Methods and Algorithms

MSc Data Science

Spring 2026

Finance Use Case: House Price Prediction

- Banks need accurate property valuations for mortgage decisions
- Insurers must estimate replacement costs
- Investors evaluate real estate portfolios

Why Linear Regression?

- Interpretable coefficients (how much does each feature matter?)
- Regulatory requirement for explainable models
- Fast, well-understood baseline

Linear regression: the workhorse of quantitative finance since the 1800s

The Model in Matrix Form

$$\mathrm{y} = \mathrm{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

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

Matrix notation enables elegant derivations and efficient computation

The Design Matrix X

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

- 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)

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

Expanding:

$$L(\boldsymbol{\beta}) = \mathrm{y}^\top \mathrm{y} - 2\boldsymbol{\beta}^\top \mathrm{X}^\top \mathrm{y} + \boldsymbol{\beta}^\top \mathrm{X}^\top \mathrm{X}\boldsymbol{\beta} \tag{4}$$

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

Taking the Derivative

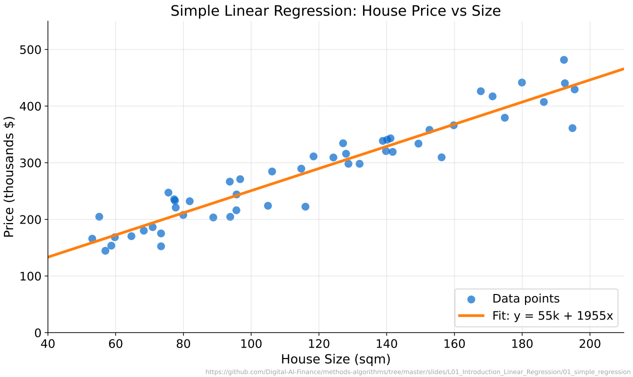
$$\frac{\partial L}{\partial \boldsymbol{\beta}} = -2\mathsf{X}^\top \mathsf{y} + 2\mathsf{X}^\top \mathsf{X}\boldsymbol{\beta} \tag{5}$$

Setting to Zero:

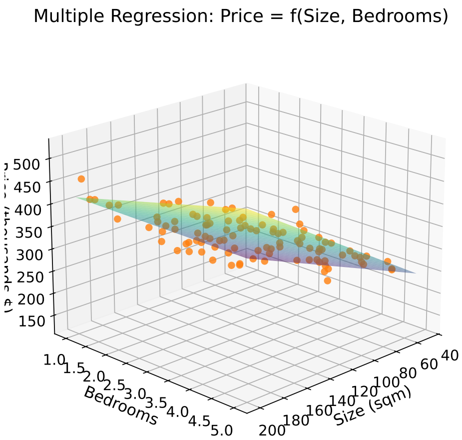
$$\mathsf{X}^\top \mathsf{X}\hat{\boldsymbol{\beta}} = \mathsf{X}^\top \mathsf{y} \tag{6}$$

$$\hat{\boldsymbol{\beta}} = (\mathsf{X}^\top \mathsf{X})^{-1}\mathsf{X}^\top \mathsf{y} \tag{7}$$

This is the closed-form OLS solution – the "normal equation"



The fitted line minimizes vertical distances squared



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L01_Introduction_Linear_Regression/02_multiple_regression_3d

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

$$\nabla L(\boldsymbol{\beta}) = -2\mathsf{X}^\top(\mathsf{y} - \mathsf{X}\boldsymbol{\beta}) = -2\mathsf{X}^\top \mathsf{r} \tag{8}$$

where $\mathsf{r} = \mathsf{y} - \mathsf{X}\boldsymbol{\beta}$ 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

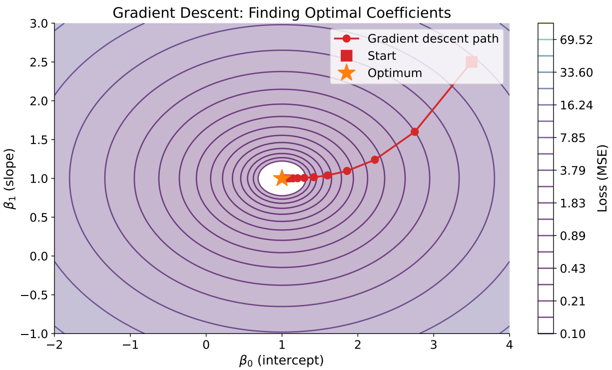
Update Rule

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

Algorithm:

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

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



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L01_Introduction_Linear_Regression/04_gradient_descent

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:

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

Use mini-batch of size m:

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

- $m = 1$: 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

$$R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{12}$$

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

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

Properties:

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

Use $R^2_{\mathbf{adj}}$ when comparing models with different p

Error Metrics in Original Units

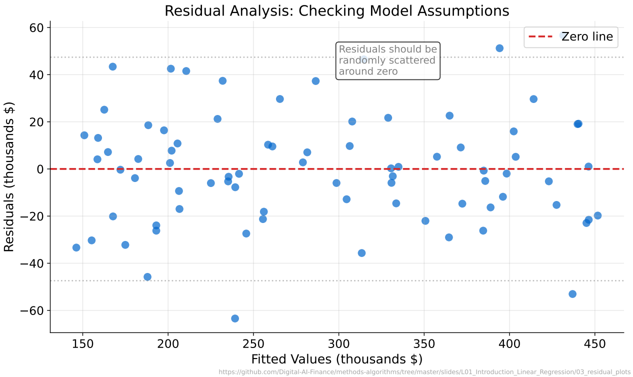
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \tag{14}$$

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

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

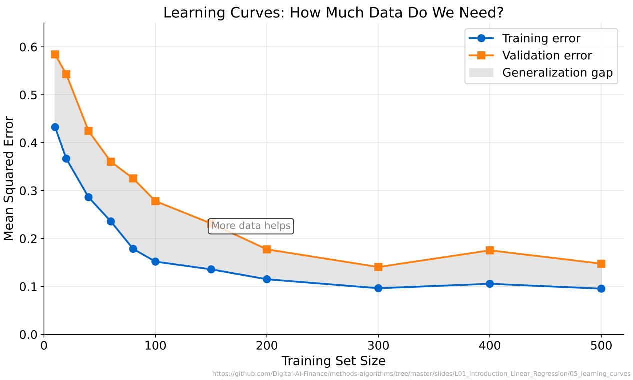
Evaluating Generalization

- 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

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

λ controls strength of regularization

L2 Penalty: Sum of Squared Coefficients

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

Closed-Form Solution:

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

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

Ridge adds λ to diagonal – stabilizes inversion

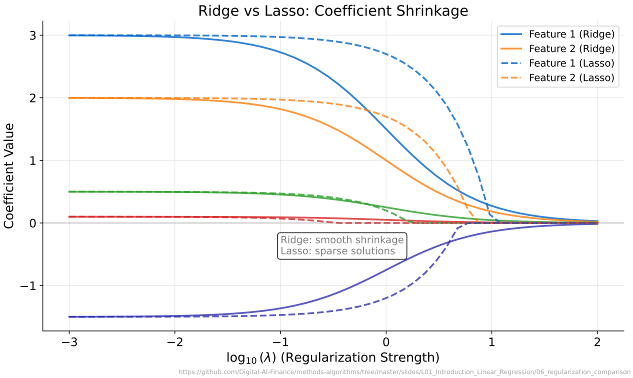
L1 Penalty: Sum of Absolute Coefficients

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

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

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

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`

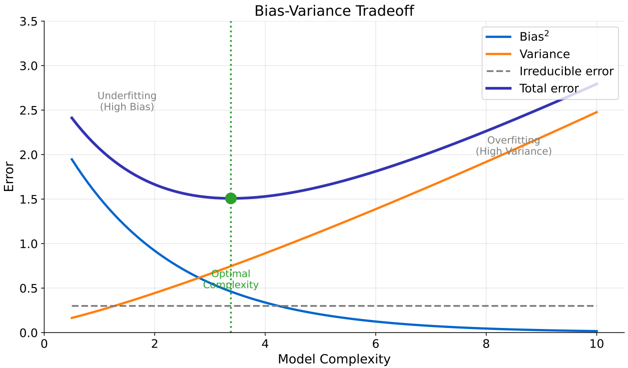
Larger λ = more regularization = simpler model

Expected Prediction Error

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

- **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



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L01_Introduction_Linear_Regression/07_bias_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:

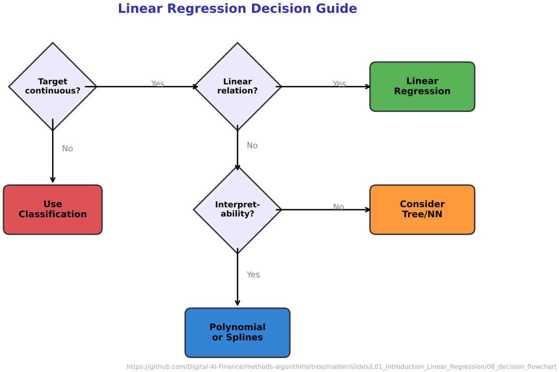
- Use CV to find optimal λ
- Regularization almost always helps when p is large

Regularization trades a little bias for a lot of variance reduction

Open the Colab Notebook

- Exercise 1: Implement OLS from scratch
- Exercise 2: Use scikit-learn LinearRegression
- Exercise 3: Compare with gradient descent

Link: `https://colab.research.google.com/` [TBD]



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

$$\text{Model:} \quad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{22}$$

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

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

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

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

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

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/`