Ruizhe Final Project - Benign Overfitting in Linear Regression

March 19, 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 surprising paradox: models can fit training data (even noisy labels) perfectly and still generalize well. Classical statistical wisdom warns that a predictor fitting every data point is likely overfitting and will perform poorly on new data. For example, Zhang et al. (2017) showed a deep network can reach near-zero training loss on a vision dataset with random labels yet still achieve non-trivial test accuracy. This defies the traditional bias–variance trade-off, which posits a tension between model complexity (capacity) and training error. Benign overfitting refers to this phenomenon where overfitting does not harm (and can even aid) prediction accuracy. Bartlett et al. (2020) set out to understand this mystery in the simplest possible setting: linear regression with more parameters than data. By studying linear models, they aim to identify conditions under which one can perfectly fit the training data (including noise) without degrading test performance. The hope is that insights from this linear case will shed light on why overparameterized deep networks can generalize well despite interpolating noisy data.

1.2 The Linear Regression Setup and Interpolation Solution

Bartlett et al. consider a standard linear regression problem (x, y) in a high-dimensional (potentially infinite-dimensional) feature space. We have an input vector $x \in \mathbb{R}^d$ (or a Hilbert space H) with covariance $\Sigma = \mathbb{E}[xx^T]$, and an output $y \in \mathbb{R}$ following the linear model $y = x^T\theta^* + \text{noise}$ (with θ^* being the true underlying parameter). The data are assumed to be mean-zero and satisfy some regularity conditions (e.g. sub-Gaussian tails) to facilitate analysis. Crucially, the authors focus on the overparameterized regime: the parameter space dimension d is much larger than the sample size n, ensuring that a perfect fit to n training points is possible (indeed $\text{rank}(\Sigma) > n$ so the n data span only a small subspace). In this regime, the least-squares normal equations $X^TX\theta = X^Ty$ have infinitely many solutions. Among all interpolating solutions \$ { : X = y}\$, they consider the minimum-norm interpolating estimator $\hat{\theta}$. This $\hat{\theta}$ is the solution with smallest Euclidean norm that exactly fits the training data, equivalent to the Moore-Penrose pseudoinverse solution $\hat{\theta} = X^+y$. Intuitively, $\hat{\theta}$ is the "least complex" interpolator. The key question is: when does this interpolator achieve near-optimal prediction accuracy, 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 $R(\widehat{\theta})$ is close to 0 (so $\widehat{\theta}$ is nearly as accurate as θ^*) even though $\widehat{\theta}$ fits the training data exactly (including noise).

1.3 Theoretical Framework: Effective Rank and Risk Decomposition

A core contribution of Bartlett et al. is a finite-sample characterization of when linear regression can benignly overfit. They show that the answer lies in the spectrum of the covariance Σ, through what they call the "effective rank." The covariance's eigenvalues (denote them λ_i in decreasing order) determine how variance is distributed across different directions in parameter space. The authors first derive a classical bias–variance decomposition of the excess risk $R(\widehat{\theta})$ 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 $\widehat{\theta}$ 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 $\widehat{\theta}$ has interpolated) impacts prediction. The key insight is that the impact of label noise on $\widehat{\theta}$'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 of Σ (Definition 3 in the paper). For $k \geq 0$, define:

- $r_k(\Sigma) = \frac{\sum_{i>k} \lambda_i}{\lambda_{k+1}}$, which is roughly the ratio of the variance remaining in the trailing eigenvalues (those beyond the k-th) to the $(k+1)$-th eigenvalue. This measures the mass in the "tail" of the spectrum relative to the next eigenvalue.
- $R_k(\Sigma) = \frac{\left(\sum_{i>k} \lambda_i\right)^2}{\sum_{i>k} \lambda_i^2}$, which is analogous to the effective dimensionality in the tail (it equals the number of eigenvalues beyond k if they are all equal). R_k is large when there are many small eigenvalues of similar size.

