Linear Regression: From Least Squares to Bayesian Inference

MEAM/EE/CBE/MSE 4600: AI for Science and Engineering

Week 4: Linear and Logistic Regression (Part 1)

Instructor: Paris Perdikaris
Reference: Murphy, K.P. (2022). *Probabilistic Machine Learning: An Introduction*, Chapter 11

Overview and Motivation

Linear regression is one of the oldest and most fundamental techniques in statistics and machine learning. Despite its simplicity, it remains remarkably useful and serves as a foundation for understanding more complex models.

Why study linear regression?

1. **Ubiquitous in practice**: Predicting house prices, stock returns, sensor readings, physical quantities
2. **Interpretable**: Coefficients have direct physical meaning (e.g., "each additional square foot adds $200 to house price")
3. **Foundation for neural networks**: A neural network with no hidden layers and identity activation is exactly linear regression
4. **Connects our estimation theory to practice**: We will see MLE leads to least squares, MAP leads to ridge/lasso, Bayesian inference leads to uncertainty quantification
5. **Analytically tractable**: Closed-form solutions enable deep understanding of what algorithms are actually doing

Connection to previous lectures:

- In Week 3, we learned MLE, MAP, and Bayesian inference for simple distributions (Bernoulli, Gaussian)
- Now we apply these same principles to the *supervised learning* setting where we want to predict y from $\mathbf{x}$
- The mathematical machinery is identical—only the model changes

Lecture 1: Linear Regression Fundamentals and Maximum Likelihood

1. The Linear Regression Model

1.1 Problem Setup

Supervised Learning Goal: Given input features $\mathbf{x} \in \mathbb{R}^D$ and a target $y \in \mathbb{R}$, learn a function $f(\mathbf{x})$ that predicts y well on new, unseen inputs.

The Linear Assumption: We assume the expected value of y is a linear function of $\mathbf{x}$:

$$\mathbb{E}[y|\mathbf{x}] = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + \mathbf{w}^\top \mathbf{x}$$

This is a strong assumption! It says that:

- Each feature x_d contributes additively to the prediction
- The contribution of each feature is proportional to its value (scaled by w_d)
- There are no interaction terms like $x_1 \cdot x_2$

Probabilistic Formulation: To apply our estimation framework, we need a full probability model. We assume observations are corrupted by Gaussian noise:

$$\boxed{p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + \mathbf{w}^\top \mathbf{x}, \sigma^2)}$$

where $\boldsymbol{\theta} = (w_0, \mathbf{w}, \sigma^2)$ are all the model parameters.

Equivalent Formulation: We can write this as:

$$y = w_0 + \mathbf{w}^\top \mathbf{x} + \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

> **Interpretation:** The target y equals a deterministic linear function of the features, plus random Gaussian noise. The noise ϵ captures everything our linear model cannot explain: measurement error, missing features, inherent randomness, model misspecification, etc.

1.2 Terminology

Term	Symbol	Meaning

Weights (regression coefficients)	$\mathbf{w} = (w_1, \ldots, w_D)^\top$	Each w_d is the expected change in y per unit change in x_d, holding other features fixed
Bias (intercept/offset)	w_0	The predicted output when all inputs are zero; captures the "baseline"
Noise variance	σ^2	Irreducible uncertainty—variance of predictions even with perfect knowledge of $\mathbf{w}$
Features (covariates, predictors)	$\mathbf{x}$	Input variables we observe and use for prediction
Target (response, label)	y	Output variable we want to predict
Residual	$r_n = y_n - \hat{y}_n$	Difference between observed and predicted value

1.3 Matrix Notation

Working with individual examples is cumbersome. Matrix notation makes everything cleaner and enables efficient computation.

Given N training examples $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, we stack them into matrices:

Design Matrix (with bias column):

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1D} \\ 1 & x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$$

Each row is one training example (with a leading 1 for the bias). Each column (after the first) corresponds to one feature.

Target Vector:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N$$

Weight Vector (including bias):

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix} \in \mathbb{R}^{D+1}$$

The column of 1s in $\mathbf{X}$ absorbs the bias term into the weight vector, so we can write everything uniformly.

The model for all examples simultaneously:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

where $\hat{y}_n = \mathbf{w}^\top \mathbf{x}_n$ is the predicted value for example n.

> **Why this notation matters:** The expression $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ computes all N predictions in one matrix-vector multiplication. This is not just notational convenience—it enables vectorized computation that runs 100-1000x faster than loops in Python/NumPy.

---

2. Maximum Likelihood Estimation (Least Squares)

2.1 Setting Up the Likelihood

We want to find the parameters $\mathbf{w}$ that best explain our observed data. Following our MLE framework from Week 3, we maximize the likelihood of the data.

Step 1: Write the likelihood for one example

For a single observation (x_n, y_n):

$$p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \mathbf{w}^\top \mathbf{x}_n)^2}{2\sigma^2}\right)$$

Step 2: Write the likelihood for all examples

Assuming the examples are independent and identically distributed (i.i.d.):

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2)$$

We can also write this in matrix form as:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}_N)$$

This is a multivariate Gaussian with mean $\mathbf{X}\mathbf{w}$ and diagonal covariance $\sigma^2 \mathbf{I}_N$ (independent noise for each example).

