2$, and $\sigma_\varepsilon = 0.2$.

[11] Jimmy Ba et al. "High-Dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation". In: *Advances in Neural Information Processing Systems*. May 2022

Two-layer random network after one step GD training

- consider first step gradient update on $\mathbf{W}$ as $\mathbf{W}_1 = \mathbf{W}_0 + \sqrt{N}\eta_0 \mathbf{G}_0$, with $\mathbf{G}_0 = \frac{1}{n}\left(\left(\frac{1}{\sqrt{N}}\boldsymbol{\beta}_0\left(\mathbf{y}^\top - \frac{1}{\sqrt{N}}\boldsymbol{\beta}_0^\top \sigma(\mathbf{W}_0\mathbf{X})\right)\right) \odot \sigma'(\mathbf{W}_0\mathbf{X})\right)\mathbf{X}^\top$
- **key observation** made in [Ba+22]: under standard assumption and for Gaussian $\mathbf{W}_0, \boldsymbol{\beta}_0$ and $\mathbf{X}$, the first step gradient $\mathbf{G}_0$ is approximately of **rank one**!

$$\left\| \mathbf{G}_0 - \frac{\mathbb{E}[\sigma'(\xi)]}{n\sqrt{N}} \boldsymbol{\beta}_0 \mathbf{y}^\top \mathbf{X}^\top \right\| \to 0. \tag{21}$$

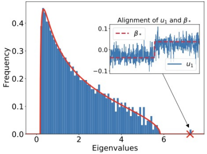


Figure 3: Main: empirical singular values of $\boldsymbol{W}_1$ (blue) vs. analytic prediction (red). Subfigure: overlap between $\boldsymbol{u}_1$ and the teacher vector $\boldsymbol{\beta}_* \propto [-\mathbf{1}_{d/2}; \mathbf{1}_{d/2}] \in \mathbb{R}^d$. We set $\sigma = \tanh$, $f^*(\boldsymbol{x}) = \mathrm{ReLU}(\langle \boldsymbol{x}, \boldsymbol{\beta}_* \rangle)$, $\eta = 2, \psi_1 = 4, \psi_2 = 2$, and $\sigma_\varepsilon = 0.2$.

[11] Jimmy Ba et al. "High-Dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation". In: *Advances in Neural Information Processing Systems*. May 2022

Two-layer random network after one step GD training

- consider first step gradient update on $\mathbf{W}$ as $\mathbf{W}_1 = \mathbf{W}_0 + \sqrt{N}\eta_0 \mathbf{G}_0$, with $\mathbf{G}_0 = \frac{1}{n}\left(\left(\frac{1}{\sqrt{N}}\boldsymbol{\beta}_0\left(\mathbf{y}^\top - \frac{1}{\sqrt{N}}\boldsymbol{\beta}_0^\top \sigma(\mathbf{W}_0\mathbf{X})\right)\right) \odot \sigma'(\mathbf{W}_0\mathbf{X})\right)\mathbf{X}^\top$
- **key observation** made in [Ba+22]: under standard assumption and for Gaussian $\mathbf{W}_0, \boldsymbol{\beta}_0$ and $\mathbf{X}$, the first step gradient $\mathbf{G}_0$ is approximately of **rank one**!

$$\left\| \mathbf{G}_0 - \frac{\mathbb{E}[\sigma'(\xi)]}{n\sqrt{N}} \boldsymbol{\beta}_0 \mathbf{y}^\top \mathbf{X}^\top \right\| \to 0. \tag{21}$$

- result obtained by (kind of conditioned on $\mathbf{X}, \mathbf{y}$ and $\boldsymbol{\beta}_0$) and playing with the randomness in $\mathbf{W}_0$

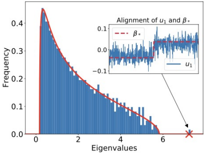


Figure 3: Main: empirical singular values of $\boldsymbol{W}_1$ (blue) vs. analytic prediction (red). Subfigure: overlap between $\boldsymbol{u}_1$ and the teacher vector $\boldsymbol{\beta}_* \propto [-\mathbf{1}_{d/2}; \mathbf{1}_{d/2}] \in \mathbb{R}^d$. We set $\sigma = \tanh$, $f^*(\boldsymbol{x}) = \mathrm{ReLU}(\langle \boldsymbol{x}, \boldsymbol{\beta}_* \rangle)$, $\eta = 2$, $\psi_1 = 4$, $\psi_2 = 2$, and $\sigma_\varepsilon = 0.2$.

[11] Jimmy Ba et al. "High-Dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation". In: *Advances in Neural Information Processing Systems*. May 2022

Two-layer random network after one step GD training

- consider first step gradient update on $\mathbf{W}$ as $\mathbf{W}_1 = \mathbf{W}_0 + \sqrt{N}\eta_0 \mathbf{G}_0$, with $\mathbf{G}_0 = \frac{1}{n}\left(\left(\frac{1}{\sqrt{N}}\boldsymbol{\beta}_0\left(\mathbf{y}^\top - \frac{1}{\sqrt{N}}\boldsymbol{\beta}_0^\top \sigma(\mathbf{W}_0\mathbf{X})\right)\right) \odot \sigma'(\mathbf{W}_0\mathbf{X})\right)\mathbf{X}^\top$
- **key observation** made in [Ba+22]: under standard assumption and for Gaussian $\mathbf{W}_0, \boldsymbol{\beta}_0$ and $\mathbf{X}$, the first step gradient $\mathbf{G}_0$ is approximately of **rank one**!

$$\left\| \mathbf{G}_0 - \frac{\mathbb{E}[\sigma'(\xi)]}{n\sqrt{N}}\boldsymbol{\beta}_0 \mathbf{y}^\top \mathbf{X}^\top \right\| \to 0. \tag{21}$$

- result obtained by (kind of conditioned on $\mathbf{X}, \mathbf{y}$ and $\boldsymbol{\beta}_0$) and playing with the randomness in $\mathbf{W}_0$
- built upon this, results on **generalization** can be obtained, etc.

