Double Descent Demystified

A Comprehensive Tutorial with Mathematical Derivations and Absurd Examples

Tutorial based on ICLR 2024 Blog Post

December 18, 2025

Abstract

This tutorial provides a comprehensive mathematical treatment of the double descent phenomenon in machine learning. We focus on ordinary linear regression and use singular value decomposition (SVD) to reveal three interpretable factors that cause test loss to spike at the interpolation threshold. The exposition includes detailed derivations, geometric intuition, and intentionally absurd examples to make the mathematics memorable and fun.

Contents

1 Introduction: The Mystery of Double Descent

Imagine you're training a model. Classical wisdom says: too simple → underfit, too complex → overfit. The sweet spot is somewhere in the middle. This is the famous bias-variance tradeoff, giving us a U-shaped test error curve.

But wait! Modern deep learning throws a wrench in this story. Make your model even MORE complex (more parameters than data points), and suddenly... the test error drops again! This creates a "double descent" curve:

$$\text{Test Error} = \underbrace{\text{U-shaped}}_{\text{Classical}} + \underbrace{\text{Another descent}}_{\text{Modern ML}} \tag{1}$$

Funny Example

The Pizza Analogy:
You're trying to predict pizza delivery times based on distance and traffic.

- **Underparameterized (2 data points, 1 parameter):** You can only capture "more distance = more time". Your model is too simple. It's like trying to explain pizza delivery with just "distance matters" – you miss traffic, weather, if the delivery guy stopped to pet a dog, etc.
- **At interpolation threshold (2 data points, 2 parameters):** NOW you have exactly as many knobs as data points. Your model memorizes: "123 Main St took 15 min, 456 Elm took 25 min." But what about 789 Oak St? Your model freaks out because it's trying to extrapolate from a knife's edge of information. **This is where things explode!**
- **Overparameterized (2 data points, 100 parameters):** Wait, more parameters should be worse, right? WRONG! With many parameters, your model can't just memorize specific addresses. It's forced to find patterns like "oh, eastern addresses tend to take longer." It learns useful features!

2 Mathematical Setup

2.1 Notation and Problem Definition

We have N training samples with features $\vec{x}_n \in \mathbb{R}^D$ and targets $y_n \in \mathbb{R}$.

Matrix notation:

$$X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{2}$$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{N \times 1} \tag{3}$$

Goal: Learn $\hat{\vec{\beta}} \in \mathbb{R}^D$ such that:

$$y_n \approx \vec{x}_n \cdot \hat{\vec{\beta}} \tag{4}$$

Key parameters:

- $P = D$: Number of parameters (in linear regression, $P = D$)
- N: Number of training samples
- Interpolation threshold: $N = P = D$
- Underparameterized: $N > P$
- Overparameterized: $N < P$

Funny Example

The Dating Profile Example:
You're trying to predict compatibility scores based on profile features.

- $D = 3$ features: [loves_dogs, reads_books, likes_hiking]
- $N = 5$ past dates with scores
- $P = 3$ parameters to learn (one weight per feature)

Since $N = 5 > P = 3$, you're **underparameterized**. You have more data than parameters – life is good!
But if you only went on $N = 2$ dates and try to learn $P = 3$ weights... that's **overparameterized**. You're trying to learn 3 things from 2 examples. At $N = 3$, you're exactly at the interpolation threshold – the danger zone!

2.2 The Two Regimes

2.2.1 Underparameterized Regime ($N > P$)

More data than parameters. We solve:

$$\hat{\vec{\beta}}_{\text{under}} = \arg\min_{\vec{\beta}} \|X\vec{\beta} - Y\|_2^2 \tag{5}$$

Solution (Ordinary Least Squares):

Key Equation

$$\hat{\vec{\beta}}_{\text{under}} = (X^T X)^{-1} X^T Y \tag{6}$$

This uses the **second moment matrix** $X^T X \in \mathbb{R}^{D \times D}$.

