Lec3: Regularized Regression Methods

Theory and Applications

Xin Xing
VT

Roughness Penalty Methods

Core Idea: Control function class by penalizing the residual sum of squares (RSS)

Penalized RSS

$$\text{PRSS}(f;\lambda) = \text{RSS}(f) + \lambda J(f)$$

- $J(f)$: User-selected roughness penalty functional
- $\lambda \geq 0$: Regularization parameter
- $J(f)$ is large for functions that vary too rapidly over small regions

Part 1: LASSO Regression

Introduction to Lasso

- **Lasso** (**Least Absolute Shrinkage and Selection Operator**) is a regularization method related to ridge regression.
- It penalizes the absolute values of coefficients, in contrast to ridge which penalizes their squares.
- It can produce *sparse* solutions, setting some coefficients to zero, effectively performing variable selection.

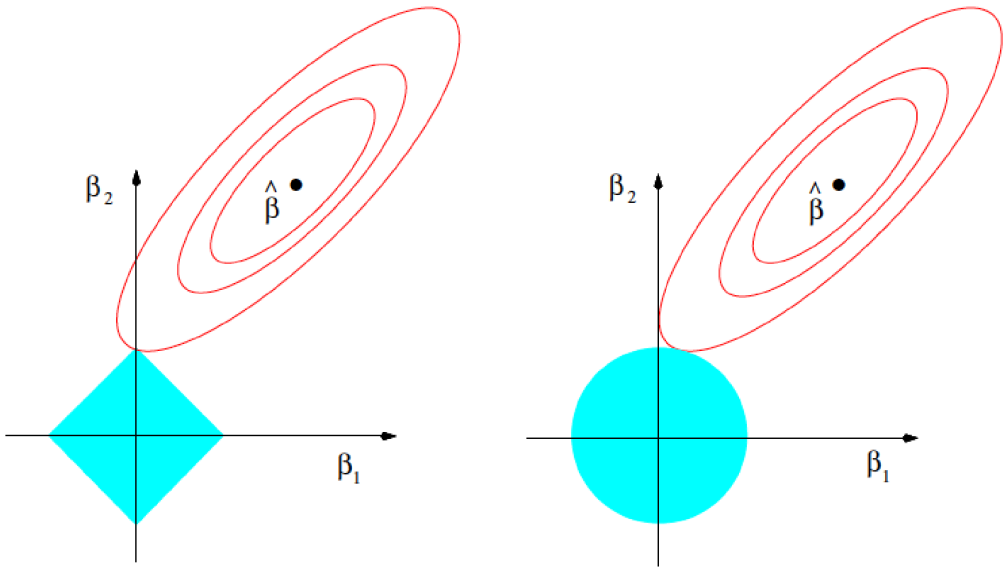
Lasso Objective

Constrained form:

$$\text{minimize} \quad \sum_{n=1}^{N} \frac{1}{2}\left(y_n - \beta x_n\right)^2 \quad \text{subject to} \quad \sum_{i=1}^{p} |\beta_i| \leq s. \tag{21}$$

- Compared to ridge ($\sum \beta_i^2 \leq t^2$), here the constraint is $\sum |\beta_i| \leq s$.
- The ℓ_1 norm constraint has *corners* (in 2D, a diamond shape).
- If the solution hits a corner, some β_j will be zero.

Geometric Interpretation



From ESL

- **Left**: Lasso constraint region $\sum |\beta_i| \leq t$ (diamond in 2D).
- **Right**: Ridge constraint region $\sum \beta_i^2 \leq t^2$ (disk in 2D).
- **Red contours**: Least squares error.
- **Question:** What happens as s (or t) increases?
- **Question:** Where does the solution lie for a fixed s?

LASSO: Active Set

Active Set Definition

The active set $\mathcal{A}$ is the set of indices where $\hat{\beta}_j \neq 0$:

$$\mathcal{A} = \{j : \hat{\beta}_j \neq 0\}$$

Properties

- $X_{\mathcal{A}}$: Matrix with columns corresponding to active variables
- $\beta_{\mathcal{A}}$: Vector of non-zero coefficients
- Active set size: $|\mathcal{A}| \leq \min(n, p)$

Subgradient Definition

Definition

A subgradient of a function f at point x_0 is any vector g that satisfies:

$$f(x) \geq f(x_0) + g(x - x_0) \quad \text{for all } x$$

Key Properties:

- For differentiable functions, the only subgradient is the gradient itself
- A function can have multiple subgradients at non-differentiable points
- The set of all subgradients at a point is called the subdifferential

Example: Absolute Value Function

Consider $f(x) = |x|$:

At $x \neq 0$:

- If $x > 0$: unique subgradient is $+1$
- If $x < 0$: unique subgradient is -1

At $x = 0$:

- Any value in $[-1, 1]$ is a subgradient

This example shows how a non-differentiable function can have multiple subgradients at its non-differentiable point ($x = 0$)