2.2 Deriving the Negative Log-Likelihood

Taking the log makes the product become a sum:

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \sum_{n=1}^{N} \log \mathcal{N}(y_n|\mathbf{w}^\top \mathbf{x}_n, \sigma^2)$$

$$= \sum_{n=1}^{N} \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{(y_n - \mathbf{w}^\top \mathbf{x}_n)^2}{2\sigma^2} \right]$$

$$= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

The **negative log-likelihood (NLL)** is:

$$\text{NLL}(\mathbf{w}, \sigma^2) = \frac{N}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

2.3 The Key Insight: MLE = Least Squares

To find the MLE for $\mathbf{w}$, we minimize the NLL. Looking at the expression above:

- The first term $\frac{N}{2} \log(2\pi\sigma^2)$ does not depend on $\mathbf{w}$
- The second term is proportional to $\sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$

the second term is proportional to $\sum_{n=1}^{N}(y_n - \mathbf{w}^\top \mathbf{x}_n)^2$

Therefore, minimizing NLL w.r.t. $\mathbf{w}$ is equivalent to minimizing the **Residual Sum of Squares (RSS)**:

$$\boxed{\text{RSS}(\mathbf{w}) = \sum_{n=1}^{N}(y_n - \mathbf{w}^\top \mathbf{x}_n)^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2}$$

> **This is the fundamental connection:** Maximum likelihood estimation with a Gaussian noise model is mathematically equivalent to **ordinary least squares (OLS)**—minimizing the sum of squared prediction errors. The probabilistic and optimization viewpoints lead to the same solution!

2.4 Deriving the Closed-Form Solution

Let us find the minimizer of RSS. First, expand in matrix form:

$$\text{RSS}(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Expanding the product:

$$= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{w} - \mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Since $\mathbf{y}^\top \mathbf{X}\mathbf{w}$ is a scalar, it equals its transpose $\mathbf{w}^\top \mathbf{X}^\top \mathbf{y}$:

$$= \mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Now take the gradient with respect to $\mathbf{w}$. Using matrix calculus identities:

- $\nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{a}) = \mathbf{a}$ for any vector $\mathbf{a}$
- $\nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{A}\mathbf{w}) = 2\mathbf{A}\mathbf{w}$ for any symmetric matrix $\mathbf{A}$

Therefore:

$$\nabla_{\mathbf{w}} \text{RSS} = -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{w}$$

Setting the gradient to zero:

$$-2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{0}$$

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

These are the famous **normal equations**. If $\mathbf{X}^\top \mathbf{X}$ is invertible, we can solve:

$$\boxed{\hat{\mathbf{w}}_{\text{MLE}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}}$$

The matrix $\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ is called the **(left) Moore-Penrose pseudo-inverse** of $\mathbf{X}$.

2.5 When Does the Inverse Exist?

The matrix $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{(D+1)\times(D+1)}$ is invertible if and only if $\mathbf{X}$ has **full column rank**, meaning its $D+1$ columns are linearly independent.

This requires:

1. $N \geq D + 1$: At least as many examples as parameters
2. No feature is a linear combination of others (no perfect multicollinearity)
3. The constant column (for bias) is not redundant

What if $\mathbf{X}^\top \mathbf{X}$ is singular?

- The solution is not unique—infinitely many $\mathbf{w}$ minimize RSS
- Regularization (ridge regression) fixes this by adding $\lambda \mathbf{I}$
- Use the pseudo-inverse via SVD

2.6 Verifying We Found a Minimum

The Hessian of RSS with respect to $\mathbf{w}$ is:

$$\mathbf{H} = \nabla^2_{\mathbf{w}} \text{RSS} = 2\mathbf{X}^\top \mathbf{X}$$

For this to be a minimum, we need $\mathbf{H}$ to be positive semi-definite. For any vector $\mathbf{v} \neq \mathbf{0}$:

$$\mathbf{v}^\top (\mathbf{X}^\top \mathbf{X})\mathbf{v} = (\mathbf{X}\mathbf{v})^\top (\mathbf{X}\mathbf{v}) = \|\mathbf{X}\mathbf{v}\|_2^2 \geq 0$$

This is always non-negative! And if $\mathbf{X}$ has full column rank, then $\mathbf{X}\mathbf{v} \neq \mathbf{0}$ for $\mathbf{v} \neq \mathbf{0}$, so the Hessian

is strictly positive definite.

> **Conclusion:** The RSS objective is a convex quadratic function. If $\mathbf{X}$ has full column rank, there is a **unique global minimum** given by the normal equations. No local minima to worry about!

2.7 MLE for the Noise Variance

After finding $\hat{\mathbf{w}}_{\text{MLE}}$, we can also estimate σ^2 by minimizing the NLL:

$$\frac{\partial}{\partial \sigma^2}\text{NLL} = \frac{N}{2\sigma^2} - \frac{1}{2(\sigma^2)^2}\sum_{n=1}^{N}(y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n)^2 = 0$$

Solving:

$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n)^2 = \frac{\text{RSS}(\hat{\mathbf{w}})}{N}$$

> **Note:** This is the **biased** MLE for variance (same issue as sample variance). The unbiased estimate divides by $N - (D+1)$ instead of N, accounting for the degrees of freedom lost in estimating $D+1$ parameters.

---

3. Geometric Interpretation

3.1 Regression as Projection

The normal equations have a beautiful geometric interpretation that provides deep insight.

Recall the normal equations: $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$

This says that the **residual vector** $\mathbf{r} = \mathbf{y} - \mathbf{X}\hat{\mathbf{w}}$ is **orthogonal** to every column of $\mathbf{X}$.

What does this mean geometrically?

Think of $\mathbf{y} \in \mathbb{R}^N$ as a point in N-dimensional space. The columns of $\mathbf{X}$ span a $(D+1)$-dimensional subspace (assuming full column rank). Any prediction $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ must lie in this subspace.

Least squares finds the point in the column space of $\mathbf{X}$ that is closest to $\mathbf{y}$.

The closest point is the **orthogonal projection** of $\mathbf{y}$ onto the column space. The residual vector points

from this projection back to $\mathbf{y}$, perpendicular to the subspace.

3.2 The Projection (Hat) Matrix

The predicted values can be written as:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \mathbf{P}\mathbf{y}$$

where the **projection matrix** is:

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$$

This is also called the **hat matrix** because it puts the "hat" on $\mathbf{y}$ to produce $\hat{\mathbf{y}}$.

