Report

High-dimensional analysis of double descent for linear regression with random projections : Francis Bach 2023

Department of Statistics
Ludwig-Maximilians-Universität München

Sahibnoor Singh

February 24, 2025

Supervised by Jana Gauss

Abstract

This report introduces the double descent phenomenon in linear regression and explains why classical methods alone do not fully capture its underlying mechanisms. Then I explore Francis Bach's high-dimensional analysis, which takes advantage of ridge regression, random matrix theory, and random projections to offer a clear mathematical explanation of double descent. This framework highlights the role of implicit regularization in helping over-parameterized models maintain strong generalization. I also present experimental results that illustrate double descent under random projections. Finally, I connect these insights to neural networks, emphasizing how they can inform faster and more accurate kernel-based methods in modern deep learning.

Contents

1 Introduction **1**
1.1 Overview of the Double Descent Phenomenon 1

2 Double Descent in Linear Regresssion **2**
2.1 Classical Linear Regression Setup 2
2.2 Observing Double Descent 2
2.3 Intuition for Why Classical Methods Don't Fully Explain It 3

3 High-Dimensional Analysis (Francis Bach's Work) **4**
3.1 Ridge Regression Framework 4
3.2 Ridge Regression Analysis in High-Dimensional Settings 4
3.2.1 Problem Setup and Estimator 4
3.2.2 Fixed Design Analysis 4
3.2.3 Random Design Analysis 5
3.2.4 Asymptotic Risk Decomposition 5
3.2.5 Key Observations 5
3.3 Self-Induced Regularization 6
3.4 Analysis of Self-induced Regularization 6
3.5 Random projections 7

4 Experiments and Illustrations **8**
4.0.1 Illustration of the Double Descent Phenomenon 8

5 Unifying Grokking and Double Descent **9**
5.1 Overview and Motivation 9
5.2 Key Claims 9
5.3 Model of Pattern Learning 9
5.4 Epoch-wise vs. Model-wise Perspective 10
5.5 Empirical Findings 10
5.6 Implications for Understanding Generalization 10

6 Application to Neural Networks **11**
6.1 Optimization under Interpolation 11
6.2 Why SGD? 11
6.3 Exponential convergence of m-SGD 11
6.4 Minibatch size? 12
6.5 SGD is (much) faster than GD 12
6.6 The power of interpolation 12
6.7 Learning from deep learning: fast and effective kernal machines 13
6.8 From classical statistics to modern ML 13

7 Conclusion **14**

1 Introduction

1.1 Overview of the Double Descent Phenomenon

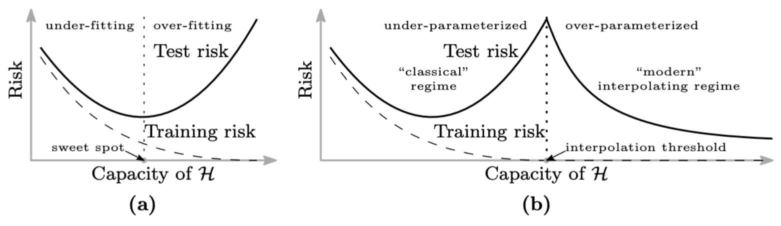


Figure 1: **Curves for training risk (dashed line) and test risk (solid line)**. (a) The classical U-shaped risk curve arising from the bias-variance trade-off. (b) The double descent risk curve, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk. Belkin et al. (2019)

In classical statistical learning, there is a principle called the bias–variance trade-off. It says that if we make our model more complex—by adding more parameters or features—we can usually reduce "bias" (the tendency to miss important patterns), but we also increase "variance" (the tendency to fit random quirks of the training data). According to traditional thinking, once the model becomes too large, it starts to overfit, meaning it focuses too much on the noise in the training set rather than the true underlying patterns. As a result, its performance on new, unseen data gets worse. This leads to the traditional belief that after a certain point, making a model bigger will only hurt its accuracy and reliability—so, "larger models are worse."
However, modern deep learning challenges this idea. Today's neural networks often have millions or even billions of parameters, easily enough to memorize an entire training dataset (including any random labels). Yet, in practice, these huge models actually perform better on many real tasks than smaller models. In fact, many practitioners now follow the rule of thumb that "larger models are better." This contrast between the older theoretical view and what we see in modern applications has motivated a lot of research into understanding why very large, over-parameterized networks still manage to generalize and work so well in practice.
In Figure 1(b), the under-parameterized regime, where the model complexity is small compared to the number of samples, the test error as a function of model complexity follows the U-like behavior predicted by the classical bias/variance tradeoff. However, once model complexity is sufficiently large to interpolate i.e., achieve (close to) zero training error, then increasing complexity only decreases test error, following the modern intuition of "bigger models are better". This phenomenon was first postulated in generality by Belkin et al. (2019) who named it **"double descent"**, and demonstrated it for decision trees, random features, and 2-layer neural networks with '2 loss, on a variety of learning tasks including MNIST and CIFAR-10.
This report introduces the double descent phenomenon in **linear regression**. We examine why classical methods alone fail to fully explain this phenomenon and explore the theoretical advancements provided by Bach (2023). We explore Bach (2023), which uses ridge regression, random matrix theory, and random projections to provide a clear mathematical explanation for double descent. His framework highlights how **implicit regularization** enables over-parameterized models to generalize effectively. We also present experiments demonstrating double descent under random projections and connect these findings to neural networks, showing how they inform faster, more accurate kernel-based methods in deep learning

2 Double Descent in Linear Regresssion

2.1 Classical Linear Regression Setup

In Dr. Roland Speicher (2023), we assume simplest neural network, without hidden layer and without a non-linearity.

