

Methods and Algorithms – MSc Data Science

MSc Data Science

- 1 Mathematical Foundations
- 2 Gradient Descent
- 3 Model Evaluation
- 4 Regularization
- 5 Bias-Variance Tradeoff
- 6 Decision Framework
- 7 Summary

The Model in Matrix Form

$$y = X\beta + \epsilon \quad (1)$$

- $y \in \mathbb{R}^n$: Response vector
- $X \in \mathbb{R}^{n \times (p+1)}$: Design matrix (with intercept column)
- $\beta \in \mathbb{R}^{p+1}$: Coefficient vector
- $\epsilon \in \mathbb{R}^n$: Error vector

Matrix notation enables elegant derivations and efficient computation

The Design Matrix X

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \quad (2)$$

- First column of 1s for intercept β_0
- Each row is one observation
- Each column (after first) is one feature

n observations, p features, p + 1 parameters

Classical Assumptions for Valid Inference

- ① **Linearity:** $E[y|X] = X\beta$ (correct functional form)
- ② **Exogeneity:** $E[\varepsilon|X] = 0$ (no omitted variables)
- ③ **Homoscedasticity:** $\text{Var}(\varepsilon|X) = \sigma^2 I$ (constant variance)
- ④ **No multicollinearity:** $\text{rank}(X) = p + 1$ (full rank)
- ⑤ **Normality** (for inference): $\varepsilon \sim N(0, \sigma^2 I)$

Violations? Robust standard errors, transformations, regularization

Assumptions 1-4 needed for unbiased estimates; 5 for t-tests and CIs

Sum of Squared Residuals (SSR)

$$L(\beta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = (y - X\beta)^\top (y - X\beta) \quad (3)$$

Expanding:

$$L(\beta) = y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X\beta \quad (4)$$

Quadratic function in β – has unique minimum (if X full rank)

Taking the Derivative

$$\frac{\partial L}{\partial \beta} = -2X^T y + 2X^T X\beta \quad (5)$$

Setting to Zero:

$$X^T X \hat{\beta} = X^T y \quad (6)$$

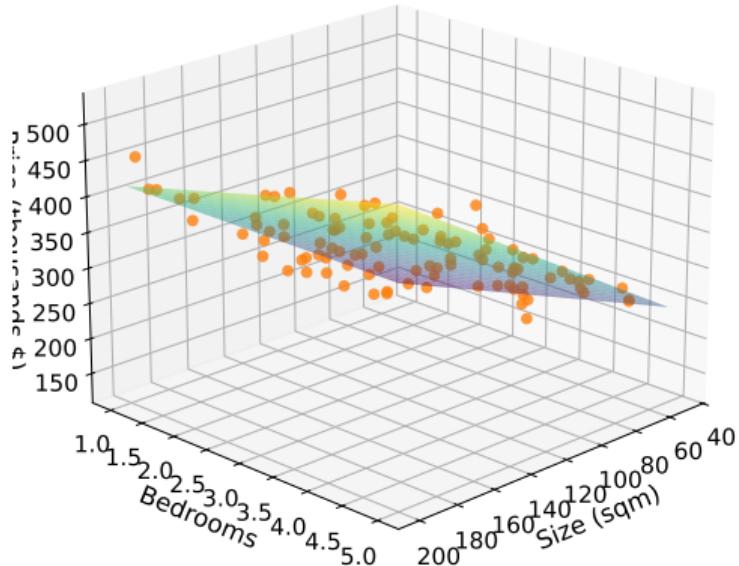
$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (7)$$

This is the closed-form OLS solution – the “normal equation”



The fitted line minimizes vertical distances squared

Multiple Regression: Price = f(Size, Bedrooms)



With 2 features, we fit a plane; with p features, a hyperplane

Normal Equation Limitations

- Computing $(X^\top X)^{-1}$ is $O(p^3)$
- Memory: Need to store $p \times p$ matrix
- For large p (millions of features): infeasible

Gradient Descent Advantages

- Memory efficient: process one sample at a time
- Scales to big data (SGD)
- Generalizes to non-linear models

For $p > 10,000$, gradient descent usually faster

Gradient of the Loss Function

$$\nabla L(\beta) = -2X^\top(y - X\beta) = -2X^\top r \quad (8)$$

where $r = y - X\beta$ is the residual vector.