LASSO: Subgradient Conditions

KKT Conditions with Subgradient

For any j:

$$-x_j^T(y - X\hat{\beta}) + \lambda \partial \|\hat{\beta}\|_1 / \partial \hat{\beta}_j = 0$$

$$\text{where } \partial \|\hat{\beta}\|_1 / \partial \hat{\beta}_j = \begin{cases} 1 & \text{if } \hat{\beta}_j > 0 \\ -1 & \text{if } \hat{\beta}_j < 0 \\ [-1, 1] & \text{if } \hat{\beta}_j = 0 \end{cases}$$

Implications

- When $\hat{\beta}_j \neq 0$: Subgradient is uniquely determined
- When $\hat{\beta}_j = 0$: Subgradient is set-valued
- Captures non-differentiability at zero

LASSO: Active Set Analysis

Conditions by Set

For active set ($\hat{\beta}_j \neq 0$):

$$-x_j^T(y - X\hat{\beta}) + \lambda \text{ sign}(\hat{\beta}_j) = 0$$

For inactive set ($\hat{\beta}_j = 0$):

$$|x_j^T(y - X\hat{\beta})| \leq \lambda$$

LASSO Bias: Active Set Focus

Expected Value

$$\begin{aligned} E[\hat{\beta}_{\mathcal{A}}|X] &= (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1}(X_{\mathcal{A}}^T E[y] - \lambda s_{\mathcal{A}}) \\ &= (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1}(X_{\mathcal{A}}^T X_{\mathcal{A}}\beta_{\mathcal{A}} - \lambda s_{\mathcal{A}}) \\ &= \beta_{\mathcal{A}} - \lambda(X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1} s_{\mathcal{A}} \end{aligned}$$

Key Insights

- Bias only depends on active variables due to sparsity
- Inactive variables ($\hat{\beta}_j = 0$) have exact zero coefficients
- Bias magnitude proportional to λ and inverse correlation of active predictors

LASSO Variance Calculation

Variance Formula

For the active set $\mathcal{A}$, assuming $\text{Var}(y) = \sigma^2 I$:

$$\begin{aligned}\text{Var}(\hat{\beta}_{\mathcal{A}}|X) &= \sigma^2 (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1} \\ &= \sigma^2 (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1} X_{\mathcal{A}}^T X_{\mathcal{A}} (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1}\end{aligned}$$

Key Properties

- Variance depends only on active predictors ($X_{\mathcal{A}}$)
- Sparsity reduces variance through dimension reduction
- Trade-off: Large λ increases bias but decreases variance
- For inactive variables ($j \notin \mathcal{A}$): $\text{Var}(\hat{\beta}_j|X) = 0$

LASSO: Bias-Variance Tradeoff

Mean Squared Error Decomposition

For the active set $\mathcal{A}$:

$$\begin{aligned} \text{MSE}(\hat{\beta}) &= \mathbb{E}[(\hat{\beta} - \beta)^T(\hat{\beta} - \beta)] \\ &= \text{Bias}(\hat{\beta})^T\text{Bias}(\hat{\beta}) + \text{tr}(\text{Var}(\hat{\beta})) \\ &= \lambda^2 s_{\mathcal{A}}^T (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-2} s_{\mathcal{A}} + \sigma^2 \text{tr}((X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1}) \end{aligned}$$

Effect of Regularization Parameter λ

- **As** $\lambda \uparrow$:
  - Bias increases ($\propto \lambda^2$)
  - Variance decreases (smaller active set)
  - More coefficients shrink to exactly zero
- **As** $\lambda \downarrow$:
  - Approaches OLS solution
  - Lower bias but higher variance
  - More variables enter the model

Simulation Study Setup

Data Generation

- Sample size: $n = 100$
- Number of predictors: $p = 20$
- True coefficients: $\beta = (3, 1.5, 0, 0, 2, 0, \ldots, 0)$
- Noise level: $\sigma = 1$
- Number of simulation runs: 500
- λ sequence: 20 values from 0.01 to 10 (log scale)

Simulation Process

- For each λ value:
  - Generate data: $X_{n \times p}$ from $\mathcal{N}(0, 1)$
  - Generate response: $y = X\beta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$
  - Fit LASSO and store coefficients
- Calculate empirical bias and variance