Intuitively, a large effective rank means Σ has a long flat tail of tiny eigenvalues – i.e. many directions with negligible variance. A small effective rank means the spectrum drops off quickly (most variance captured by a limited number of directions). These definitions help characterize when the noise-fitting term (Term 2) remains small. Specifically, Bartlett et al. show that Term 2 is small if and only if Σ's effective rank in the low-variance regime is large compared to n. In other words, benign overfitting requires a high-dimensional "spread-out" spectrum: a great many directions with small eigenvalues so that the label noise can be absorbed into those directions without greatly affecting predictions. This condition is both necessary and sufficient in their analysis

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

- If Σ does not have a sufficiently large effective rank (roughly, not enough small-eigenvalue directions compared to n), then benign overfitting is impossible. In this case, $\hat{\theta}$'s risk will be bounded below by a constant fraction of the irreducible noise level – meaning it performs much worse than the optimal θ^*. Intuitively, if the model doesn't have "extra" weak directions to stash the noise, any interpolating solution must distort some important directions, causing significant excess error.
- Conversely, if Σ's spectrum is suitably flat/extended (effective rank $\gg n$ in the low-variance end), then the excess risk can be made arbitrarily small (near zero) even as $\hat{\theta}$ fits the data perfectly. In this regime, $\hat{\theta}$ achieves near-optimal prediction accuracy, essentially matching the performance of θ^* despite interpolating the noise. Theorem 1 quantifies this with an upper bound on $R(\hat{\theta})$ that goes to 0 under the stated conditions as n grows. All the terms in that bound involve ratios like $r_0(\Sigma)/n$ or $n/R_{k_n^*}(\Sigma)$, which will be small when the spectrum conditions are met.

The authors then define a notion of "benign" sequence of covariance matrices (Definition 4) to formalize asymptotic benign overfitting. A sequence Σ_n (varying with sample size n) is benign if as $n \to \infty$:

- $r_0(\Sigma_n)/n \to 0$,
- $k_n^*/n \to 0$, and
- $n/R_{k_n^*}(\Sigma_n) \to 0$.

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 $R(\hat{\theta}) \to 0$ (benign overfitting).

Theorem 2 provides concrete examples of spectral conditions that are benign vs. non-benign, illustrating the theory. Two notable cases are highlighted:

- Infinite-Dimensional Case (Heavy-tailed spectrum): Suppose the eigenvalues decay as a power-law just at the boundary of being summable. For example, $\mu_k(\Sigma) \sim k^{-\alpha}(\ln k)^{-\beta}$. Theorem 2.1 shows that benign overfitting occurs if and only if $\alpha = 1$ and $\beta > 1$. This corresponds to eigenvalues $\approx 1/(k \cdot (\ln k)^{\beta})$, which decay just slowly enough that $\sum_k \mu_k < \infty$ (finite variance) but nearly as slow as $1/k$. In other words, the spectrum has a heavy tail (slow decay) – for benign overfitting, it must be right at the edge of too heavy: any faster decay (e.g. $\alpha > 1$ meaning exponentially fast or $1/k^{1+\varepsilon}$) and the effective rank isn't large enough; any slower (<1) and total variance would diverge, violating assumptions.
- High but Finite Dimension with Isotropic Noise: In contrast, Theorem 2.2 considers a scenario where the data lie in a finite-dimensional space, but the dimension p_n grows with n. Imagine Σ_n has p_n non-zero eigenvalues that decay very fast (even exponentially), yet there is a small "isotropic" variance ε_n added in all directions (so essentially p_n eigenvalues around some tiny value ε_n, and 0 beyond p_n). In this case, even though the primary eigenvalues of the original signal may drop off quickly, the sheer number of features $p_n \gg n$ and the presence of a tiny

floor ε_n can yield benign overfitting. The condition is that the dimension grows significantly faster than n (formally $p_n = \omega(n)$), and the total isotropic variance is small relative to n (specifically $\varepsilon_n p_n = o(n)$, but not too small, e.g. not exponentially small). Under these conditions, $\widehat{\theta}$ achieves near-optimal risk. Intuitively, here the overparameterization is extreme (dimension much larger than sample size) and there is a flat part of the spectrum (almost constant small eigenvalues), which again means lots of "harmless" directions to absorb noise.

