Neural Networks, Scaling Laws and Effective Field Theories

Zhengkang "Kevin" Zhang (University of Utah)

[ZZ, 2405.19398] + [Banta, Cai, Craig, ZZ, 2305.02334]

Trillion

1.1 Curve Fitting with at Least a ~~Million~~ Parameters

If at any point Machine Learning seems confusing, complicated, jargon-filled, etc, then just remember... it's really just curve fitting, or 'regression', with a very, very large number of parameters.

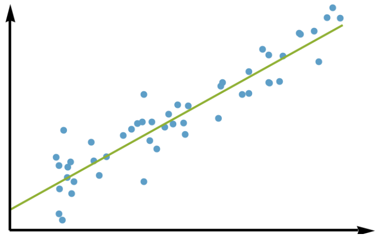
[J. Kaplan, "Notes on Contemporary Machine Learning for Physicists."]

Let's start simple...

Observe data: $\{x_\alpha, y_\alpha\}$ $(\alpha = 1, \ldots, T)$.

Linear regression:

$f_{\{\theta\}}(x) = \theta_0 + \theta_1 x$ (**2** parameters)

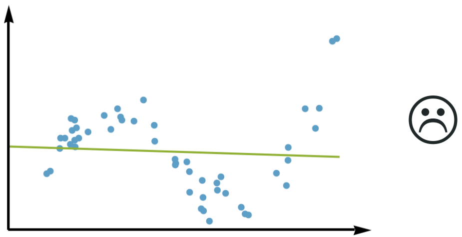


☺

Minimize loss $\mathcal{L} = \frac{1}{2}\sum_\alpha \left[f_{\{\theta\}}(x_\alpha) - y_\alpha\right]^2 \Rightarrow$ fit parameters $\{\theta\}$

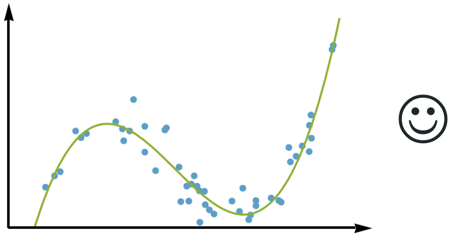
Linear regression:

$$f_{\{\theta\}}(x) = \theta_0 + \theta_1\, x \quad (\mathbf{2}\ \text{parameters})$$



Cubic regression:

$$f_{\{\theta\}}(x) = \theta_0 + \theta_1\, x + \theta_2\, x^2 + \theta_3\, x^3 \quad (\mathbf{4}\ \text{parameters})$$

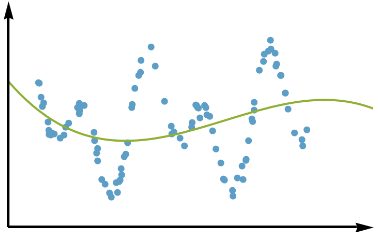


Linear regression:

$f_{\{\theta\}}(x) = \theta_0 + \theta_1 \, x$ (**2** parameters)

Cubic regression:

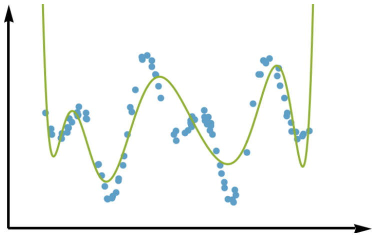
$f_{\{\theta\}}(x) = \theta_0 + \theta_1 \, x + \theta_2 \, x^2 + \theta_3 \, x^3$ (**4** parameters)



☹

10th degree polynomial regression?

$f_{\{\theta\}}(x) = \theta_0 + \theta_1 \, x + \cdots + \theta_{10} \, x^{10}$ (**11** parameters)



??

Generalize:

$f_{\{\theta\}}(x) = \sum_j \theta_j \, \varphi_j(x)$

This is known as a **linear model**: linear combination of feature functions.

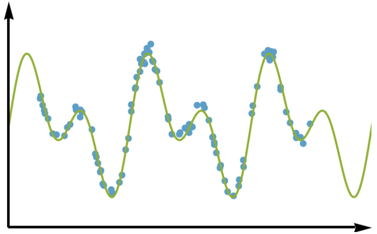
Adjust coefficients $\{\theta\}$ (model parameters) to fit data.

In the examples above, we picked: $\varphi_j(x) = x^j$.

Not always the best choice of feature functions.

Alternative: $\varphi_j(x) = \sin(j\pi x)$.

$f_{\{\theta\}}(x) = \theta_0 + \theta_1 \sin(\pi x) + \theta_2 \sin(2\pi x)$



☺

But how do we know which is better to use?

$\varphi_j(x) = x^j$ or $\varphi_j(x) = \sin(j\pi x)$ or something else?

Additional complication: "curse of dimensionality."

L_1 pixels

L_2 pixels

$$\vec{x} = (\,(r_1, g_1, b_1), (r_2, g_2, b_2), \ldots\,) \in \mathbb{R}^{3L_1L_2}$$

$$\Downarrow$$

$$f(\vec{x}) = \text{“cat”}$$

What are the useful features in such high-dimensional input space?

Beyond human's capability?

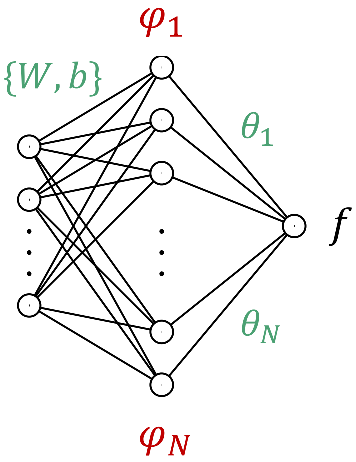
Machines can help!

Solution offered by modern AI/ML:

- ❑ Random features (a LOT of them).
- ❑ Deep neural networks capable of building useful features according to data (learning features from data).

Let's look at the simplest neural network...

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$



input → hidden layer → output

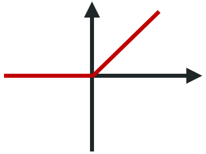
$$f_{\{\theta,W,b\}}(\vec{x}) = \sum_{j=1}^{N} \theta_j \, \varphi_j(\vec{x})$$

$$\varphi_j(\vec{x}) = \sigma\left(\sum_{k=1}^{d} W_{jk} \, x_k + b_j\right)$$

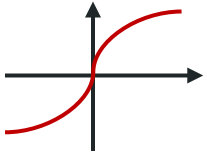
weights biases

activation function

e.g. ReLU



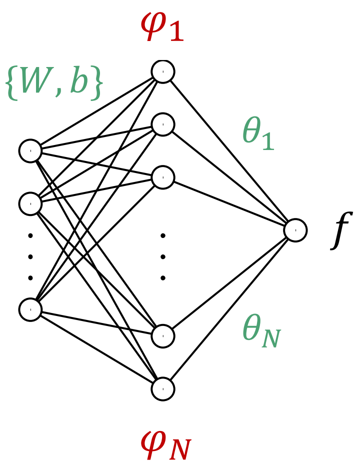
tanh



linear transformation → nonlinear activation function → linear transformation

Let's look at the simplest neural network...