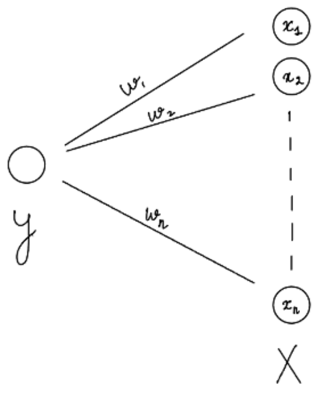


Figure 2: Dr. Roland Speicher (2023)

Here, $y = f(x) = wx$ and $Y = wX$, where (X, Y) are given and we look for the "best" w to model this. This is then nothing else but the classical problem of "linear regression".
Assume we have a linear relation, but have noise in the measurements, thus we see

$$\hat{Y} = wX + N,$$

where N is some Gaussian noise. We want to find the best $\hat{w}$ such that we have

$$\hat{Y} = \hat{w}X.$$

Note that $X \in \mathbb{R}^{p\times n}$, $\hat{w} \in \mathbb{R}^{1\times p}$ and $\hat{Y} \in \mathbb{R}^{1\times n}$, i.e. we have a system of n linear equations for p variables. This is under-determined for $n < p$ and over-determined for$n > p$. These two cases will correspond to the two regimes in our double descent picture.

2.2 Observing Double Descent

Combining all the results from Dr. Roland Speicher (2023), we get expected error

$$E\left[\|w - \hat{w}\|^2\right] = \begin{cases} \sigma^2 \frac{\gamma}{1-\gamma}, & \gamma < 1, \\ \sigma^2 \frac{1}{\gamma-1} + \|w\|^2 \left(1 - \frac{1}{\gamma}\right), & \gamma > 1. \end{cases} \quad (1)$$

where $\gamma = \frac{p}{n}$. and σ^2 is variance of noise as we assume gaussian noise.
This equation describes how the mean squared error (MSE) of the estimator $\hat{w}$ depends on the parameter γ. When $\gamma < 1$, the MSE is simply $\sigma^2 \frac{\gamma}{1-\gamma}$, showing that in this "high-noise" regime, the error depends only on the noise variance σ^2. By contrast, when $\gamma > 1$, the MSE has two parts: one piece is $\sigma^2 \frac{1}{\gamma-1}$, which still comes from the noise, and the other is $\|w\|^2 \left(1 - \frac{1}{\gamma}\right)$, reflecting the bias introduced by shrinking w. Here, $\|w\|$ is the norm of the true parameter, and γ can be interpreted as a signal-to-noise ratio or a shrinkage factor. Essentially, once $\gamma > 1$, the true parameter's magnitude starts to influence the MSE, indicating that the estimator's reduced shrinkage in this regime yields a bias term that depends on $\|w\|$.

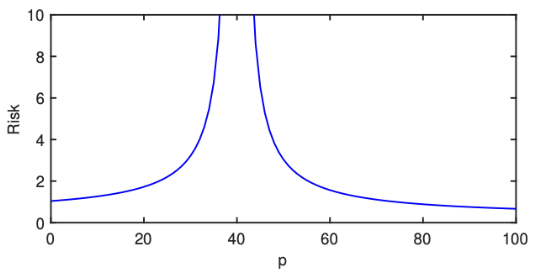


Figure 3: Belkin et al. (2020), Plot of risk as a function of p as Number of parameters. Here, $\|w\|^2 = 1, \sigma^2 = \frac{1}{25}, n = 40, D = 100$, where D is total number of parameters

This graph shows how test error (vertical axis) changes as we increase the number of parameters p(horizontal axis). When p is small, the test error stays at a moderate level. As p nears the number of training samples n, the error suddenly spikes, forming a peak. Surprisingly, once p becomes larger than n, the error goes back down again, demonstrating that very large models can sometimes perform better than models with fewer parameters.

2.3 Intuition for Why Classical Methods Don't Fully Explain It

Once a model has enough parameters to perfectly fit its training data, one would normally expect severe overfitting and poor performance on new inputs. Surprisingly, though, making the model even larger can sometimes lead to better test results instead of worse ones. This puzzling pattern, often called "double descent," isn't fully explained by the basic formula that shows a rise and then a drop in error.Traditionally, regularization (such as ridge or L1 penalties) is used to combat overfitting by penalizing large parameter values, forcing the model toward simpler solutions that generalize better. Surprisingly, though, when models become even larger—exceeding the number of training samples—test performance can improve again instead of getting worse, is due to some missing piece. The formula 1 tells us **what happens**, but not **why** it happens. The missing piece is **implicit regularization**—even with no explicit penalty, standard optimization routines (such as gradient descent) usually lean toward solutions with smaller norms or other structured qualities. This "hidden" bias can keep models from latching onto noise, allowing over-parameterized systems to retain strong generalization. In his recent work, Bach (2023) uses concepts like ridge regression, random matrix theory, and random projections to mathematically clarify how these implicit constraints emerge in high-dimensional spaces. By analyzing the geometry of the solution space and the behavior of optimization algorithms, Bach (2023) shows how these methods gravitate toward well-regularized solutions—despite the model's vast capacity—thus offering a deeper explanation of why the test error eventually decreases again, even when the number of parameters far exceeds the number of data points.

In the following sections, we will discuss these ideas in more depth and show how they connect to practical methods for constructing fast and accurate models. we will delve into Bach (2023), examine the experimental results that demonstrate implicit regularization, and explore how random projections can further illustrate and harness double descent in practice.

3 High-Dimensional Analysis (Francis Bach's Work)

3.1 Ridge Regression Framework

Key Reference: Bach (2023)
Ridge regression serves as a foundational framework for understanding how the double descent phenomenon emerges in high-dimensional settings used in this paper.

Ridge Regression extends ordinary least squares (OLS) by introducing **L2 regularization** to mitigate overfitting. The loss function combines the OLS objective with a penalty term proportional to the squared magnitude of the coefficients:

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where y_i is the target value, β_0 is the intercept, β_j are coefficients, x_{ij} are features, and $\lambda \geq 0$ controls regularization strength. In matrix form (assuming centered data), the estimator minimizes $\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|^2$, leading to the closed-form solution:

$$\hat{\beta}_{\text{Ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y},$$

where $\mathbf{I}$ is the identity matrix. The $\lambda\mathbf{I}$ term ensures invertibility even with multicollinearity. As λ increases, coefficients shrink toward zero, introducing bias but reducing variance. Unlike OLS, Ridge regression balances the bias-variance trade-off, favoring generalizability over training-data precision. Optimal λ is chosen via cross-validation, and feature standardization is critical due to Ridge's sensitivity to scale. This method is particularly effective in high-dimensional settings or when predictors are correlated, stabilizing coefficient estimates while preserving interpretability.