Properties of the projection matrix:

1. **Symmetric:** $\mathbf{P}^\top = \mathbf{P}$
2. **Idempotent:** $\mathbf{P}^2 = \mathbf{P}$ (projecting twice is the same as projecting once)
3. **Eigenvalues:** All eigenvalues are 0 or 1
4. **Trace:** $\text{tr}(\mathbf{P}) = D + 1$ (the dimension of the column space)

The residuals are:

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I} - \mathbf{P})\mathbf{y}$$

where $\mathbf{I} - \mathbf{P}$ projects onto the orthogonal complement (the "residual space").

3.3 Formulas for Simple Linear Regression

For the 1D case $y = w_0 + w_1 x + \epsilon$, the solution has a particularly elegant form:

$$\hat{w}_1 = \frac{\sum_{n=1}^{N}(x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N}(x_n - \bar{x})^2} = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} = \rho_{XY}\frac{\sigma_Y}{\sigma_X}$$

$$\hat{w}_0 = \bar{y} - \hat{w}_1\bar{x}$$

where $\bar{x} = \frac{1}{N}\sum_n x_n$ and $\bar{y} = \frac{1}{N}\sum_n y_n$ are the sample means, and ρ_{XY} is the sample correlation.

> **Intuition:**
>
> - The slope $\hat{w}_1$ equals the covariance divided by variance. If X and Y covary positively, the slope is positive.
> - The regression line always passes through the "center of mass" $(\bar{x}, \bar{y})$.
> - If X and Y are perfectly correlated ($\rho = \pm 1$), all points lie exactly on the regression line.

---

4. Measuring Goodness of Fit

How do we know if our linear model is any good?

4.1 Residual Analysis

Residuals: $r_n = y_n - \hat{y}_n$

If the model is well-specified, residuals should look like random noise:

- **Mean zero:** $\frac{1}{N}\sum_n r_n \approx 0$ (automatically true for OLS with intercept)
- **No patterns:** Residuals vs. x or vs. $\hat{y}$ should show no systematic structure
- **Constant variance:** Spread of residuals should not depend on x (homoscedasticity)
- **Normality:** For inference, residuals should be approximately Gaussian

Residual plots reveal problems:

- **Curved pattern** implies the true relationship is nonlinear; add polynomial terms or use a different model
- **Fan/funnel shape** implies heteroscedastic noise; consider weighted least squares or variance-stabilizing transform
- **Outliers** are influential points that may distort the fit; consider robust regression
- **Autocorrelation** means for time series, residuals should not be correlated across time

4.2 Coefficient of Determination (R-squared)

The R^2 **score** (also called coefficient of determination) measures the proportion of variance in y explained by the model:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\sum_n (y_n - \hat{y}_n)^2}{\sum_n (y_n - \bar{y})^2}$$

$$\text{RSS} = \sum_n (y_n - \hat{y}_n)^2$$

Decomposition of variance:

$$\underbrace{\sum_n (y_n - \bar{y})^2}_{\text{TSS: Total}} = \underbrace{\sum_n (\hat{y}_n - \bar{y})^2}_{\text{ESS: Explained}} + \underbrace{\sum_n (y_n - \hat{y}_n)^2}_{\text{RSS: Residual}}$$

So: $R^2 = \frac{\text{ESS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$

Interpretation:

- $R^2 = 1$: Perfect fit—model explains all variance in y
- $R^2 = 0$: Model is no better than simply predicting $\bar{y}$ for everything
- $R^2 < 0$: Model is **worse** than the mean baseline (only possible on test data or without intercept)

Connection to correlation: For simple linear regression, $R^2 = \rho_{XY}^2$ (squared correlation coefficient).

> **Caution:** R^2 always increases (or stays the same) when you add more features, even if those features are just noise! For model comparison with different numbers of features, use **adjusted** R^2 or cross-validation.

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(N - 1)}{N - D - 1}$$

4.3 Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2} = \sqrt{\frac{\text{RSS}}{N}}$$

Advantages:

- Same units as y, so directly interpretable ("average error is 5 degrees")
- Comparable across different datasets with same target
- Penalizes large errors more than small errors (due to squaring)

Related metrics:

- **MAE (Mean Absolute Error):** $\frac{1}{N} \sum_{n=1}^{N} |y_n - \hat{y}_n|$ — more robust to outliers

- **MAE (Mean Absolute Error):** $\frac{1}{N}\sum_n |y_n - y_n|$ — more robust to outliers
- **MSE (Mean Squared Error):** RMSE^2 — equals $\hat{\sigma}^2_{\text{MLE}}$

---

5. Basis Function Regression (Feature Engineering)

5.1 The Limitation of Linear Models

A straight line (or hyperplane) cannot capture nonlinear relationships. If the true relationship is $y = \sin(x) + \epsilon$, no linear model will fit well.

But we can fix this while keeping the mathematical framework!

5.2 The Key Idea: Transform the Inputs

Instead of predicting from raw inputs $\mathbf{x}$, we first transform them using **basis functions** $\boldsymbol{\phi}(\mathbf{x})$:

$$p(y|\mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(y|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \sigma^2)$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^M$ maps the original D features to M transformed features.

> **Critical insight:** The model is **nonlinear in the inputs $\mathbf{x}$** but remains **linear in the parameters $\mathbf{w}$**. All our closed-form solutions still apply!

This is called **basis function regression** or **feature engineering**.