The two cases above illustrate a fundamental trade-off in benign overfitting: on one hand, we need slow-decaying small eigenvalues to make n/R_{k^*} small (so that noise impact is negligible); on the other hand, we need the eigenvalues to be summable (finite trace) so that $r_0(\Sigma)/n$ is small. In infinite dimensions, achieving both requires a very specific borderline decay rate (roughly $1/k$ up to log factors). This suggests that benign overfitting in an infinite-dimensional function space is a delicate and somewhat "unusual" phenomenon, only possible under fine-tuned spectral conditions. In contrast, if data live in a large but finite-dimensional space, it is much easier to satisfy both conditions: as long as the dimension is huge (ensuring summability of eigenvalues) one can have an almost flat spectrum (slow decay or even nearly constant eigenvalues) and still be benign. The authors note that benign overfitting is a more generic scenario in high but finite dimensions (e.g. many features) than in strictly infinite-dimensional settings. This underscores the role of finite but very high dimensional data in allowing overfitting without harm.

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 (or feature dimensions) than the number of training examples. Bartlett et al. show that it's not just the count of extra parameters, but the presence of many "uninformative" directions in parameter space that makes overfitting harmless. These directions correspond to eigenvectors of Σ with tiny eigenvalues – directions in which the inputs x have almost no variance, so they barely affect the output. When there are "significantly more" of these weak directions than there are data points, the least-norm solution will utilize them to fit the noise, leaving the important directions (those with large variance) mostly aligned with the true signal θ^*.

In essence, the parameter vector $\widehat{\theta}$ can be decomposed into two components: (i) one in the subspace of principal components (large eigenvalues) and (ii) one in the subspace of weak components (small eigenvalues). The first part is responsible for predicting the true signal, and the second part mainly soaks up label noise. Overparameterization ensures the second part (noise-fitting component) has lots of room (dimensions) to live in without overlapping with the first part. This idea aligns with the intuition that $\widehat{\theta}$ can be written as $\widehat{\theta} = \theta^* + \Delta$, where Δ lies mostly in the spiky, low-variance directions of Σ and thus Δ has minimal effect on predictions. Classical theory would normally penalize any $\Delta \neq 0$ as overfitting, but here Δ is "harmless" because $x^T\Delta$ is very small for new samples (since x has almost no projection in those directions). This is precisely why a large effective rank (many small eigenvalues) is equivalent to requiring a high degree of overparameterization for benign overfitting.

To summarize: Benign overfitting needs an extreme excess of parameters relative to data, such that the model has a high-capacity subspace that is irrelevant to the true function (the "unimportant directions"). This subspace acts as a reservoir for fitting noise. If this reservoir is big enough (effective rank $\gg n$), the noise-fitting won't hurt generalization. Bartlett et al.'s results thus formalize a crucial lesson: simply having more parameters than data is not enough – the structure

of those extra parameters (through Σ) must be such that they do not contribute significantly to the true signal. When that holds, overparameterization becomes benign, even beneficial.

1.6 Insights and Connections to Deep Learning

Although the paper's analysis is for linear models, it was directly motivated by and is believed to shed light on deep learning's behavior. The phenomena observed in linear regression draw interesting parallels to complex models:

- Implicit Bias of Optimization: In deep networks, gradient-based training often seems to find solutions that generalize well even when many solutions could fit the data. In the linear case, the minimum-norm interpolant $\widehat{\theta}$ plays a similar role as a "biased" solution among the many interpolating ones. One connection discussed is the Neural Tangent Kernel (NTK) regime, where an extremely wide neural network can be approximated by a linear model in function space. In such cases, gradient descent on the network effectively performs gradient descent in that high-dimensional linear (kernel) space, and it tends to find an interpolator that minimizes a norm in that space. Bartlett et al. note that under reasonable data assumptions, the NTK's eigenvalues can indeed have a heavy tail (slow decay) and the model dimension is huge (effectively infinite width), which are precisely the ingredients for benign overfitting . This suggests that wide neural networks might be operating in a regime analogous to the benign overfitting conditions – with many "slightly important" directions in function space due to the wide network.