Derivation:

$$\frac{\partial}{\partial \vec{\beta}} \|X\vec{\beta} - Y\|_2^2 = \frac{\partial}{\partial \vec{\beta}} (X\vec{\beta} - Y)^T (X\vec{\beta} - Y) \tag{7}$$

$$= \frac{\partial}{\partial \vec{\beta}} (\vec{\beta}^T X^T X\vec{\beta} - 2Y^T X\vec{\beta} + Y^T Y) \tag{8}$$

$$= 2X^T X\vec{\beta} - 2X^T Y = 0 \tag{9}$$

$$\Rightarrow \vec{\beta} = (X^T X)^{-1} X^T Y \tag{10}$$

2.2.2 Overparameterized Regime ($N < P$)

More parameters than data! The problem is **ill-posed** (infinitely many solutions). We pick the **minimum norm** solution:

$$\hat{\vec{\beta}}_{\text{over}} = \arg\min_{\vec{\beta}} \|\vec{\beta}\|_2^2 \quad \text{subject to} \quad X\vec{\beta} = Y \tag{11}$$

Why minimum norm? This is what gradient descent implicitly finds!
Solution (using Lagrangian):

$$\mathcal{L}(\vec{\beta}, \vec{\lambda}) = \frac{1}{2}\|\vec{\beta}\|_2^2 + \vec{\lambda}^T (Y - X\vec{\beta}) \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{\beta}} = \vec{\beta} - X^T\vec{\lambda} = 0 \Rightarrow \vec{\beta} = X^T\vec{\lambda} \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{\lambda}} = Y - X\vec{\beta} = 0 \Rightarrow Y = XX^T\vec{\lambda} \tag{14}$$

Key Equation

$$\hat{\vec{\beta}}_{\text{over}} = X^T (XX^T)^{-1} Y \tag{15}$$

This uses the **Gram matrix** $XX^T \in \mathbb{R}^{N \times N}$.

Funny Example

The Cooking Recipe Analogy:
Underparameterized: You've cooked 100 meals ($N = 100$) and want to learn the effect of 3 ingredients ($P = 3$): salt, sugar, spice. You have way more data than parameters – you can clearly see "add more salt → tastier food".
Overparameterized: You've only cooked 3 meals ($N = 3$) but trying to learn 100 ingredient weights ($P = 100$). Yikes! But here's the trick: you pick the *simplest* recipe (minimum norm) that explains your 3 meals. Maybe "just use salt, ignore the other 99 ingredients." This prevents overfitting!

3 Singular Value Decomposition (SVD): The Magic Lens

3.1 SVD Basics

Every matrix $X \in \mathbb{R}^{N \times D}$ can be decomposed:

$$X = USV^T \tag{16}$$

where:

- $U \in \mathbb{R}^{N \times R}$: Left singular vectors (orthonormal)
- $S \in \mathbb{R}^{R \times R}$: Diagonal matrix with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > 0$
- $V \in \mathbb{R}^{D \times R}$: Right singular vectors (orthonormal)
- $R = \text{rank}(X) \leq \min(N, D)$

Key properties:

$$X^T X = VS^2V^T \quad \text{(eigendecomposition of second moment matrix)} \tag{17}$$
$$XX^T = US^2U^T \quad \text{(eigendecomposition of Gram matrix)} \tag{18}$$

Mathematical Insight

Why SVD is powerful for understanding double descent:
The SVD reveals the *directions of variation* in your data:

- σ_1 (largest): Direction with most variance
- σ_R (smallest): Direction with least variance
- At interpolation threshold, $\sigma_R \approx 0^+$ (tiny but non-zero!)

The reciprocals $1/\sigma_r$ appear in our formulas, so small $\sigma_r \to$ HUGE $1/\sigma_r \to$ DIVERGENCE!

3.2 Moore-Penrose Pseudoinverse

For a diagonal matrix S with entries $\sigma_1, \ldots, \sigma_R$:

$$S^+ = \text{diag}\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_R}\right) \tag{19}$$

If $\sigma_r > 0$, then $(S^+)_{rr} = 1/\sigma_r$. If $\sigma_r = 0$, then $(S^+)_{rr} = 0$.

Funny Example

The Compass Analogy:
Imagine you're navigating with a compass that measures direction strength:

- **North:** Strong signal ($\sigma_1 = 10$)
- **East:** Weak signal ($\sigma_2 = 0.1$)
- **South:** Super weak ($\sigma_3 = 0.001$)

To "invert" your journey, you need to divide by these signals:

- North: $1/10 = 0.1$ (easy!)
- East: $1/0.1 = 10$ (getting risky...)
- South: $1/0.001 = 1000$ (**BOOM! Explodes!**)

Small singular values $\rightarrow$ huge inverses $\rightarrow$ numerical chaos!