Intuition:

- Gradient points in direction of steepest ascent
- We move opposite to gradient (steepest descent)
- Scale by learning rate α

Gradient is a $p + 1$ dimensional vector

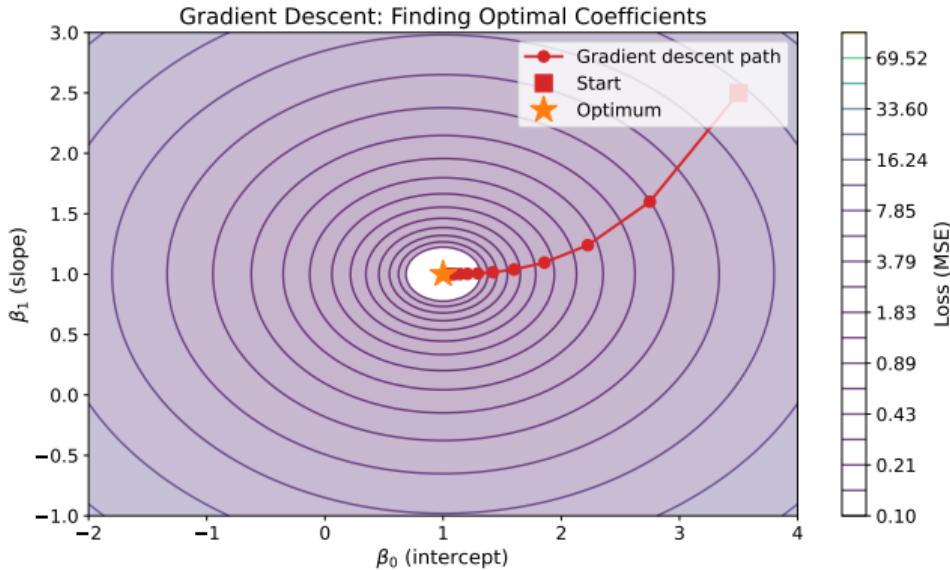
Update Rule

$$\beta^{(t+1)} = \beta^{(t)} - \alpha \nabla L(\beta^{(t)}) \quad (9)$$

Algorithm:

- ① Initialize $\beta^{(0)}$ (often zeros or random)
- ② Compute gradient $\nabla L(\beta^{(t)})$
- ③ Update: $\beta^{(t+1)} = \beta^{(t)} - \alpha \nabla L$
- ④ Repeat until convergence

Convergence: $\|\beta^{(t+1)} - \beta^{(t)}\| < \epsilon$ or max iterations



Contours show loss surface; path shows optimization trajectory

The Critical Hyperparameter

- **Too small:** Slow convergence, many iterations
- **Too large:** Divergence, oscillation
- **Just right:** Fast, stable convergence

Practical Approaches:

- Start with $\alpha = 0.01$ or 0.001
- Learning rate schedules (decay over time)
- Adaptive methods: Adam, AdaGrad, RMSprop

For OLS, optimal $\alpha = 1/\lambda_{\max}(X^\top X)$

Mini-Batch Gradient Descent

Instead of full gradient:

$$\nabla L(\beta) = -\frac{2}{n} \mathbf{X}^\top \mathbf{r} \quad (10)$$

Use mini-batch of size m :

$$\nabla L_B(\beta) = -\frac{2}{m} \mathbf{X}_B^\top \mathbf{r}_B \quad (11)$$

- $m = 1$: Stochastic GD (noisy but fast)
- $m = n$: Batch GD (stable but slow)
- $m \in [32, 256]$: Mini-batch (good tradeoff)

SGD: Process data once per epoch, update many times

Coefficient of Determination

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad (12)$$

Interpretation:

- Proportion of variance explained by model
- $R^2 = 0$: Model no better than mean
- $R^2 = 1$: Perfect fit
- $R^2 = 0.7$: 70% of variance explained

R^2 always increases with more features – use Adjusted R^2

Penalizing Model Complexity

$$R_{\text{adj}}^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \quad (13)$$

Properties:

- Adjusts for number of predictors p
- Can decrease when adding irrelevant features
- Better for model comparison

