Ruizhe Final Project - Benign Overfitting in Linear Regression

March 20, 2025

github repo link: https://github.com/RuizheJiang/PSTAT234-Final-Project—Benign-overfitting-in-linear-regression

1 Benign Overfitting in Linear Regression – Summary and Analysis

1.1 Motivation: Deep Learning and the Overfitting Paradox

Modern deep learning has revealed a remarkable paradox: neural networks can perfectly fit training data—even with noisy labels—while still generalizing well to unseen data. This contradicts classical statistical wisdom, which warns that predictors fitting every training point are likely overfitting and should perform poorly on new data. Zhang et al. (2017) demonstrated this phenomenon by showing deep networks achieving near-zero training loss on vision datasets with randomized labels while maintaining non-trivial test accuracy. This behavior challenges the traditional bias-variance trade-off, which suggests an inherent tension between model complexity and training error.

"Benign overfitting" describes situations where perfect fits to training data, including noise, do not harm prediction accuracy. In their paper, Bartlett et al. (2020) investigate this phenomenon in the simplest possible setting: linear regression with more parameters than data points. By analyzing overparameterized linear models, they identify specific conditions where interpolating training noise doesn't degrade test performance. Their findings aim to provide insights into why heavily overparameterized deep networks can generalize effectively despite fitting noisy training data perfectly.

1.2 The Linear Regression Setup and Interpolation Solution

Bartlett et al. consider a linear regression problem in high-dimensional space with input vector $x \in \mathbb{R}^d$ (or Hilbert space H) having covariance $\Sigma = \mathbb{E}[xx^T]$, and output $y \in \mathbb{R}$ following $y = x^T\theta^* + \text{noise}$. The data are assumed mean-zero with sub-Gaussian properties.

The authors focus on overparameterized settings where dimension d exceeds sample size n, ensuring perfect fits are possible ($\text{rank}(\Sigma) > n$). This creates infinitely many solutions to the normal equations $X^TX\theta = X^Ty$. Among all interpolating solutions, they study the minimum-norm estimator $\hat{\theta}$ - the solution with smallest Euclidean norm that fits the training data exactly. This corresponds to the Moore-Penrose pseudoinverse solution $\hat{\theta} = X^+y$.

To evaluate generalization, they define excess risk as $R(\theta) := \mathbb{E}[(y - x^T\theta)^2 - (y - x^T\theta^*)^2]$. Benign overfitting occurs when $R(\hat{\theta})$ approaches 0, meaning $\hat{\theta}$ performs nearly as well as the optimal θ^* despite fitting all training noise.

To evaluate generalization, the authors define the excess risk of an estimator θ as the difference in mean squared error compared to the Bayes-optimal predictor θ^*: $R(\theta) := \mathbb{E}[(y - x^T\theta)^2 - (y - x^T\theta^*)^2]$, i.e. how much larger the MSE is for θ than for the true θ^*. Benign overfitting in this context means $R(\hat{\theta})$ is close to 0 (so $\hat{\theta}$ is nearly as accurate as θ^*) even though $\hat{\theta}$ fits the training data exactly (including noise).

1.3 Theoretical Framework: Effective Rank and Risk Decomposition

A core contribution of Bartlett et al. is characterizing when linear regression can benignly overfit. They show that the answer lies in the spectrum of the covariance Σ through what they call "effective rank." The covariance's eigenvalues λ_i (in decreasing order) determine how variance distributes across different directions in parameter space. The authors derive a bias-variance decomposition of the excess risk $R(\hat{\theta})$ into two parts:

- Term 1: Error from estimating the signal θ^* using n samples. This term is controlled by the total "scale" of the problem. If the overall variance (trace of Σ) is not too large relative to n, then the component of $\hat{\theta}$ aligned with θ^* will be estimated accurately (i.e., θ^*'s contribution isn't overly distorted when $\sum_i \lambda_i$ is small compared to n.)
- Term 2: Error from fitting the label noise. This term captures how random noise in y (which $\hat{\theta}$ has interpolated) impacts prediction. The key insight is that label noise's impact depends on how "spread out" the covariance's small eigenvalues are. If many directions have tiny variance (small λ_i), the noise can be "hidden" there with minimal penalty to prediction accuracy.

- Term 1: Error from estimating the signal θ^* using n samples. This term is controlled by the total "scale" of the problem. If the overall variance (trace of Σ) is not too large relative to n, then the component of $\hat{\theta}$ aligned with θ^* will be estimated accurately (In other words, θ^*'s contribution isn't overly distorted by sampling noise when $\sum_i \lambda_i$ is small compared to n.)
- Term 2: Error from fitting the label noise. This term is more novel – it captures how the random noise in y (which $\hat{\theta}$ has interpolated) impacts prediction. The key insight is that the impact of label noise on $\hat{\theta}$'s risk depends on how "spread out" the covariance's small eigenvalues are. If there are many directions in parameter space with tiny variance (small λ_i), the noise can be "hidden" in those directions with little penalty to prediction accuracy

They formalize this via two notions of effective rank (for $k \geq 0$):

- $r_k(\Sigma) = \frac{\sum_{i>k} \lambda_i}{\lambda_{k+1}}$: Measures the ratio of variance in trailing eigenvalues to the $(k+1)$-th eigenvalue.
- $R_k(\Sigma) = \frac{(\sum_{i>k} \lambda_i)^2}{\sum_{i>k} \lambda_i^2}$: Represents effective dimensionality in the tail (equals the number of eigenvalues beyond k if all equal).