Key Metrics

For Each λ

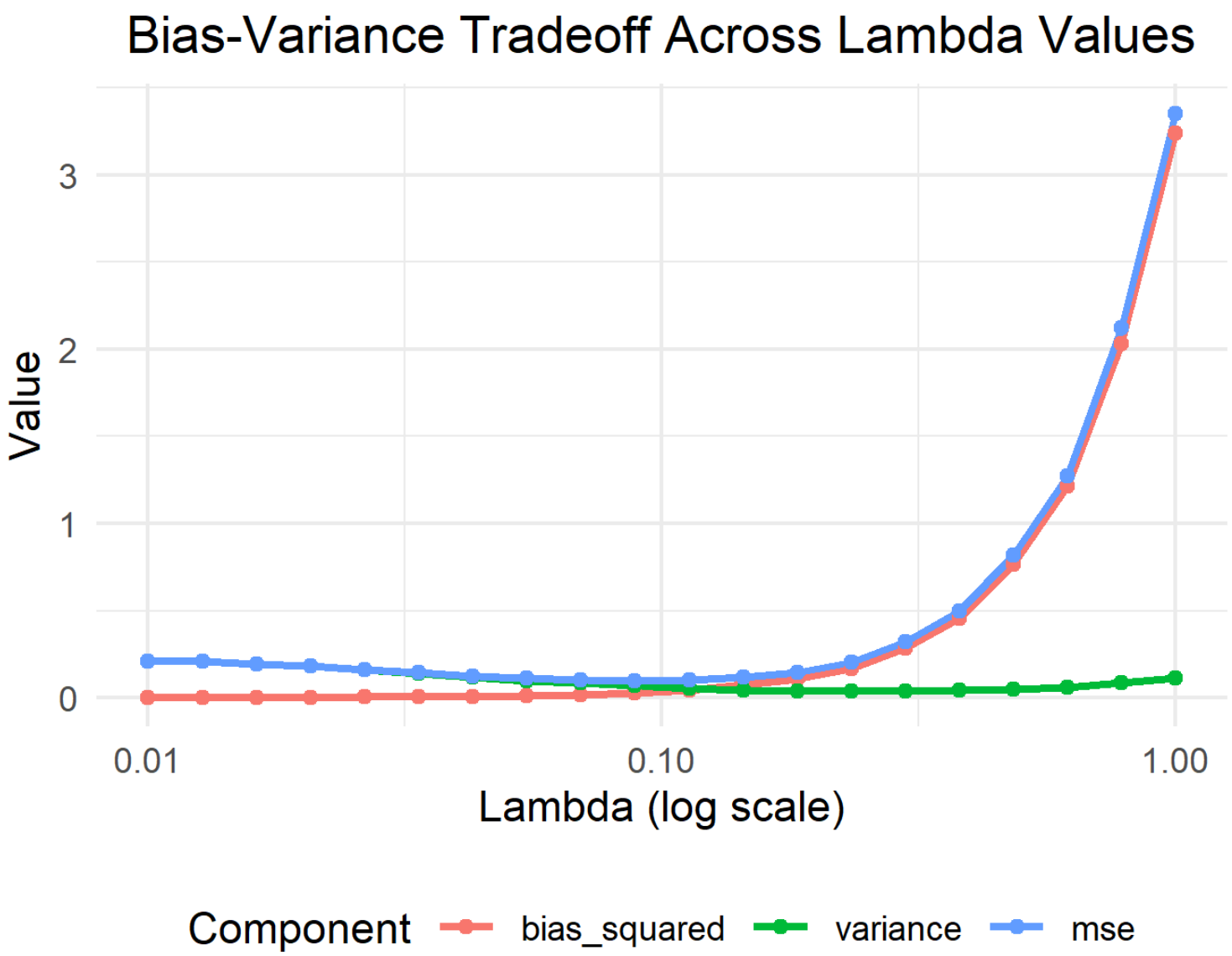
Empirical calculations:

$$\text{Bias}^2 = \|\mathbb{E}[\hat{\beta}] - \beta\|_2^2$$
$$\text{Variance} = \mathbb{E}[\|\hat{\beta} - \mathbb{E}[\hat{\beta}]\|_2^2]$$
$$\text{MSE} = \text{Bias}^2 + \text{Variance}$$