4 The Critical Equation: Why Double Descent Happens

4.1 Decomposing the Target

The true relationship between features and targets:

$$Y = X\vec{\beta}^* + E \tag{20}$$

where:

- $\vec{\beta}^* \in \mathbb{R}^D$: The ideal linear parameters
- $E \in \mathbb{R}^{N\times 1}$: Residual errors (noise, model misspecification, missing features)

4.2 Predictions in Both Regimes

Using SVD $X = USV^T$:

Underparameterized prediction:

$$\begin{aligned}
\hat{y}_{\text{test}}^{\text{under}} &= \vec{x}_{\text{test}}^T (X^T X)^{-1} X^T Y && (21)\\
&= \vec{x}_{\text{test}}^T V S^{-2} V^T V S U^T (X\vec{\beta}^* + E) && (22)\\
&= \vec{x}_{\text{test}}^T V S^{-1} U^T (X\vec{\beta}^* + E) && (23)
\end{aligned}$$

Overparameterized prediction:

$$\begin{aligned}
\hat{y}_{\text{test}}^{\text{over}} &= \vec{x}_{\text{test}}^T X^T (XX^T)^{-1} Y && (24)\\
&= \vec{x}_{\text{test}}^T V S U^T (U S U^T)^{-1} (X\vec{\beta}^* + E) && (25)\\
&= \vec{x}_{\text{test}}^T V S U^T U S^{-2} U^T (X\vec{\beta}^* + E) && (26)\\
&= \vec{x}_{\text{test}}^T V S^{-1} U^T (X\vec{\beta}^* + E) && (27)
\end{aligned}$$

Mathematical Insight

Key Insight: Both regimes share a common term!

$$\text{Common term} = \vec{x}_{\text{test}}^T V S^{-1} U^T E \tag{28}$$

This term causes the divergence at the interpolation threshold.

4.3 The Prediction Error

Ideal prediction: $y_{\text{test}}^* = \vec{x}_{\text{test}} \cdot \vec{\beta}^*$

Underparameterized error:

Key Equation

$$\hat{y}_{\text{test}}^{\text{under}} - y_{\text{test}}^* = \underbrace{\vec{x}_{\text{test}}^T V S^{-1} U^T E}_{\text{DIVERGENCE TERM}} = \sum_{r=1}^{R} \frac{1}{\sigma_r} (\vec{x}_{\text{test}} \cdot \vec{v}_r)(\vec{u}_r \cdot E) \tag{29}$$

Overparameterized error:

Key Equation

$$\hat{y}_{\text{test}}^{\text{over}} - y_{\text{test}}^* = \underbrace{\vec{x}_{\text{test}}^T V S^{-1} U^T E}_{\text{SAME DIVERGENCE}} + \underbrace{\vec{x}_{\text{test}}^T (I - VV^T)\vec{\beta}^*}_{\text{BIAS TERM}} \tag{30}$$

5 The Three Factors of Double Descent

The divergence term reveals three factors that must *all* be present:

Key Equation

$$\boxed{\text{Divergence} = \sum_{r=1}^{R} \underbrace{\frac{1}{\sigma_r}}_{\text{Factor 1}} \cdot \underbrace{(\vec{x}_{\text{test}} \cdot \vec{v}_r)}_{\text{Factor 2}} \cdot \underbrace{(\vec{u}_r \cdot E)}_{\text{Factor 3}}} \tag{31}$$

5.1 Factor 1: Small Singular Values in Training Features

What it means: The training data X has some directions with very little variance.

Why it matters: Small $\sigma_r \to$ huge $1/\sigma_r \to$ amplification of errors!

When it occurs: Near the interpolation threshold! As $N \to D$, the last few singular values become tiny.

Funny Example

The Pancake Analogy:
You're measuring pancake quality with 3 features: [thickness, diameter, fluffiness].

- **1 pancake:** Variance exists in only 1 direction (say, this pancake is thick). $\sigma_1 > 0$, but $\sigma_2 = \sigma_3 = 0$.
- **2 pancakes:** Second pancake adds a second direction (this one's also fluffy). But it's probably also a bit thick, so the second direction has *less* variance than the first. $\sigma_1 > \sigma_2 > 0$, $\sigma_3 = 0$.
- **3 pancakes (interpolation threshold!):** Third pancake adds diameter variation, but it's probably also thick and fluffy. The third direction has *even less* variance. σ_3 is tiny! This is the danger zone.
- **4+ pancakes:** Now each dimension gets more samples, σ_3 grows, and we're safe.