Use R_{adj}^2 when comparing models with different p

Error Metrics in Original Units

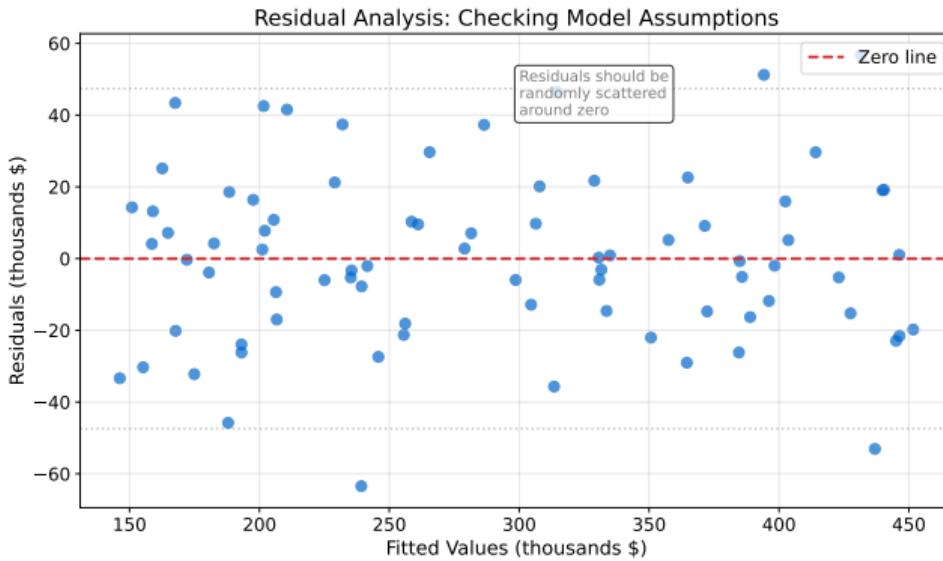
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2} \quad (14)$$

$$\text{MAE} = \frac{1}{n} \sum |y_i - \hat{y}_i| \quad (15)$$

Comparison:

- RMSE: Penalizes large errors more (sensitive to outliers)
- MAE: More robust, easier to interpret
- Units: Same as target variable (e.g., dollars)

Report both for comprehensive evaluation



Good: random scatter. Bad: patterns indicate model misspecification

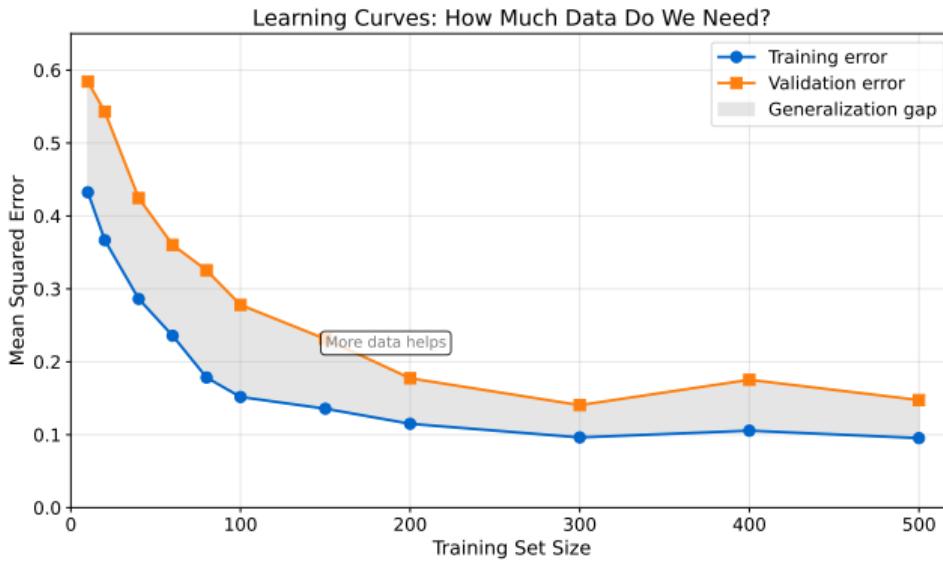
Evaluating Generalization

- Never evaluate on training data alone
- Split: 70-80% train, 20-30% test
- Report test set metrics

Cross-Validation (K-Fold):

- Split into K folds (typically $K = 5$ or 10)
- Train on $K - 1$ folds, validate on 1
- Repeat K times, average results

CV gives more reliable estimate with limited data



Gap between curves indicates overfitting; convergence shows saturation

When Models Memorize Instead of Learn

- High-dimensional data ($p \approx n$ or $p > n$)
- Coefficients become very large
- Perfect fit on training data, poor generalization