5.3 Polynomial Basis Functions

The most common example. For a 1D input x, a degree-d polynomial uses:

$$\boldsymbol{\phi}(x) = [1, x, x^2, x^3, \ldots, x^d]^\top \in \mathbb{R}^{d+1}$$

Example: Quadratic regression ($d = 2$)

$$\hat{y} = w_0 + w_1 x + w_2 x^2$$

This can model parabolic relationships. The design matrix becomes:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}$$

$$\mathbf{\Phi} = \begin{bmatrix} \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

And the solution is identical in form: $\hat{\mathbf{w}} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{y}$

For multivariate inputs, we can include:

- Pure powers: $x_1^2, x_2^2, \ldots$
- Interaction terms: $x_1 x_2, x_1 x_3, \ldots$
- Higher-order interactions: $x_1^2 x_2, x_1 x_2 x_3, \ldots$

The number of features grows rapidly with degree!

5.4 Other Basis Functions

Radial Basis Functions (RBF):

$$\phi_j(x) = \exp\left(-\frac{(x - \mu_j)^2}{2s^2}\right)$$

Each basis function is a Gaussian "bump" centered at μ_j with width s. Using M bumps at different locations can approximate any smooth function.

> **Connection:** RBF networks are a type of single-layer neural network. Gaussian process regression (Week 10) can be viewed as RBF regression with infinitely many basis functions.

Fourier Basis (for periodic functions):

$$\phi_j(x) = \sin(j\omega x), \quad \phi_{j+M}(x) = \cos(j\omega x)$$

Excellent for modeling periodic phenomena (daily temperature cycles, seasonal patterns).

B-splines (for smoothing): Piecewise polynomials joined smoothly at "knots." Provide local control—changing one coefficient only affects the function near that knot.

5.5 The Bias-Variance Tradeoff

Increasing model complexity (more basis functions) has competing effects:

Model Complexity	Bias	Variance	Training Error	Test Error

Too simple (underfitting)	High	Low	High	High
Just right	Moderate	Moderate	Low	**Low**
Too complex (overfitting)	Low	High	Very Low	High

Example: Polynomial degree

- $d = 1$ (line): High bias if true function is curved
- $d = 3$: Often a good tradeoff
- $d = 15$: Can interpolate training data perfectly but oscillates wildly between points

> **The fundamental challenge of machine learning:** We cannot directly observe test error. We must use validation data or cross-validation to estimate it, then select the complexity that minimizes estimated test error.

---

6. Practical Considerations

6.1 Feature Standardization

Before fitting, it is often essential to **standardize** features:

$$\tilde{x}_{nd} = \frac{x_{nd} - \bar{x}_d}{s_d}$$

where $\bar{x}_d$ is the mean of feature d and s_d is its standard deviation.

Why standardize?

1. **Interpretability:** Coefficients become comparable across features (each measures "effect per standard deviation")
2. **Numerical stability:** Avoids ill-conditioning when features have very different scales
3. **Required for regularization:** L1/L2 penalties treat all coefficients equally, so features should be on equal footing
4. **Faster convergence:** For iterative methods like gradient descent

> **Important:** Compute mean and std on **training data only**, then apply the same transformation to

> **Important:** Compute mean and std on **training data only**, then apply the same transformation to test data. Otherwise you leak information from the future!

6.2 Computational Considerations

Do NOT compute $(\mathbf{X}^\top \mathbf{X})^{-1}$ explicitly!

Problems:

- $\mathbf{X}^\top \mathbf{X}$ may be ill-conditioned (small eigenvalues cause numerical instability)
- Matrix inversion is slow: $O(D^3)$
- May not even be invertible

Better approaches:

1. **Solve the linear system directly:** Instead of $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$, solve $(\mathbf{X}^\top \mathbf{X})\mathbf{w} = \mathbf{X}^\top \mathbf{y}$ using Cholesky decomposition.
2. **QR decomposition (recommended for $N > D$):** Decompose $\mathbf{X} = \mathbf{QR}$ where $\mathbf{Q}$ is orthogonal and $\mathbf{R}$ is upper triangular. Then: $\mathbf{w} = \mathbf{R}^{-1}\mathbf{Q}^\top \mathbf{y}$ (solve by back-substitution) Cost: $O(ND^2)$
3. **SVD (most robust, handles rank-deficient cases):** Decompose $\mathbf{X} = \mathbf{USV}^\top$. Then: $\mathbf{w} = \mathbf{VS}^{-1}\mathbf{U}^\top \mathbf{y}$ Cost: $O(ND^2)$ but with larger constant
4. **Iterative methods (for very large D):** Conjugate gradient, GMRES—only require matrix-vector products, good for sparse $\mathbf{X}$.

> **In practice:** `numpy.linalg.lstsq` and `sklearn.linear_model.LinearRegression` use SVD-based methods. Trust the library!

6.3 Interpreting Coefficients

Each coefficient w_d represents the expected change in y per unit change in x_d, **holding all other features constant**.

This is the "partial" interpretation: The effect of x_d after controlling for everything else.

Challenges with interpretation:

1. **Correlated features (multicollinearity):** If x_1 and x_2 are highly correlated, their individual coefficients become unstable. Small changes in data can flip signs. But $w_1 + w_2$ may still be stable.
2. **Different scales:** A coefficient of 1000 for "house size in sq ft" vs 0.1 for "number of bedrooms"

does not mean size matters more. Standardize first for fair comparison.

3. **Causation vs correlation:** $w_d \neq 0$ does not mean x_d causes y. There may be confounders.

Lecture 1 Summary

1. **Linear regression** assumes $\mathbb{E}[y|\mathbf{x}] = \mathbf{w}^\top \mathbf{x}$ with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
2. **MLE with Gaussian noise = Least Squares**: Maximizing likelihood is equivalent to minimizing $\text{RSS} = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$.
3. **Closed-form solution**: $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ (the normal equations).
4. **Geometric interpretation**: $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto the column space of $\mathbf{X}$.
5. **Basis functions** (e.g., polynomials) enable nonlinear modeling while keeping the model linear in parameters.
6. **Model complexity** controls the bias-variance tradeoff—too simple underfits, too complex overfits.
7. **Always standardize features** before fitting; use numerically stable algorithms (QR, SVD).