Intuitively, large effective rank means Σ has many directions with negligible variance. Bartlett et al. prove that Term 2 is small if and only if Σ's effective rank in the low-variance regime significantly exceeds n. This means benign overfitting requires extreme overparameterization with a heavy-tailed spectrum—many directions with small eigenvalues where noise can be absorbed without affecting predictions.

In short, to benignly overfit, the model must be extremely overparameterized and the data distribution must have a heavy-tailed covariance (lots of small eigenvalues). This ensures the estimator can memorize noise in "safe" directions that barely affect its predictions. If instead the data had

only a few relevant directions (fast-decaying eigenvalues), then fitting noise along those directions would significantly hurt accuracy.

1.4 Main Results: When is Overfitting Benign?

Using their framework, Bartlett et al. provide a rigorous characterization through finite-sample bounds on $R(\widehat{\theta})$ (Theorem 1). This theorem gives nearly matching upper and lower bounds on the excess risk of the minimum-norm interpolating estimator in terms of effective ranks of Σ. While technical, its implications can be summarized as:

- If Σ lacks sufficient effective rank (not enough small-eigenvalue directions compared to n), benign overfitting becomes impossible. In this case, $\widehat{\theta}$'s risk remains bounded below by a constant fraction of the irreducible noise level—performing significantly worse than the optimal θ^*. Intuitively, without enough "extra" weak directions to absorb noise, any interpolating solution must distort important directions, causing substantial excess error.
- Conversely, when Σ's spectrum is suitably flat/extended (effective rank $\gg n$ in the low-variance end), the excess risk approaches zero even as $\widehat{\theta}$ perfectly fits the data. Here, $\widehat{\theta}$ achieves near-optimal prediction accuracy despite interpolating noise. The upper bound on $R(\widehat{\theta})$ involves ratios like $r_0(\Sigma)/n$ or $n/R_{k_n^*}(\Sigma)$, which become small when spectrum conditions are met.

The authors define "benign" covariance sequences (Definition 4) to formalize asymptotic benign overfitting. A sequence Σ_n is benign if as $n \to \infty$: - $r_0(\Sigma_n)/n \to 0$ - $k_n^*/n \to 0$ - $n/R_{k_n^*}(\Sigma_n) \to 0$

Here k^* represents the index of the first "large" effective rank: $k^* = \min\{k : r_k(\Sigma) \geq bn\}$ for constant $b > 1$. These conditions ensure the spectrum has enough small eigenvalues while the overall scale remains controlled.

Here k is the index of the first "large" effective rank: $k = \min k : r_k(\Sigma) \geq bn$ for some constant $b > 1$. In essence, these conditions ensure that the spectrum has a sufficiently large bulk of small eigenvalues (making R_{k^*} huge) while the overall scale of Σ remains controlled. Under these conditions, Theorem 1 guarantees $R(\widehat{\theta}) \to 0$ (benign overfitting).

Theorem 2 illustrates this theory with concrete examples:

1. Infinite-Dimensional Case (Heavy-tailed spectrum): When eigenvalues decay as $\mu_k(\Sigma) \sim k^{-\alpha}(\ln k)^{-\beta}$, benign overfitting occurs if and only if $\alpha = 1$ and $\beta > 1$. This represents eigenvalues decaying just slowly enough to remain summable, but nearly as slow as $1/k$.
2. High but Finite Dimension with Isotropic Noise: When data lie in a finite-dimensional space with dimension p_n growing faster than n, even with rapidly decaying primary eigenvalues, adding small "isotropic" variance in all directions enables benign overfitting. The conditions are that $p_n = \omega(n)$ and $\varepsilon_n p_n = o(n)$ but not exponentially small.

These examples highlight a fundamental trade-off: achieving both slow eigenvalue decay (for small n/R_{k^*}) and summability (for small $r_0(\Sigma)/n$). In infinite dimensions, this requires a precise decay rate at the edge of summability. In contrast, finite but high-dimensional spaces allow much more flexibility, making benign overfitting more naturally achievable when dimension significantly exceeds sample size.

1.5 Overparameterization and Effective Rank: Why So Many Parameters?

A clear message from this work is that overparameterization is essential for benign overfitting. In practical terms, overparameterization means the model has far more parameters than training examples. Bartlett et al. demonstrate that beyond just counting extra parameters, what matters is having many "uninformative" directions in parameter space. These directions correspond to eigenvectors of Σ with tiny eigenvalues—directions where inputs x have almost no variance and barely affect output. When these weak directions significantly outnumber data points, the least-norm solution uses them to fit noise, leaving important directions (those with large variance) primarily aligned with the true signal θ^*.