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$



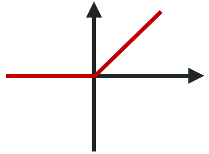
input → hidden layer → output

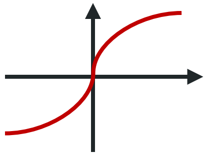
$$f_{\{\theta,W,b\}}(\vec{x}) = \sum_{j=1}^{N} \theta_j \, \varphi_j(\vec{x})$$

$$\varphi_j(\vec{x}) = \sigma\Big(\sum_{k=1}^{d} \underset{\text{weights}}{W_{jk}} \, x_k + \underset{\text{biases}}{b_j}\Big)$$

activation function

e.g. ReLU tanh





Draw $\{W_{jk}, b_j\}$ randomly from Gaussian distributions ⇒ random features.

Fit $\{\theta_j\}$ to data: linear model w/ (a lot of) random features.

$$f_{\{\theta,W,b\}}(\vec{x}) = \sum_{j=1}^{N} \theta_j\, \varphi_j(\vec{x})$$

$$\varphi_j(\vec{x}) = \sigma\left(\sum_{k=1}^{d} W_{jk}\, x_k + b_j\right)$$

In reality, evolve all of $\{\theta_j, W_{jk}, b_j\}$ by gradient descent.

Loss function: $\mathcal{L} = \frac{1}{2} \sum_{\alpha=1}^{T} \left(f_{\{\theta,W,b\}}(\vec{x}_\alpha) - y_\alpha\right)^2$

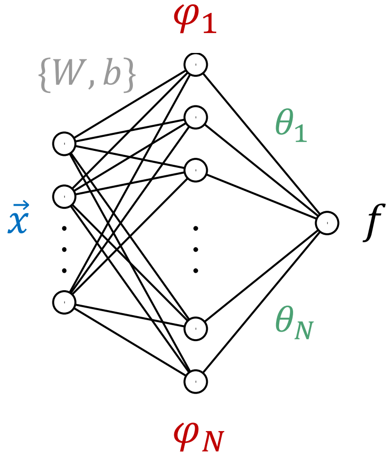
- $\theta_j(t+1) = \theta_j(t) - \eta \frac{\partial \mathcal{L}}{\partial \theta_j}(t) = \theta_j(t) - \eta \sum_{\alpha=1}^{T} (f(\vec{x}_\alpha; t) - y_\alpha)\, \varphi_j(\vec{x}_\alpha; t)$
- $W_{jk}(t+1) = W_{jk}(t) - \eta \frac{\partial \mathcal{L}}{\partial W_{jk}}(t) = W_{jk}(t) - \eta \sum_{\alpha=1}^{T} (f(\vec{x}_\alpha; t) - y_\alpha)\, \theta_j(t)\, \sigma_j'(\vec{x}_\alpha; t)\, (\vec{x}_\alpha)_k$
- $b_j(t+1) = b_j(t) - \eta \frac{\partial \mathcal{L}}{\partial b_j}(t) = b_j(t) - \eta \sum_{\alpha=1}^{T} (f(\vec{x}_\alpha; t) - y_\alpha)\, \theta_j(t)\, \sigma_j'(\vec{x}_\alpha; t)$

However, must have $\theta_j \sim \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ s.t. $f(\vec{x}) \sim \mathcal{O}(1)$.

Large N limit: only $\{\theta_j\}$ evolve, $\{W_{jk}, b_j\}$ are "frozen."

So indeed:

wide neural network ≈ linear model w/ a LOT of random features.

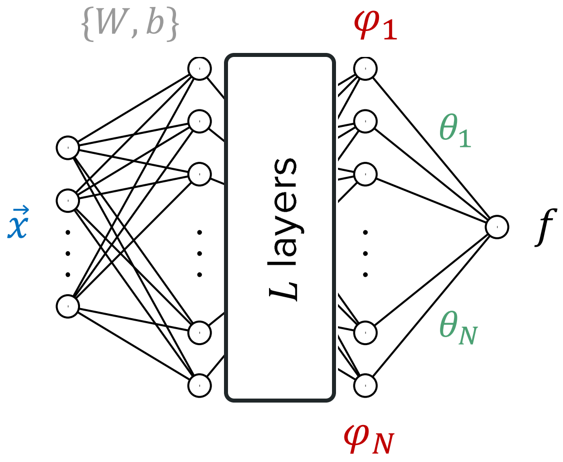


This is **Part 1** of how modern AI/ML works (approximately).

Part 2: stack L layers $\Rightarrow$ **deep** neural network.

Feature functions φ_j evolve according to data at $\mathcal{O}\left(\frac{L}{N}\right)$.

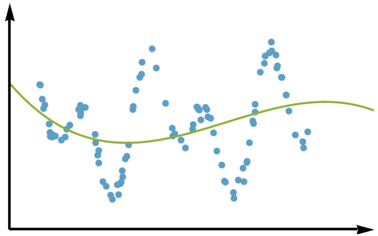
Emergent scale: depth-to-width ratio. [Roberts, Yaida, Hanin, 2106.10165]

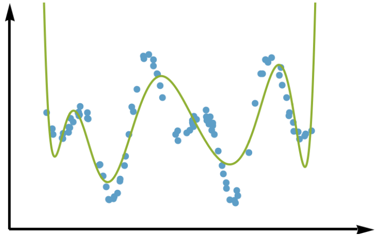


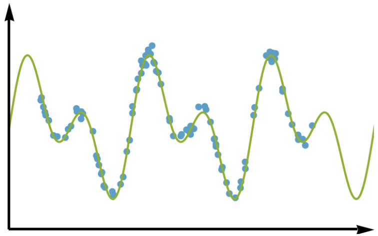
Note: must stay in $L \ll N$ regime b/c statistical fluctuations also accumulate as $\frac{L}{N}$.

Recap: (a physicist's (biased) understanding of) how AI/ML works

- ❑ Wide neural network ≈ linear model w/ a LOT of random features.
- ❑ Deep neural network learns features from data.







Recap: (a physicist's (biased) understanding of) how AI/ML works

- ❑ Wide neural network ≈ linear model w/ a LOT of random features.
- ❑ Deep neural network learns features from data.

Some behaviors of deep neural networks appear (at leading order) not to rely on their capability to learn features.

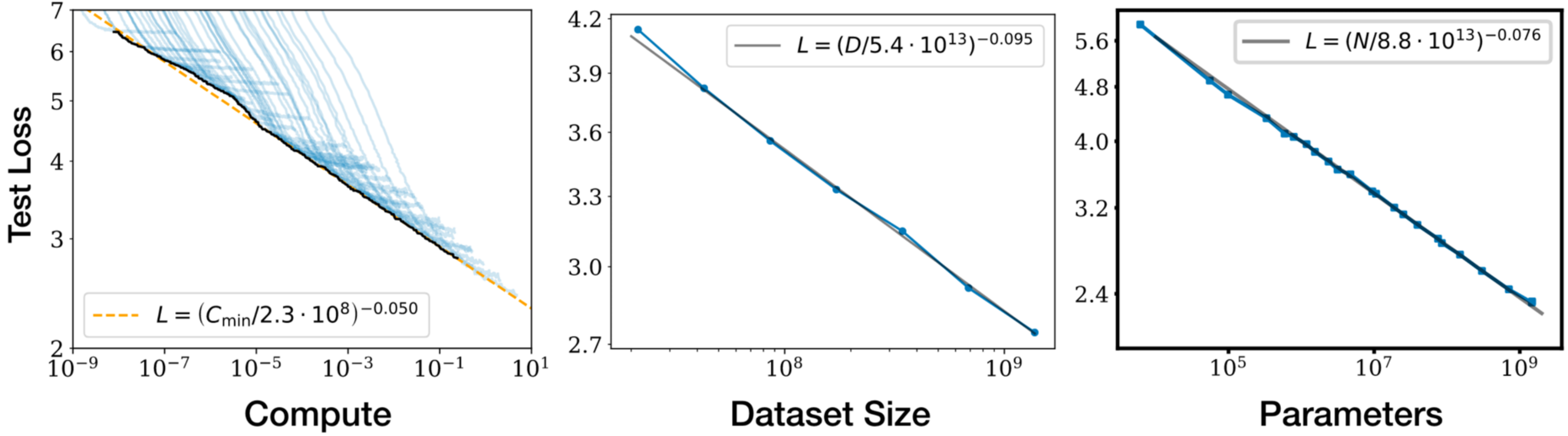
They are consequences of the "magic of large numbers" (of random features).