Lecture 2: Regularization and Bayesian Linear Regression

7. The Overfitting Problem

7.1 What Goes Wrong with MLE?

With many features or high-degree polynomials, MLE can **overfit**:

Symptoms:

- Training error is very low (even zero)
- Test error is much higher
- Coefficients become very large in magnitude
- Model is extremely sensitive to small changes in training data

Why does this happen?

The MLE objective only cares about fitting the training data. With enough parameters, you can fit anything—including the noise! The model "memorizes" rather than "learns."

Extreme example: With N data points and a degree-$(N-1)$ polynomial, you can interpolate exactly through all points. Training RSS = 0. But the polynomial oscillates wildly between points.

7.2 Regularization: The Idea

Add a penalty for model complexity to the objective function:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\underbrace{\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2}_{\text{Data fit}} + \underbrace{\lambda \cdot C(\mathbf{w})}_{\text{Complexity penalty}} \right]$$

The complexity penalty $C(\mathbf{w})$ discourages large or many non-zero coefficients. The hyperparameter $\lambda \geq 0$ controls the tradeoff:

- $\lambda = 0$: Pure MLE (no regularization)
- $\lambda \to \infty$: Ignore data, minimize complexity only

> **From Week 3:** This is exactly MAP estimation! The penalty corresponds to $-\log p(\mathbf{w})$ for some prior.

---

8. Ridge Regression (L2 Regularization)

8.1 The Objective

Ridge regression penalizes the squared L2 norm of the weights:

$$\boxed{\hat{\mathbf{w}}_{\text{ridge}} = \arg\min_{\mathbf{w}} \left[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right]}$$

where $\|\mathbf{w}\|_2^2 = \sum_{d=1}^{D} w_d^2 = \mathbf{w}^\top \mathbf{w}$.

> **Convention:** We typically do NOT penalize the bias term w_0. This is achieved by centering the data (subtracting means) and fitting without intercept, or by excluding w_0 from the penalty.

8.2 Bayesian Interpretation: Gaussian Prior

Ridge regression is **MAP estimation** with a Gaussian prior on the weights:

$$p(\mathbf{w}) = \prod_{d=1}^{D}$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \tau^2\mathbf{I}) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \tau^2)$$

Derivation:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\max_{\mathbf{w}} \left[\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \log p(\mathbf{w})\right]$$

The log-prior is:

$$\log p(\mathbf{w}) = -\frac{D}{2}\log(2\pi\tau^2) - \frac{1}{2\tau^2}\|\mathbf{w}\|_2^2$$

Combining with the log-likelihood and dropping constants:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min_{\mathbf{w}} \left[\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{1}{2\tau^2}\|\mathbf{w}\|_2^2\right]$$

Defining $\lambda = \sigma^2/\tau^2$:

$$= \arg\min_{\mathbf{w}} \left[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2\right]$$

> **The prior variance τ^2 encodes our belief about typical coefficient magnitudes.** Small τ^2 (strong prior) leads to large λ leads to more shrinkage. Large τ^2 (weak prior) leads to small λ leads to closer to MLE.

8.3 Closed-Form Solution

Taking the gradient of the ridge objective:

$$\nabla_{\mathbf{w}} J = -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\mathbf{w} + 2\lambda\mathbf{w} = \mathbf{0}$$

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^\top\mathbf{y}$$

$$\boxed{\hat{\mathbf{w}}_{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}}$$

Key properties:

1. Adding $\lambda\mathbf{I}$ ensures the matrix is **always invertible** (eigenvalues increased by $\lambda > 0$)
2. This stabilizes the solution even with multicollinearity or $D > N$
3. As $\lambda \to 0$: $\hat{\mathbf{w}}_{\text{ridge}} \to \hat{\mathbf{w}}_{\text{MLE}}$
4. As $\lambda \to \infty$: $\hat{\mathbf{w}}_{\text{ridge}} \to \mathbf{0}$

8.4 What Does Ridge Regression Do?

Shrinkage toward zero: All coefficients are pulled toward zero, with larger λ causing more shrinkage.

Using SVD for insight: Let $\mathbf{X} = \mathbf{USV}^\top$ where $\mathbf{S} = \text{diag}(s_1, \ldots, s_D)$ contains the singular values.

The MLE predictions are:

$$\hat{\mathbf{y}}_{\text{MLE}} = \mathbf{X}\hat{\mathbf{w}}_{\text{MLE}} = \sum_{j=1}^{D} \mathbf{u}_j \mathbf{u}_j^\top \mathbf{y}$$

The ridge predictions are:

$$\hat{\mathbf{y}}_{\text{ridge}} = \sum_{j=1}^{D} \mathbf{u}_j \underbrace{\frac{s_j^2}{s_j^2 + \lambda}}_{\text{shrinkage factor}} \mathbf{u}_j^\top \mathbf{y}$$

Interpretation of shrinkage factors:

- Directions with large singular values ($s_j^2 \gg \lambda$): shrinkage approximately 1 (kept)
- Directions with small singular values ($s_j^2 \ll \lambda$): shrinkage approximately 0 (discarded)

> **Ridge regression shrinks the "noisy" directions (small singular values) more than the "signal" directions (large singular values).** This is why it prevents overfitting!

8.5 Effective Degrees of Freedom

How complex is a ridge model? We can measure this by the **effective degrees of freedom**:

$$D \qquad s^2$$

$$\text{df}(\lambda) = \text{tr}(\mathbf{P}_\lambda) = \sum_{j=1} \frac{s_j}{s_j^2 + \lambda}$$

where $\mathbf{P}_\lambda = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top$ is the ridge projection matrix.