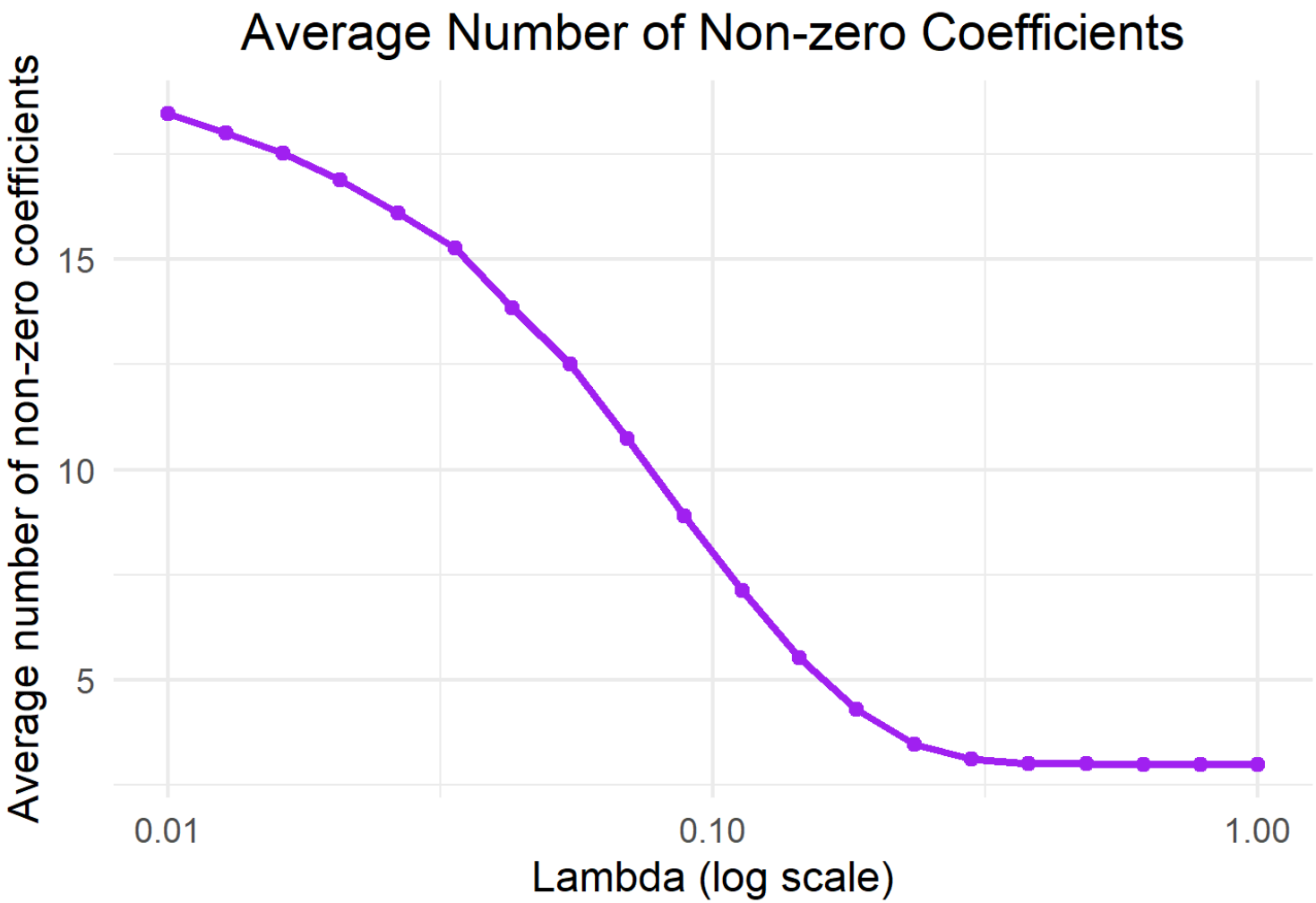
Additional Analysis

- Model sparsity: Average number of non-zero coefficients
- Optimal λ selection based on minimum MSE
- Trade-off between model complexity and accuracy

Bias-Variance Tradeoff Results



Model Sparsity Analysis



Key Findings

Observations

- As λ increases:
  - Bias increases due to stronger shrinkage
  - Variance decreases as the model becomes sparser
  - Number of non-zero coefficients decreases
- Optimal λ balances:
  - Prediction accuracy (MSE)
  - Model sparsity
  - Bias-variance tradeoff

Practical Implications

- Choose λ based on specific goals:
  - Minimum MSE for the best prediction
  - Sparsity for interpretability
  - Cross-validation in practice

Part 2: Ridge Regression

Introduction to Ridge Regression

- **Ridge regression** is a regularization technique that addresses multicollinearity and overfitting
- Also known as **Tikhonov regularization** or ℓ_2 regularization
- Shrinks coefficients toward zero but does not set them exactly to zero
- Particularly useful when predictors are highly correlated

Key Difference from OLS

Ridge regression trades a small increase in bias for a large decrease in variance

Ridge Regression Objective

Constrained form:

$$\text{minimize} \quad \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq t$$

Lagrangian (penalized) form:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- $\lambda \geq 0$: Regularization parameter (controls amount of shrinkage)
- ℓ_2 penalty: $\sum_{j=1}^{p} \beta_j^2$

Matrix Formulation

Using centered variables: Assume y and X are centered (mean 0)