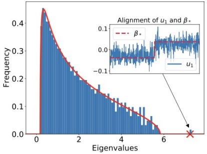


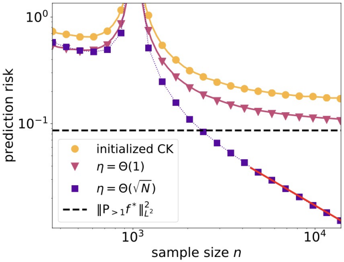
Figure 3: Main: empirical singular values of $\boldsymbol{W}_1$ (blue) vs. analytic prediction (red). Subfigure: overlap between $\boldsymbol{u}_1$ and the teacher vector $\boldsymbol{\beta}_* \propto [-\mathbf{1}_{d/2}; \mathbf{1}_{d/2}] \in \mathbb{R}^d$. We set $\sigma = \tanh$, $f^*(\boldsymbol{x}) = \mathrm{ReLU}(\langle \boldsymbol{x}, \boldsymbol{\beta}_* \rangle)$, $\eta = 2, \psi_1 = 4, \psi_2 = 2$, and $\sigma_\varepsilon = 0.2$.

[11] Jimmy Ba et al. "High-Dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation". In: *Advances in Neural Information Processing Systems*. May 2022

Discussion on the step size and its impact

- since $\|\mathbf{W}_0\| = O(1)$, $\|\mathbf{W}_0\|_F = \sqrt{N}$, and $\sqrt{N}\|\mathbf{G}_0\| = O(1)$, $\sqrt{N}\|\mathbf{G}_0\|_F = O(1)$, may consider:

1. small step $\eta = O(1)$ (same order in spectral norm): improve over initial CK, but not as good as optimal linear model
2. large step $\eta = O(\sqrt{N})$ (same order in Frobenius norm): improve over a class of **nonlinear** model, match **neural scaling law** in some cases



[12] Jimmy Ba et al. "High-Dimensional Asymptotics of Feature Learning: How One Gradient Step Improves the Representation". In: *Advances in Neural Information Processing Systems*. May 2022

Outline

Conclusion and take-away message

Take-away message:

- basics in ML and DL

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

1. from shallow to deep random NNs

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

1. from shallow to deep random NNs
2. from random to non-so-random NNs

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

1. from shallow to deep random NNs
2. from random to non-so-random NNs

- what is a good theory for DNN?

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

1. from shallow to deep random NNs
2. from random to non-so-random NNs

- what is a good theory for DNN?
- Model and data/task dependent, can be used to **guild DNN model design**.

Conclusion and take-away message

Take-away message:

- basics in ML and DL
- DL theory: **optimization+generalization**
- some not so fantastic story on neural tangent kernel and double descent
- opportunities in RMT for DL:

1. from shallow to deep random NNs
2. from random to non-so-random NNs

- what is a good theory for DNN?
- Model and data/task dependent, can be used to **guild DNN model design**.
- A recent (short) review focusing on RMT4DL: Zhenyu Liao and Michael W. Mahoney. *Random Matrix Theory for Deep Learning: Beyond Eigenvalues of Linear Models*. 2025. arXiv: 2201.04753 [cs, math]

RMT for machine learning: from theory to practice!

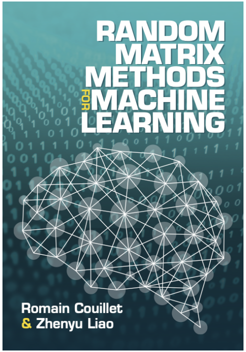
Random matrix theory (RMT) for machine learning:

- **change of intuition** from small to large dimensional learning paradigm!
- **better understanding** of existing methods: why they work if they do, and what the issue is if they do not
- **improved novel methods** with performance guarantee!

RMT for machine learning: from theory to practice!

Random matrix theory (RMT) for machine learning:

- **change of intuition** from small to large dimensional learning paradigm!
- **better understanding** of existing methods: why they work if they do, and what the issue is if they do not
- **improved novel methods** with performance guarantee!

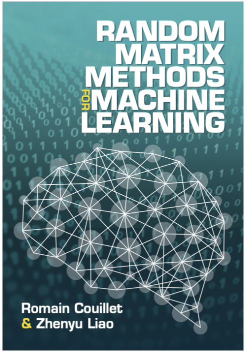


- book "*Random Matrix Methods for Machine Learning*"
- by Romain Couillet and **Zhenyu Liao**
- Cambridge University Press, 2022
- a pre-production version of the book and exercise solutions at `https://zhenyu-liao.github.io/book/`
- MATLAB and Python codes to reproduce all figures at `https://github.com/Zhenyu-LIAO/RMT4ML`

RMT for machine learning: from theory to practice!

Random matrix theory (RMT) for machine learning:

- **change of intuition** from small to large dimensional learning paradigm!
- **better understanding** of existing methods: why they work if they do, and what the issue is if they do not
- **improved novel methods** with performance guarantee!



- book "*Random Matrix Methods for Machine Learning*"
- by Romain Couillet and **Zhenyu Liao**
- Cambridge University Press, 2022
- a pre-production version of the book and exercise solutions at `https://zhenyu-liao.github.io/book/`
- MATLAB and Python codes to reproduce all figures at `https://github.com/Zhenyu-LIAO/RMT4ML`

Thank you! Q & A?