Double Descent
The Success Recipe Behind Large Neural Networks

Leonidas Ntrekos

AUEB

January 1, 2026

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**
- In every introduction course to machine learning lectures, we learn that:

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**
- In every introduction course to machine learning lectures, we learn that:
  - Overfitting $\Rightarrow$ poor generalization

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**
- In every introduction course to machine learning lectures, we learn that:
  - Overfitting $\Rightarrow$ poor generalization
  - Underfitting $\Rightarrow$ unable to capture data complexity

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**
- In every introduction course to machine learning lectures, we learn that:
  - Overfitting $\Rightarrow$ poor generalization
  - Underfitting $\Rightarrow$ unable to capture data complexity
- Empirically, we have, however, discovered that over-parameterized models often generalize extremely well

Motivation

- In recent years, machine learning and artificial intelligence community has shifted its focus to scaling large neural networks to a great extent
- Modern neural networks have **billions of parameters**
- In every introduction course to machine learning lectures, we learn that:
  - Overfitting $\Rightarrow$ poor generalization
  - Underfitting $\Rightarrow$ unable to capture data complexity
- Empirically, we have, however, discovered that over-parameterized models often generalize extremely well
- This contradicts the classical **bias–variance trade-off**

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance
- "Good" Models: Moderate of both

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance
- "Good" Models: Moderate of both
- The classical machine learning approach implies that when increasing model complexity:

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance
- "Good" Models: Moderate of both
- The classical machine learning approach implies that when increasing model complexity:
  - Training error decreases monotonically

Bias–Variance Trade-Off

For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance
- "Good" Models: Moderate of both
- The classical machine learning approach implies that when increasing model complexity:
  - Training error decreases monotonically
  - Test error follows a U-shaped curve

Bias–Variance Trade-Off

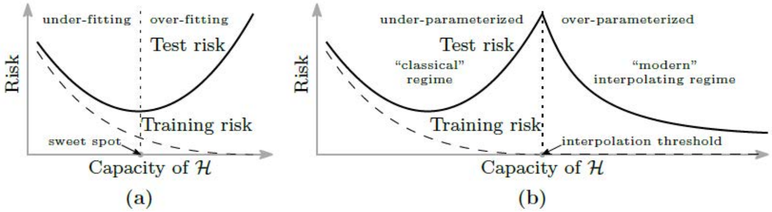
For regression:

$$\mathrm{MSE} = \mathbb{E}(Y - \hat{f}(X))^2$$

$$= \underbrace{\mathrm{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{\mathrm{Bias}^2(\hat{f}(X))}_{\text{Bias}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

- Simple models: high bias, low variance
- Flexible models: low bias, high variance
- "Good" Models: Moderate of both
- The classical machine learning approach implies that when increasing model complexity:
  - Training error decreases monotonically
  - Test error follows a U-shaped curve
- The goal has been to find the optimal point where both training and test error are simultaneously minimized: **bias–variance sweet spot**

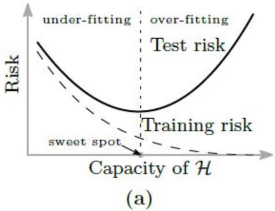
Double Descent Curve



Train and test error vs model complexity [Belkin et al., 2019]

- Double Descent (right plot) is observed in modern ML models

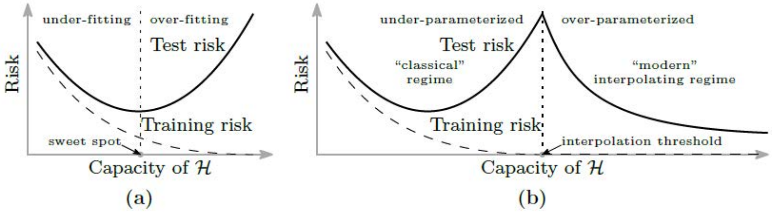
Double Descent Curve



Train and test error vs model complexity [Belkin et al., 2019]

- Double Descent (right plot) is observed in modern ML models
- Test error decreases again **after interpolation** (the point where parameters are equal to training data observations)

