Introduction & Linear Regression

Deep Dive: Mathematics and Implementation

Methods and Algorithms

MSc Data Science

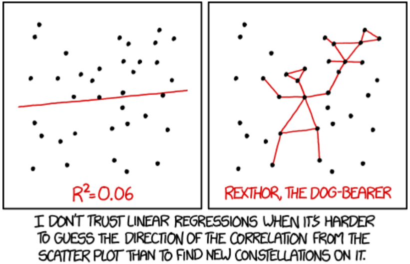
Spring 2026

Outline

Today's Deep Dive

In the overview, we learned that linear regression fits the best line through data. Now we go deeper:

- How do we **derive** the solution mathematically?
- How do we **optimize** when data is too large for a closed-form solution?
- How do we **know** if our model is any good?



XKCD #1725 by Randall Munroe (CC BY-NC 2.5) – The math behind the "best line"

Learning Objectives

By the end of this session, you will be able to:

1. **Derive** the OLS estimator and prove its optimality under Gauss-Markov assumptions
2. **Analyze** gradient descent convergence and evaluate learning rate selection
3. **Evaluate** regression diagnostics to identify assumption violations
4. **Compare** regularization strategies (Ridge, Lasso, Elastic Net) for different problem structures

Finance Applications: Property valuation, asset pricing (CAPM)

Foundation for all supervised learning methods

Definition

The linear model in matrix form:

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

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

Design Matrix

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

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

Matrix notation enables elegant derivations and efficient computation

What Must Be True for OLS to Work?

Classical Assumptions for Valid Inference

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

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

What Are We Minimizing?

Sum of Squared Residuals (SSR)

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

Expanding:

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

Quadratic in β – has unique minimum if X is full rank

How Do We Solve for the Best Coefficients?

Taking the Derivative

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

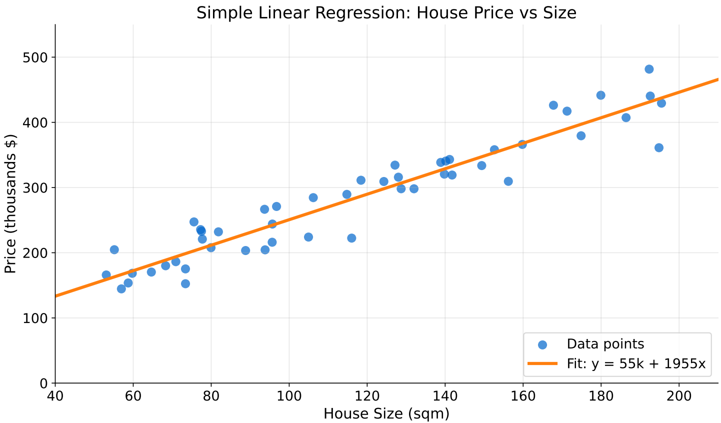
Setting to Zero:

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

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

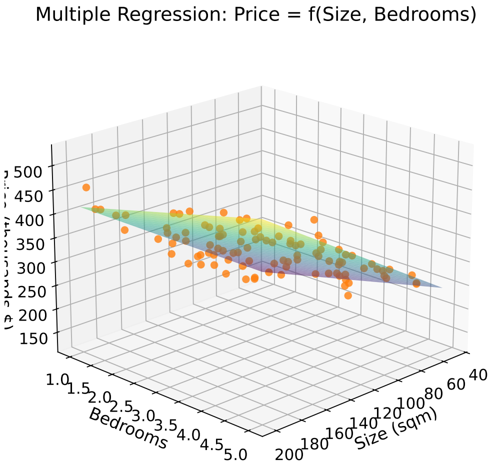
This is the closed-form OLS solution

What Does the Best Fit Look Like?



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L01_Introduction_Linear_Regression/01_simple_regression

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

Why Gradient Descent?

Normal Equation

- Computing $(\mathbf{X}^\top\mathbf{X})^{-1}$ is $O(p^3)$
- Must store $p \times p$ matrix
- Exact solution but slow for large p

Gradient Descent

- 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

How Does Each Step Work?

Gradient of the Loss

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

- Gradient points in direction of steepest ascent
- We move *opposite* to gradient

Update Rule

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

Algorithm:

1. Initialize $\boldsymbol{\beta}^{(0)}$ (zeros or random)
2. Compute gradient ∇L
3. Update $\boldsymbol{\beta}$
4. Repeat until convergence

Convergence: $\|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t\| < \varepsilon$ **or max iterations**

How Do We Choose the Step Size?