The parameter vector $\widehat{\theta}$ can be decomposed into two components: one in the principal components subspace (large eigenvalues) responsible for predicting the true signal, and another in the weak components subspace (small eigenvalues) that mainly captures label noise. Overparameterization provides the noise-fitting component sufficient dimensions to exist without interfering with the signal component. This aligns with the intuition that $\widehat{\theta} = \theta^* + \Delta$, where Δ primarily occupies the low-variance directions of Σ and thus minimally affects predictions. While classical theory would penalize any non-zero Δ as overfitting, here Δ becomes "harmless" because $x^T\Delta$ remains very small for new samples (as x has minimal projection in those directions).

To summarize: Benign overfitting requires an extreme excess of parameters relative to data, creating a high-capacity subspace irrelevant to the true function. This subspace serves as a reservoir for fitting noise. When large enough (effective rank $\gg n$), this reservoir prevents noise-fitting from harming generalization. Bartlett et al.'s results formalize a crucial insight: simply having more parameters than data isn't sufficient—these extra parameters must be structured (through Σ) to contribute minimally to the true signal. Under these conditions, overparameterization becomes not just benign but potentially beneficial.

1.6 Insights and Connections to Deep Learning

Although focused on linear models, this paper was directly motivated by deep learning phenomena and offers valuable insights into neural networks' behavior. The findings in linear regression reveal intriguing parallels to complex neural networks and may help explain their surprising generalization abilities.

In deep networks, gradient-based optimization appears to implicitly favor solutions that generalize well despite many possible data-fitting alternatives. This mirrors how the minimum-norm interpolant in linear regression represents a "biased" choice among infinitely many solutions. A particularly relevant connection appears in the Neural Tangent Kernel (NTK) regime, where extremely wide networks can be approximated by linear models in a high-dimensional function space. When training such networks, gradient descent effectively operates in this kernel space, finding interpolators that minimize certain norms. Under reasonable data assumptions, the NTK's eigenvalues often exhibit heavy tails with slow decay rates and effectively infinite dimensions—precisely the conditions identified for benign overfitting in linear models. This suggests wide neural networks might operate in a similar regime, with numerous "slightly important" directions in function space.

The results further suggest that networks exhibiting benign overfitting likely have learned representations with approximately flat eigenvalue spectra extending to very high ranks. Rather than concentrating variance in a few directions, such networks distribute it across many dimensions in weight or activation space. Some researchers view deep networks as finite-dimensional approxima-

tions to infinite-dimensional function classes, but the linear analysis indicates a surprising advantage to finite but large capacity. While infinite-dimensional models require very specific spectral conditions to generalize well, high-but-finite models can do so across broader conditions. This implies the finite width of real neural networks might actually facilitate benign overfitting by constraining how quickly eigenvalues decay.

The paper stops short of claiming deep networks definitely possess the covariance structure needed for benign overfitting, instead providing theoretical clues and direction. Many assumptions in the linear setting—such as independent features or model linearity—don't hold in realistic deep learning scenarios. Nevertheless, the analysis suggests a potential explanation for deep learning's generalization "mystery": networks might implicitly achieve something akin to minimum-norm solutions in high-dimensional spaces with slow spectral decay, effectively channeling noise into directions that minimally affect outputs. Confirming this hypothesis for actual neural networks remains an important open problem for future research.

1.7 Conclusion

Bartlett et al. (2020) provide a clear mathematical framework for understanding benign overfitting in linear regression. Their analysis highlights the crucial interplay between model capacity (overparameterization) and data distribution (covariance spectrum). For benign overfitting to occur, a linear model needs sufficient "wiggle room" to absorb noise in directions that minimally impact predictions. This manifests as a covariance with many small eigenvalues (large effective rank) combined with a parameter space vastly exceeding the sample size.

When these conditions are satisfied, the minimum-norm interpolator can match the performance of the true model despite fitting training noise perfectly, resolving the apparent paradox of overfitting without harming generalization. Though derived in a simplified linear setting, these insights draw compelling parallels to deep neural networks and potentially explain why modern overparameterized models can generalize effectively despite seemingly excessive capacity.

2 Benign Overfitting in Linear Regression: Experimental Verification

Modern over-parameterized models can perfectly fit training data yet still generalize well, a phenomenon known as benign overfitting. In classical theory, a model that interpolates noisy data would severely overfit and perform poorly on new data. However, Bartlett et al. (2020) established conditions under which linear regression can interpolate noise benignly, achieving near-optimal test accuracy despite fitting training data perfectly.

Two key insights from their analysis are that benign overfitting requires: (1) significant overparameterization, where the model has substantially more parameters than samples, particularly many "uninformative" directions in parameter space that have minimal effect on prediction; and (2) slowly decaying eigenvalues in the feature covariance spectrum, creating a high effective rank with many small-eigenvalue directions where label noise can be hidden with minimal impact on predictions. When these conditions are met, the minimum-norm interpolating solution distributes fitted noise across many weak directions, limiting its harmful effect on test performance.