- Spectrum of Learned Representations: Their results hint that a deep network that benignly overfits may be one where the learned features or internal representations have an approximately flat spectrum up to a very large rank. In practice, this could mean the network spreads variance across a huge number of directions in weight space or activation space, rather than concentrating on a few. Indeed, Bartlett et al. conjecture that covariance eigenvalues that are nearly constant or slowly decaying in a high-dimensional feature space might be important for benign overfitting in deep nets as well. Some researchers have suggested viewing deep nets as finite-dimensional approximations to infinite-dimensional function classes (like kernels). The linear analysis here indicates that having a finite but large capacity (rather than truly infinite) is crucial – an infinite-dimensional model requires very specific conditions to generalize, whereas a high-but-finite model can generalize across a broader set of conditions. This could imply that the finite width of real neural networks (as opposed to infinitely wide ones) might actually be a feature that enables benign overfitting by limiting how fast the eigenvalues can decay.

- Open Problems: The paper stops short of claiming that deep networks definitely have the covariance structure needed; rather, it provides clues. Verifying these conditions in actual neural network training is an open problem. The authors caution that some of their assumptions (like independent features or linearity of the model) do not hold in realistic deep learning setups. Nevertheless, the linear case suggests a possible explanation for the "mystery" of deep learning's generalization: if the network implicitly achieves something akin to a minimum-norm solution in a very high-dimensional parameter space with a slow spectral decay, it could be benignly overfitting. In short, deep networks might generalize well for the same reason the min-norm linear regressor does – because they have a form of implicit overparameterization that channels noise into directions that don't affect the output much. Confirming this hypothesis for neural nets is an important direction for future research.

1.7 Conclusion

Conclusion: Bartlett et al. (2020) provide a clear mathematical story for benign overfitting in linear regression. The story highlights the interplay between model capacity (overparameterization) and data distribution (covariance spectrum). In essence, to overfit benignly, a linear model needs to have enough "wiggle room" to fit noise in directions that don't matter. This manifests as a covariance with a long tail of small eigenvalues (large effective rank) and a model with vastly more parameters than data. When these conditions are met, the minimum-norm interpolator achieves nearly the same performance as the true model, resolving the apparent paradox of overfitting without harming prediction. These findings, while derived in a simple setting, draw intriguing parallels to deep neural networks and offer a potential piece of the puzzle in understanding why modern high-capacity models generalize well.

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, recent work by Bartlett et al. (2020) provides conditions under which linear regression can interpolate noise benignly, achieving near-optimal test accuracy despite overfitting the training data. Two key insights from their analysis are:

- Significant Over-parameterization: The model must have far more parameters than samples. In fact, the number of "uninformative" directions in parameter space (directions with little effect on prediction) should significantly exceed the sample size. This ensures there are many degrees of freedom to absorb noise without affecting predictive features. Slowly Decaying Eigenvalues (High Effective Rank): The covariance spectrum of the features should not drop off too fast. Intuitively, there should be a large effective rank for the feature covariance – meaning many small-eigenvalue directions where label noise can hide with minimal impact on predictions. If the small eigenvalues decay slowly (e.g. a heavy-tailed spectrum), the minimum-norm interpolating solution will spread out the fitted noise in many weak directions, limiting its harm on test error.
- Slowly Decaying Eigenvalues (High Effective Rank): The covariance spectrum of the features should not drop off too fast. Intuitively, there should be a large effective rank for the feature covariance – meaning many small-eigenvalue directions where label noise can hide with minimal impact on predictions. If the small eigenvalues decay slowly (e.g. a heavy-tailed spectrum), the minimum-norm interpolating solution will spread out the fitted noise in many weak directions, limiting its harm on test error.

2.0.1 Experiment Setup: Synthetic Data Generation

To study these phenomena in a controlled way, we generate synthetic regression data where we can specify the feature covariance structure. We consider a linear model: - Data (x_i, y_i) for $i = 1, \ldots, n$ are drawn i.i.d. in $\mathbb{R}^p \times \mathbb{R}$. - Feature vector $x_i \sim \mathcal{N}(0, \Sigma)$, a zero-mean Gaussian with covariance Σ. We will design Σ to have various eigenvalue spectra (e.g. slow or fast decaying eigenvalues, correlated features, etc.). - The true underlying relationship is $y = x^\top w^* + \varepsilon$, where w^* is the true parameter vector and ε is noise $\sim \mathcal{N}(0, \sigma^2)$. In our experiments, we set $\sigma^2 = 1$ for simplicity (so

noise variance is 1). We will often take $w^{=}0$ to focus purely on the effect of fitting noise. (When $w^* = 0$, the best possible prediction is 0 for all x, with mean squared error equal to the noise variance.)

