Lecture 2: Linear Regression

Intuition $\rightarrow$ OLS $\rightarrow$ Ridge/Lasso $\rightarrow$ evaluation & diagnostics.

Luu Minh Sao Khue

October 11, 2025

Novosibirsk State University

Learning Objectives

- Explain simple & multiple linear regression.
- Derive OLS and interpret projection geometry.
- Understand Ridge/Lasso and bias–variance trade-off.
- Use R^2, RMSE, residuals, leverage, VIF.
- Implement models in Python and validate properly.

Intuition

Scatter a feature vs. target; trend is roughly linear. We fit a line/plane minimizing squared vertical errors.

- Simple: one predictor; Multiple: many predictors.
- Links optimization, geometry, and statistics.
- Regularization stabilizes estimates with many/collinear features.

Simple Linear Regression (SLR)

Population

$$\min_{\beta_0,\beta_1} \mathbb{E}\left[(Y - \beta_0 - \beta_1 X)^2\right]$$

Solution

$$\beta_1^* = \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}(X)}, \quad \beta_0^* = \mathbb{E}[Y] - \beta_1^* \mathbb{E}[X]$$

- Sample estimators replace expectations with sample means.
- Prediction: $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$.

Multiple Linear Regression (MLR)

Model

$$y = X\beta + \varepsilon, \quad X \in \mathbb{R}^{n\times(p+1)}$$

Objective

$$J(\beta) = \|y - X\beta\|_2^2$$

Closed form

$$\hat{\beta} = (X^\top X)^{-1}X^\top y \quad \text{(if } X^\top X \text{ invertible)}$$

- Geometry: $\hat{y} = X\hat{\beta}$ is the projection of y onto $\mathrm{col}(X)$; residuals r orthogonal to columns.

If $X^\top X$ is Singular

- Causes: $p > n$, exact collinearity.
- Minimum-norm LS solution:

$$\hat{\beta} = X^+ y \quad \text{(Moore–Penrose pseudoinverse).}$$

- Ridge ensures $(X^\top X + \lambda I)$ is invertible.

OLS Assumptions & BLUE

- Model: $y = X\beta + \varepsilon$.
- Assumptions: $E[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2 I$, independence.
- Gauss–Markov: OLS is BLUE (min variance among linear unbiased estimators).
- Normality is only needed for exact t/F inference, not for unbiasedness.

Why Regularization?

- High variance with many/collinear predictors.
- Add penalty on coefficient size to stabilize estimation.
- Bias–variance trade-off: small bias can reduce test error.

Ridge Regression (ℓ_2)

Objective

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

Solution

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

- Shrinks coefficients; no exact zeros.
- Helps with multicollinearity; improves conditioning.
- Standardize features; do not penalize intercept.

Lasso Regression (ℓ_1)

Objective

$$\min_{\beta} \; \|y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

- No closed form; solved via coordinate descent.
- Encourages sparsity (feature selection).
- Standardize features; do not penalize intercept.

Ridge vs. Lasso

	Ridge	Lasso
Penalty	ℓ_2	ℓ_1
Closed form	Yes	No
Sparsity	No	Yes
Use case	Multicollinearity	Many features & selection

Elastic Net: $\|y - X\beta\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$ combines stability and sparsity.

Tuning: choose λ via cross-validation.

Evaluation: R^2 & Error Metrics

R^2

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}, \quad \text{RSS} = \sum_i (y_i - \hat{y}_i)^2$$

- Adjusted R^2: $1 - \frac{(1-R^2)(n-1)}{n-p-1}$ (p: # predictors, no intercept).
- $\text{RMSE} = \sqrt{\text{RSS}/n}$, RSE $\hat{\sigma} = \sqrt{\text{RSS}/(n-p-1)}$.
- CI (mean) vs. PI (new obs): PI is wider (adds noise variance).

Diagnostics: Residuals, Leverage, VIF

- Residuals: zero mean, no pattern, constant variance.
- Hat matrix $H = X(X^\top X)^{-1}X^\top$; leverage h_{ii}.
- Influential points: Cook's distance.
- Multicollinearity: $\mathrm{VIF}_j = 1/(1 - R_j^2)$.

Algorithm: Gradient Descent

Loss

$$J(\beta) = \frac{1}{n}\|y - X\beta\|_2^2$$

Update

$$\beta^{(t+1)} = \beta^{(t)} - \eta \frac{2}{n} X^\top (X\beta^{(t)} - y)$$

- Use when p is large or data stream in.
- Tune η; stop on small improvement.

Python Demo: OLS (scikit-learn)

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([[1], [2], [3], [4]])
y = np.array([2, 4, 6, 8])

model = LinearRegression().fit(X, y)
print("coef:", model.coef_, "intercept:", model.intercept_)
print("R2:", model.score(X, y))
```

Python Demo: Ridge & Lasso

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge, Lasso

X = np.array([[1],[2],[3],[4],[5]])
y = np.array([2.2,4.1,5.9,8.2,10.1])

scaler = StandardScaler().fit(X)
Xs = scaler.transform(X)

ridge = Ridge(alpha=1.0).fit(Xs, y)
lasso = Lasso(alpha=0.1).fit(Xs, y)

print("ridge coef:", ridge.coef_)
print("lasso coef:", lasso.coef_)
```

Notation

- Statistics/econometrics: β for coefficients.
- General ML/optimization: θ for parameters.
- Neural nets/engineering: w (weights), b (bias).

We use β for regression coefficients in this course.

Practical Tips

- Standardize/center features; do not penalize intercept.
- Use CV or hold-out for model/penalty selection.
- Check residuals/leverage before trusting coefficients.
- Avoid extrapolation; beware of data leakage in preprocessing.

Summary

- OLS: projection view, BLUE under classical assumptions.
- Ridge/Lasso: control complexity; bias–variance trade-off.
- Metrics: R^2, adjusted R^2, RMSE; CI vs. PI.
- Diagnostics: residuals, leverage, multicollinearity (VIF).

Exercises

1. Derive $\nabla_\beta J(\beta)$ and implement GD.
2. Compare OLS vs. Ridge/Lasso with CV on a toy set.
3. Compute VIF and identify multicollinearity.

Further Reading

- Stanford CS229 Lecture Notes (Regression).
- ISLR (James et al.), Ch. 3; ESL (HTF), Ch. 3.
- MIT 6.036: Linear Models.