Mathematical reason: Consider N data points $\vec{x}_1, \ldots, \vec{x}_N$ in $\mathbb{R}^D$. They span an R-dimensional subspace where $R = \min(N, D)$. Near $N = D$:

- First datum: 1 direction, $\sigma_1 \sim O(1)$
- Second datum: likely overlaps with first, $\sigma_2 < \sigma_1$
- ...
- N-th datum: likely overlaps with all previous, $\sigma_N \ll \sigma_1$

5.2 Factor 2: Test Features in Training Feature Subspace

What it means: The test point $\vec{x}_{\text{test}}$ has a large projection onto the weak directions $\vec{v}_r$ (the ones with small σ_r).

Why it matters: If $\vec{x}_{\text{test}} \cdot \vec{v}_r$ is large and σ_r is small, the error explodes!

Geometric picture: The model must extrapolate along directions where it has little information.

Funny Example

The GPS Analogy:
Your GPS training data has great coverage of North-South roads (σ_1 large) but terrible coverage of East-West roads (σ_2 tiny).
If your test point is on a North-South road: $\vec{x}_{\text{test}} \cdot \vec{v}_1$ is large, $\vec{x}_{\text{test}} \cdot \vec{v}_2$ is small. You're fine!
If your test point is on an East-West road: $\vec{x}_{\text{test}} \cdot \vec{v}_2$ is large, and you have to extrapolate along a direction you barely trained on. **Disaster!**
Even worse: your test point is on a diagonal road (both components large). You're extrapolating in EVERY poorly-covered direction. **Maximum chaos!**

5.3 Factor 3: Residual Errors from Best Possible Model

What it means: Even the ideal model $\vec{\beta}^*$ makes errors $E = Y - X\vec{\beta}^*$ that project onto $\vec{u}_r$.

Why it matters: These errors get amplified by $1/\sigma_r$ and leak into the predictions!

Sources of E:

1. **Label noise:** $y_n = \vec{x}_n \cdot \vec{\beta}^* + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
2. **Model misspecification:** True relation is nonlinear but we use linear model
3. **Missing features:** $y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ but we only observe x_1, x_2

Funny Example

The Fortune Teller Analogy:
You're predicting lottery numbers (good luck!).

- **No noise case:** If lottery numbers were deterministic (say, always [1,2,3,4,5]), the "best possible model" would be perfect. $E = 0 \to$ no divergence!
- **Noisy case:** Lottery numbers are random. Even the best model makes errors. Those errors E exist in every direction, including the weak directions with small σ_r. When you try to invert $(1/\sigma_r)$, you massively amplify the noise. **Boom!**
- **Model misspecification:** True lottery formula is "multiply yesterday's numbers by π then take modulo 100" (nonlinear!), but you use a linear model. You'll have residual errors E that can't be eliminated.

Absurd example: You're trying to predict whether it will rain by measuring ice cream sales. The "best linear model" is terrible because ice cream sales don't cause rain (correlation $\neq$ causation). Huge residuals $E \to$ double descent apocalypse!

6 Why the Interpolation Threshold?

6.1 Evolution of Smallest Singular Value

As we increase N from 1 to D and beyond:

$$\sigma_{\min}(N) = \begin{cases} \text{Doesn't exist} & N = 1 \text{ (rank 1)} \\ \text{Small} & N = 2 \\ \text{Smaller} & N = 3 \\ \vdots & \\ \text{Tiny!} & N = D - 1 \\ \text{Smallest!} & N = D \text{ (interpolation threshold)} \\ \text{Growing...} & N = D + 1 \\ \text{Stabilizes} & N \gg D \end{cases} \tag{32}$$

Intuition: With $N < D$, you can't even reach the D-dimensional space. With $N = D$, you *barely* reach it, so the last direction has almost zero variance. With $N > D$, each direction gets more samples and variance increases.

Funny Example

The Party Guests Analogy:
You're throwing a party in a 3D room (3 dimensions). You want to understand how guests spread out.

- **1 guest:** They're just a point. You know 1 direction (where they are), but have NO idea about the other 2 dimensions.
- **2 guests:** They define a line. You know 2 directions now, but the second direction (along the line) has way less "spread" than if they were far apart.
- **3 guests (interpolation threshold):** They *barely* define a plane. If they're almost collinear (which is likely!), the third direction has almost zero variance. You're teetering on the edge!
- **4+ guests:** Now you have redundancy. The 3D space is well-covered. All three dimensions have solid variance.

At exactly 3 guests in 3D, you're in the **danger zone** – tiny perturbations cause huge changes!