Example: **neural scaling laws**.

Neural scaling laws

Many ML models exhibit power law scaling of performance.

[Kaplan et al, 2001.08361] [Sharma, Kaplan, 2004.10802] [Bahri et al, 2102.06701]



Solvable random feature model to understand the physics? [Maloney, Roberts, Sully, 2210.16859]

Outline

- ✅ How AI/ML works.

  Wide and deep neural networks, magic of large numbers + feature learning.

- ○ Simple model of neural scaling laws.

  Teacher-student setup, fit random projections with random features.

- ○ Solution.

  Effective theory, planar diagrams.

- ○ Duality.

  Symmetry of scaling laws, neural networks ↔ field theories.

Teacher-student setup

input | (unknown to student) | label

Train: $\vec{x}_\alpha \longrightarrow$ teacher model $\longrightarrow y_\alpha$

$\vec{x}_\alpha \longrightarrow$ student model $f_\theta \longrightarrow f_\theta(\vec{x}_\alpha)$

$\left.\begin{matrix} y_\alpha \\ f_\theta(\vec{x}_\alpha) \end{matrix}\right\}$ compare

update parameters $\theta \rightarrow \theta^\star$ to minimize $\mathcal{L} = \frac{1}{2} \sum_{\alpha=1}^{T} \left(f_\theta(\vec{x}_\alpha) - y_\alpha\right)^2$

Test: $\hat{\vec{x}}_\beta \longrightarrow$ teacher model $\longrightarrow \hat{y}_\beta$

$\hat{\vec{x}}_\beta \longrightarrow$ student model $f_{\theta^\star} \longrightarrow f_{\theta^\star}(\hat{\vec{x}}_\beta)$

$\left.\begin{matrix} \hat{y}_\beta \\ f_{\theta^\star}(\hat{\vec{x}}_\beta) \end{matrix}\right\}$ compare

evaluate performance: $\hat{\mathcal{L}} = \frac{1}{2} \sum_{\beta=1}^{\hat{T}} \left(f_{\theta^\star}(\hat{\vec{x}}_\beta) - \hat{y}_\beta\right)^2$ (test loss)

Input: $\vec{x}_\alpha \,, \hat{\vec{x}}_\beta \in \mathbb{R}^M$

Training set: $x = (\vec{x}_1 \,, \dots , \vec{x}_T) = \begin{pmatrix} x_{11} & \cdots & x_{1T} \\ \vdots & \ddots & \vdots \\ x_{M1} & \cdots & x_{MT} \end{pmatrix} \in \mathbb{R}^{M \times T}$

Test set: $\hat{x} = \left(\hat{\vec{x}}_1 \,, \dots , \hat{\vec{x}}_{\hat{T}}\right) = \begin{pmatrix} \hat{x}_{11} & \cdots & \hat{x}_{1\hat{T}} \\ \vdots & \ddots & \vdots \\ \hat{x}_{M1} & \cdots & \hat{x}_{M\hat{T}} \end{pmatrix} \in \mathbb{R}^{M \times \hat{T}}$

Drawn from Gaussian distributions with

$$\langle x_{I\alpha} \rangle = \langle \hat{x}_{I\beta} \rangle = 0 \,, \quad \langle x_{I_1\alpha_1} x_{I_2\alpha_2} \rangle = \Lambda_{I_1 I_2} \delta_{\alpha_1\alpha_2} \,, \quad \langle \hat{x}_{I_1\beta_1} \hat{x}_{I_2\beta_2} \rangle = \Lambda_{I_1 I_2} \delta_{\beta_1\beta_2} \,.$$

Input: $\vec{x}_\alpha \, , \hat{\vec{x}}_\beta \in \mathbb{R}^M$

Drawn from Gaussian distributions with

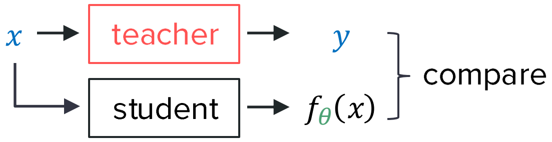
$$\langle x_{I\alpha} \rangle = \langle \hat{x}_{I\beta} \rangle = 0\,, \quad \langle x_{I_1\alpha_1} x_{I_2\alpha_2} \rangle = \Lambda_{I_1 I_2} \delta_{\alpha_1\alpha_2}\,, \quad \langle \hat{x}_{I_1\beta_1} \hat{x}_{I_2\beta_2} \rangle = \Lambda_{I_1 I_2} \delta_{\beta_1\beta_2}\,.$$

Eigenvalues of Λ : $\quad \lambda_I = \lambda_+ \, I^{-(1+\alpha)} \quad (\alpha > 0)\,.$

Power law in input data spectrum (observed in natural language and image data sets)

$\xrightarrow{\text{ML models w/ certain properties}}$ Power law scaling of test loss. [Maloney, Roberts, Sully, 2210.16859]

Teacher model: random projection



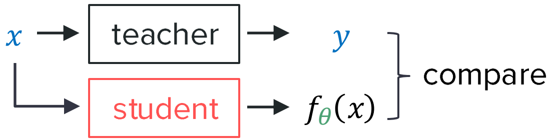
Training set: $y_\alpha \equiv y(\vec{x}_\alpha) = \sum_{I=1}^{M} w_I \, x_{I\alpha}$ i.e. $y = w\, x$

Test set: $\hat{y}_\beta \equiv \hat{y}\left(\vec{\hat{x}}_\beta\right) = \sum_{I=1}^{M} w_I \, \hat{x}_{I\beta}$ i.e. $\hat{y} = w\, \hat{x}$

with w drawn from Gaussian distribution with

$$\langle w_I \rangle = 0 \,, \; \left\langle w_{I_1} w_{I_2} \right\rangle = \frac{\sigma_w^2}{M} \, \delta_{I_1 I_2} \,.$$

Student model: wide neural network



$\approx$ linear model with a lot of random features.