Ridge Objective in Matrix Form

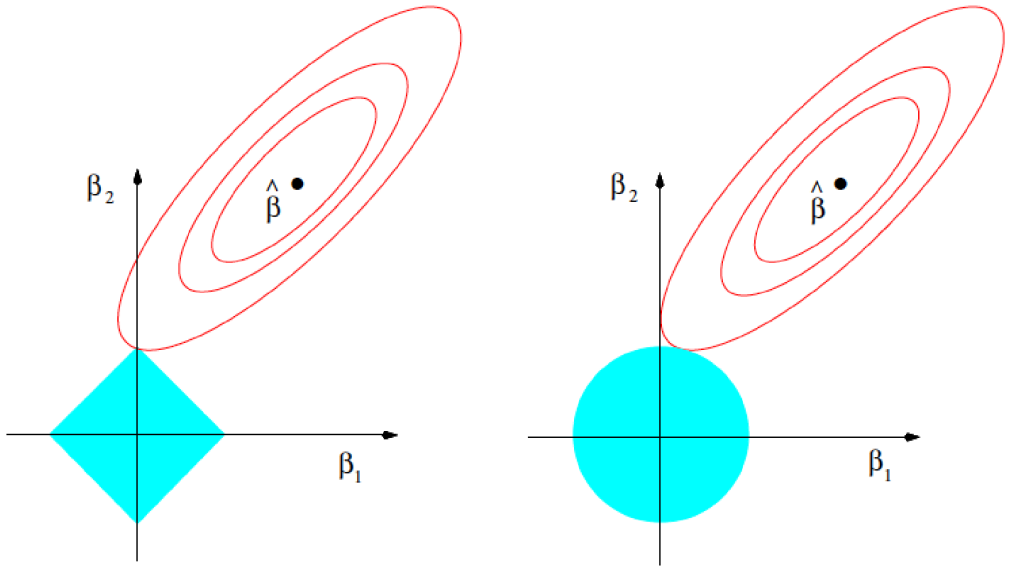
$$\text{RSS}(\beta) + \lambda\|\beta\|_2^2 = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$$

Closed-form solution:

$$\hat{\beta}^{\text{ridge}} = (X^TX + \lambda I)^{-1}X^Ty$$

- Adding λI to X^TX ensures invertibility even when $p > n$
- Solution exists and is unique for any $\lambda > 0$

Geometric Interpretation



From ESL

- **Right panel**: Ridge constraint region $\sum \beta_i^2 \leq t^2$ (circular in 2D)
- **Red contours**: OLS error ellipses
- Solution occurs where contour touches the constraint region
- Smooth boundary $\Rightarrow$ coefficients rarely exactly zero

Comparison: Ridge vs. LASSO Constraints

Ridge (ℓ_2):

- Circular constraint
- Smooth boundary
- No exact zeros
- All variables retained

LASSO (ℓ_1):

- Diamond constraint
- Corners at axes
- Exact zeros common
- Variable selection

Key Insight

The shape of the constraint region determines whether we get sparse solutions (LASSO) or dense solutions (Ridge)

Ridge Solution via SVD

Using Singular Value Decomposition: $X = UDV^T$

Ridge Estimator

$$\hat{\beta}^{\text{ridge}} = V(D^2 + \lambda I)^{-1} D U^T y = \sum_{j=1}^{p} \frac{d_j}{d_j^2 + \lambda} u_j^T y \cdot v_j$$

where:

- d_j: Singular values of X
- u_j, v_j: Left and right singular vectors

Shrinkage factor: $\frac{d_j^2}{d_j^2+\lambda}$

- Small d_j (weak signal) $\Rightarrow$ more shrinkage
- Large d_j (strong signal) $\Rightarrow$ less shrinkage

Bias of Ridge Regression

Expected Value

$$\begin{aligned} E[\hat{\beta}^{\text{ridge}}|X] &= E[(X^T X + \lambda I)^{-1} X^T y|X] \\ &= (X^T X + \lambda I)^{-1} X^T X \beta \\ &= (X^T X + \lambda I)^{-1} (X^T X + \lambda I - \lambda I)\beta \\ &= \beta - \lambda (X^T X + \lambda I)^{-1} \beta \end{aligned}$$

Bias

$$\text{Bias}(\hat{\beta}^{\text{ridge}}) = -\lambda (X^T X + \lambda I)^{-1} \beta$$

- Ridge estimator is **biased** (unlike OLS)
- Bias increases with λ
- Trade-off: Accept bias to reduce variance

Variance of Ridge Regression

Variance Formula

Assuming $\text{Var}(y) = \sigma^2 I$:

$$\begin{aligned}\text{Var}(\hat{\beta}^{\text{ridge}}|X) &= \text{Var}((X^T X + \lambda I)^{-1} X^T y|X) \\ &= (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} \sigma^2\end{aligned}$$

Comparison with OLS variance:

$$\text{Var}(\hat{\beta}^{\text{OLS}}|X) = (X^T X)^{-1} \sigma^2$$

- Ridge variance $\leq$ OLS variance (element-wise)
- Variance decreases as λ increases
- Greatest reduction when $X^T X$ has small eigenvalues (multicollinearity)

Mean Squared Error

MSE Decomposition

$$\begin{aligned} \text{MSE}(\hat{\beta}^{\text{ridge}}) &= E[(\hat{\beta}^{\text{ridge}} - \beta)^T(\hat{\beta}^{\text{ridge}} - \beta)] \\ &= \text{Bias}(\hat{\beta}^{\text{ridge}})^T\text{Bias}(\hat{\beta}^{\text{ridge}}) + \text{tr}(\text{Var}(\hat{\beta}^{\text{ridge}})) \end{aligned}$$

Effect of λ:

- $\lambda = 0$: Ridge = OLS (unbiased, high variance)
- $\lambda \to \infty$: $\hat{\beta}^{\text{ridge}} \to 0$ (high bias, zero variance)
- Optimal λ: Minimizes MSE by balancing bias and variance