3.2 Ridge Regression Analysis in High-Dimensional Settings

Key Reference: Bach (2023)
This section provides a detailed explanation of ridge regression as analyzed in the paper, focusing on high-dimensional asymptotic regimes. The goal is to characterize the excess risk in terms of bias and variance, leveraging tools from random matrix theory.

3.2.1 Problem Setup and Estimator

Consider the linear regression model with n observations and d-dimensional covariates. The response vector $y \in \mathbb{R}^n$ follows:

$$y = X\theta_* + \varepsilon,$$

where $X \in \mathbb{R}^{n \times d}$ is the design matrix with i.i.d. sub-Gaussian rows, $\theta_* \in \mathbb{R}^d$ is the true parameter, and $\varepsilon \in \mathbb{R}^n$ is noise with $\mathbb{E}[\varepsilon] = 0$ and $\text{var}(\varepsilon) = \sigma^2$. The ridge regression estimator is defined as:

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \left\{ \frac{1}{n} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2 \right\},$$

which has the closed-form solution:

$$\hat{\theta} = (X^\top X + n\lambda I)^{-1} X^\top y.$$

3.2.2 Fixed Design Analysis

Under a *fixed design* (deterministic X), the excess risk decomposes into squared bias and variance:

$$\mathbb{E}_\varepsilon \left[(\hat{\theta} - \theta_*)^\top \widehat{\Sigma} (\hat{\theta} - \theta_*) \right] = \underbrace{\lambda^2 \theta_*^\top (\widehat{\Sigma} + \lambda I)^{-2} \widehat{\Sigma} \theta_*}_{\text{Bias}^2} + \underbrace{\frac{\sigma^2}{n} \text{tr} \left[\widehat{\Sigma}^2 (\widehat{\Sigma} + \lambda I)^{-2} \right]}_{\text{Variance}},$$

where $\widehat{\Sigma} = \frac{1}{n} X^\top X$ is the empirical covariance matrix. The degrees of freedom $\mathrm{df}_1(\lambda) = \mathrm{tr}[\widehat{\Sigma}(\widehat{\Sigma} + \lambda I)^{-1}]$ and $\mathrm{df}_2(\lambda) = \mathrm{tr}[\widehat{\Sigma}^2(\widehat{\Sigma} + \lambda I)^{-2}]$ govern the trade-off between bias and variance.

3.2.3 Random Design Analysis

For a *random design* (stochastic X), the analysis incorporates high-dimensional asymptotics where $n, d \to \infty$ with $d/n \to \gamma > 0$. A key result from random matrix theory is the emergence of an *implicit regularization parameter* $\kappa(\lambda) \geq \lambda$, defined via the self-consistency equation:

$$\kappa(\lambda) \left(1 - \frac{1}{n} \mathrm{df}_1(\kappa(\lambda))\right) \sim \lambda,$$

where $\mathrm{df}_1(\kappa) = \mathrm{tr}[\Sigma(\Sigma + \kappa I)^{-1}]$. This parameter $\kappa(\lambda)$ accounts for additional regularization induced by the randomness of X.

3.2.4 Asymptotic Risk Decomposition

Using Proposition 3 from the paper, the asymptotic excess risk under random design is:

$$\mathbb{E}_\varepsilon \left[\mathcal{R}(\hat{\theta})\right] \sim \mathcal{R}^{(\mathrm{bias})}(\hat{\theta}) + \mathbb{E}_\varepsilon \left[\mathcal{R}^{(\mathrm{var})}(\hat{\theta})\right],$$

where:

$$\mathbb{E}_\varepsilon \left[\mathcal{R}^{(\mathrm{var})}(\hat{\theta})\right] \sim \frac{\sigma^2}{n} \mathrm{df}_2(\kappa(\lambda)) \cdot \frac{1}{1 - \frac{1}{n}\mathrm{df}_2(\kappa(\lambda))},$$

$$\mathcal{R}^{(\mathrm{bias})}(\hat{\theta}) \sim \kappa(\lambda)^2 \theta_*^\top \Sigma(\Sigma + \kappa(\lambda) I)^{-2} \theta_* \cdot \frac{1}{1 - \frac{1}{n}\mathrm{df}_2(\kappa(\lambda))}.$$

Here, $\mathrm{df}_2(\kappa) = \mathrm{tr}[\Sigma^2(\Sigma + \kappa I)^{-2}]$. The factor $(1 - \frac{1}{n}\mathrm{df}_2(\kappa(\lambda)))^{-1}$ amplifies the risk when $\mathrm{df}_2(\kappa(\lambda))$ approaches n, signaling overfitting.

3.2.5 Key Observations

- **Self-Induced Regularization:** The implicit parameter $\kappa(\lambda)$ enhances explicit regularization λ, particularly in over-parameterized regimes ($\gamma > 1$). For $\lambda \to 0$, $\kappa(0) > 0$ ensures stability even without explicit regularization.
- **Bias-Variance Trade-off:** The bias term depends on the alignment of θ_* with eigenvectors of Σ, while the variance term grows with $\mathrm{df}_2(\kappa(\lambda))$. When $\mathrm{df}_2(\kappa(\lambda)) \approx n$, variance explodes, reflecting the "peak" in double descent.
- **Isotropic Covariance Example:** If $\Sigma = \sigma I$, solving $\kappa(\lambda)$ explicitly gives:

$$\kappa(\lambda) = \frac{1}{2}\left(\lambda - \sigma(1-\gamma) + \sqrt{(\sigma(1-\gamma) - \lambda)^2 + 4\lambda\sigma}\right).$$

  This illustrates distinct behaviors for $\gamma < 1$ (under-parameterized) and $\gamma > 1$ (over-parameterized).