- $\lambda = 0$: $\text{df} = D$ (all parameters free, like OLS)
- $\lambda \to \infty$: $\text{df} \to 0$ (all parameters fully constrained)

This gives a continuous measure of complexity between 0 and D.

---

9. Lasso Regression (L1 Regularization)

9.1 The Sparsity Problem

Ridge regression shrinks coefficients toward zero but **never sets them exactly to zero**. For high-dimensional problems, we often want **sparse** solutions where many coefficients are exactly zero.

Why sparsity?

- **Interpretability:** Easier to understand a model with 10 features than 10,000
- **Feature selection:** Automatically identifies which features matter
- **Computational efficiency:** Sparse models are cheaper to evaluate
- **Generalization:** Simpler models often generalize better

9.2 The Lasso Objective

Lasso (Least Absolute Shrinkage and Selection Operator) uses the L1 norm:

$$\boxed{\hat{\mathbf{w}}_{\text{lasso}} = \arg\min_{\mathbf{w}} \left[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 \right]}$$

where $\|\mathbf{w}\|_1 = \sum_{d=1}^{D} |w_d|$.

9.3 Bayesian Interpretation: Laplace Prior

Lasso is MAP estimation with a **Laplace (double-exponential) prior**:

$$p(\mathbf{w}) = \prod_{d=1}^{D} \text{Laplace}(w_d | 0, b) = \prod_{d=1}^{D} \frac{1}{2b} \exp\left(-\frac{|w_d|}{b} \right)$$

$$p(\mathbf{w}) = \prod_{d=1}^{D} \text{Laplace}(w_d|0,b) = \prod_{d=1}^{D} \frac{1}{2b} \exp\left(-\frac{|w_d|}{b}\right)$$

The Laplace distribution has heavier tails than Gaussian but a sharp peak at zero—it "believes" coefficients are likely to be exactly zero or large, not small.

9.4 Why Does L1 Give Sparse Solutions?

Geometric intuition: Consider the constrained formulations:

- **Ridge:** $\min \text{RSS}$ subject to $\|\mathbf{w}\|_2^2 \leq B$ (constraint region is a ball/sphere)
- **Lasso:** $\min \text{RSS}$ subject to $\|\mathbf{w}\|_1 \leq B$ (constraint region is a diamond/cross-polytope)

The optimal solution is where the RSS contours (ellipses) first touch the constraint region.

For L1 (diamond): The corners stick out along the axes. In high dimensions, it is very likely that the first contact point is at a corner, where some coordinates are exactly zero.

For L2 (sphere): The ball is smooth with no corners. Contact can happen anywhere, typically not on the axes.

9.5 Soft Thresholding

For orthonormal features ($\mathbf{X}^\top \mathbf{X} = \mathbf{I}$), the lasso solution has a simple closed form:

$$\hat{w}_d^{\text{lasso}} = \text{sign}(\hat{w}_d^{\text{MLE}}) \cdot \max(|\hat{w}_d^{\text{MLE}}| - \lambda, 0)$$

This is called **soft thresholding**:

- If $|\hat{w}_d^{\text{MLE}}| \leq \lambda$: set $\hat{w}_d = 0$ (eliminate small coefficients)
- If $|\hat{w}_d^{\text{MLE}}| > \lambda$: shrink toward zero by λ (keep but reduce large coefficients)

Compare to **hard thresholding** (subset selection):

- If $|\hat{w}_d^{\text{MLE}}| \leq \lambda$: set $\hat{w}_d = 0$
- If $|\hat{w}_d^{\text{MLE}}| > \lambda$: keep $\hat{w}_d = \hat{w}_d^{\text{MLE}}$ unchanged

> **Lasso does both selection AND shrinkage.** This introduces bias but reduces variance.

9.6 No Closed-Form Solution

Unlike ridge, there is **no closed-form solution** for lasso when features are not orthogonal. The L1 norm is not differentiable at zero.

is not differentiable at zero.

Optimization algorithms:

- **Coordinate descent:** Optimize one w_d at a time; each subproblem has a closed-form (soft thresholding)
- **Proximal gradient descent (ISTA/FISTA):** Gradient step followed by soft thresholding
- **LARS (Least Angle Regression):** Efficiently computes the entire regularization path

9.7 Ridge vs. Lasso: Comparison

Property	Ridge (L2)	Lasso (L1)
Prior	Gaussian	Laplace
Penalty	$\|\mathbf{w}\|_2^2$	$\|\mathbf{w}\|_1$
Solution	Always dense	Often sparse
Feature selection	No	Yes
Closed-form	Yes	No
Correlated features	Keeps all, shrinks together	Arbitrarily picks one
Bias	Some	More (due to shrinkage)
When to use	All features matter, some may be small	Many features irrelevant

9.8 Elastic Net: Best of Both Worlds

Combine L1 and L2 penalties:

$$\hat{\mathbf{w}}_{\text{elastic}} = \arg\min_{\mathbf{w}} \left[\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 \right]$$

Benefits:

- Sparse solutions like lasso
- Handles correlated features better than lasso (groups them together like ridge)
- Strictly convex objective (unique solution)

10. Choosing the Regularization Strength

10.1 Cross-Validation

The regularization hyperparameter λ controls model complexity. We cannot optimize it on training data (would choose $\lambda = 0$). Instead, use **validation data** or **cross-validation**.

K-Fold Cross-Validation:

1. Split data into K folds (typically $K = 5$ or 10)
2. For each candidate λ:
    - For each fold k:
        - Train on all folds except k
        - Evaluate on fold k
    - Average the K validation errors
3. Select $\lambda^* = \arg\min_\lambda \text{CV-error}(\lambda)$
4. Retrain on all data with λ^*

10.2 The Regularization Path

A **regularization path** shows how coefficients evolve as λ changes:

- At $\lambda = \lambda_{\max}$ (computed from data): all coefficients are zero
- As λ decreases: coefficients "turn on" and grow
- At $\lambda = 0$: we get the OLS solution

For lasso, this path is **piecewise linear**—coefficients change linearly between "events" where a coefficient enters or leaves the active set. The LARS algorithm efficiently computes the entire path.