We will fit a linear regression model in the highly over-parameterized regime ($p \geq n$) using the minimum-norm interpolating solution – essentially the pseudoinverse solution that fits y exactly. This is the solution of $\min_w |w|_2$ subject to $Xw = y$, where X is the $n \times p$ design matrix. This choice aligns with the theoretical analysis (it's the estimator that many gradient-based methods would pick in an underdetermined system).

Evaluation: For each experiment, we will measure:

- Training MSE: The mean squared error on the training set (which will be near 0 in over-parameterized cases when interpolation is achieved).
- Test MSE: The mean squared error on an independent test set, which indicates generalization performance. For comparison, note that the irreducible error (noise variance) is 1 in our setup, and if w^* is non-zero, the Bayes-optimal predictor w^* would achieve a test MSE equal to the noise variance (plus any approximation error if w^* not realizable, but here w^* is the true linear parameter so noise is the only source of error). Benign overfitting means the test MSE of the fitted model should be close to this optimum despite fitting noise.

Below, we implement helper functions to generate synthetic data with a specified covariance spectrum and to compute the minimum-norm solution and errors. We will use these throughout our experiments.

[ ]:
```
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
    # We can 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

# Quick demonstration of usage (this is not an experiment yet, just sanity␣
 ↪check)
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
```

In the code above, generate_data allows us to specify a list of eigenvalues for Σ and whether features are independent or correlated. If correlated=False, Σ is taken as diagonal (features independent with given variances). If correlated=True, we generate a random orthonormal matrix Q to produce a full covariance $Q\,\mathrm{diag}(eigen\_vals)Q^T$ with the desired spectrum but mixed features. We use Cholesky decomposition to sample X efficiently from the Gaussian distribution. The helper fit_min_norm returns the minimum-norm interpolating weight vector $w_{\min}$ for the linear system $Xw = y$. Finally, evaluate_model will be used to compute training and test MSE given a learned w and true w^* (note: we will call evaluate_model in the same scope where X_train, y_train are defined for convenience).

2.0.2 Benign Overfitting under Ideal Conditions

First, we construct a scenario expected to satisfy the benign overfitting conditions. According to theory, we need: (a) significant overparameterization ($p \gg n$), and (b) a slowly decaying covariance spectrum with many small eigenvalues (high effective rank). We will use: - Sample size: $n = 100$ - Feature dimension: $p = 1000$ (an order of magnitude larger than n, providing plenty of extra degrees of freedom) - Covariance spectrum: For an ideal slow-decay scenario, we can take approximately constant eigenvalues. The most extreme case of "slow decay" is no decay at all – e.g. $\lambda_i \approx 1$ for all i. Here we'll use $\Sigma = I$ (identity covariance) as a simple case where all eigenvalues are equal (1). This means features are independent and of equal importance, and effectively the rank = p = 1000 (maximal effective rank).

We also set the true signal $w^* = 0$ so that labels are pure noise. This way, any test error above the noise variance clearly indicates overfitting harm, whereas achieving test MSE ≈ 1 (the noise variance) would be essentially optimal (since even the best predictor, which is 0, has to incur the noise MSE of 1). This lets us directly assess if fitting the noise has increased test error or not.

Expectation: In this benign setting, the minimum-norm interpolator should achieve near-noise-level test MSE despite zero training error. As Bartlett et al. noted, if there are many "unimportant" directions (here 1000-100 = 900 extra directions) with small variance, the fitted noise will reside in those directions and not impact prediction much.