The ridge regression analysis reveals that high-dimensional settings induce implicit regularization through $\kappa(\lambda)$, modifying both bias and variance. This mechanism underpins the double descent curve: as model complexity increases, risk first rises due to overfitting ($m \to n$), then declines in the over-parameterized regime ($m > n$) as implicit regularization stabilizes predictions.

3.3 Self-Induced Regularization

Key References: Bach (2023)
Defining the Stieltjes Transform: The Stieltjes transform is a mathematical tool used to study the spectral distribution of a matrix. For a random matrix M with eigenvalue distribution μ, the Stieltjes transform at a point z is defined as:

$$\varphi(z) = \int \frac{1}{\lambda - z} d\mu(\lambda).$$

In the given context, the matrix M is $XX^\top$ (or $\frac{1}{n}XX^\top$), and $\phi(z)$ captures how the eigenvalues of $XX^\top$ are distributed in high dimensions.
The Key Equation: critical relationship arises in this context:

$$\kappa(\lambda) = \frac{1}{\varphi(-\lambda)},$$

where $\kappa(\lambda)$, termed the **self-induced regularization**, is the inverse of the Stieltjes transform evaluated at $-\lambda$. This equation highlights how the regularization effect in high dimensions is governed by the spectral properties of the matrix. 1. Resolvent Representation The resolvent of a matrix M, defined as $(M - zI)^{-1}$, is directly linked to the Stieltjes transform via the expression:

$$\varphi(z) = \frac{1}{d}\mathrm{tr}\left(\left(\frac{1}{n}XX^\top - zI\right)^{-1}\right).$$

As the dimensions $n, d \to \infty$, $\varphi(z)$ converges to the limiting transform of the population covariance under certain regularity conditions.
2. **Fixed-Point or Self-Consistency Condition**: In high dimensions Bach (2023), when analyzing the ridge inverse $\left(\frac{1}{n}X^\top X + \lambda I\right)^{-1}$, a self-consistency equation emerges for the trace of the resolvent. This leads to an expression for $\kappa(\lambda)$ such that:

$$\left(\frac{1}{n}X^\top X + \lambda I\right)^{-1} \simeq (\Sigma + \kappa(\lambda)I)^{-1},$$

where $\simeq$ denotes asymptotic equivalence.
3. **Identifying $\kappa(\lambda)$:** By comparing spectral properties, the relationship between $\kappa(\lambda)$ and the Stieltjes transform becomes:

$$\kappa(\lambda) = \frac{1}{\varphi(-\lambda)}.$$

This shows that $\kappa(\lambda)$ depends implicitly on the eigenvalue distribution of $XX^\top$, not just on λ.
Why $\kappa(\lambda)$ is "Self-Induced": Although the ridge penalty explicitly introduces λ, the high-dimensional behavior of $XX^\top$ magnifies or diminishes the regularization effect. As a result: - $\kappa(\lambda)$ can be larger or smaller than λ, depending on the eigenvalue spread of $XX^\top$. - This phenomenon is termed self-induced regularization because it arises naturally from the interaction between λ and the random matrix $XX^\top$, rather than being externally imposed. Conclusion The identity $\kappa(\lambda) = \frac{1}{\varphi(-\lambda)}$ illustrates how high-dimensional effects introduce an effective or self-induced regularization that extends beyond the nominal λ.

3.4 Analysis of Self-induced Regularization

Bach (2023) explores the behavior of the self-induced regularization term $\kappa(\lambda)$ under isotropic covariance assumptions. The formula for $\kappa(\lambda)$ is provided as:

$$\kappa(\lambda) = \frac{1}{2}\left(\lambda - \sigma(1-\gamma) + \sqrt{(\sigma(1-\gamma) - \lambda)^2 + 4\lambda\sigma}\right),$$

where σ and γ are parameters that capture properties of the covariance structure and dimensionality.

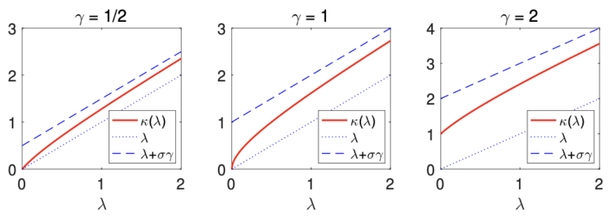


Figure 4: Bach (2023) Implicit regularization parameter $\kappa(\lambda)$ in the three regimes for isotropic covariance matrices, with $\sigma = 1$.

Observations from Plots: Three plots corresponding to different values of γ ($\gamma = \frac{1}{2}, 1, 2$): - The red curve represents $\kappa(\lambda)$. - The dotted blue curve represents λ, the nominal regularization parameter. - The dashed blue curve represents $\lambda + \sigma\gamma$, an adjusted baseline for comparison.
Across all cases, $\kappa(\lambda)$ is observed to be consistently greater than λ, indicating that the self-induced regularization amplifies the effect of the nominal λ. This amplification depends on the eigenvalue distribution and other parameters of the data matrix X.
Property of $\kappa(\lambda)$: It is always true that $\kappa(\lambda) > \lambda$, meaning that the self-induced regularization enhances the effective regularization beyond the value of the nominal penalty parameter λ.
These results highlight the impact of high-dimensional settings on regularization. Specifically, $\kappa(\lambda)$ incorporates interactions between the nominal regularization λ and the spectral properties of the covariance matrix, leading to a more complex behavior that surpasses the direct effect of λ alone.