To experimentally verify these theoretical findings, this section implements synthetic data experiments that systematically explore the conditions that enable or prevent benign overfitting in linear

regression.

2.0.1 Experiment Setup: Synthetic Data Generation

To study benign overfitting in a controlled environment, synthetic regression data is generated with specified feature covariance structures. The linear model setup includes:

- Data (x_i, y_i) for $i = 1, \ldots, n$ drawn i.i.d. in $\mathbb{R}^p \times \mathbb{R}$.
- Feature vectors $x_i \sim \mathcal{N}(0, \Sigma)$, zero-mean Gaussian with covariance Σ designed with various eigenvalue spectra (slow or fast decaying, correlated features).
- True relationship $y = x^T w^* + \varepsilon$, with true parameter w^* and noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. For simplicity, $\sigma^2 = 1$ and often $w^* = 0$ to focus purely on noise-fitting effects. (When $w^* = 0$, the optimal prediction is 0 for all inputs, with MSE equal to the noise variance.)

The experimental framework uses the minimum-norm interpolating solution—the pseudoinverse solution that exactly fits y, solving $\min_w |w|_2$ subject to $Xw = y$. This aligns with the theoretical analysis and represents what many gradient-based methods would converge to in underdetermined systems.

For evaluation, each experiment measures:

- Training MSE: The mean squared error on the training set (nearly 0 in overparameterized cases).
- Test MSE: The mean squared error on an independent test set, indicating generalization. For reference, the irreducible error (noise variance) is 1, and the Bayes-optimal predictor would achieve this MSE. Benign overfitting occurs when the fitted model's test MSE approaches this optimal value despite interpolating noise.

Helper functions implement data generation with specified covariance spectra, compute minimum-norm solutions, and evaluate performance. The generate_data function allows specifying eigenvalues for Σ and whether features are independent (diagonal covariance) or correlated (using random orthonormal transformation to mix features while preserving the spectrum). Cholesky decomposition enables efficient sampling from multivariate Gaussian distributions, while fit_min_norm computes the minimum-norm interpolating weights for the system $Xw = y$.

[ ]:
```
import numpy as np

def generate_data(n, p, eigen_vals, w_star=None, correlated=True,␣
 ↪random_state=None):
    if random_state is not None:
        np.random.seed(random_state)
    eigen_vals = np.array(eigen_vals)
    p = len(eigen_vals)
    # Construct covariance matrix Sigma = Q * diag(eigen_vals) * Q^T
    if correlated:
        # random orthonormal matrix for eigenvectors
        Q, _ = np.linalg.qr(np.random.randn(p, p))
    else:
        # Use identity as eigenvector basis (so features independent)
        Q = np.eye(p)
```

```
    Sigma = Q @ np.diag(eigen_vals) @ Q.T
    # Generate data: X ~ N(0, Sigma) with n samples
    # Sample X by drawing Z ~ N(0, I_p) and then transforming: X = Z * Sigma^(1/
↪2).
    # Compute a Cholesky factor for Sigma for sampling:
    try:
        L = np.linalg.cholesky(Sigma)
    except np.linalg.LinAlgError:
        # If Sigma is nearly singular, add a tiny diagonal for stability
        L = np.linalg.cholesky(Sigma + 1e-10 * np.eye(p))
    # Each row of X is generated as (Z_i * L) where Z_i ~ N(0, I_p)
    Z = np.random.randn(n, p)  # n x p standard normal
    X = Z @ L.T                # X will be n x p, with Cov(X_row) = Sigma
    # True parameter w_star (if not given, default to zero vector)
    if w_star is None:
        w_star = np.zeros(p)
    # Sample noise
    noise = np.random.randn(n)
    # Generate labels: y = X w* + noise
    y = X.dot(w_star) + noise
    return X, y, w_star, Sigma

def fit_min_norm(X, y):
    # Find the minimum-norm solution (pseudoinverse solution) for X w = y.
    w_min_norm, *_ = np.linalg.lstsq(X, y, rcond=None)
    return w_min_norm

def compute_mse(y_pred, y_true):
    return np.mean((y_pred - y_true)**2)

def evaluate_model(w_hat, w_true, Sigma):
    # Use global X_train, y_train if available (in this context, we'll call␣
↪evaluate_model inside the same cell where X, y are defined)
    train_pred = X_train.dot(w_hat)
    train_mse = compute_mse(train_pred, y_train)
    # Generate a large test set for reliable estimate
    n_test = 10000
    Zt = np.random.randn(n_test, Sigma.shape[0])
    X_test = Zt @ np.linalg.cholesky(Sigma).T
    y_test = X_test.dot(w_true) + np.random.randn(n_test)
    test_pred = X_test.dot(w_hat)
    test_mse = compute_mse(test_pred, y_test)
    return train_mse, test_mse

n, p = 50, 100
eigs = np.ones(p)  # identity covariance (all eigenvalues 1)
```

```
X_train, y_train, w_true, Sigma = generate_data(n, p, eigs, w_star=None,␣
 ↪correlated=False, random_state=42)
w_hat = fit_min_norm(X_train, y_train)
train_mse, test_mse = compute_mse(X_train.dot(w_hat), y_train),␣
 ↪compute_mse(X_train.dot(w_hat), y_train)
print(f"Train MSE (should be ~0 if p>n): {train_mse:.2e},  Test MSE (example):␣
 ↪{test_mse:.2f}")
```

```
Train MSE (should be ~0 if p>n): 8.41e-30,  Test MSE (example): 0.00
```