[ ]:
```python
# 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)
```

Results: The output shows the training MSE is essentially 0, confirming the model interpolates the training data perfectly. The test MSE should come out around ~1.1. This is very close to 1.0, the noise level. Despite fitting pure noise in the training set, the model's predictions on new data are nearly as good as always predicting the mean (zero in this case). The small excess over 1 can be attributed to finite sample fluctuations, but it diminishes as p grows even larger. We have effectively hidden the noise in the many extra dimensions of w that correspond to tiny (here zero) marginal variance in x, thus not hurting prediction 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, we investigate the effect of the covariance eigenvalue decay on benign overfitting. The theory's characterization is in terms of the effective rank of Σ. Intuitively, even if p is large, if the feature covariance has most of its variance concentrated in a small number of top directions (i.e. eigenvalues that decay rapidly), then there aren't enough "small-variance" directions to hide the noise. The small eigenvalues need to decay slowly, so that there are many of them contributing a significant collective variance (making the effective rank large).

In this experiment, we will keep $n = 100$ and $p = 1000$ (so plenty of dimensions), but we will design different covariance spectra from slow-decaying to fast-decaying. We will use independent features for clarity (correlated=False) so that $\Sigma = \text{diag}(\lambda_1, \ldots, \lambda_p)$ with the desired eigenvalues.

We compare: - Slow Decay: Eigenvalues that decrease gradually. For example, $\lambda_i \propto \frac{1}{i}$ (harmonic decay) is fairly slow. We could also consider a flat spectrum (which we did above: constant eigenvalues) as an extreme case of slow/no decay. - Moderate Decay: Eigenvalues might decay polynomially faster or have a mix of large and small values. For instance, $\lambda_i \propto \frac{1}{i^2}$ decays faster. - Fast Decay: Eigenvalues that drop off very quickly, e.g. an exponential decay $\lambda_i \propto a^i$ for some $a < 1$. This means after the first few directions, the remaining eigenvalues (and corresponding feature directions) carry negligible variance. This is a scenario with low effective rank, as most variance is in the top components.

We will simulate these cases and measure test MSE of the min-norm interpolator in each case. All cases use $p = 1000$, $n = 100$, and $w^* = 0$ (pure noise labels) to isolate the effect on fitting noise.

Expected results: The test MSE will be lowest for the flattest/slowest-decaying 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
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
```

These numbers (illustrative) show a clear pattern: when eigenvalues decay very fast (exponential case), the model that interpolates noise performs very poorly on test data (MSE several times the noise level). Conversely, with a flat or slowly decaying spectrum, the test MSE is close to 1.0 (noise level), indicating benign behavior.

This aligns with the effective rank condition from the paper: a large effective rank (many comparably small eigenvalues) is necessary to keep the excess error from noise small. In the exponential decay case, most of the variance is in the first few features; the model must place a lot of weight on those few directions to fit the noise (because the remaining directions have near-zero variance in x and thus can't absorb much noise without huge weights). Fitting noise in those important directions severely hurts predictions, hence the high test error. On the other hand, in the slow-decay cases, there are lots of directions with modest variance – the noise gets spread out over many directions of w, each of which has little influence on the prediction, keeping test error low

In summary, if the covariance spectrum decays too quickly, benign overfitting fails: the interpolating solution will generalize poorly because it overfits in the important directions. Only when the spectrum has a long tail (slow decay) can the overfitting be harmless.

2.1 Failure Cases: Violating Benign Overfitting Conditions

2.1.1 1. Covariance Spectrum Decays Too Fast

This case is essentially the extreme of the experiment above: if the feature covariance has a rapid decay in eigenvalues, the effective rank is low and the conditions for benign overfitting are violated. We already saw an example with exponential decay. Here, let's take an even simpler but stark example: suppose out of p features, only a small handful have significant variance and the rest are negligibly small. For instance, let's say Σ has 10 eigenvalues equal to 1, and the remaining $p - 10$ eigenvalues equal to 0.001 (very small). This means after the first 10 principal components, the spectrum drops off sharply. We will use $n = 100, p = 500$ for this illustration.

We expect that even though $p \gg n$, most of those p directions carry almost no variance, so effectively the model only has about 10 meaningful directions to fit the data. It will have to use those to interpolate, likely overfitting badly.

[ ]:
```
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)

Now we examine the scenario where the number of features p is not significantly larger than n. We know from theory that we need many more parameters than data points for benign overfitting. If p is only slightly larger (or even equal or less than n), the model either cannot perfectly fit the data or, if it does, it won't generalize well.