3.5 Random projections

This section considers the use of a random projection matrix $S \in \mathbb{R}^{d\times m}$, sampled independently from X, under the assumptions (A5) and (A6)(Bach (2023).
For linear regression, we project each covariate $x \in \mathbb{R}^d$ to $S^\top x \in \mathbb{R}^m$. The projected estimator $\hat{\eta} \in \mathbb{R}^m$ minimizes $\|y - XS\hat{\eta}\|_2^2$. The solution is then mapped back to $\mathbb{R}^d$ as $\hat{\theta} = S\hat{\eta}$. This approach is distinct from applying the random projection directly to the response variable y and X, a method often referred to as **"sketching."**
The asymptotic performance of the random projection-based estimator is characterized under different regimes, assuming (A1)–(A6)(Bach (2023)):
Under-Parameterized Regime ($m < n$):
Variance Term:

$$\mathbb{E}[\mathcal{R}^{(\mathrm{var})}(\hat{\theta})] \sim \frac{\sigma^2 m}{n-m} \cdot \frac{1}{1-\frac{m}{n}} = \frac{\sigma^2 m}{n-m}.$$

Bias Term:

$$\mathcal{R}^{(\mathrm{bias})}(\hat{\theta}) \sim \frac{1}{1-\frac{m}{n}} \kappa_m \theta_*^\top \Sigma(\Sigma + \kappa_m I)^{-1}\theta_*.$$

Here, κ_m is defined such that $\mathrm{df}_1(\kappa_m) \sim m$.
Over-Parameterized Regime ($m \geq n$): Variance Term:

$$\mathbb{E}[\mathcal{R}^{(\mathrm{var})}(\hat{\theta})] \sim \frac{\sigma^2}{n} \cdot \mathrm{df}_2(\kappa_n) \cdot \frac{1}{1-\frac{1}{n}\mathrm{df}_2(\kappa_n)} + \frac{\sigma^2 m}{n-m}.$$

Bias Term:

$$\mathcal{R}^{(\mathrm{bias})}(\hat{\theta}) \sim \frac{1}{1-\frac{1}{n}\mathrm{df}_2(\kappa_n)} \kappa_n \theta_*^\top \Sigma(\Sigma + \kappa_n I)^{-2}\theta_* + \frac{n}{m-n}.$$

Under-Parameterized Regime ($m < n$):

- In this regime, the traditional bias and variance terms are scaled by the factor $1 - \frac{m}{n}$, leading to a potential for catastrophic overfitting when m approaches n.

- Specifically, as m increases from 0 to n, the variance term increases, while the bias term exhibits both a decreasing component $\kappa_m \theta_*^\top \Sigma(\Sigma + \kappa_m I)^{-1}\theta_*$ and an increasing component $(1 - \frac{m}{n})^{-1}$.
- In certain cases, such as when θ_* and Σ are isotropic, the overall performance improves consistently. However, in many practical scenarios, the traditional U-shaped curve of bias-variance tradeoff is observed in the under-parameterized regime.

Over-Parameterized Regime ($m \geq n$):

- In this regime, as $m \to \infty$, the behavior aligns with the limit of ridge regression as the regularization parameter $\lambda \to 0$. Here, κ_n converges to the value denoted as $\kappa(0)$.
- Both bias and variance terms decrease monotonically as m increases. This suggests that, in the over-parameterized regime, increasing m improves performance, making it optimal to take m as large as possible.
- Notably, to achieve the performance at $m = \infty$, it suffices to solve $\hat{\theta} = X^\top (XX^\top)^{-1} y$, without needing to explicitly work in the high-dimensional space m.

Double Descent Phenomenon: Combining the under-parameterized and over-parameterized regimes, the analysis reveals the emergence of a **double descent curve** in many settings.
This analysis highlights the impact of random projections on the variance and bias of estimators, particularly in high-dimensional settings.

4 Experiments and Illustrations

4.0.1 Illustration of the Double Descent Phenomenon

Key Reference: Bach (2023)
This section examines the double descent phenomenon using a fixed covariance matrix Σ of size d. The eigenvectors of Σ are uniformly random, and the eigenvalues are either proportional to $1/k$ for $k \in \{1, \dots, d\}$ (non-isotropic) or constant (isotropic). The matrix Σ is normalized such that $\mathrm{tr}(\Sigma) = 1$. A vector $\theta_* \in \mathbb{R}^d$ is sampled from a standard Gaussian distribution and then normalized to satisfy $\theta_*^\top \Sigma \theta_* = 1$.
Experimental Setup for the prediction problem:

- Forty replications of Z and S are generated using Rademacher random variables.
- Empirical performance is plotted for both the bias and variance.
- The bounds are computed using $\kappa_m^{-1} = \mathbb{E}\left[\mathrm{tr}((S^\top \Sigma S)^{-1})\right]$.

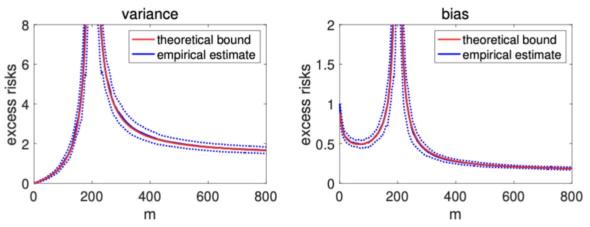


Figure 5: Bach (2023)

Non-Isotropic Covariance Matrix (Figure 5): The results show a U-shaped curve for the bias term, consistent with the traditional double descent behavior. The U-shape emerges as the number of projections (m) varies relative to the sample size (n).

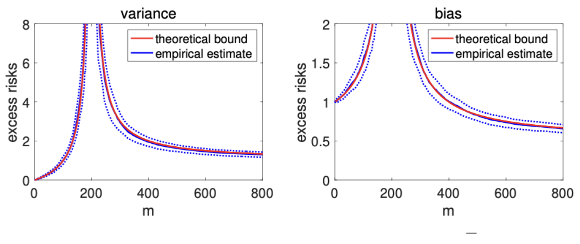


Figure 6: Bach (2023)

Isotropic Covariance Matrix (Figure 6): For the isotropic case, no U-shaped curve is observed for the bias term. Consequently, when combining bias and variance, there is no U-shaped behavior in this setting.
The experimental results align closely with the asymptotic limits derived in Bach (2023), demonstrating a strong match between theoretical predictions and empirical observations in both isotropic and non-isotropic cases.