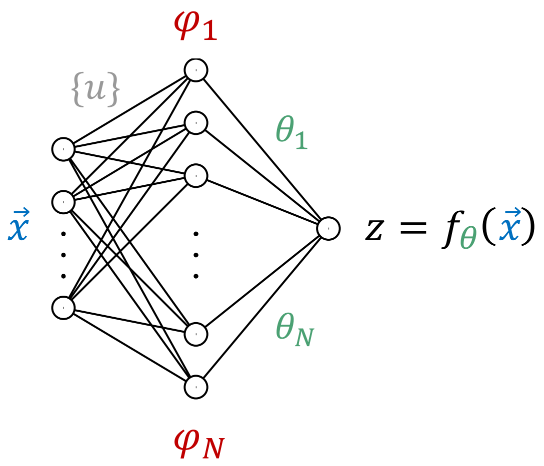
$z = \sum_{j=1}^{N} \theta_j \, \varphi_j$, $\varphi_j(\vec{x}) = \sum_{I=1}^{M} u_{jI} \vec{x}_I$ (omit biases & nonlinear activation function)

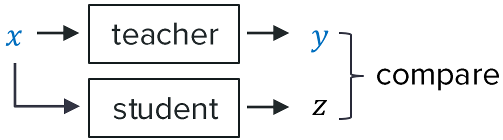
i.e. $z = \theta \, \varphi$, $\varphi = u \, x$.

Draw u from Gaussian. Train θ .

$\langle u_{jI} \rangle = 0$, $\langle u_{j_1 I_1} u_{j_2 I_2} \rangle = \frac{\sigma_u^2}{M} \, \delta_{j_1 j_2} \delta_{I_1 I_2}$.

Recall: Training does not change u in $N \to \infty$ limit.





Training

Loss function: $\mathcal{L} = \frac{1}{2}(\|z - y\|^2 + \gamma\|\theta\|^2)$

mean squared error   regulator (penalizes large $|\theta_j|$)

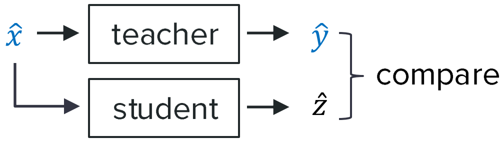
Minimize over training data set $\Rightarrow$ $\theta^\star = y\,\varphi^T q = y\,Q\,\varphi^T$

where $q = \frac{1}{\gamma + \varphi\varphi^T} \in \mathbb{R}^{N\times N}$, $Q = \frac{1}{\gamma + \varphi^T\varphi} \in \mathbb{R}^{T\times T}$

\# of features   # of training data points (samples)

γ ("ridge parameter") ensures invertibility $\Rightarrow$ unique solution.

Test



Evaluate student's performance after training by (per-sample) test loss.

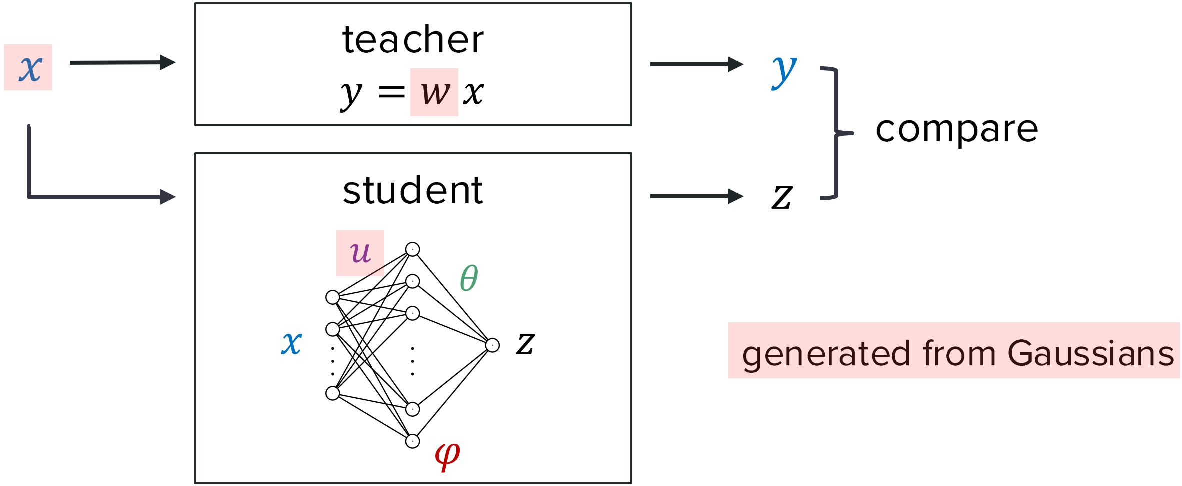
$$\hat{\mathcal{L}}(\gamma) = \frac{1}{2\hat{T}} \|\hat{z}(\gamma) - \hat{y}\|^2 \quad \text{with} \quad \hat{z}(\gamma) = \theta^\star(\gamma)\hat{\varphi} = y\,\varphi^T\,q(\gamma)\,\hat{\varphi} = y\,Q(\gamma)\,\varphi^T\hat{\varphi}\,.$$

Function of ridge parameter γ (coefficient of $\|\theta\|^2$ regularization term in training loss).

- $\gamma \to 0$ (ridgeless limit): solved in [Maloney, Roberts, Sully, 2210.16859] .
- General γ : solved in [**ZZ**, 2405.19398] $\Rightarrow$ role of regularization in ML + new scaling laws.

Recap: the model

[Maloney, Roberts, Sully, 2210.16859]



Outline

- How AI/ML works.

  Wide and deep neural networks, magic of large numbers + feature learning.

- Simple model of neural scaling laws.

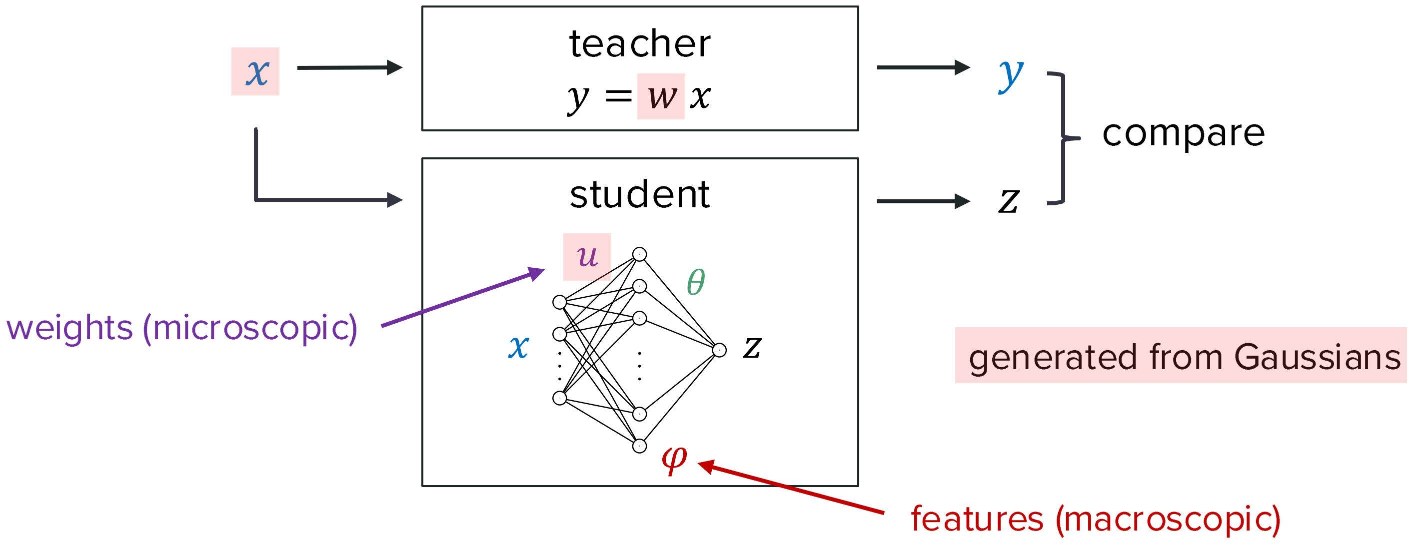
  Teacher-student setup, fit random projections with random features.