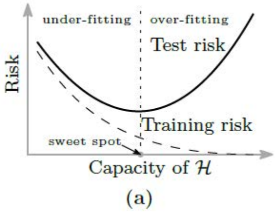
Double Descent Curve



Train and test error vs model complexity [Belkin et al., 2019]

- Double Descent (right plot) is observed in modern ML models
- Test error decreases again **after interpolation** (the point where parameters are equal to training data observations)
- Occurs when:

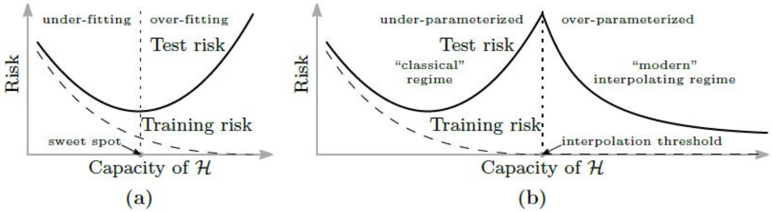
Double Descent Curve



Train and test error vs model complexity [Belkin et al., 2019]

- Double Descent (right plot) is observed in modern ML models
- Test error decreases again **after interpolation** (the point where parameters are equal to training data observations)
- Occurs when:
  - Parameters are greatly increased

Double Descent Curve



Train and test error vs model complexity [Belkin et al., 2019]

- Double Descent (right plot) is observed in modern ML models
- Test error decreases again **after interpolation** (the point where parameters are equal to training data observations)
- Occurs when:
  - Parameters are greatly increased
  - Model fits training data perfectly and even beyond

Example I: Natural Cubic Splines

- To illustrate the Double Descent phenomenon, we will do a simple simulation

Example I: Natural Cubic Splines

- To illustrate the Double Descent phenomenon, we will do a simple simulation
- Simulate data: $x \sim U[-5, 5]$,

Example I: Natural Cubic Splines

- To illustrate the Double Descent phenomenon, we will do a simple simulation
- Simulate data: $x \sim U[-5, 5]$,
- Transform: $y = \sin(x) + \varepsilon$, where $\varepsilon \sim N(0, 0.3^2)$

Example I: Natural Cubic Splines

- To illustrate the Double Descent phenomenon, we will do a simple simulation
- Simulate data: $x \sim U[-5, 5]$,
- Transform: $y = \sin(x) + \varepsilon$, where $\varepsilon \sim N(0, 0.3^2)$
- Based on 20 training samples (with noise from a Gaussian) to estimate 1000 points from a uniform distribution that are transformed to sine function

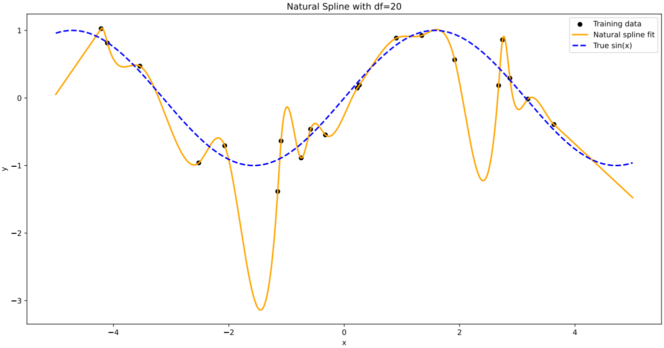
Example I: Natural Cubic Splines

- To illustrate the Double Descent phenomenon, we will do a simple simulation
- Simulate data: $x \sim U[-5, 5]$,
- Transform: $y = \sin(x) + \varepsilon$, where $\varepsilon \sim N(0, 0.3^2)$
- Based on 20 training samples (with noise from a Gaussian) to estimate 1000 points from a uniform distribution that are transformed to sine function
- Fit a spline regression model with d degrees of freedom,

$$f(x_i) = \sum_{j=1}^{d} \hat{\beta}_j \, N_j(x_i),$$

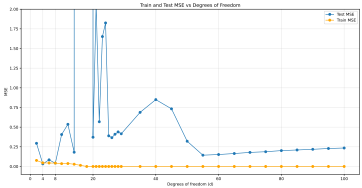
where $N_j(\cdot)$ are cubic spline basis functions and the coefficients $\hat{\beta}_j$ are estimated from the data.