5 Unifying Grokking and Double Descent

Reference: Davies et al. (2023), *Unifying Grokking and Double Descent.*

5.1 Overview and Motivation

Grokking refers to a surprisingly delayed generalization phenomenon where a model (often a small transformer on a modular arithmetic task) maintains high training accuracy but very low test accuracy for many epochs, only to "suddenly" generalize perfectly after much more training. Double descent, on the other hand, often describes test performance that first improves, then worsens (overfitting), and finally improves once again as model complexity or training progresses. Despite seeming different, Davies et al. (2023) posits that *both* grokking and double descent can emerge from the *same* underlying training dynamics.

5.2 Key Claims

Claim 1. Grokking, like epoch-wise double descent, arises when there is a *slowly learned* pattern that generalizes well yet is preceded by *faster, poorly generalizing* patterns. Early in training, the model relies on these simpler or more direct patterns (often memorization) that fit the training set but do not generalize. Over long training, a better pattern eventually *overtakes* the memorization-based solutions, producing the abrupt jump in test accuracy typical of grokking.
Claim 2. Delayed generalization can also happen as a function of *model size*, not just training epochs. In other words, adding capacity (e.g. more embedding dimensions in a transformer) can replicate the same "descent-overfit-rediscent" phenomenon that appears over training steps. The authors show "model-wise grokking" by moving horizontally along a plot of model size (instead of time) and observing delayed generalization.

5.3 Model of Pattern Learning

To unify the two phenomena, Davies et al. (2023) sketches a *pattern learning* viewpoint:

- Assume each model essentially learns different "patterns" or solution strategies at different speeds, each pattern having a certain *maximum predictiveness* (how many data points it can correctly classify) and a *generalization fraction* (its performance on unseen data).

- Patterns that are quick to learn but do not generalize (like memorization) will dominate early in training, driving test accuracy down if those patterns conflict with other parts of the data distribution.
- Patterns that are slow to learn but *do* generalize well eventually appear, causing a late improvement in test accuracy.

By tuning the strengths and learning rates of these patterns, one can produce either a double descent (where intermediate solutions degrade test accuracy before improving again) *or* a stark grokking effect (where test accuracy is stuck near chance and then abruptly jumps to perfect).

5.4 Epoch-wise vs. Model-wise Perspective

- **Epoch-wise:** Increasing training steps at fixed model size can lead from underfitting (test accuracy not too high) to overfitting (test accuracy drops) to a second improvement (double descent). Or, in a more extreme version, to grokking, where test accuracy languishes at chance levels for a long period before an abrupt rise.
- **Model-wise:** The paper shows the same sort of behavior when one keeps training steps fixed but increases model dimensionality. A neural net might lack capacity at small widths (underfit), then overfit as it gains more capacity, and finally discover a well-generalizing solution at some larger size, revealing a second improvement in test performance.

5.5 Empirical Findings

Transformer on Modular Division. Extending the grokking setup of Power et al. (2022), the authors use small decoder-only transformers trained on modular arithmetic tasks. By plotting test and training losses across both epochs (vertical axis) and model embedding dimension (horizontal axis), they show:

1. A "delayed generalization" (the hallmark of grokking) over long training at *fixed* dimension (vertical slices).
2. A similar delayed effect when moving horizontally (increasing dimension at a fixed number of training steps), thus illustrating *model-wise grokking*.

This confirms that slow-emerging patterns—improving generalization late—are a consistent mechanism for both grokking and double descent.

5.6 Implications for Understanding Generalization

- **Unified Framework:** Grokking and double descent need not be seen as distinct: the authors' pattern-based model captures both by mixing fast but poor-generalizing patterns with slow but strong-generalizing ones.
- **Practical Tuning:** The results suggest that *when* a model discovers better patterns can depend heavily on training durations, learning rates, weight decay, and model size. Interactions among these may cause abrupt performance jumps.
- **Safety Considerations:** From an AI safety perspective, the paper highlights that we cannot always predict when a model will switch from memorization to truly generalizing. More robust theories of how and when networks shift to better patterns could be crucial for controlling or auditing model behavior.

By proposing that "grokking" is essentially an *extreme* manifestation of the same type of delayed transition that underpins double descent, this paper helps unify two apparently distinct training phenomena. It shows that slow-learning but strong-generalizing patterns can take over either *late in training* or at *higher model capacity*, creating a sharp improvement in test accuracy that was initially blocked by simpler, poorly generalizing patterns.

6 Application to Neural Networks

6.1 Optimization under Interpolation

Bassily et al. (2018) contrasts two regimes of model fitting—“classical” under-parameterized training versus “modern” interpolation—by focusing on how optimization behaves in each case. In the classical (under-parameterized) regime, there are typically many local minima, and using stochastic gradient descent (SGD) with a fixed step size can fail to converge reliably. Intuitively, when the model does not have enough parameters to capture the entire training set perfectly, the loss landscape is rugged, and it is easy to get stuck in suboptimal solutions.
By contrast, in the modern interpolation (often over-parameterized) regime, the model has enough capacity to fit the training data exactly, making “every local minimum” effectively a global minimum. Recent theoretical work, Kawaguchi et al. (2022), Levine and Feizi (2019), Soltanolkotabi et al. (2022), Heckel and Soltanolkotabi (2016), Xu and Du (2023), Ma et al. (2018) shows that in these highly parameterized settings the loss surface is surprisingly well-behaved, and SGD can converge as fast as full gradient descent, even when using a fixed step size. This shift in perspective—from under-parameterized models with complicated optimization landscapes to over-parameterized models where interpolation is possible—helps explain phenomena like “double descent.” As a model transitions from under- to over-parameterized, its error on a test set may first decrease, then worsen (as overfitting takes hold), and finally improve again once the model has enough parameters to reach the interpolation regime and leverage the favorable optimization landscape.

6.2 Why SGD?

Stochastic Gradient Descent (SGD) addresses the full optimization problem,

$$\min_w L(w) = \frac{1}{n}\sum_i L_i(w)$$

by only sampling a small batch of terms (losses) at a time. Instead of computing the gradient using the entire dataset (as in full Gradient Descent), it updates the parameters more frequently using a noisy but far cheaper approximation to the true gradient.
In practice, this not only reduces computational cost—since you do not have to wait to sum over all n training examples—but also provides beneficial “noise” that can help the optimizer escape sharp minima. Though theoretically its convergence rate (on paper) appears slower than that of batch Gradient Descent (it shrinks the error like $1/t$ rather than e^{-t}), in practice it scales very well with large datasets and high-dimensional parameter spaces.
All major neural network frameworks use SGD and its variants because it is a good balance of efficiency, memory usage, and an implicit regularization effect that tends to prevent overfitting in modern deep learning models.