2.0.2 Benign Overfitting under Ideal Conditions

First, an experiment scenario is constructed to satisfy the theoretical conditions for benign overfitting. According to theory, two key ingredients are required: (1) significant overparameterization ($p \gg n$), and (2) a slowly decaying covariance spectrum with many small eigenvalues (high effective rank). The setup includes:

- Sample size: $n = 100$
- Feature dimension: $p = 1000$ (ten times larger than n, providing ample extra degrees of freedom)
- Covariance spectrum: Approximately constant eigenvalues, using $\Sigma = I$ (identity covariance) where all eigenvalues equal 1. This represents the most extreme case of "slow decay" (no decay at all), with independent features of equal importance and maximal effective rank (rank = 1000).

The true signal parameter is set to $w^* = 0$, making labels pure noise. This setup allows direct assessment of overfitting effects - any test error above the noise variance (1.0) clearly indicates harmful overfitting, while test MSE close to 1 would be optimal (since even the best possible predictor would incur this error due to irreducible noise).

In this benign setting, theory predicts the minimum-norm interpolator should achieve near-noise-level test MSE despite having zero training error. With 900 extra directions (1000-100) of small variance, the fitted noise should be contained in these directions without significantly impacting prediction accuracy.

[ ]:
```
# Benign scenario: p >> n and flat (slow-decay) spectrum
n = 100
p = 1000
eigen_vals = np.ones(p)   # all eigenvalues = 1 (identity covariance)
X_train, y_train, w_true, Sigma = generate_data(n, p, eigen_vals, w_star=np.
 ↪zeros(p), correlated=False, random_state=1)
w_hat = fit_min_norm(X_train, y_train)
train_mse = compute_mse(X_train.dot(w_hat), y_train)
# Estimate test MSE on a large test set
n_test = 10000
X_test = np.random.randn(n_test, p)  # since Sigma=I, we can sample standard␣
 ↪normal for features
y_test = X_test.dot(w_true) + np.random.randn(n_test)
test_mse = compute_mse(X_test.dot(w_hat), y_test)
```

```
print(f"Train MSE = {train_mse:.2e}")
print(f"Test MSE  = {test_mse:.3f} (noise variance = 1.0)")
```

```
Train MSE = 3.35e-30
Test MSE  = 1.085 (noise variance = 1.0)
```

The experimental results confirm theoretical predictions. When running the code, the training MSE is essentially zero (approximately 3.35e-30), verifying that the model perfectly interpolates the training data. The test MSE is approximately 1.085, remarkably close to the optimal noise level of 1.0. This demonstrates the key insight of benign overfitting: despite fitting pure noise in the training set, the model's predictions on new data are nearly as good as the optimal predictor (which simply outputs zero in this case).

The small excess over the noise floor (0.085) can be attributed to finite sample effects, which diminish as the feature dimension p increases further. What's happening is that the minimum-norm solution effectively "hides" the fitted noise in the many extra dimensions of the weight vector that correspond to directions in input space with negligible variance, thus minimizing their impact on new samples.

To illustrate the effect of over-parameterization more clearly, let's vary the model dimension p and see how test performance changes. We'll keep $n = 100$ fixed and use the same identity covariance (eigenvalues all 1), and examine test MSE for different ratios of p/n.

[ ]:
```
import math

n = 100
p_values = [100, 150, 300, 500, 1000, 2000]  # from equal to n up to 20x n
test_mse_results = []
for p in p_values:
    # average test MSE over a few random trials for stability
    trials = 3
    mse_sum = 0.0
    for seed in range(trials):
        X_train, y_train, w_true, Sigma = generate_data(n, p, np.ones(p),␣
 ↪w_star=np.zeros(p), correlated=False, random_state=seed)
        w_hat = fit_min_norm(X_train, y_train)
        # Compute test MSE
        X_test = np.random.randn(10000, p)  # sampling from N(0,I) for test
        y_test = X_test.dot(w_true) + np.random.randn(10000)
        mse_sum += compute_mse(X_test.dot(w_hat), y_test)
    avg_test_mse = mse_sum / trials
    test_mse_results.append(avg_test_mse)
    print(f"p = {p:4d}, Test MSE  {avg_test_mse:.2f}")
```

```
p =  100, Test MSE  27813.00
p =  150, Test MSE  3.37
p =  300, Test MSE  1.51
p =  500, Test MSE  1.30
p = 1000, Test MSE  1.11
```

```
p = 2000, Test MSE  1.07
```

We expect a trend: when p is only equal to or slightly above n, test MSE will be significantly higher than 1 (indicating harmful overfitting), but as p grows much larger than n, test MSE drops toward 1. The printed results confirm this.