Effective Degrees of Freedom

Ridge DF

$$\text{df}(\lambda) = \text{tr}[X(X^T X + \lambda I)^{-1} X^T] = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

- Measures model complexity
- $\lambda = 0$: $\text{df}(\lambda) = p$ (OLS)
- $\lambda \to \infty$: $\text{df}(\lambda) \to 0$
- Decreases monotonically with λ
- Used in model selection criteria (AIC, BIC)

Interpretation

Larger $\lambda \Rightarrow$ more shrinkage $\Rightarrow$ lower effective degrees of freedom

Choosing the Regularization Parameter

Methods for selecting λ:

1. **Cross-Validation**
   - K-fold CV: Minimize prediction error on held-out data
   - Most common: 5-fold or 10-fold CV
2. **Generalized Cross-Validation (GCV)**

$$\text{GCV}(\lambda) = \frac{n \cdot \text{RSS}(\lambda)}{[n - \text{df}(\lambda)]^2}$$

3. **Information Criteria**
   - AIC, BIC based on effective degrees of freedom
4. **Ridge Trace**
   - Plot $\hat{\beta}_j(\lambda)$ vs. λ
   - Choose λ where coefficients stabilize

Standardization

Important Preprocessing

Always standardize predictors before applying ridge regression!

Standardization:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\text{sd}(x_j)}$$

Why?

- Ridge penalty is not scale-invariant
- Variables on different scales would be penalized differently
- Standardization ensures equal penalization
- Intercept β_0 is typically not penalized (fit after centering)

Ridge Regression Properties

Advantages

- Handles multicollinearity effectively
- Always has a unique solution (even when $p > n$)
- Computationally efficient (closed-form solution)
- Reduces prediction error through bias-variance tradeoff
- Stable estimates when predictors are correlated

Disadvantages

- Does not perform variable selection (all p predictors retained)
- Biased estimates (not suitable when unbiased estimates needed)
- Interpretation can be difficult with many predictors
- Choice of λ requires tuning

Bayesian Interpretation of Ridge

Ridge regression has a Bayesian interpretation:

Prior Distribution

Assume independent Gaussian priors on coefficients:

$$\beta_j \sim \mathcal{N}(0, \tau^2), \quad j = 1, \ldots, p$$

Posterior Mode

With likelihood $y|X, \beta \sim \mathcal{N}(X\beta, \sigma^2 I)$, the posterior mode is:

$$\hat{\beta}^{\text{ridge}} = \arg\max_{\beta} \{\log p(y|X, \beta) + \log p(\beta)\}$$

which corresponds to ridge with $\lambda = \sigma^2/\tau^2$

Ridge = Maximum a posteriori (MAP) estimate under Gaussian prior

When to Use Ridge Regression

Ridge is preferred when:

- Predictors are highly correlated (multicollinearity)
- All predictors are believed to be relevant
- Goal is prediction rather than interpretation
- p is close to or exceeds n
- Stable coefficient estimates are needed

LASSO is preferred when:

- Variable selection is desired
- Sparse solutions are believed to exist
- Interpretability is important
- Many predictors are irrelevant

Elastic Net

Combines both: $(1-\alpha)\|\beta\|_2^2/2 + \alpha\|\beta\|_1$ for $\alpha \in [0,1]$

Part 3: Kernel Ridge Regression

Motivation for Kernel Methods

Limitation of Linear Ridge Regression:

- Linear models can only capture linear relationships
- Real-world data often has nonlinear patterns

Solution Approaches:

1. **Feature engineering:** Manually create nonlinear features $\phi(x)$
   - Example: $x \mapsto (x, x^2, x^3, \ldots)$
2. **Kernel trick:** Implicitly work in high-dimensional feature space
   - No explicit feature construction needed
   - Computationally efficient for certain kernels

Feature Space Mapping

Explicit feature mapping: $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$

Ridge Regression in Feature Space

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta^T \phi(x_i))^2 + \lambda \|\beta\|^2 \right\}$$

Problems with explicit mapping:

- Feature dimension D can be very large (even infinite!)
- Computing and storing $\phi(x)$ explicitly is expensive
- Solution: Use the **kernel trick**

The Kernel Trick

Key Insight: We don't need $\phi(x)$ explicitly, only inner products!

Kernel Function

$$K(x, x') = \langle \phi(x), \phi(x') \rangle = \phi(x)^T \phi(x')$$

Advantages:

- Compute $K(x, x')$ directly without computing $\phi(x)$
- Work in very high (even infinite) dimensional spaces
- Efficient computation for many kernels

Example

Polynomial kernel: $K(x, x') = (1 + x^T x')^d$
Equivalent to $O(d^p)$ dimensional feature space, computed in $O(d)$ time!