6.3 Exponential convergence of m-SGD

Bassily et al. (2018), presents a theoretical result showing that mini-batch SGD (“m-SGD”) can converge to the optimal parameter exponentially fast in the interpolation regime, provided the loss function satisfies certain smoothness and strong convexity conditions.
Theorem: exponential convergence of m–SGD in interpolation regime

$$EL(w_{t+1}) \leq \frac{\lambda}{2}(1-\eta^*(m)\alpha)^t \|w_1 - w^*\| \tag{2}$$

$$\eta^*(m) = \frac{m}{\beta + \lambda(m-1)} \tag{3}$$

The theorem states that with a suitable choice of step size $\eta^*(m)$, the expected loss after t iterations decays like

$$(1-\eta^*(m)\,\alpha)^t,$$

which is an exponential rate. In other words, each iteration multiplies the distance to the optimum by a constant factor smaller than 1, so the model converges quickly.
This result generalizes analyses (like the case $m = 1$) to mini-batch settings, highlighting the efficiency of small-batch updates for convex problems when the data can be perfectly fit ("interpolation").

6.4 Minibatch size?

Ma et al. (2018), highlights that there is an "optimal" mini-batch size m^* that balances the computational efficiency of batching with how effectively each update step reduces the loss. If the batch size m is smaller than m^*, then one iteration of m-SGD essentially achieves the same progress as m iterations of (batch size 1) SGD—leading to a near-linear speedup. Past this critical threshold m^*, the benefits of larger batches "saturate," so each iteration is not much better than a single step of full-batch gradient descent. Increasing m beyond m^* can thus waste computation without corresponding gains in convergence speed. A key insight is that this m^* (derived from β/λ in Ma et al. (2018)) depends more on the optimization properties (smoothness and strong convexity) than on the particular dataset, and choosing it wisely can yield a significant reduction in total training time.

6.5 SGD is (much) faster than GD

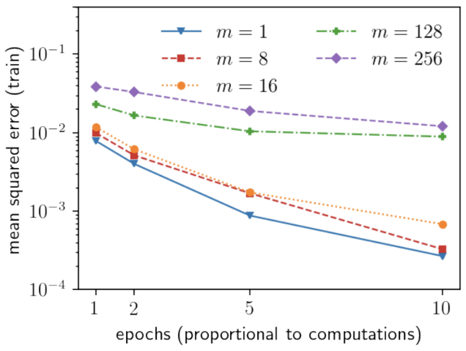


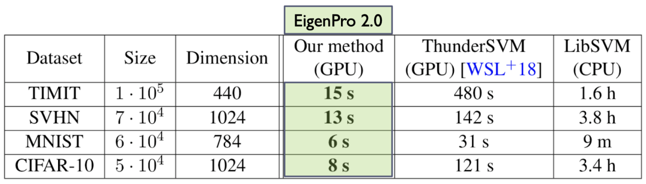
Figure 7: Ma et al. (2018)

Ma et al. (2018), shows an empirical result on a real dataset, demonstrating that using a mini-batch size around $m^* = 8$ can yield similar per-step improvements as full-batch Gradient Descent (GD), but at a fraction of the computational cost. Figure 7 plots the mean squared training error over epochs (on a log scale), comparing different batch sizes m. We see that $m = 8$ (the red dashed curve) achieves nearly the same slope in error reduction per "unit" of computation as the blue curve for $m = 1$ (stochastic updates) or the not-shown full-batch GD. This nicely illustrates how a well-chosen mini-batch size can approach the convergence rate of GD while being much more efficient to run, making SGD "much faster" in practical settings.

6.6 The power of interpolation

Modern deep learning pipelines leverage over-parameterized models (allowing interpolation), efficient mini-batch SGD, and specialized parallel hardware such as GPUs—all to achieve massive speedups over classical, full-batch gradient descent on CPUs. By choosing a "sweet spot" batch size m^*, SGD reduces the per-iteration cost from $\mathcal{O}(n)$ to $\mathcal{O}(n/m^*)$, and further taking advantage of GPU acceleration yields an additional factor of around 100.
As a result, on a problem with $n = 10^6$ and $m^* = 8$, SGD on a GPU can be roughly 10^7 times faster than running full-batch gradient descent on a CPU. This scalability is a key reason why today's deep learning systems can train enormous models on large datasets.

6.7 Learning from deep learning: fast and effective kernal machines



Dataset	Size	Dimension	EigenPro 2.0 Our method (GPU)	ThunderSVM (GPU) [WSL+18]	LibSVM (CPU)
TIMIT	$1 \cdot 10^5$	440	**15 s**	480 s	1.6 h
SVHN	$7 \cdot 10^4$	1024	**13 s**	142 s	3.8 h
MNIST	$6 \cdot 10^4$	784	**6 s**	31 s	9 m
CIFAR-10	$5 \cdot 10^4$	1024	**8 s**	121 s	3.4 h

Figure 8: Belkin (2019)

Hui et al. (2018) demonstrates an efficient kernel-based learning method called **EigenPro 2.0**, which applies insights from deep learning (such as mini-batch updates) to classical kernel machines. The table compares the training times of EigenPro on a GPU against those of ThunderSVM (GPU) and LibSVM (CPU) on standard benchmark datasets (TIMIT, SVHN, MNIST, CIFAR-10). EigenPro shows dramatically faster training—on the order of seconds—while more conventional kernel methods can take minutes or hours to process datasets of similar size.
Furthermore, EigenPro does not require extensive hyperparameter tuning, which simplifies its use. These results show that by combining the strengths of GPU acceleration and modern optimization approaches, kernel methods can be made both fast and effective for large-scale learning problems.