- Solution.

  Effective theory, planar diagrams.

- Duality.

  Symmetry of scaling laws, neural networks ↔ field theories.



Expected test loss: $\langle \hat{\mathcal{L}} \rangle = \frac{1}{Z} \int dx\, d\hat{x}\, du\, \hat{\mathcal{L}}\, e^{-S[x,\hat{x},u]}$

where $S[x,\hat{x},u] = \frac{1}{2}\,\mathrm{tr}\left(x^T \Lambda^{-1} x + \hat{x}^T \widehat{\Lambda}^{-1} \hat{x} + u \Sigma^{-1} u^T\right)$ w/ $\widehat{\Lambda} = \Lambda$, $\Sigma = \frac{\sigma_u^2}{M} \mathbf{1}_M$

Here's the key: $\hat{\mathcal{L}}$ depends on u only via $\varphi = u\,x$, $\hat{\varphi} = u\,\hat{x}$.

Integrate out weights $\Rightarrow$ effective theory of features

[**ZZ**, 2405.19398] (inspired by [Roberts, Yaida, Hanin, 2106.10165])

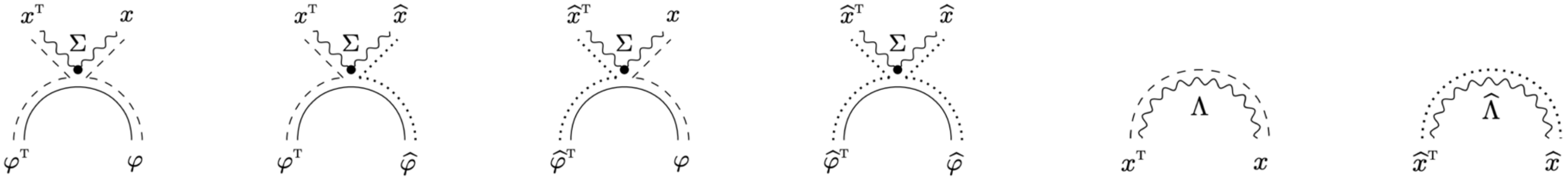
$$\frac{1}{Z_{\text{eff}}}\, e^{-S_{\text{eff}}[x,\widehat{x},\varphi,\widehat{\varphi}]} = \frac{1}{Z}\int du\,\delta(\varphi - ux)\,\delta(\widehat{\varphi} - u\widehat{x})\, e^{-S[x,\widehat{x},u]}$$

$$S_{\text{eff}}[x,\widehat{x},\varphi,\widehat{\varphi}] = \frac{1}{2}\,\mathrm{tr}\big(x^{\mathrm{T}}\Lambda^{-1}x\big) + \frac{1}{2}\,\mathrm{tr}\big(\widehat{x}^{\mathrm{T}}\,\widehat{\Lambda}^{-1}\widehat{x}\big) + \frac{1}{2}\,\mathrm{tr}\left[\begin{pmatrix}\varphi & \widehat{\varphi}\end{pmatrix}\mathbf{K}^{-1}\begin{pmatrix}\varphi^{\mathrm{T}}\\ \widehat{\varphi}^{\mathrm{T}}\end{pmatrix}\right] + N\,\log\det(2\pi\mathbf{K})$$

$$\mathbf{K} = \begin{pmatrix} x^{\mathrm{T}}\Sigma\, x & x^{\mathrm{T}}\Sigma\,\widehat{x} \\ \widehat{x}^{\mathrm{T}}\Sigma\, x & \widehat{x}^{\mathrm{T}}\Sigma\,\widehat{x}\end{pmatrix}$$

can be written as ghost action, but won't need explicitly

Feynman rules:



Highlights of the calculation

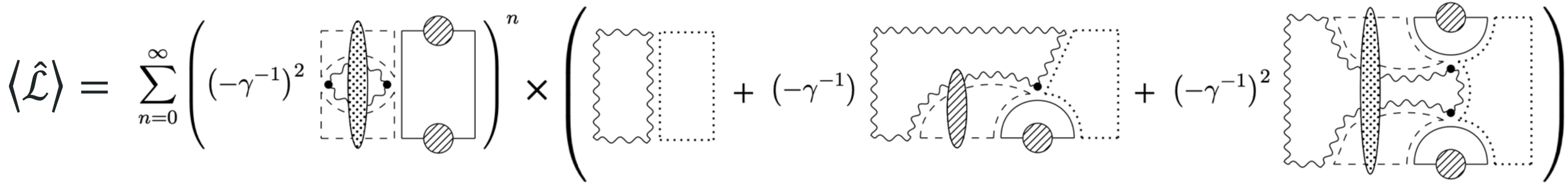
[**ZZ**, 2405.19398]

Matrix variables $\Rightarrow$ double line notation.

Large N, T, M limit $\Rightarrow$ planar diagrams only (as in large-N field theory).

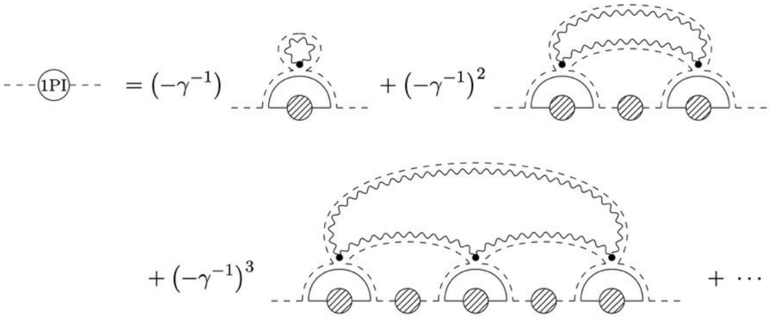
# of features, # of training samples, input dimension

Factorized structure:



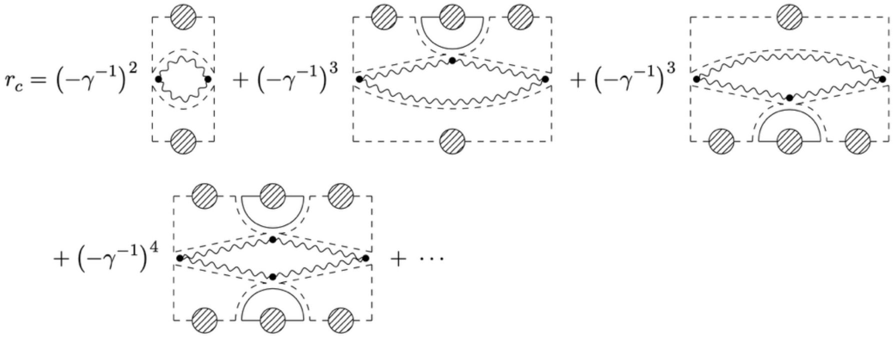
Each blob = geometric series. All-order resummation possible!