Representer Theorem

Fundamental result for kernel ridge regression:

Theorem (Representer Theorem)

The solution to kernel ridge regression can be written as:

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

for some coefficients $\alpha_1, \ldots, \alpha_n$

Implication:

- Solution lies in the span of kernel evaluations at training points
- Don't need to work with infinite-dimensional β
- Only need to find n coefficients α_i

Dual Formulation of Kernel Ridge Regression

Primal form (in feature space):

$$\min_{\beta} \|y - \Phi\beta\|^2 + \lambda\|\beta\|^2$$

where $\Phi = [\phi(x_1), \ldots, \phi(x_n)]^T$ is the $n \times D$ feature matrix

Dual form (using representer theorem):

Kernel Ridge Regression Dual

$$\hat{\alpha} = (K + \lambda I)^{-1} y$$

where K is the $n \times n$ kernel matrix: $K_{ij} = K(x_i, x_j)$

Prediction for new point x:

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, x_i) = k(x)^T \hat{\alpha}$$

where $k(x) = [K(x, x_1), \ldots, K(x, x_n)]^T$

Comparing Primal and Dual

Primal (Linear Ridge):

- Solve: $(X^T X + \lambda I)^{-1} X^T y$
- Complexity: $O(d^3)$
- Prediction: $O(d)$
- Best when $d < n$

Dual (Kernel Ridge):

- Solve: $(K + \lambda I)^{-1} y$
- Complexity: $O(n^3)$
- Prediction: $O(n)$
- Best when $n < d$
- Works for $d = \infty$!

Key Trade-off

Kernel methods trade feature dimension complexity for sample size complexity

Common Kernel Functions

1. Linear Kernel

$$K(x, x') = x^T x'$$

Equivalent to standard ridge regression (no feature transformation)

2. Polynomial Kernel

$$K(x, x') = (x^T x' + c)^d$$

Parameters: degree d, constant c (often $c = 1$)

3. Gaussian (RBF) Kernel

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

Parameter: bandwidth σ (controls smoothness)

More Kernel Functions

4. Laplacian Kernel

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{\sigma}\right)$$

Similar to RBF but uses ℓ_1 distance

5. Sigmoid Kernel

$$K(x, x') = \tanh(\kappa x^T x' + c)$$

Inspired by neural networks

6. String Kernels

For discrete/structured data (text, sequences, graphs)

Note: Any valid kernel must satisfy Mercer's condition (positive semi-definite)

Gaussian (RBF) Kernel - Deep Dive

Most popular kernel in practice:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

- **Infinite dimensional feature space**
- **Universal approximator:** Can approximate any continuous function
- **Locality:** $K(x, x') \approx 0$ when $\|x - x'\|$ is large
- **Smoothness:** Infinitely differentiable

Effect of bandwidth σ:

- Small σ: More local, flexible, risk of overfitting
- Large σ: More global, smooth, risk of underfitting

Kernel Matrix Properties

The kernel (Gram) matrix: $K \in \mathbb{R}^{n \times n}$

$$K_{ij} = K(x_i, x_j)$$

Properties

- **Symmetric:** $K = K^T$ (since $K(x, x') = K(x', x)$)
- **Positive semi-definite:** All eigenvalues ≥ 0
- **Full rank for most kernels:** Ensures $(K + \lambda I)$ is invertible

Computational considerations:

- Need to compute and store $n \times n$ matrix
- Memory: $O(n^2)$
- Matrix inversion: $O(n^3)$
- Becomes expensive for large n (say $n > 10{,}000$)

Kernel Ridge Regression Algorithm

Training (given λ, kernel K):

1. Compute kernel matrix: $K_{ij} = K(x_i, x_j)$ for all i, j
2. Solve linear system: $\hat{\alpha} = (K + \lambda I)^{-1} y$
3. Store $\hat{\alpha}$ and training data $\{x_i\}_{i=1}^n$

Prediction (for new point x):

1. Compute kernel vector: $k(x) = [K(x, x_1), \ldots, K(x, x_n)]^T$
2. Predict: $\hat{f}(x) = k(x)^T \hat{\alpha} = \sum_{i=1}^n \hat{\alpha}_i K(x, x_i)$

Note

Must store all training points for prediction (non-parametric method)

Hyperparameter Selection

Two key hyperparameters:

1. **Regularization parameter λ**
   - Controls bias-variance tradeoff
   - Larger $\lambda \Rightarrow$ smoother solution
2. **Kernel parameters** (e.g., σ for RBF)
   - Control complexity of feature space
   - Affect model flexibility

Selection methods:

- Cross-validation (most common)
- Grid search over parameter ranges
- Bayesian optimization for efficiency
- Information criteria (AIC, BIC)

Comparison: Linear vs. Kernel Ridge

	Linear Ridge	**Kernel Ridge**
Feature space	$\mathbb{R}^d$	$\mathcal{H}$ (can be ∞-dim)
Training complexity	$O(d^3)$	$O(n^3)$
Prediction complexity	$O(d)$	$O(n)$
Memory	$O(d)$	$O(n^2)$
Linearity	Linear only	Nonlinear
Interpretability	High	Low
Flexibility	Low	High