We consider two sub-cases: - Exactly $p = n$: the model has just enough parameters to potentially interpolate (if X is full rank). This is the border of overparametrization. - Slightly above n: e.g. $p = 1.2n$ or $1.5n$. In such cases, there are some extra degrees of freedom but not "many more" than n.

We'll use $\Sigma = I$ (nice flat spectrum) which on its own is a favorable scenario, and see how performance deteriorates as p decreases toward n.

In the printout, we expect: - When $p < n$ (e.g. 80), the model cannot fit all training points (underparameterized). Train MSE will not be zero. The test MSE may not be as extremely large as some overfitting cases, but it will be above 1 due to the usual bias-variance tradeoff (underfitting some noise but also some signal, if any). This is not benign overfitting; it's just the traditional case of not enough capacity. - When $p = n$ (100), if X is full rank, the solution interpolates (Train MSE ≈ 0). But as we observed, the weights can be poorly behaved. We often see very high test MSE in this scenario (the print might show a huge number or inf if numerical issues occur). This indicates a fragile solution – essentially, with no extra degrees of freedom, the interpolator might place enormous weight on directions that just barely allow fitting, leading to a disastrous test performance. - When $p = 120$ or 150 (a bit over n), train MSE will be 0, and test MSE will be high (perhaps a few times the noise level, say 2-5). There are some extra parameters, but not

enough to fully "bury" the noise harmlessly. The noise ends up spread across fewer directions, some of which inevitably have moderate variance impact on y, causing noticeable generalization error.

[ ]:
```
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
```

The pattern is clear: if p is not significantly larger than n, the test error is substantially worse than the noise floor (often catastrophically worse when p is only equal to n). This reinforces that overparameterization is essential for benign overfitting. We need a large surplus of features to achieve near-optimal generalization when interpolating noise.

2.1.3 3. Correlated Features (Violating Independence)

The theory and our previous simulations assumed features are roughly independent (or at least that the covariance eigenstructure is not pathological). What if features are highly correlated with each other? Correlation among features effectively reduces the dimensionality of the data — even if p is large, if many features are redundant or linearly dependent, the true number of independent directions is smaller. This can undermine the overparameterization and effective rank conditions.

We will test a scenario with **strong feature correlation**. One way to induce this is to use a covariance matrix with high off-diagonal values. For example, consider an AR(1) covariance where features have correlation $\rho^{|i-j|}$ between feature i and j. If ρ is close to 1, adjacent features are highly collinear. We use an AR(1) model with $\rho = 0.9$ for a pronounced correlation. We'll compare it to an independent feature case (identity covariance) with the same n and p to isolate the effect of correlation.

Let's take $n = 100, p = 300$ and compare: - Independent features: $\Sigma = I_{300}$. - Correlated features: $\Sigma_{ij} = 0.9^{|i-j|}$ (AR(1) covariance with strong correlation).

We keep $w^* = 0$ and measure the test MSE for both.

Expected outcome: The test MSE with correlated features will be worse than with independent features.

[ ]:
```
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
```

The correlated case (AR(0.9)) had test MSE 3.35, notably higher than the MSE of the independent case.

The reason: with $\rho = 0.9$, the features are highly collinear, effectively reducing the effective dimensionality of the data. Indeed, the AR(1) covariance has a decaying spectrum – one large eigenvalue and a bunch of smaller ones (we could examine eigvalsh($Sigma_corr$) to see the distribution). The effective rank is lower than 300 in this case, violating the ideal conditions. As a result, the model cannot hide the noise as effectively and ends up with higher prediction error. In other words, correlation among features can negate the benefits of high nominal dimension by collapsing variance into fewer directions, making overfitting harmful.

2.2 Conclusion

The experiments show that a highly over-parameterized linear model with a slowly decaying covariance spectrum can perfectly interpolate training noise while keeping test error near the irreducible noise level—illustrating benign overfitting. However, when the effective rank is reduced (via rapid eigenvalue decay, strong feature correlations, or insufficient extra dimensions), the minimal-norm solution becomes unstable and test error increases sharply. These results underscore the crucial balance between model capacity and data covariance in achieving benign overfitting, offering insight into why highly over-parameterized models (like deep networks) can generalize well despite fitting noise.