Here’s what it looks like…



$$= -\mathbb{1}_T\, N\langle q\rangle \sum_{n=0}^{\infty}\big(-\gamma NT\langle q\rangle\langle Q\rangle\big)^n \operatorname{tr}\big[(\Lambda\Sigma)^{n+1}\big]$$

$$= -\mathbb{1}_T\, N\langle q\rangle \operatorname{tr}\left[\frac{\Lambda\Sigma}{1+\gamma NT\langle q\rangle\langle Q\rangle\Lambda\Sigma}\right].$$



$$= \sum_{n_1,n_2=0}^{\infty}(-\gamma^{-1})^{2+n_1+n_2}\big(\gamma N\langle q\rangle\big)^{n_1+n_2}\big(\gamma T\langle Q\rangle\big)^{2+n_1+n_2}\operatorname{tr}\big[(\Lambda\Sigma)^{2+n_1+n_2}\big]$$

$$= T^2\langle Q\rangle^2 \operatorname{tr}\left\{(\Lambda\Sigma)^2\left[\sum_{n=0}^{\infty}(-\gamma\xi\Lambda\Sigma)^n\right]^2\right\}$$

$$= T^2\langle Q\rangle^2 \operatorname{tr}\left[\frac{(\Lambda\Sigma)^2}{(1+\gamma\xi\Lambda\Sigma)^2}\right],$$

Result

[**ZZ**, 2405.19398]

$$l = \mathrm{tr}\left[\frac{\widehat{\Lambda}}{(1+\gamma\xi\Lambda\Sigma)^2}\right]$$

$$\boxed{\langle\widehat{\mathcal{L}}\rangle = \frac{\sigma_w^2}{2M}\,\frac{l}{\big(1-\gamma^2 N\langle q\rangle^2 r_c\big)\big(1-\gamma^2 T\langle Q\rangle^2 v\big) - \frac{\gamma^2\xi^2}{NT}\,r_d'^2}}$$

$$r_c = T^2\langle Q\rangle^2\,\mathrm{tr}\left[\frac{(\Lambda\Sigma)^2}{(1+\gamma\xi\Lambda\Sigma)^2}\right] \qquad v = N^2\langle q\rangle^2\,\mathrm{tr}\left[\frac{(\Lambda\Sigma)^2}{(1+\gamma\xi\Lambda\Sigma)^2}\right] \qquad r_d' = \mathrm{tr}\left[\frac{\Lambda\Sigma}{(1+\gamma\xi\Lambda\Sigma)^2}\right]$$

$\xi, \langle q\rangle, \langle Q\rangle$ are defined by: $\xi = NT\langle q\rangle\langle Q\rangle$ $\quad \gamma\xi\,\mathrm{tr}\left(\frac{\Lambda\Sigma}{1+\gamma\xi\Lambda\Sigma}\right) = N\big(1-\gamma\langle q\rangle\big) = T\big(1-\gamma\langle Q\rangle\big)$

Result

[**ZZ**, 2405.19398]

$$\boxed{\langle\widehat{\mathcal{L}}\rangle = \frac{\sigma_w^2}{2M}\,\frac{l}{\big(1-\gamma^2 N\langle q\rangle^2 r_c\big)\big(1-\gamma^2 T\langle Q\rangle^2 v\big) - \frac{\gamma^2\xi^2}{NT}\, r_d'^2}}$$

Function of ridge parameter γ (coefficient of $||\theta||^2$ regularization term in training loss).

In practice, wish to set $\gamma = \gamma^\star$, optimal value at which $\langle\hat{\mathcal{L}}\rangle$ is minimized.

Scaling law of the optimal test loss:

power-law exponent of data spectrum

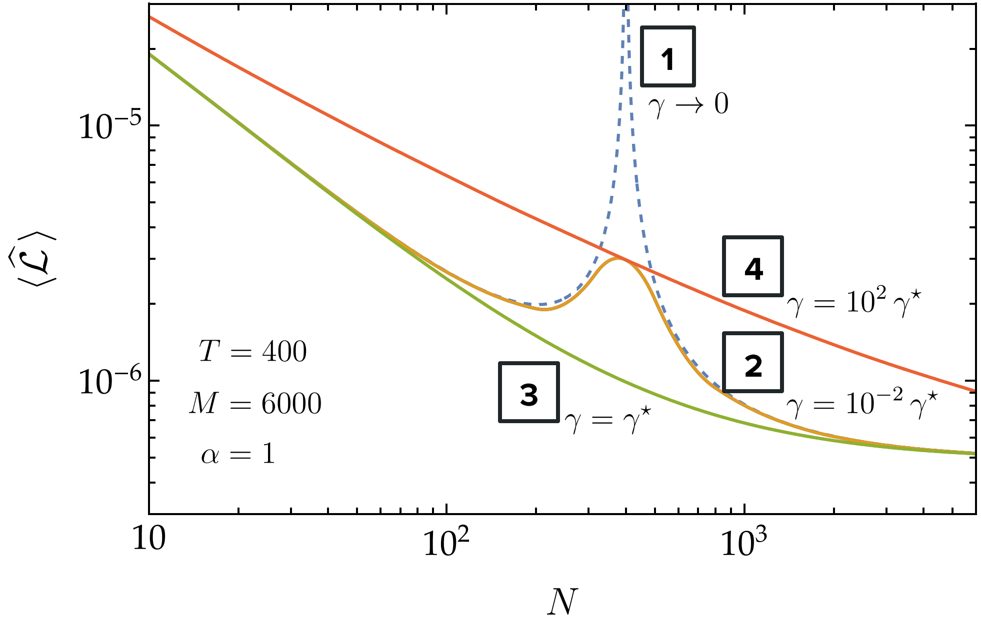
$$\langle\widehat{\mathcal{L}}\rangle(\gamma^\star) \simeq \frac{C\sigma_w^2\lambda_+}{2M}\left[\frac{\frac{\pi}{1+\alpha}}{\sin\!\left(\frac{\pi}{1+\alpha}\right)}\right]^{1+\alpha} \underbrace{\left(\frac{1+\nu}{2}\right)^{1+\alpha}\left[\frac{1+(1+\alpha)\,\nu}{2+\alpha}\right]^{-1}}_{\text{correction factors}} \underbrace{\left(\frac{1}{N}+\frac{1}{T}\right)^{\alpha}}_{\text{overall scaling}}$$

(involves α and a function $\nu(N,T)$)

(conjectured in [Maloney, Roberts, Sully, 2210.16859])

Role of regularization

Fix T (training dataset size), look at scaling with N (model size). Same with $N \leftrightarrow T$.