The trend confirms that significant overparameterization is crucial: as the number of features grows well beyond the sample size, the test error drops to the noise floor. This matches the theory that a large surplus of parameters (here, hundreds more dimensions than data points) is required for benign overfitting

2.0.3 The Role of Effective Rank: Varying the Covariance Spectrum

Next, the experiment examines how covariance eigenvalue decay affects benign overfitting. According to theory, effective rank of Σ plays a crucial role in determining whether interpolation harms generalization. Even with high dimensionality, if feature variance concentrates in a few dominant directions (rapidly decaying eigenvalues), there won't be enough "small-variance" directions to hide fitted noise. For benign overfitting, small eigenvalues must decay slowly enough to create many directions with modest variance, collectively providing a large effective rank.

This experiment maintains $n = 100$ and $p = 1000$ while varying eigenvalue decay patterns. Independent features are used for clarity (diagonal covariance matrix), allowing direct control over the spectrum. Three distinct patterns are compared:

- Slow Decay: Eigenvalues decreasing gradually, such as $\lambda_i \propto \frac{1}{i}$ (harmonic decay). The flat spectrum from the previous experiment (constant eigenvalues) represents an extreme case of no decay.
- Moderate Decay: Eigenvalues decaying polynomially faster, like $\lambda_i \propto \frac{1}{i^2}$, or having mixed large and small values.
- Fast Decay: Eigenvalues dropping rapidly, as in exponential decay $\lambda_i \propto a^i$ for some $a < 1$. In this scenario, eigenvalues beyond the first few directions become negligible, creating a low effective rank with variance concentrated in top components.

All simulations use the same dimensions ($p = 1000$, $n = 100$) and $w^* = 0$ (pure noise labels) to isolate the effect of eigenvalue decay on noise-fitting. Theory predicts that test MSE will be lowest for the flattest spectrum and highest for the fastest-decaying spectrum.

[ ]:
```
n = 100
p = 1000
# Define different eigenvalue spectra
eigen_spectra = {
    "flat (all equal)": np.ones(p),
    "slow (1/i)": 1.0 / (np.arange(p) + 1),              # harmonic decay
    "moderate (1/i^2)": 1.0 / ((np.arange(p) + 1)**2),   # polynomial decay␣
 ↪(faster)
    "fast (exp decay)": 0.95**(np.arange(p))             # exponential decay␣
 ↪with ratio 0.95
}
# Function to get test MSE for a given spectrum
```

```python
def get_test_mse_for_spectrum(eigs):
    X_train, y_train, w_true, Sigma = generate_data(n, p, eigs, w_star=np.
 ↪zeros(p), correlated=False)
    w_hat = fit_min_norm(X_train, y_train)
    # Evaluate test error on large test set
    X_test = np.random.randn(10000, p) * np.sqrt(eigs)  # since features␣
 ↪independent, we can multiply std dev
    # (Alternatively, generate from Sigma via generate_data for test)
    y_test = X_test.dot(w_true) + np.random.randn(10000)
    test_mse = compute_mse(X_test.dot(w_hat), y_test)
    return test_mse

# Calculate test MSE for each spectrum
for desc, eigs in eigen_spectra.items():
    mse = get_test_mse_for_spectrum(eigs)
    print(f"Spectrum: {desc:20s} -> Test MSE   {mse:.2f}")
```

```
Spectrum: flat (all equal)     -> Test MSE   1.15
Spectrum: slow (1/i)           -> Test MSE   1.36
Spectrum: moderate (1/i^2)     -> Test MSE   2.00
Spectrum: fast (exp decay)     -> Test MSE   4.96
```

The experimental results reveal a clear pattern: models interpolating noise with rapidly decaying eigenvalues (exponential case) perform poorly on test data, with MSE several times the noise level. In contrast, models with flat or slowly decaying spectra achieve test MSE close to 1.0 (the noise level), confirming benign overfitting.

These findings directly support the paper's effective rank condition. A large effective rank, created by many comparably small eigenvalues, is essential for controlling excess error from noise fitting. In the exponential decay scenario, variance concentrates in the first few features, forcing the model to place substantial weight on these important directions to fit noise. Since remaining directions have near-zero variance, they cannot effectively absorb noise without requiring enormous weights. This noise fitting in critical directions severely impacts prediction accuracy.

Conversely, with slow-decay spectra, variance spreads across many directions. The minimum-norm solution distributes fitted noise over numerous directions of modest variance, each having minimal influence on predictions. This spreading effect keeps test error remarkably close to optimal despite perfect training fits.

The experiment confirms a fundamental principle: if the covariance spectrum decays too quickly, benign overfitting becomes impossible. The interpolating solution generalizes poorly because it must overfit in the directions that matter most for prediction. Only when the spectrum exhibits a long tail (slow decay) can interpolation occur without harming generalization.