Learning Rate α

- **Too small**: Slow convergence, many iterations
- **Too large**: Divergence, oscillation
- **Practical**: Start with $\alpha = 0.01$, use adaptive methods (Adam)

Mini-Batch SGD

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

SGD is how we train all modern ML models, not just linear regression

Why Must We Scale Features?

Standardization: Zero Mean, Unit Variance

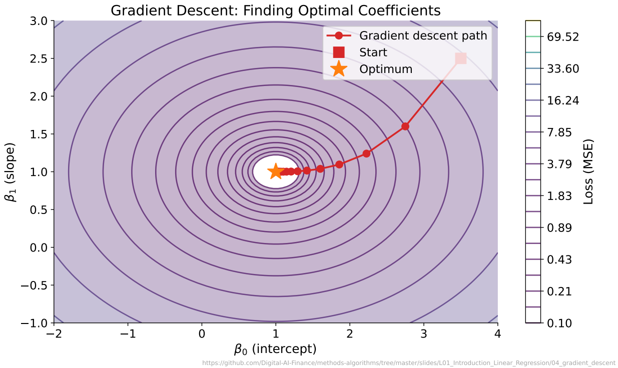
$$z_j = \frac{x_j - \mu_j}{\sigma_j} \tag{10}$$

Why Scale Features?

1. **Coefficient comparison**: After scaling, $|\beta_j|$ reflects relative importance
2. **Gradient descent**: Converges faster with similar feature scales
3. **Regularization**: Fair penalty across all features

Always standardize for regularization; optional for OLS if only predicting

How Does Gradient Descent Converge?



- The loss surface is a paraboloid for OLS – gradient descent follows the steepest path
- Learning rate controls step size; too large causes oscillation

For OLS, gradient descent always converges to the global minimum (convexity)

How Much Variance Does Our Model Explain?

R^2: Coefficient of Determination

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{11}$$

- Proportion of variance explained
- $R^2 = 0$: no better than mean
- $R^2 = 1$: perfect fit
- Always increases with more features

Adjusted R^2

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{12}$$

- Penalizes for number of predictors p
- Can decrease with irrelevant features
- Better for model comparison

Use adjusted R^2 when comparing models with different numbers of features

How Far Off Are Our Predictions?

Error Metrics in Original Units

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \tag{13}$$

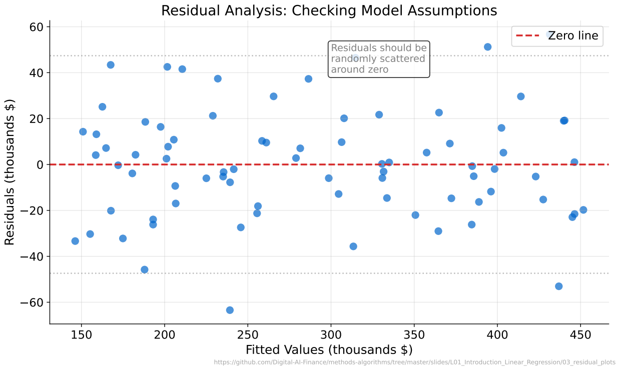
$$\text{MAE} = \frac{1}{n}\sum|y_i - \hat{y}_i| \tag{14}$$

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

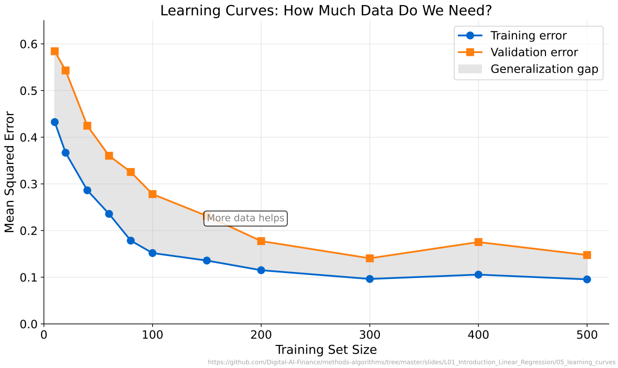
What Do the Residuals Tell Us?



- Random scatter = assumptions satisfied; patterns = model misspecification
- Funnel shape = heteroscedasticity; use robust standard errors
- Curvature = missing nonlinear terms; consider polynomial features

Residual plots are the single most important diagnostic – always plot them first

What Do Learning Curves Tell Us?