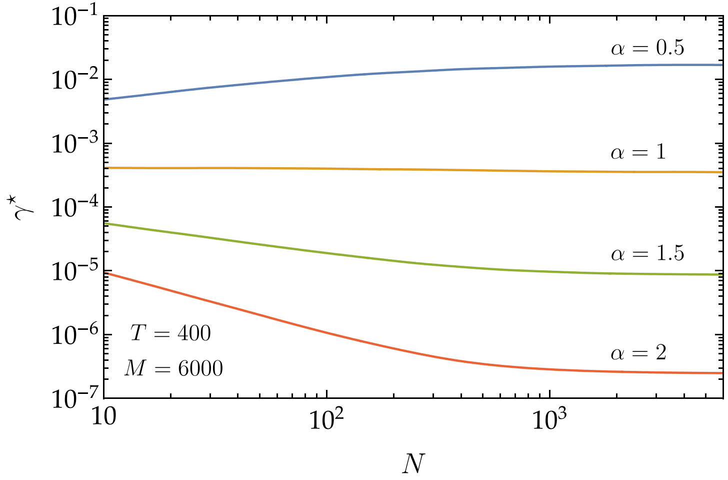
1. Unregularized (singular at $N = T$).
2. Under-regularized.
3. Optimally regularized.
4. Over-regularized.

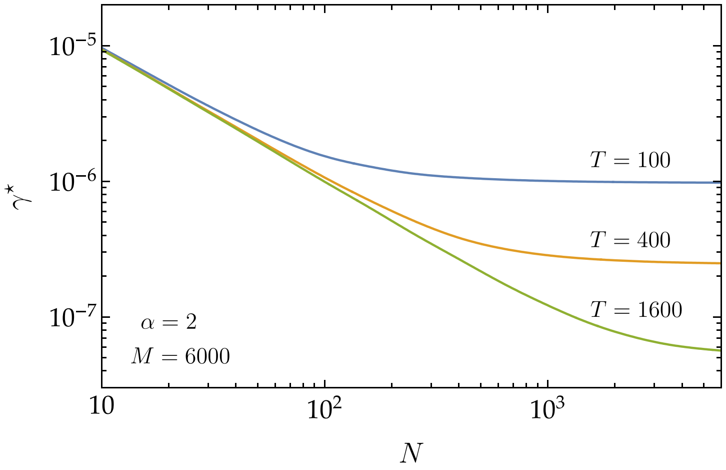
Power-law ($N < T$) followed by **plateau** ($N > T$), as observed in practical ML settings.

More scaling laws

Optimal ridge parameter also exhibits scaling laws (w/ different exponent).

$$\gamma^\star \simeq \frac{\sigma_u^2 \lambda_+}{M} \left[\frac{\frac{\pi}{1+\alpha}}{\sin\left(\frac{\pi}{1+\alpha}\right)} \right]^{1+\alpha} \frac{2}{\alpha(\alpha+1)} \left(\frac{1+\nu}{2} \right)^{\alpha-1} \left(\frac{1}{N} + \frac{1}{T} \right)^{\alpha-1}$$





Outline

- ✅ **How AI/ML works.**

  Wide and deep neural networks, magic of large numbers + feature learning.

- ✅ **Simple model of neural scaling laws.**

  Teacher-student setup, fit random projections with random features.

- ✅ **Solution.**

  Effective theory, planar diagrams.

- ○ **Duality.**

  Symmetry of scaling laws, neural networks ↔ field theories.

Neural scaling laws appear to be (approximately) **symmetric** between N and T.

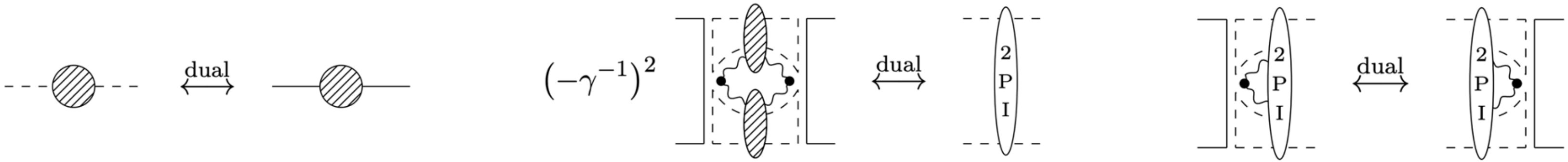
\# of features   \# of training samples

Our simple model indeed predicts $\langle \hat{\mathcal{L}} \rangle (N, T) = \langle \hat{\mathcal{L}} \rangle (T, N)$.

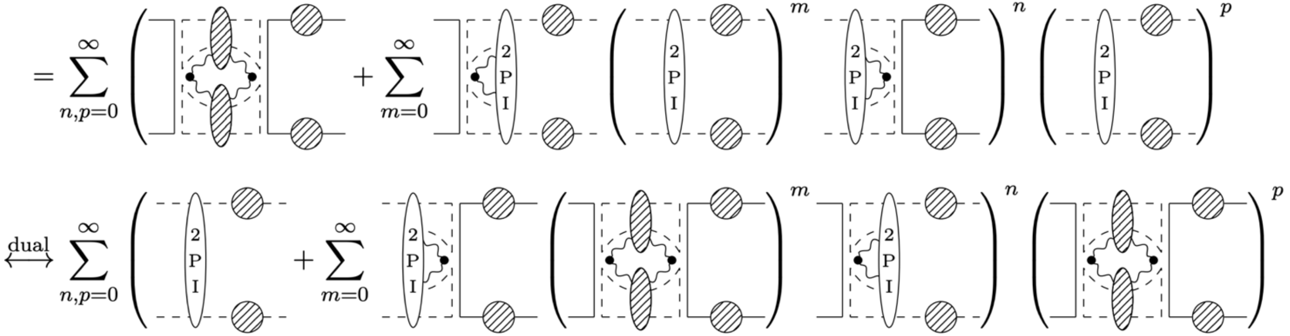
$$\langle \widehat{\mathcal{L}} \rangle = \frac{\sigma_w^2}{2M} \frac{l}{\left(1 - \gamma^2 N \langle q \rangle^2 r_c\right)\left(1 - \gamma^2 T \langle Q \rangle^2 v\right) - \frac{\gamma^2 \xi^2}{NT}\, r_d'^2}$$

There is actually a duality transformation that exchanges $N \leftrightarrow T$ and maps various quantities onto each other.

And it goes like this...

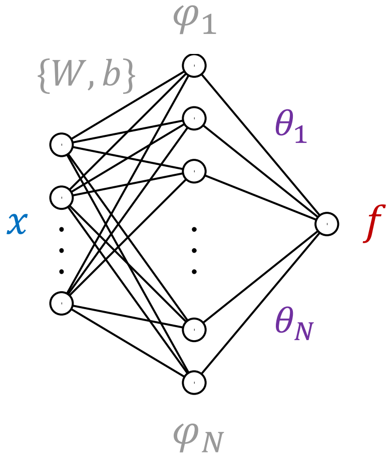


$$\mathcal{R} \cdot \frac{\mathcal{L}_1 + 2\mathcal{L}_2 + \mathcal{L}_3'}{\mathcal{L}_1 + 2\mathcal{L}_2 + \mathcal{L}_{3c}'}$$



Upshot: diagrams-level duality $\Rightarrow$ sum of all diagrams is self-dual.

Take this notion of duality further?