2.1 Failure Cases: Violating Benign Overfitting Conditions

2.1.1 1. Covariance Spectrum Decays Too Fast

Building on previous findings, this experiment examines an extreme case where benign overfitting conditions are deliberately violated through rapid eigenvalue decay. While the exponential decay

case already demonstrated problems, here a more dramatic scenario is constructed: a stark "cliff" in the eigenvalue spectrum.

The setup uses a covariance matrix where only 10 eigenvalues equal 1.0, with the remaining 490 eigenvalues set to 0.001 (very small). This creates a sharp drop-off after the first 10 principal components, effectively concentrating almost all variance in just a handful of directions. Despite maintaining high dimensionality ($p = 500$ with $n = 100$), most dimensions carry negligible variance.

In this setting, theory predicts that overparameterization alone won't save generalization. Even though $p \gg n$ provides many parameters, the effective model capacity is severely constrained - only about 10 meaningful directions exist for fitting the data. The minimum-norm solution must still interpolate through these few important directions, likely leading to poor generalization despite the nominal high dimension.

[ ]:
```
n = 100
p = 500
# Construct eigenvalues: first 10 = 1, rest = 0.001
eigs_fast = np.array([1.0]*10 + [0.001]*(p-10))
X_train, y_train, w_true, Sigma = generate_data(n, p, eigs_fast, w_star=np.
 ↪zeros(p), correlated=False, random_state=0)
w_hat = fit_min_norm(X_train, y_train)
train_mse = compute_mse(X_train.dot(w_hat), y_train)
# Test on a new dataset
X_test, y_test, _, _ = generate_data(10000, p, eigs_fast, w_star=np.zeros(p),␣
 ↪correlated=False, random_state=1)
test_mse = compute_mse(X_test.dot(w_hat), y_test)
print(f"Train MSE = {train_mse:.2e}")
print(f"Test MSE  = {test_mse:.2f}")
```

```
Train MSE = 3.70e-29
Test MSE  = 1.46
```

Test MSE is larger than 1.0. This happens because effectively the model had only ~10 effective dimensions with significant variance to use for fitting – it had to contort those directions to fit random noise, which leads to large random projections on new data (high error). This confirms that a fast-decaying spectrum breaks benign overfitting.

2.1.2 2. Insufficient Overparameterization (p not n)

This experiment investigates another critical violation of benign overfitting conditions - when feature dimension isn't sufficiently larger than sample size. Theory indicates that substantial overparameterization is essential for benign overfitting. When p is only marginally larger than n (or equal/less), interpolation either becomes impossible or leads to poor generalization.

The investigation focuses on two key scenarios:

- Exactly $p = n$: The borderline case where the model has precisely enough parameters to potentially interpolate the training data (assuming X has full rank). This represents the threshold between under- and over-parameterization.

- Slightly above n: Cases where $p = 1.2n$ or $1.5n$, providing some extra degrees of freedom but far fewer than the "many more than n" required by theory.

To isolate the effect of dimensionality, the experiment uses the identity covariance $\Sigma = I$ (a flat spectrum with all eigenvalues equal to 1), which otherwise creates ideal conditions for benign overfitting. This allows direct observation of how generalization deteriorates solely due to insufficient parameter count as p approaches n.

The experimental outcomes follow a predictable pattern across different dimensionality regimes:

- When $p < n$ (e.g., 80 features with 100 samples), the model cannot interpolate the training data, resulting in non-zero training MSE. Test MSE exceeds the noise level due to standard bias-variance tradeoff constraints.
- When $p = n$ (exactly 100 features), the solution theoretically interpolates the training data (zero training MSE) but produces catastrophically high test MSE. With no freedom beyond exact fitting, the interpolator places extreme weights on certain directions, creating an unstable solution that generalizes poorly.
- When p is slightly above n (120 or 150 features), training MSE reaches zero but test MSE remains substantially higher than the noise level (2-5×). These extra parameters prove insufficient to properly distribute fitted noise into harmless directions.

[ ]:
```python
import numpy as np

n = 100
for p in [80, 100, 120, 150]:
    X_train, y_train, w_true, Sigma = generate_data(n, p, np.ones(p), w_star=np.
 ↪zeros(p), correlated=False, random_state=42)
    w_hat = fit_min_norm(X_train, y_train)
    # Compute test error
    X_test = np.random.randn(10000, p)  # since Sigma=I
    y_test = X_test.dot(w_true) + np.random.randn(10000)
    test_mse = compute_mse(X_test.dot(w_hat), y_test)
    train_mse = compute_mse(X_train.dot(w_hat), y_train)
    print(f"p={p:3d} -> Train MSE {train_mse:.2e}, Test MSE {test_mse:.2f}")
```

```
p= 80 -> Train MSE 1.94e-01, Test MSE 6.12
p=100 -> Train MSE 1.18e-28, Test MSE 118.29
p=120 -> Train MSE 6.36e-30, Test MSE 4.22
p=150 -> Train MSE 4.29e-30, Test MSE 2.96
```

These results clearly demonstrate that insufficient overparameterization prevents benign overfitting. When feature dimension isn't substantially larger than sample size, test error remains significantly above the noise floor—with particularly catastrophic performance when $p = n$. The experiments confirm that substantial feature surplus is necessary to achieve near-optimal generalization while interpolating noisy training data.