7 Geometric Interpretation

7.1 Underparameterized Geometry

In the underparameterized regime, we use $X^T X$:

$$X^T X = V S^2 V^T = \sum_{r=1}^{R} \sigma_r^2 \vec{v}_r \vec{v}_r^T \tag{33}$$

This is the **empirical covariance matrix** (up to scaling by $1/N$). The eigenvectors $\vec{v}_r$ are the principal components – directions of variance.

7.2 Overparameterized Geometry: Representation Learning!

In the overparameterized regime, we use XX^T and create an *internal representation*:

Key Equation

$$\hat{\vec{x}}_{\text{test}} = X^T (XX^T)^{-1} X \vec{x}_{\text{test}} = \sum_{r=1}^{R} \vec{v}_r \vec{v}_r^T \vec{x}_{\text{test}} = V V^T \vec{x}_{\text{test}} \tag{34}$$

This is an **orthogonal projection** of $\vec{x}_{\text{test}}$ onto the row space of X!

The bias term becomes:

$$\text{Bias} = \vec{x}_{\text{test}}^T(I - VV^T)\vec{\beta}^* = (\vec{x}_{\text{test}} - \hat{\vec{x}}_{\text{test}})^T\vec{\beta}^* \tag{35}$$

Interpretation: The model can only "see" the projection $\hat{\vec{x}}_{\text{test}}$, not the full $\vec{x}_{\text{test}}$. Information about $\vec{\beta}^*$ in the orthogonal directions $(I - VV^T)\vec{\beta}^*$ is lost!

Funny Example

The Shadow Analogy:
You're a 3D creature trying to understand a 2D world (flatland).

- Your training data X lives in a 2D plane (the row space).
- A test point $\vec{x}_{\text{test}}$ is in 3D.
- Your model projects it onto the 2D plane: $\hat{\vec{x}}_{\text{test}} = VV^T\vec{x}_{\text{test}}$ (the shadow).
- The model can ONLY see the shadow, not the full 3D point!
- If $\vec{\beta}^*$ has components in the 3D direction (perpendicular to the plane), that information is lost. Bias!

Absurd version: You're trying to predict a 3D dinosaur's weight by looking at its 2D shadow. If the dinosaur is oriented to minimize its shadow (thin edge facing the light), your prediction is garbage!

8 Ablation Experiments: Testing Our Understanding

To verify the three factors, we can ablate (remove) each one and check if the divergence disappears.

8.1 Ablation 1: Remove Small Singular Values

Set $\sigma_r = 0$ for all $\sigma_r <$ threshold.
Prediction: Divergence should decrease or disappear.
Why: No small $\sigma_r \to$ no huge $1/\sigma_r \to$ no amplification!

8.2 Ablation 2: Project Test Data onto Leading Modes

Replace $\vec{x}_{\text{test}}$ with $\sum_{r=1}^{k}(\vec{x}_{\text{test}} \cdot \vec{v}_r)\vec{v}_r$ where $k \ll R$ (keep only top k modes).
Prediction: Divergence should decrease.
Why: Test data no longer projects onto the weak modes $\vec{v}_r$ with small σ_r.

8.3 Ablation 3: Remove Residual Errors

Create a noiseless, perfectly linear dataset: $Y_{\text{new}} = X\vec{\beta}_{\text{fit}}$ where $\vec{\beta}_{\text{fit}}$ is fit on the full dataset.

Prediction: Divergence should disappear entirely.
Why: $E = 0 \to$ divergence term is zero!

9 Adversarial Examples

9.1 Adversarial Test Examples

To maximize test error, make $\vec{x}_{\text{test}} \cdot \vec{v}_r$ large for the mode r with smallest σ_r:

$$\vec{x}_{\text{test}}^{\text{adv}} = \alpha \vec{v}_R \quad \text{(where } \sigma_R \text{ is smallest)} \tag{36}$$

Effect: Test MSE explodes!

Funny Example

Adversarial Test = Finding Your Model's Blind Spot
Your GPS model has terrible East-West coverage ($\vec{v}_2$ weak direction). An adversarial test point is just... asking for directions on an East-West road! Your model panics because it has to extrapolate wildly.
Absurd version: You trained a "cat detector" only on photos of orange tabbies. An adversarial test image is a black cat. Your model outputs "NEGATIVE 5000% CAT" because it's never seen this direction in feature space!

9.2 Adversarial Training Data (Dataset Poisoning)

To maximize test error via training data, manipulate the residuals E to align with $\vec{u}_r$ for small σ_r:

$$E^{\text{adv}} = \gamma \vec{u}_R \quad \text{(where } \sigma_R \text{ is smallest)} \tag{37}$$

Effect: Training error unchanged, test error explodes by 1-3 orders of magnitude!