- Training error increases with more data (harder to memorize)
- Test error decreases then plateaus (diminishing returns from more data)
- The gap between curves reveals overfitting vs underfitting

Learning curves are a diagnostic tool – they tell you whether to get more data or a better model

Which Coefficients Are Statistically Significant?

t-Test for Coefficient β_j

Standard error:

$$\text{SE}(\hat{\beta}_j) = \hat{\sigma}\sqrt{[(\mathbf{X}^\top \mathbf{X})^{-1}]_{jj}} \tag{15}$$

Test statistic:

$$t_j = \frac{\hat{\beta}_j}{\text{SE}(\hat{\beta}_j)} \sim t_{n-p-1} \tag{16}$$

Reject $H_0 : \beta_j = 0$ if p-value < 0.05

95% Confidence Interval

$$\hat{\beta}_j \pm t_{n-p-1,\, 0.975} \times \text{SE}(\hat{\beta}_j) \tag{17}$$

- 95% of intervals contain true β_j
- CI excludes 0 $\Leftrightarrow$ significant at 5%
- Prediction intervals are wider (include σ^2)

Always check significance before interpreting coefficients

Does the Model as a Whole Explain Anything?

Null hypothesis: $H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$

$$F = \frac{R^2/p}{(1 - R^2)/(n - p - 1)} \sim F_{p,\, n-p-1} \tag{18}$$

- Tests whether the model **as a whole** explains significant variance
- Reject H_0 if p-value $< \alpha$
- Complements t-tests: possible to have no significant t-tests but a significant F-test (multicollinearity)

Check the F-statistic before interpreting individual coefficients

What Happens When We Have Too Many Features?

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

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

λ controls strength of regularization

Which Penalty Should We Choose?

Ridge (L2)

$$L + \lambda \|\boldsymbol{\beta}\|_2^2$$

- Closed-form solution
- Shrinks all toward zero
- Never exactly zero

Lasso (L1)

$$L + \lambda \|\boldsymbol{\beta}\|_1$$

- No closed-form
- Sparse solutions
- Automatic feature selection

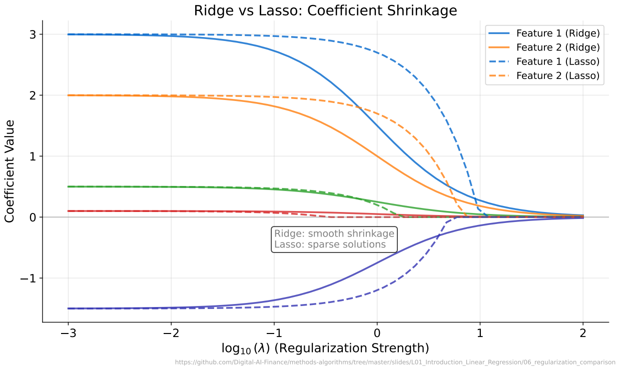
Elastic Net

$$L + \lambda\left(\alpha \|\boldsymbol{\beta}\|_1 + \frac{1-\alpha}{2} \|\boldsymbol{\beta}\|_2^2\right)$$

- Mixing parameter α
- Handles correlated features
- Best of both worlds

Ridge for many small effects, Lasso for feature selection, Elastic Net for correlated features

How Does Regularization Shrink Coefficients?



- Ridge shrinks all coefficients toward zero but never reaches it
- Lasso drives some coefficients exactly to zero (feature selection)
- Larger λ = more shrinkage = simpler model

The regularization path shows how coefficients change as λ increases

How Do We Tune the Regularization Strength?

Cross-Validation for Hyperparameter Tuning

1. **Define grid** of λ values (e.g., 10^{-4} to 10^{4}, log-spaced)
2. **K-fold CV**: For each λ, compute average validation error
3. **Select** λ with lowest CV error
4. **Refit** on full training data with chosen λ

In Practice:

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

Larger λ = more regularization = simpler model

Why Can't We Eliminate All Error?