2.1.3 3. Correlated Features (Violating Independence)

The final experiment examines how feature correlation affects benign overfitting. Previous simulations assumed roughly independent features or non-pathological covariance structures. However, real-world data often contains correlations that effectively reduce dimensionality - even with large feature counts, redundant or linearly dependent features decrease the number of truly independent directions, potentially undermining overparameterization benefits.

To investigate this effect, a scenario with strong feature correlation is constructed using an AR(1) covariance matrix. This structure imposes correlation $\rho^{|i-j|}$ between features i and j, creating high collinearity between adjacent features when ρ approaches 1. The experiment uses $\rho = 0.9$ for pronounced correlation, comparing performance against an independent feature case (identity covariance) with identical dimensions to isolate correlation effects.

The experiment uses a comparative setup with $n = 100$ samples and $p = 300$ features under two conditions:

- Independent features: $\Sigma = I_{300}$ (identity covariance matrix)
- Correlated features: $\Sigma_{ij} = 0.9^{|i-j|}$ (AR(1) covariance with strong correlation)

Both scenarios maintain $w^* = 0$ (pure noise labels) to isolate correlation effects.

Theory predicts that correlated features will result in higher test MSE, as feature correlation effectively reduces the true dimensionality despite the nominal high feature count.

[ ]:
```
import numpy.linalg as LA

def ar1_cov(p, rho):
    # Construct an AR(1) covariance matrix of size p with correlation rho
    return np.array([[rho**abs(i-j) for j in range(p)] for i in range(p)])

n = 100
p = 300
# Independent case:
Sigma_indep = np.eye(p)
# Correlated case (AR(1) with rho=0.9):
Sigma_corr = ar1_cov(p, rho=0.9)
# Generate data for each case
X_i, y_i, w_true, _ = generate_data(n, p, np.ones(p), w_star=np.zeros(p),␣
 ↪correlated=False, random_state=0)
X_c, y_c, w_true, _ = generate_data(n, p, LA.eigvalsh(Sigma_corr)[::-1],␣
 ↪w_star=np.zeros(p), correlated=True, random_state=0)
# Note: We used LA.eigvalsh to get eigenvalues of Sigma_corr, then␣
 ↪generate_data with correlated=True to actually produce data with that␣
 ↪covariance
# Fit models
w_hat_i = fit_min_norm(X_i, y_i)
w_hat_c = fit_min_norm(X_c, y_c)
# Compute test errors
```

```
X_test_i, y_test_i, _, _ = generate_data(10000, p, np.ones(p), w_star=np.
  ↪zeros(p), correlated=False, random_state=1)
X_test_c, y_test_c, _, _ = generate_data(10000, p, LA.eigvalsh(Sigma_corr)[::
  ↪-1], w_star=np.zeros(p), correlated=True, random_state=1)
test_mse_i = compute_mse(X_test_i.dot(w_hat_i), y_test_i)
test_mse_c = compute_mse(X_test_c.dot(w_hat_c), y_test_c)
print(f"Test MSE (independent features)  = {test_mse_i:.2f}")
print(f"Test MSE (correlated features)   = {test_mse_c:.2f}")
```

```
Test MSE (independent features)  = 1.45
Test MSE (correlated features)   = 3.35
```

Results confirm the prediction: the correlated case (AR(0.9)) produces a test MSE of 3.35, substantially higher than the independent features case. This occurs because high collinearity ($\rho = 0.9$) effectively reduces the data's dimensionality through its spectral properties. The AR(1) covariance creates a decaying eigenvalue spectrum with one dominant eigenvalue followed by progressively smaller ones, reducing the effective rank well below the nominal 300 dimensions.

With lower effective rank, the conditions for benign overfitting are violated. The model cannot distribute fitted noise optimally across directions, resulting in higher prediction error. This demonstrates that feature correlation can undermine the benefits of nominal high dimensionality by concentrating variance in fewer directions, transforming what would be benign overfitting into harmful overfitting.

2.2 Conclusion

These experiments demonstrate that a highly over-parameterized linear model with a slowly decaying covariance spectrum can perfectly interpolate training noise while maintaining test error near the irreducible noise level—a clear illustration of benign overfitting. However, when effective rank decreases through rapid eigenvalue decay, strong feature correlations, or insufficient feature dimensions, the minimum-norm solution becomes unstable and test performance deteriorates significantly.

The results highlight the delicate balance between model capacity and data distribution required for benign overfitting. This provides valuable insight into why heavily overparameterized models like deep neural networks can generalize well despite fitting noise—they may naturally create conditions analogous to those identified in these linear models.