When to use kernel ridge:

- Nonlinear relationships in data
- Small to moderate n (computationally feasible)
- Standard features don't work well

Kernel Ridge Regression Properties

Advantages

- Handles nonlinear relationships without explicit feature engineering
- Works in infinite-dimensional spaces
- Closed-form solution (no iterative optimization)
- Inherits ridge's stability and bias-variance properties
- Flexible through kernel choice

Disadvantages

- Computational cost: $O(n^3)$ training, $O(n)$ prediction
- Memory intensive: must store $n \times n$ kernel matrix
- Less interpretable than linear models
- Hyperparameter tuning can be challenging
- Doesn't perform variable selection

Connection to Gaussian Processes

Bayesian perspective:

Kernel ridge regression is equivalent to:

- Gaussian Process regression with prior covariance K
- Maximum a posteriori (MAP) estimation

Gaussian Process Prior

$$f \sim \mathcal{GP}(0, K)$$

where $K(x, x')$ is the kernel function

Prediction distribution:

$$f(x) \mid \{x_i, y_i\} \sim \mathcal{N}(\hat{f}(x), \sigma^2(x))$$

where $\hat{f}(x)$ is the kernel ridge prediction and $\sigma^2(x)$ is the predictive variance

Practical Considerations

1. **Computational Tricks:**
   - Cholesky decomposition: $(K + \lambda I) = LL^T$
   - Solve $LL^T\alpha = y$ via back-substitution
   - Reuse decomposition for different λ
2. **Scalability:**
   - Approximate methods for large n: Nyström, random features
   - Subset of regressors (inducing points)
   - Sparse kernel approximations
3. **Preprocessing:**
   - Standardize features before computing kernels
   - Center outputs: $\tilde{y} = y - \bar{y}$

Extensions of Penalty Methods

Penalty functionals can be constructed for:

- Functions in any dimension
- Special structures (additive models, projection pursuit)

Additive Models

Additive penalties: $J(f) = \sum_{j=1}^{p} J(f_j)$

For additive functions: $f(X) = \sum_{j=1}^{p} f_j(X_j)$

Projection Pursuit Regression

$$f(X) = \sum_{m=1}^{M} g_m(\alpha_m^T X)$$

Each g_m can have an associated roughness penalty

Bayesian Interpretation

Penalty methods have a Bayesian framework:

- Penalty J corresponds to a **log-prior**
- $\text{PRSS}(f;\lambda)$ is the **log-posterior distribution**
- Minimizing $\text{PRSS}(f;\lambda)$ = finding the **posterior mode**

Interpretation

Regularization methods express our *prior belief* that the functions we seek exhibit a certain type of smooth behavior.

Basis Functions and Dictionary Methods

Core Idea: Linear expansion of basis functions

General Form

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

where:

- h_m: Basis functions of input x
- θ_m: Linear parameters
- "Linear" refers to the action of parameters θ

Wide variety: Covers many different methods

Radial Basis Functions

Symmetric p-dimensional kernels:

RBF Form

$$f_\theta(x) = \sum_{m=1}^{M} K_{\lambda_m}(\mu_m, x)\theta_m$$

Popular choice: Gaussian kernel

$$K_\lambda(\mu, x) = e^{-\|x-\mu\|^2/2\lambda}$$

Parameters to determine:

- Centroids μ_m
- Scales λ_m

Neural Networks as Basis Functions

Single-layer feed-forward network with linear output weights:

Neural Network Model

$$f_\theta(x) = \sum_{m=1}^{M} \beta_m \sigma(\alpha_m^T x + b_m)$$

where:

- $\sigma(x) = \frac{1}{1+e^{-x}}$: Activation function (sigmoid)
- α_m: Weight directions (to be estimated)
- b_m: Bias terms (to be estimated)
- β_m: Output weights

Estimation: Determining α_m and b_m is the computational challenge

Dictionary Methods

General framework:

- Available: Possibly infinite set or *dictionary* $\mathcal{D}$ of candidate basis functions
- Goal: Choose appropriate basis functions from $\mathcal{D}$
- Build models adaptively

Challenge

Including basis function parameters (knots, centroids, directions) changes the problem from:

- Straightforward linear problem $\Rightarrow$
- Combinatorially hard nonlinear problem

Practical solutions:

- Greedy algorithms
- Two-stage processes

Summary: Regularization Approaches

1. **LASSO Regression (ℓ_1 penalty)**
   - Penalty: $\lambda \sum |\beta_j|$
   - Produces sparse solutions (variable selection)
2. **Ridge Regression (ℓ_2 penalty)**
   - Penalty: $\lambda \sum \beta_j^2$
   - Shrinks coefficients, handles multicollinearity
3. **Kernel Ridge Regression**
   - Nonlinear extension via kernel trick
   - Works in infinite-dimensional feature spaces
4. **General Framework**
   - Roughness penalties, basis functions, dictionary methods
   - All balance bias and variance

Common theme: Regularization controls model complexity to improve generalization