Interpolation - Illustration

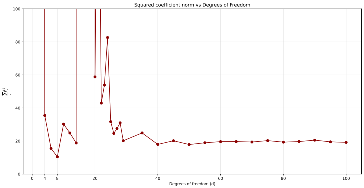


Exact fit of 20 points (interpolation)

Interpolation – Illustration

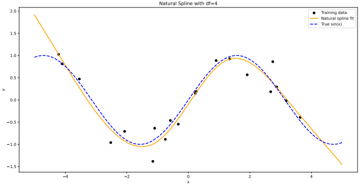


Training and test mean squared error versus model complexity

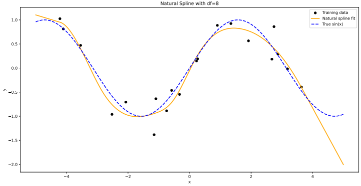


ℓ_2 norm of spline coefficients

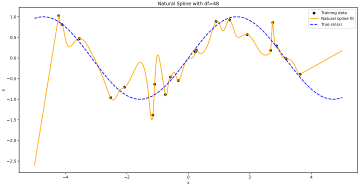
Natural Spline Fits for Increasing Degrees of Freedom



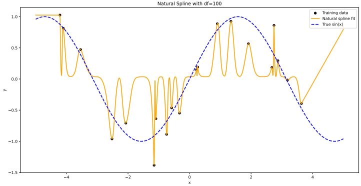
df = 4



df = 8



df = 48



df = 100

Over-Parameterized Regime

- When Degrees of freedom $>$ samples we essentially have **infinite interpolating solutions**

Over-Parameterized Regime

- When Degrees of freedom $>$ samples we essentially have **infinite interpolating solutions**
- Least squares (and various other algorithms that solve optimization problems) selects the **minimum** ℓ_2-norm solution

Over-Parameterized Regime

- When Degrees of freedom $>$ samples we essentially have **infinite interpolating solutions**
- Least squares (and various other algorithms that solve optimization problems) selects the **minimum** ℓ_2-norm solution
- The solution with the least absolute coefficient value

Over-Parameterized Regime

- When Degrees of freedom $>$ samples we essentially have **infinite interpolating solutions**
- Least squares (and various other algorithms that solve optimization problems) selects the **minimum** ℓ_2-norm solution
- The solution with the least absolute coefficient value
- This results in jiggly functions (equations) that tend to *approach* reality through many different angles as demonstrated above

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

- In practice, this is implemented using real-valued Random Fourier Features:

$$f(x) = w^\top z(x)$$

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

- In practice, this is implemented using real-valued Random Fourier Features:

$$f(x) = w^\top z(x)$$

- Where essentially transform the data into the feature map as:

$$z(x) = \sqrt{\frac{2}{N}} \cos(Wx + b)$$

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

- In practice, this is implemented using real-valued Random Fourier Features:

$$f(x) = w^\top z(x)$$

- Where essentially transform the data into the feature map as:

$$z(x) = \sqrt{\frac{2}{N}} \cos(Wx + b)$$

- $W \in \mathbb{R}^{N \times d}$ with rows $w_k \sim \mathcal{N}(0, \sigma^{-2} I_d)$

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

- In practice, this is implemented using real-valued Random Fourier Features:

$$f(x) = w^\top z(x)$$

- Where essentially transform the data into the feature map as:

$$z(x) = \sqrt{\frac{2}{N}} \cos(Wx + b)$$

- $W \in \mathbb{R}^{N \times d}$ with rows $w_k \sim \mathcal{N}(0, \sigma^{-2} I_d)$
- $b \in \mathbb{R}^N$, with $b_k \sim \mathrm{Uniform}(0, 2\pi)$

Example II: Random Fourier Features

- We demonstrate double descent on a real-world application using the MNIST dataset, using Random Fourier Features:

$$f(x) = \sum_{k=1}^{N} a_k \phi(x; v_k), \qquad \phi(x, v) := e^{i\langle v, x\rangle}$$