Let's initialize a 1 hidden layer neural network.

$$f_\theta(x) = \sum_{j=1}^{N} \theta_j \, \varphi_j(x) \qquad \text{with} \quad \theta_j \sim P(\theta)$$

(random) feature functions

Correlation functions (ignoring randomness in φ_j for now):

$$\langle f(x_1) \dots f(x_k) \rangle = \int d\theta \, P(\theta) \, f_\theta(x_1) \dots f_\theta(x_k)$$

$$= \int \mathcal{D}f \boxed{\int d\theta \, P(\theta) \, \delta(f - f_\theta)} f(x_1) \dots f(x_k) = \int \mathcal{D}f \boxed{P[f]} f(x_1) \dots f(x_k)$$

$$\langle f(x_1) \ldots f(x_k) \rangle = \int d\theta \, P(\theta) \, f_\theta(x_1) \ldots f_\theta(x_k) = \int \mathcal{D}f \, P[f] \, f(x_1) \ldots f(x_k)$$

Probability distribution in

parameter space: $P(\theta)$ w/ θ_j carrying feature index j ;

function space: $P[f]$ w/ $f_\alpha \equiv f(x_\alpha)$ carrying sample index α .

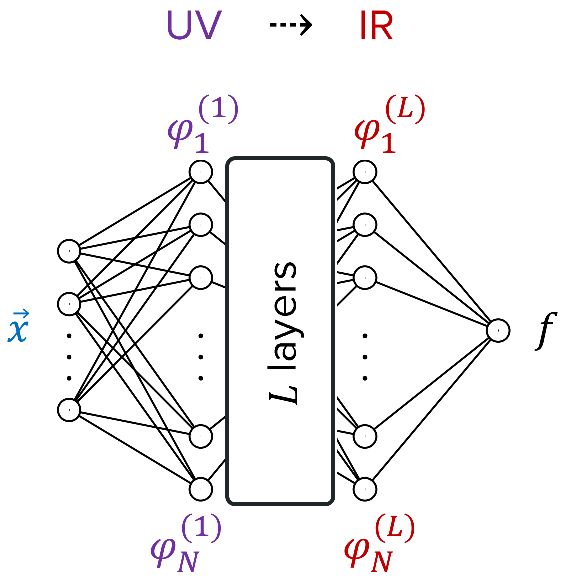
$$P[\underset{\phi}{\cancel{f}}] = \frac{1}{Z} \, e^{-S[\underset{\phi}{\cancel{f}}]}$$

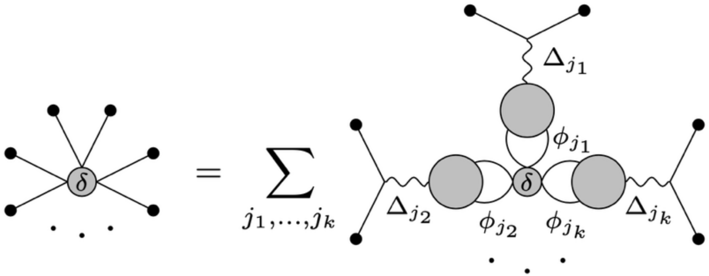
neural networks $\leftrightarrow$ field theories ?

[Halverson, Maiti, Stoner, 2008.08601 + 2106.00694] [Halverson, 2112.04527]
[Demirtas, Halverson, Maiti, Schwartz, Stoner, 2307.03223] [Howard, Klinger, Maiti, Stapleton, 2405.17538]

There's a notion of RG flow… and it has structures!

[Roberts, Yaida, Hanin, 2106.10165] [Banta, Cai, Craig, **ZZ**, 2305.02334]





$$\langle \varphi^{(\ell)}(x_1) \dots \varphi^{(\ell)}(x_{2k}) \rangle \sim \chi^k \langle \varphi^{(\ell-1)}(x_1) \dots \varphi^{(\ell-1)}(x_{2k}) \rangle$$

Ian Banta
(UCSB Physics → CCS)

Tianji Cai
(UCSB → SLAC)

Nathaniel Craig
(UCSB & KITP)

Outline

- ✅ How AI/ML works.

  Wide and deep neural networks, magic of large numbers + feature learning.

- ✅ Simple model of neural scaling laws.

  Teacher-student setup, fit random projections with random features.

- ✅ Solution.

  Effective theory, planar diagrams.

- ✅ Duality.

  Symmetry of scaling laws, neural networks $\leftrightarrow$ field theories.

Closing thoughts

AI/ML as a **tool** ⇒ amazing science.

“cat”

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

download review GitHub

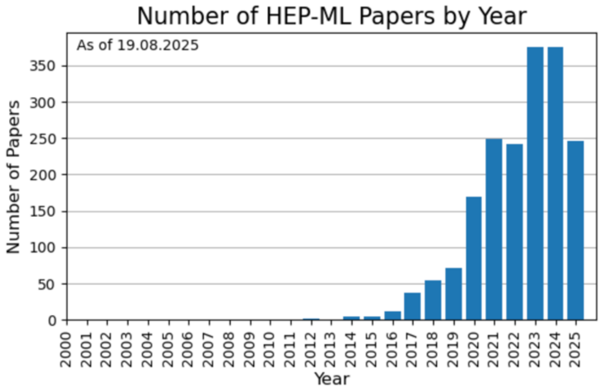
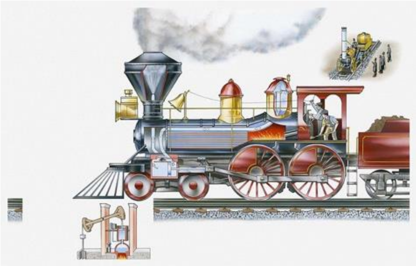
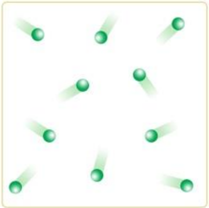
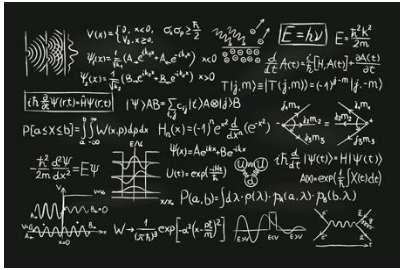


Table of contents

Closing thoughts

Better understanding of **how the tool works** ⇒ revolutionary science.

Industrial Revolution (steam engine)	⇨	thermodynamics & statistical physics	⇨	modern physics + countless new technologies
				
AI Revolution (deep neural networks)	⇨	theory of ML?	⇨	???

Physicists can (and should!) play a part in this.

Many intersections with ideas and tools in our field:
effective theories, RG, criticality, large-N field theory, Feynman diagrams, ...

Backup slides

Linear model = kernel method

Kernel: measure of similarity between data points.

$$(\varphi^T \varphi)_{\alpha_1 \alpha_2} = \sum_{j=1}^{N} \varphi_j(\vec{x}_{\alpha_1})\, \varphi_j(\vec{x}_{\alpha_2})$$

$$(\varphi^T \hat{\varphi})_{\alpha_1 \beta_2} = \sum_{j=1}^{N} \varphi_j(\vec{x}_{\alpha_1})\, \varphi_j(\hat{\vec{x}}_{\beta_2})$$

Trained model prediction: $\hat{z}_\beta = \sum_{\alpha_1 \alpha_2} y_{\alpha_1} (\gamma + \varphi^T \varphi)^{-1}_{\alpha_1 \alpha_2} (\varphi^T \hat{\varphi})_{\alpha_2 \beta}$

compare unseen test data with seen training data

Then form linear combination of seen labels according to similarity