Funny Example

Data Poisoning = Subtle Sabotage
Imagine you're poisoning a linear model for pizza delivery time prediction. You change the labels slightly so that residual errors align with the weakest training direction (maybe "distance to pizzeria along Elm Street").
The training error looks fine (small perturbations). But at test time, predictions for Elm Street addresses are catastrophically wrong!
Absurd version: You're training a model to predict exam scores from study hours. An adversary poisons your training data so that students who study exactly 3.7 hours have corrupted scores. At the interpolation threshold, predictions for 3.7-hour studiers are "you'll score -500%!"

10 Connection to Nonlinear Models

The intuition extends beyond linear regression!

10.1 Neural Tangent Kernel (NTK)

For wide neural networks, training dynamics resemble kernel regression:

$$\vec{\beta}_{\text{NTK}}(t) = K^{+}Y \tag{38}$$

where K is the NTK. Same SVD story applies!

10.2 Superposition in Autoencoders

Henighan et al. (2023) found that autoencoders:

- **Memorize data points** (use XX^T-like features) when $N < D$
- **Learn generalizing features** (use X^TX-like features) when $N > D$
- **Exhibit double descent** at the transition!

Our clarification: "Data point features" CAN generalize! The issue isn't memorization vs. generalization – it's about the spectrum.

11 Practical Implications

11.1 When to Expect Double Descent

Double descent requires **all three factors**:

1. Small (non-zero) singular values in training data
2. Test data varies in those weak directions
3. Residual errors in best model

Safe regimes:

- Far from interpolation threshold ($N \ll D$ or $N \gg D$)
- Clean, noiseless data ($E \approx 0$)
- Regularization (adds to singular values, prevents $\sigma_r \to 0$)
- Well-conditioned data (all σ_r roughly equal)

11.2 How to Avoid Double Descent

1. **Regularization:** $\hat{\vec{\beta}} = (X^TX + \lambda I)^{-1}X^TY$ replaces σ_r^2 with $\sigma_r^2 + \lambda$, preventing small singular values.
2. **Data augmentation:** More diverse data → larger smallest singular value.
3. **Feature selection:** Remove redundant features → better-conditioned X.
4. **Early stopping:** Stop training before exact interpolation.
5. **Ensembling:** Errors from small σ_r are high-variance; averaging helps.

12 Summary and Key Takeaways

Key Equation

The Double Descent Story:

1. Double descent is NOT mysterious – it's a confluence of three factors.
2. The SVD reveals everything: small $\sigma_r \to$ huge $1/\sigma_r \to$ amplified errors.
3. Interpolation threshold is the danger zone because $\sigma_{\min}$ reaches its minimum there.
4. Overparameterization can help because it forces models to learn compressed representations (projection onto row space).
5. Both underparameterized and overparameterized regimes share a common variance term; overparameterized adds a bias term.

Funny Example

The Final Absurd Analogy: Machine Learning as a Tightrope Walk

- **Underparameterized:** You're walking on a wide bridge. Safe, but boring. You can't fit complex patterns.
- **At interpolation threshold:** You're on a tightrope over a canyon. One gust of wind (small σ_r), and you're done for. **Maximum danger!**
- **Slightly overparameterized:** You fell off the tightrope... but there's a safety net! (The bias term provides stability, and the model learns useful features.)
- **Massively overparameterized:** You're walking on a cloud. So many parameters that the model finds low-dimensional structure and generalizes beautifully.

The lesson? Don't balance on the tightrope. Either stay on the bridge or jump into the cloud!

References

1. Belkin, M., et al. (2019). "Reconciling modern machine-learning practice and the classical bias-variance trade-off." *PNAS*.
2. Nakkiran, P., et al. (2021). "Deep double descent: Where bigger models and more data hurt." *Journal of Statistical Mechanics*.
3. Advani, M. S., & Saxe, A. M. (2020). "High-dimensional dynamics of generalization error in neural networks." *Neural Networks*.

4. Bartlett, P. L., et al. (2020). "Benign overfitting in linear regression." *PNAS*.

5. Hastie, T., et al. (2022). "Surprises in high-dimensional ridgeless least squares interpolation." *The Annals of Statistics*.

6. ICLR 2024 Blogpost: "Double Descent Demystified"

7. arXiv:2303.14151