10.3 One Standard Error Rule

Often the CV curve is relatively flat near the minimum. The **1-SE rule** chooses the largest λ (simplest model) whose CV error is within one standard error of the minimum:

$$\lambda_{\text{1SE}} = \max\{\lambda : \text{CV-error}(\lambda) \leq \text{CV-error}(\lambda^*) + \text{SE}(\lambda^*)\}$$

This favors simpler, more interpretable models when the performance

This favors simpler, more interpretable models when they perform comparably.

---

11. Bayesian Linear Regression

11.1 Motivation: Uncertainty Matters

MLE and MAP give **point estimates**—single "best" values for $\mathbf{w}$. But in many applications, we need to know **how confident** we are:

- **Active learning:** Which new data point would be most informative?
- **Bayesian optimization:** Where should we evaluate an expensive function next?
- **Safety-critical systems:** Is the model confident enough to act?
- **Model comparison:** Which model better explains the data?

Full Bayesian inference maintains the entire **posterior distribution** over parameters.

11.2 The Setup

Prior: $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$

Likelihood: $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I})$

Posterior (by Bayes' rule):

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$

11.3 Deriving the Posterior

Since both prior and likelihood are Gaussian, the posterior is also Gaussian (conjugacy!):

$$\boxed{p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)}$$

Posterior covariance:

$$\boldsymbol{\Sigma}_N = \left(\boldsymbol{\Sigma}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X}\right)^{-1}$$

Posterior mean:

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \left(\boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{y} \right)$$

Interpretation:

- The posterior precision (inverse covariance) is the sum of prior precision and data precision
- The posterior mean is a precision-weighted average of prior mean and data-derived mean
- More data leads to posterior concentrating leads to less uncertainty

11.4 Special Case: Zero-Mean Isotropic Prior

If $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Sigma}_0 = \tau^2 \mathbf{I}$:

$$\boldsymbol{\Sigma}_N = \left(\frac{1}{\tau^2} \mathbf{I} + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X} \right)^{-1} = \sigma^2 (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}$$

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \cdot \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{y} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

where $\lambda = \sigma^2 / \tau^2$.

> **Key insight:** The posterior mean $\boldsymbol{\mu}_N$ equals the ridge regression estimate! But now we also have the posterior covariance $\boldsymbol{\Sigma}_N$ which quantifies uncertainty.

11.5 Sequential Bayesian Updates

A beautiful property of Bayesian inference: **the posterior from one dataset becomes the prior for the next**.

If we observe data in batches $\mathcal{D}_1, \mathcal{D}_2, \ldots$:

$$p(\mathbf{w} | \mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2 | \mathbf{w}) \cdot \underbrace{p(\mathbf{w} | \mathcal{D}_1)}_{\text{new prior}}$$

This enables **online learning**: update beliefs incrementally as new data arrives, without reprocessing all historical data.

11.6 Posterior Predictive Distribution

To predict at a new input $\mathbf{x}_*$, we **marginalize** over our uncertainty in $\mathbf{w}$:

$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, \mathbf{w})\, p(\mathbf{w}|\mathcal{D})\, d\mathbf{w}$$

This is an integral of a Gaussian (y_* given $\mathbf{w}$) weighted by another Gaussian (posterior over $\mathbf{w}$). The result is also Gaussian:

$$\boxed{p(y_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(y_*|\mu_*, \sigma_*^2)}$$

Predictive mean:

$$\mu_* = \boldsymbol{\mu}_N^\top \mathbf{x}_*$$

Predictive variance:

$$\sigma_*^2 = \sigma^2 + \mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*$$

11.7 Understanding the Predictive Variance

The predictive variance has two components:

$$\sigma_*^2 = \underbrace{\sigma^2}_{\text{aleatoric}} + \underbrace{\mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*}_{\text{epistemic}}$$

1. **Aleatoric uncertainty** (σ^2): Irreducible noise in observations. Even if we knew $\mathbf{w}$ perfectly, predictions would still have this much variance. Cannot be reduced by collecting more data.
2. **Epistemic uncertainty** ($\mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*$): Uncertainty due to not knowing $\mathbf{w}$ exactly. This term:
   - Is large when $\mathbf{x}_*$ is far from training data (extrapolation is uncertain)
   - Is small when $\mathbf{x}_*$ is close to training data (interpolation is more certain)
   - Decreases as we collect more training data (epistemic uncertainty is reducible)

> **This is powerful:** The Bayesian model "knows what it does not know." Predictions far from training data come with large error bars.

11.8 Comparison of Approaches

Aspect	MLE	MAP (Ridge)	Full Bayesian
Output	Point estimate $\hat{\mathbf{w}}$	Point estimate $\hat{\mathbf{w}}$	Distribution $p(\mathbf{w}
Prediction	$\hat{y} = \hat{\mathbf{w}}^\top \mathbf{x}$	$\hat{y} = \hat{\mathbf{w}}^\top \mathbf{x}$	$p(y
Uncertainty	None	None (without extra work)	Full posterior predictive
Regularization	None (overfits)	Via prior variance	Via prior
Computation	Solve linear system	Solve linear system	Compute posterior (tractable here)
Hyperparameters	None	λ via CV	Can marginalize or use empirical Bayes

---

12. Putting It All Together

12.1 Summary of Key Formulas

MLE (Ordinary Least Squares):

$$\hat{\mathbf{w}}_{\text{MLE}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Ridge Regression (MAP with Gaussian prior):

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

Lasso (MAP with Laplace prior):

$$\hat{\mathbf{w}}_{\text{lasso}} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