Solution: Add Penalty to Loss Function

$$L_{\text{reg}}(\beta) = \|y - X\beta\|^2 + \lambda \cdot \text{penalty}(\beta) \quad (16)$$

λ controls strength of regularization

L2 Penalty: Sum of Squared Coefficients

$$L_{\text{ridge}}(\beta) = \|y - X\beta\|^2 + \lambda\|\beta\|_2^2 \quad (17)$$

Closed-Form Solution:

$$\hat{\beta}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y \quad (18)$$

- Shrinks all coefficients toward zero
- Never sets coefficients exactly to zero
- Always invertible (even when $p > n$)

Ridge adds λ to diagonal – stabilizes inversion

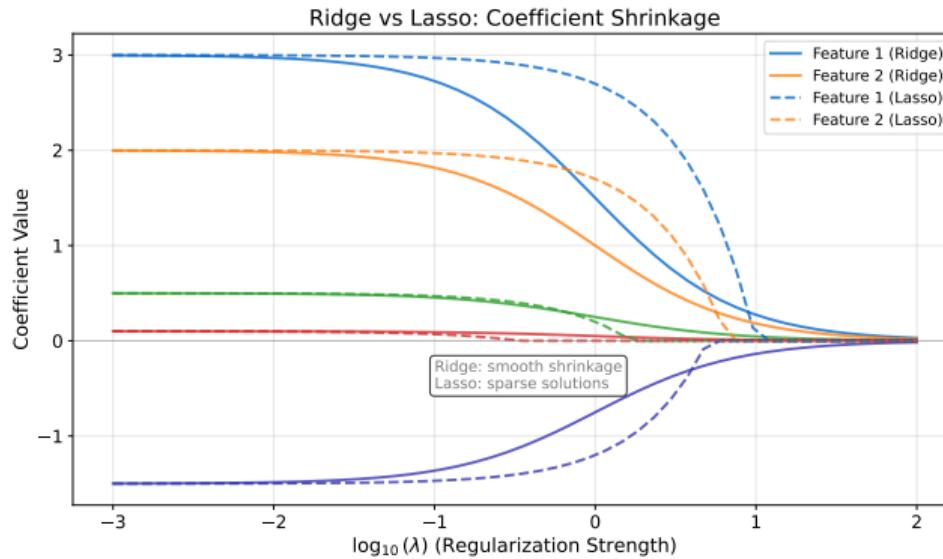
L1 Penalty: Sum of Absolute Coefficients

$$L_{\text{lasso}}(\beta) = \|y - X\beta\|^2 + \lambda\|\beta\|_1 \quad (19)$$

Properties:

- Produces sparse solutions (some $\beta_j = 0$)
- Automatic feature selection
- No closed-form solution (use coordinate descent)

Lasso: Least Absolute Shrinkage and Selection Operator



Ridge: smooth shrinkage. Lasso: sparse (feature selection)

Combining L1 and L2 Penalties

$$L_{\text{elastic}}(\beta) = \|y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2 \quad (20)$$

Benefits:

- Sparsity from L1
- Stability from L2 (handles correlated features)
- Two hyperparameters to tune

Often best of both worlds for correlated features

Cross-Validation for Hyperparameter Tuning

- ① Define grid of λ values (e.g., 10^{-4} to 10^4)
- ② For each λ , perform K-fold CV
- ③ Select λ with lowest CV error
- ④ Refit on full training data

In Practice:

- `sklearn.linear_model.RidgeCV`
- `sklearn.linear_model.LassoCV`

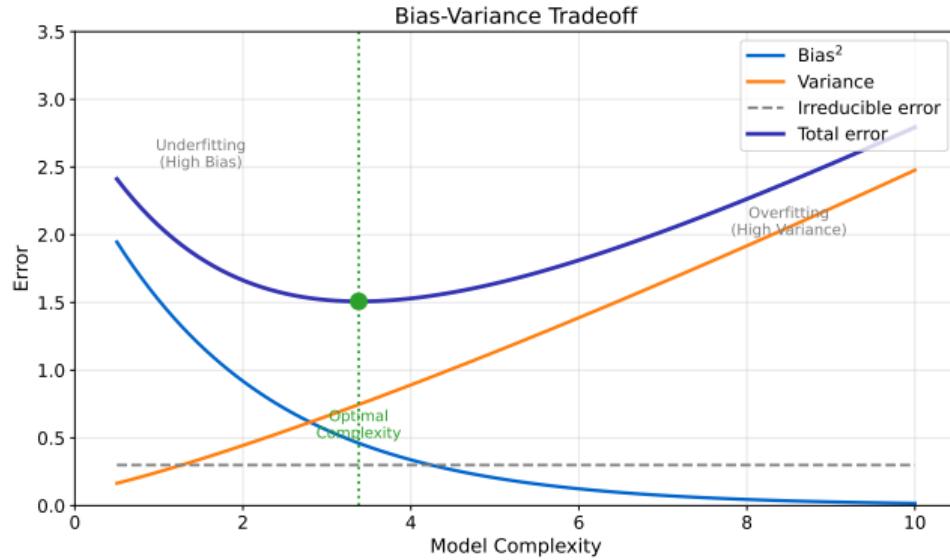
Larger λ = more regularization = simpler model

Expected Prediction Error

$$E[(y - \hat{f}(x))^2] = \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f}) + \sigma^2 \quad (21)$$

- **Bias:** Error from wrong assumptions (underfitting)
- **Variance:** Error from sensitivity to training data (overfitting)
- σ^2 : Irreducible noise in data

We can't reduce irreducible error – focus on bias and variance



Optimal complexity minimizes total error

How Regularization Helps

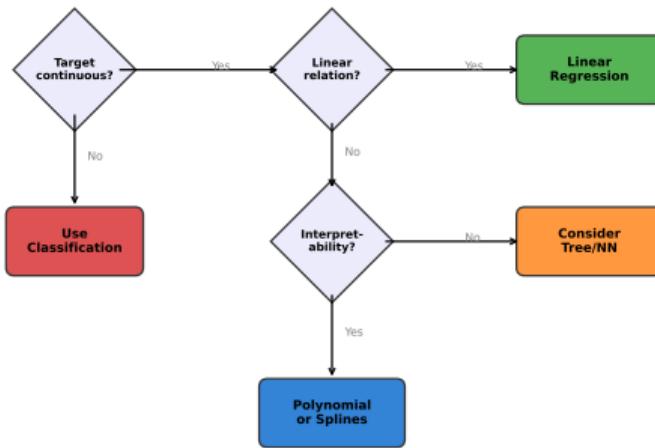
- Increasing λ : **increases bias, decreases variance**
- Decreasing λ : decreases bias, increases variance
- Optimal λ : minimizes total error

In Practice:

- Use CV to find optimal λ
- Regularization almost always helps when p is large

Regularization trades a little bias for a lot of variance reduction

Linear Regression Decision Guide



Use this framework when choosing regression methods

Use When:

- Continuous target variable
- Approximate linear relationships
- Interpretability is critical
- Inference on coefficients needed
- Fast prediction required

Avoid When:

- Target is categorical
- Strong non-linear patterns
- Many outliers present
- Features highly correlated
- Prediction accuracy paramount

When in doubt, linear regression is a strong baseline

$$\text{Model: } \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (22)$$

$$\text{OLS Solution: } \hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \quad (23)$$

$$\text{Gradient: } \nabla L = -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \quad (24)$$

$$\text{GD Update: } \boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha \nabla L \quad (25)$$

$$\text{Ridge: } \hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \quad (26)$$

$$R^2 : 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \quad (27)$$

- ① Linear regression minimizes squared error – closed form or GD
- ② Matrix notation enables efficient computation
- ③ Gradient descent scales to large datasets
- ④ Regularization (Ridge/Lasso) prevents overfitting
- ⑤ The bias-variance tradeoff guides model complexity
- ⑥ Always evaluate on held-out test data

Next Session: Logistic Regression for Classification

- James, Witten, Hastie, Tibshirani (2021). *Introduction to Statistical Learning*. Chapter 3.
- Hastie, Tibshirani, Friedman (2009). *Elements of Statistical Learning*. Chapter 3.
- Bishop (2006). *Pattern Recognition and Machine Learning*. Chapter 3.

Online Resources:

- scikit-learn: https://scikit-learn.org/stable/modules/linear_model.html
- Stanford CS229: <https://cs229.stanford.edu/>