6.8 From classical statistics to modern ML

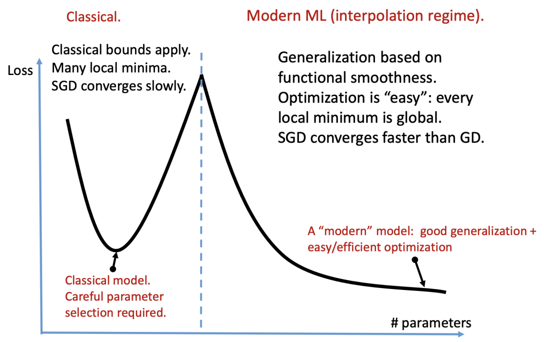

Classical.
Modern ML (interpolation regime).
Classical bounds apply.
Many local minima.
SGD converges slowly.
Loss
Generalization based on functional smoothness.
Optimization is "easy": every local minimum is global.
SGD converges faster than GD.
A "modern" model: good generalization + easy/efficient optimization
Classical model.
Careful parameter selection required.
# parameters

Figure 9: Belkin (2019)

Figure 9 contrasts the "classical" under-parameterized setting against the "modern" over-parameterized (interpolation) regime. On the left, classical models have fewer parameters than data points can fully exploit, and they can get stuck in many local minima. In this region, careful parameter tuning and classical statistical bounds are critical, and stochastic gradient descent tends to converge more slowly. Moving past the vertical dotted line into the over-parameterized regime, the model can perfectly fit (interpolate) the training data. Surprisingly, this often makes optimization "easier," because the loss surface no longer has problematic local minima—every local minimum is essentially a global optimum. Consequently, modern deep networks not only fit very flexibly but also allow SGD to converge quickly, sometimes even faster than full gradient descent in practice.

7 Conclusion

In this report, we explore the intriguing phenomenon of double descent, beginning with its introduction and subsequent observation within the framework of linear regression. Although initial observations effectively described the occurrence of double descent, they failed to provide a mathematical explanation for its emergence. To bridge this gap, we delved into the insights presented in Francis Bach's paper, "High-dimensional analysis of double descent for linear regression with random projections." This study illuminated the role of self-induced regularization and the impact of random design on ridge regression in high-dimensional settings. It became evident that the self-induced bias in over-parameterized regimes contributes to a reduction in test error, thereby explaining the double descent phenomenon.
Furthering our understanding, we examined random projections and reformulated the random design in the context of high-dimensional assumptions. The results mirrored those of ridge regression analysis, reinforcing the underlying principles driving double descent. To validate our theoretical findings, we conducted experiments based on the equations formulated for random projections, illustrating the phenomenon of double descent in practice.
Building on these results, we further explored the unification of grokking and double descent, two phenomena that challenge classical intuitions about generalization. Just as double descent illustrates how increasing model capacity can resolve over-fitting through implicit regularization, grokking reflects a delayed but analogous emergence of generalization after prolonged training. This connection suggests that both phenomena may arise from a shared principle: the interplay between model complexity, optimization dynamics, and high-dimensional data geometry.
Finally, we briefly touched upon the applications of double descent in neural networks, highlighting the power of interpolation, the superior speed of Stochastic Gradient Descent (SGD) compared to Gradient Descent (GD), and the effectiveness of kernel machines. These applications underscore the broader relevance of double descent in modern machine learning paradigms.
Overall, this report not only elucidates the double descent phenomenon in linear regression through both theoretical analysis and experimental validation, but also connects these insights to practical applications in neural networks. By bridging empirical observations with mathematical explanations, we contribute to a deeper understanding of model behavior in high-dimensional and over-parameterized settings.

References

Bach, F. (2023). High-dimensional analysis of double descent for linear regression with random projections.
URL: *https://arxiv.org/abs/2303.01372*

Bassily, R., Belkin, M. and Ma, S. (2018). On exponential convergence of sgd in non-convex overparametrized learning.
URL: *https://arxiv.org/abs/1811.02564*

Belkin, M. (2019). *From classical statistics to modern machine learning*, Simons Institute: Frontiers of Deep Learning.
URL: *https://simons.berkeley.edu/sites/default/files/docs/14172/slides1.pdf*

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* .

Belkin, M., Hsu, D. and Xu, J. (2020). Two models of double descent for weak features, *SIAM Journal on Mathematics of Data Science* **2**(4): 1167–1180.
URL: *http://dx.doi.org/10.1137/20M1336072*

Davies, X., Langosco, L. and Krueger, D. (2023). Unifying grokking and double descent.
URL: *https://arxiv.org/abs/2303.06173*

Dr. Roland Speicher, D. J. H. (2023). *High-Dimensional Analysis: Random Matrices and Machine Learning*, Saarland UniversityFaculty of Mathematics and Computer Science Department of Mathematics.

Heckel, R. and Soltanolkotabi, M. (2016). Generalized line spectral estimation via convex optimization.
URL: *https://arxiv.org/abs/1609.08198*

Hui, L., Ma, S. and Belkin, M. (2018). Kernel machines beat deep neural networks on mask-based single-channel speech enhancement.
URL: *https://arxiv.org/abs/1811.02095*

Kawaguchi, K., Bengio, Y. and Kaelbling, L. (2022). *Generalization in Deep Learning*, Cambridge University Press, p. 112–148.
URL: *http://dx.doi.org/10.1017/9781009025096.003*

Levine, A. and Feizi, S. (2019). Robustness certificates for sparse adversarial attacks by randomized ablation.
URL: *https://arxiv.org/abs/1911.09272*

Ma, S., Bassily, R. and Belkin, M. (2018). The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning.
URL: *https://arxiv.org/abs/1712.06559*

Soltanolkotabi, M., Javanmard, A. and Lee, J. D. (2022). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks.
URL: *https://arxiv.org/abs/1707.04926*

Xu, W. and Du, S. S. (2023). Over-parameterization exponentially slows down gradient descent for learning a single neuron.
URL: *https://arxiv.org/abs/2302.10034*

Declaration of authorship

Munich, February 24, 2025
Location, date

Sahibnoor Singh

Name