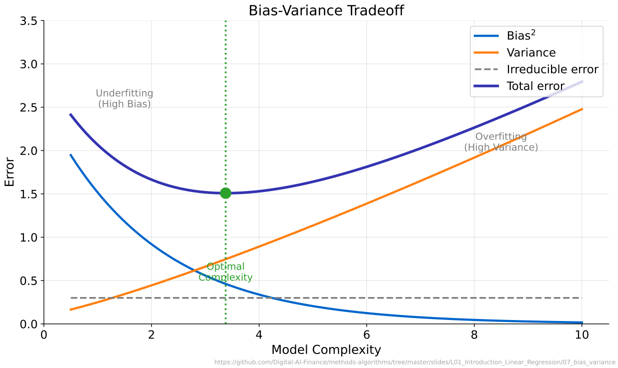
Expected Prediction Error

$$E[(y - \hat{f}(x))^2] = \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f}) + \sigma^2 \quad (20)$$

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

We cannot reduce irreducible error – focus on bias and variance

Where Is the Sweet Spot?



- Total error $=$ Bias^2 $+$ Variance $+$ Irreducible noise
- The minimum total error occurs at intermediate model complexity
- Regularization controls where we sit on this curve

The optimal model balances underfitting (high bias) and overfitting (high variance)

How Do We Detect Correlated Features?

Variance Inflation Factor

$$\text{VIF}_j = \frac{1}{1 - R_j^2} \tag{21}$$

where R_j^2 is from regressing x_j on all other features.

- $\text{VIF} = 1$: No correlation
- $\text{VIF} > 5$: Moderate concern
- $\text{VIF} > 10$: Serious multicollinearity

Remedies

- Remove highly correlated features
- Use Ridge regression ($\lambda > 0$ stabilizes)
- Apply PCA before regression

Why it matters:

- Inflated standard errors
- Unstable coefficient estimates
- Misleading significance tests

Always check VIF before trusting coefficient estimates

How Do We Know the Model Generalizes?

Simple Split

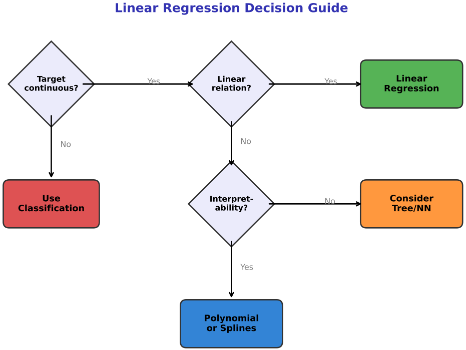
- 70–80% train, 20–30% test
- Report test set metrics
- Fast but high variance estimate

K-Fold Cross-Validation

- Split into K folds (typically 5 or 10)
- Train on $K-1$ folds, validate on 1
- More reliable with limited data

Always evaluate on data the model has never seen

When Should You Use Linear Regression?



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L01_Introduction_Linear_Regression/08_decision_flowchart

- Start with linear regression as baseline before trying complex models
- If assumptions hold and relationships are approximately linear, OLS is hard to beat
- If $p > n$, use regularization; if nonlinear patterns, consider tree-based methods

The simplest model that meets your accuracy threshold is usually the best choice

Key Equations Summary

Model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$
$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Optimization

$$\nabla L = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha\nabla L$$

Regularization

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$
$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Keep this slide as a reference sheet

Key Takeaways

1. Linear regression minimizes squared error – closed form or gradient descent
2. Matrix notation enables efficient computation and elegant derivations
3. Gradient descent scales to large datasets where normal equations fail
4. Regularization (Ridge/Lasso/Elastic Net) prevents overfitting
5. The bias-variance tradeoff guides model complexity decisions

Next Session: Logistic Regression for Classification

Foundation for all supervised learning – next: Logistic Regression

Until Next Time...

"I used to think fitting a line was trivial.

Then I learned about heteroscedasticity, multicollinearity, regularization, and the bias-variance tradeoff.

Now I think fitting a line is an art."

— Adapted from XKCD #1725

XKCD #1725 callback – Next session: Logistic Regression turns this line into a curve

Appendix: Advanced Topics and Proofs

These slides are supplementary material for self-study

Finance Application: The CAPM

Capital Asset Pricing Model – Linear Regression in Finance

$$R_i - R_f = \alpha_i + \beta_i(R_m - R_f) + \varepsilon_i \tag{22}$$

- R_i: Return of asset i
- R_f: Risk-free rate (e.g., T-bill)
- R_m: Market return (e.g., S&P 500)
- β_i: Systematic risk (market sensitivity)

Interpretation: $\beta = 1.2$ means 10% market rise $\Rightarrow$ 12% expected asset rise