- In practice, this is implemented using real-valued Random Fourier Features:

$$f(x) = w^\top z(x)$$

- Where essentially transform the data into the feature map as:

$$z(x) = \sqrt{\frac{2}{N}} \cos(Wx + b)$$

- $W \in \mathbb{R}^{N\times d}$ with rows $w_k \sim \mathcal{N}(0, \sigma^{-2} I_d)$
- $b \in \mathbb{R}^N$, with $b_k \sim \mathrm{Uniform}(0, 2\pi)$
- Inner products $z(x)^\top z(x')$ approximate the Gaussian kernel

MNIST Dataset

- MNIST: well-established benchmark dataset for image classification

MNIST Dataset

- MNIST: well-established benchmark dataset for image classification
- Handwritten digits labeled from 0 to 9 ($K = 10$ classes)

MNIST Dataset

- MNIST: well-established benchmark dataset for image classification
- Handwritten digits labeled from 0 to 9 ($K = 10$ classes)
- Grayscale images of size 28×28 pixels ($d = 784$)

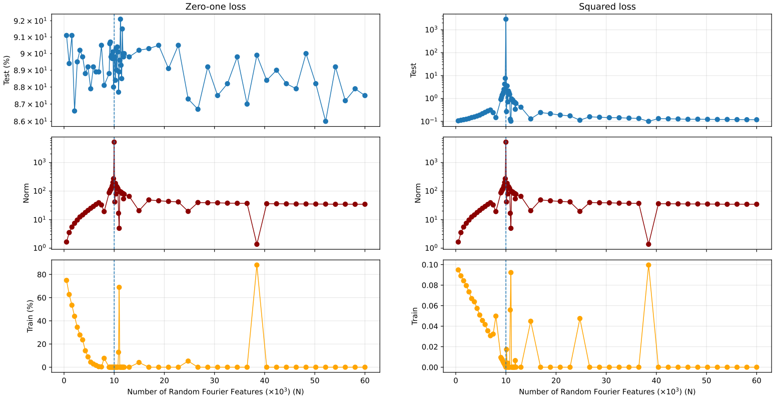
MNIST Dataset

- MNIST: well-established benchmark dataset for image classification
- Handwritten digits labeled from 0 to 9 ($K = 10$ classes)
- Grayscale images of size 28×28 pixels ($d = 784$)
- $n = 1000$ training samples and 1000 test samples in the following example

MNIST Dataset

- MNIST: well-established benchmark dataset for image classification
- Handwritten digits labeled from 0 to 9 ($K = 10$ classes)
- Grayscale images of size 28×28 pixels ($d = 784$)
- $n = 1000$ training samples and 1000 test samples in the following example
- In the classification setting, we additionally report the zero–one loss, which equals 1 for an incorrect prediction and 0 for a correct one.

RFF Results



Training/test error and coefficient norms vs N

Key Takeaways

- Double descent contradicts classical intuition as it has no application in large models

Key Takeaways

- Double descent contradicts classical intuition as it has no application in large models
- Interpolation does not imply poor generalization anymore

Key Takeaways

- Double descent contradicts classical intuition as it has no application in large models
- Interpolation does not imply poor generalization anymore
- Minimum-norm solutions play a crucial role as the key requirement to yield such results

Key Takeaways

- Double descent contradicts classical intuition as it has no application in large models
- Interpolation does not imply poor generalization anymore
- Minimum-norm solutions play a crucial role as the key requirement to yield such results
- SGD acts as an implicit regularizer and is widely used to estimate the parameters of neural networks

Key Takeaways

- Double descent contradicts classical intuition as it has no application in large models
- Interpolation does not imply poor generalization anymore
- Minimum-norm solutions play a crucial role as the key requirement to yield such results
- SGD acts as an implicit regularizer and is widely used to estimate the parameters of neural networks
- The examples share a common approach to neural networks, essentially estimating a function f through the sum of multiple simpler functions

Conclusion

- Double **describes/illustrates** why modern large (to the point of over-parameterization) models perform exceptionally
- The main idea is that, by using too many parameters, selecting the minimum-norm solution leads to predictors that approximate the true underlying behavior well.
- This is what Large Language Models essentially are, over-fitted models that have been trained on the whole internet with billions of parameters and are able to **approximate** the truth (with a pinch of randomness)
- Full theoretical understanding is yet to be found

References

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019).
Reconciling modern machine-learning practice and the classical bias–variance trade-off.
Proceedings of the National Academy of Sciences, 116(32):15849–15854.