Bayesian Posterior:

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\Sigma}_N = \left(\boldsymbol{\Sigma}_0^{-1} + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X} \right)^{-1}$$

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \left(\boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{y} \right)$$

Posterior Predictive:

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_N^\top \mathbf{x}_*, \sigma^2 + \mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*)$$

12.2 When to Use What

Use MLE (Least Squares) when:

- Abundant data relative to features ($N \gg D$)
- Features are well-conditioned (not highly correlated)
- Interpretability of raw coefficients is important
- Speed matters and regularization is unnecessary

Use Ridge when:

- Multicollinearity is present
- D is large or $D > N$
- All features are expected to contribute (no sparsity)
- Want closed-form solution with regularization

Use Lasso when:

- Many features are expected to be irrelevant (sparsity)
- Feature selection/interpretation is important
- Willing to accept more bias for simpler model

Use Elastic Net when:

- Features are correlated but some are irrelevant
- Want sparsity with stability

Use Full Bayesian when:

- Uncertainty quantification is critical
- Data is scarce and prior knowledge is valuable
- Sequential/online learning is needed
- Making decisions under uncertainty (active learning, optimization)

---

13. Practice Problems

Problem 1: MLE Derivation

Starting from the Gaussian likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I})$:

(a) Write out the log-likelihood and show it reduces to minimizing RSS.

(b) Derive the normal equations by setting the gradient to zero.

(c) Verify the Hessian is positive semi-definite.

Problem 2: Simple Linear Regression

Given data points: $(1, 2), (2, 4), (3, 5), (4, 4), (5, 5)$

(a) Compute sample means $\bar{x}$ and $\bar{y}$.

(b) Compute the MLE estimates $\hat{w}_0$ and $\hat{w}_1$ using the formulas for simple linear regression.

(c) Compute R^2 for your fitted model.

(d) What is the predicted value at $x = 6$?

(e) Compute the residuals and verify they sum to zero.

Problem 3: Ridge Regression

(a) Show that the ridge objective $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$ corresponds to MAP estimation with prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \tau^2\mathbf{I})$. What is the relationship between λ and τ^2?

(b) Derive the closed-form ridge solution by setting the gradient to zero.

(c) What happens to $\hat{\mathbf{w}}_{\text{ridge}}$ as $\lambda \to 0$? As $\lambda \to \infty$?

(d) Why does adding $\lambda \mathbf{I}$ fix the invertibility problem?

Problem 4: Ridge vs. Lasso Geometry

Consider minimizing RSS subject to:

- Ridge: $\|\mathbf{w}\|_2^2 \leq B$
- Lasso: $\|\mathbf{w}\|_1 \leq B$

(a) Sketch the constraint regions in 2D (w_1-w_2 plane).

(b) Sketch typical RSS contours (ellipses).

(c) Explain geometrically why lasso tends to give sparse solutions.

(d) If two features are perfectly correlated ($x_1 = x_2$), what happens with ridge vs. lasso?

Problem 5: Bayesian Linear Regression

Consider a 1D model $y = wx + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 = 1)$.

Prior: $p(w) = \mathcal{N}(0, \tau^2 = 4)$

You observe one data point: $(x = 1, y = 2)$.

(a) Write the likelihood $p(y|x, w)$.

(b) Compute the posterior $p(w|\mathcal{D})$ using the conjugate update formulas.

(c) What is the posterior mean? Compare to the MLE.

(d) What is the posterior variance? How does it compare to the prior variance?

(e) Compute the predictive distribution $p(y_*|x_* = 2, \mathcal{D})$.

(f) Decompose the predictive variance into aleatoric and epistemic components.

Problem 6: Basis Functions

(a) Write the design matrix $\mathbf{\Phi}$ for quadratic regression ($y = w_0 + w_1 x + w_2 x^2$) with data points $x \in \{-1, 0, 1, 2\}$.

(b) Compute $\mathbf{\Phi}^\top \mathbf{\Phi}$. Is it invertible?

(c) If we increased the polynomial degree to $d = N - 1 = 3$, what would happen to the training

error?

(d) What is the expected test error in this case, and why?

Lecture 2 Summary

1. **Ridge regression** = MAP with Gaussian prior = L2 regularization. Shrinks all coefficients toward zero; closed-form solution always exists.
2. **Lasso** = MAP with Laplace prior = L1 regularization. Produces sparse solutions; enables automatic feature selection.
3. **Elastic net** combines L1 + L2, getting sparsity with stability for correlated features.
4. **Cross-validation** is the standard method to select λ; the regularization path shows how coefficients evolve.
5. **Bayesian linear regression** with Gaussian prior gives a Gaussian posterior. The posterior mean equals the ridge estimate.
6. **Posterior predictive** distribution provides calibrated uncertainty: aleatoric (noise) + epistemic (parameter uncertainty).
7. **Epistemic uncertainty** is large far from training data and decreases with more data; aleatoric uncertainty is irreducible.
8. **Choose your method** based on: amount of data, expected sparsity, need for uncertainty quantification, computational constraints.

Looking Ahead

Week 4 (continued): Logistic Regression

- Classification instead of regression
- Bernoulli likelihood with sigmoid link function
- Cross-entropy loss
- No closed-form solution leads to gradient descent / Newton's method

Week 5: Variational Inference

- Approximate intractable posteriors with tractable distributions

- The Evidence Lower Bound (ELBO)

Week 6: Monte Carlo Sampling

- MCMC methods for sampling from complex posteriors
- When closed-form Bayesian inference is not possible

Week 10: Gaussian Process Regression

- Bayesian nonparametric regression
- Infinite-dimensional generalization of Bayesian linear regression

---

These notes draw from Chapter 11 of Murphy, K. P. (2022). Probabilistic Machine Learning: An Introduction. MIT Press.