CAPM: The original factor model – basis for portfolio management

Why OLS is Special

Under assumptions 1–4 (linearity, exogeneity, homoscedasticity, full rank):

OLS is BLUE – Best Linear Unbiased Estimator

- **Best**: Lowest variance among all linear unbiased estimators
- **Linear**: Estimator is a linear function of $\mathbf{y}$
- **Unbiased**: $E[\hat{\beta}] = \beta$

Gauss-Markov justifies why OLS is the default for linear regression

Gauss-Markov: Proof Sketch

Goal: Show OLS $\hat{\boldsymbol{\beta}}$ has minimum variance among all linear unbiased estimators.
Proof: Let $\tilde{\boldsymbol{\beta}} = \mathbf{C}\mathbf{y}$ be any other linear unbiased estimator. Write $\mathbf{C} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' + \mathbf{D}$ for some matrix $\mathbf{D}$.

- Unbiasedness requires $E[\tilde{\boldsymbol{\beta}}] = \boldsymbol{\beta}$, which forces $\mathbf{DX} = \mathbf{0}$
- Then: $\mathrm{Var}(\tilde{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1} + \sigma^2\mathbf{DD}'$
- Since $\mathbf{DD}' \succeq \mathbf{0}$:

$$\mathrm{Var}(\tilde{\boldsymbol{\beta}}) - \mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2\mathbf{DD}' \succeq \mathbf{0} \tag{23}$$

Therefore OLS has the smallest variance among all linear unbiased estimators. $\square$

Any deviation from OLS adds noise without reducing bias

OLS and Maximum Likelihood Estimation

Under normality $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, the likelihood is:

$$L(\boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2}{2\sigma^2}\right) \tag{24}$$

Maximizing the log-likelihood:

- $\frac{\partial \ell}{\partial \boldsymbol{\beta}} = 0$ gives the **same normal equations as OLS**
- $\hat{\sigma}^2_{\text{MLE}} = \frac{\text{RSS}}{n}$ (biased; OLS uses $n - p - 1$)
- This connection enables likelihood ratio tests, AIC/BIC model comparison

Key insight: OLS $\equiv$ MLE under normality $\Rightarrow$ all MLE properties transfer to OLS.

OLS = MLE under normality – bridges to Logistic Regression (L02)

Formal Assumption Diagnostic Tests

Assumption	**Test**	H_0
Homoscedasticity	Breusch-Pagan	Constant variance
	White test	Constant variance (robust)
No autocorrelation	Durbin-Watson	No serial correlation
Normality	Shapiro-Wilk	Residuals are normal
	Jarque-Bera	Skewness=0, kurtosis=3
Functional form	Ramsey RESET	No omitted nonlinearities

When assumptions fail:

- Heteroscedasticity $\Rightarrow$ White/HC robust standard errors
- Autocorrelation $\Rightarrow$ Newey-West standard errors, GLS
- Non-normality $\Rightarrow$ Bootstrap inference (large n: CLT helps)

Regulators require formal test results – "the residuals looked fine" is insufficient

Hat Matrix and Cook's Distance

The **hat matrix** maps observed to fitted values: $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ where

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \tag{25}$$

Leverage: $h_{ii} \in [1/n, 1]$ measures how far $\mathbf{x}_i$ is from the center of the design space. High leverage ($h_{ii} > 2p/n$): observation *could* be influential.
Cook's Distance combines leverage and residual:

$$D_i = \frac{e_i^2}{p \cdot \text{MSE}} \cdot \frac{h_{ii}}{(1 - h_{ii})^2} \tag{26}$$

- $D_i > 1$: observation substantially changes $\hat{\beta}$ when removed
- Critical in finance: a single crisis observation can distort the entire model

Investigate points with $D_i > 4/n$ or leverage $h_{ii} > 2(p+1)/n$

Convergence Rates for Gradient Descent

- **Convex**: $O(1/t)$ convergence rate (sub-linear)
- **Strongly convex**: $O(\rho^t)$ where $\rho < 1$ (linear rate)

Learning Rate Condition:

$$\eta < \frac{2}{L} \tag{27}$$

where L is the Lipschitz constant of ∇L.

For OLS specifically, the optimal learning rate is:

$$\eta^* = \frac{1}{\lambda_{\max}(\mathbf{X}^\top \mathbf{X})} \tag{28}$$

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

References and Resources

Textbooks

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

Online Resources

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

See also Andrew Ng's CS229 lectures for implementation-focused treatment