Linear Regression

AERO 689: Introduction to Machine Learning for Aerospace Engineers

Dr. Raktim Bhattacharya

Texas A&M University - Aerospace Engineering

Learning Objectives

- Apply linear regression to aerospace performance prediction
- Understand least squares method and gradient descent
- Implement drag coefficient prediction from wind tunnel data
- Validate models using aerospace-specific metrics

The Fuel Crisis Challenge

The $100 Million Question

Scenario: An airline operates 200 aircraft

- **Fuel cost**: $50M+ annually per aircraft type
- **Challenge**: Predict fuel consumption for flight planning
- **Current method**: Simplified performance charts
- **ML opportunity**: Precise models using real flight data

The Fuel Crisis Challenge (contd.)

Real Impact

- 1% fuel savings = \$100M+ industry-wide annually
- Better range predictions = route optimization
- Accurate payload calculations = safety + efficiency
  Question for class: *What factors affect aircraft fuel consumption?*

From Wind Tunnel to Flight - The Data Challenge

Traditional Approach: Empirical Models

Parabolic Drag Polar

$$C_D = C_{D_0} + KC_L^2$$

- **Problem**: Assumes perfect conditions
- **Reality**: Real flights have weather, weight variations, engine degradation
- **Solution**: ML to learn from actual operational data

From Wind Tunnel to Flight (contd.)

Available Data Sources

1. **Wind Tunnel Data**: Controlled, precise, limited conditions
2. **Flight Test Data**: Real conditions, expensive to collect
3. **Operational Data**: Massive scale, noisy, representative

The Linear Regression Framework

- **Goal**: Predict drag coefficient (CD) from flight parameters
- **Input features**: Angle of attack (α), Mach number (M), Reynolds number (Re)
- **Output**: Drag coefficient for performance calculations

Mathematical Foundation: The Linear Model

General Form

For n samples and d features, the linear regression model is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_d x_{id} + \epsilon_i$$

where:

- y_i: response variable (e.g., drag coefficient)
- x_{ij}: j-th feature of i-th sample (e.g., Mach, α, Re)
- β_j: regression coefficients (parameters to learn)
- ϵ_i: error term (noise, unmodeled physics)

Mathematical Foundation (contd.)

Vector Notation

$$y = X\beta + \epsilon$$

where $X \in \mathbb{R}^{n\times(d+1)}$ is the **design matrix** with augmented 1's for intercept

Matrix Formulation

Design Matrix Structure

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Matrix Formulation: Aerospace Example

Drag Prediction

Design matrix:

$$X = \begin{bmatrix} 1 & \alpha_1 & M_1 & \mathsf{Re}_1 & \alpha_1^2 & \alpha_1 M_1 & \cdots \\ 1 & \alpha_2 & M_2 & \mathsf{Re}_2 & \alpha_2^2 & \alpha_2 M_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Parameters: $\beta = \begin{bmatrix} C_{D_0} & k_\alpha & k_M & k_{\mathsf{Re}} & k_{\alpha^2} & k_{\alpha M} & \cdots \end{bmatrix}^T$

Targets: $y = \begin{bmatrix} C_{D_1} & C_{D_2} & \cdots & C_{D_n} \end{bmatrix}^T$

Key Insight: What "Linear" Means

"Linear Regression" = Linear in Parameters

The model:

$$y = \beta_0\phi_0(x) + \beta_1\phi_1(x) + \cdots + \beta_d\phi_d(x)$$

Where:

- $\phi_j(x)$ are **basis functions** (can be nonlinear!)
- β_j are **coefficients** (what we solve for)
- Model is **linear** in β_j, **not** in x

Key Insight: Why It Matters

Linearity in β implies:

- Closed-form solution exists
- Can model complex nonlinear phenomena
- Optimization remains convex (one global minimum)

Aerospace Example: Drag Model

Drag coefficient model:

$$C_D = \beta_0 + \beta_1\alpha + \beta_2\alpha^2 + \beta_3 M^2 + \beta_4(\alpha M)$$

Analysis:

- **Nonlinear function** of α and M (parabola, interactions)
- **Linear combination** of terms: $\beta_0 \cdot 1 + \beta_1 \cdot \alpha + \beta_2 \cdot \alpha^2 + \cdots$
- **Linear in coefficients**: doubling β_2 doubles the α^2 contribution

Matrix form: $y = X\beta^*$ where

$$X = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & M_1^2 & \alpha_1 M_1 \\ 1 & \alpha_2 & \alpha_2^2 & M_2^2 & \alpha_2 M_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Real Data: Noisy Measurements

Wind Tunnel Data Example

Key observations:

- Data points scatter around true parabolic relationship
- Noise represents: sensor precision limits, flow unsteadiness, model simplification
- **This is why we need the error term** ϵ_i in our model!

The Optimization Problem

Objective: Minimize Squared Error

Residual Sum of Squares (RSS):

$$\text{RSS}(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - x_i^T \beta)^2$$

Matrix form:

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

The Optimization Problem (contd.)

Optimization goal:

$$\beta^* = \arg\min_{\beta} \mathsf{RSS}(\beta)$$

Physical interpretation: *Find aircraft model parameters that best match observed performance data*

Derivation: Normal Equations

Step 1: Expand the objective function

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

Step 2: Take derivative with respect to β

$$\frac{\partial \text{RSS}}{\partial \beta} = -2X^T y + 2X^T X\beta$$

Derivation: Normal Equations (contd.)

Step 3: Set to zero and solve

$$X^T X\beta^* = X^T y$$

Normal Equations

Step 4: Solution (if $X^T X$ is invertible)

$$\beta^* = (X^T X)^{-1} X^T y$$

Geometric Interpretation: Understanding the Error

What is the Error?

Definition: The error (residual) is the difference between observed data and our prediction:

$$r = y - \hat{y} = y - X\beta$$

Our goal: Minimize the **length** of this error vector:

$$\min_{\beta} \|r\|^2 = \min_{\beta} \|y - X\beta\|^2$$

Geometric Interpretation (contd.)

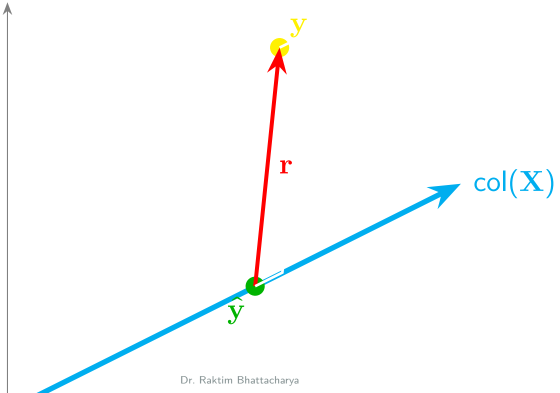
Key Geometric Insight

- **Column space col(X)**: Space spanned by basis functions (all possible predictions $X\beta$)
- **Data y**: Our actual observations (usually not in col(X) due to noise)
- **Question**: What is the **best** representation of y in the feature space?
  Answer: *Error is orthogonal to feature space* $X^T r = 0$

Why Projection Minimizes Error

The Fundamental Geometric Principle

Projection Theorem: The shortest distance from a point to a subspace is achieved by the **perpendicular projection**.



Why Projection Minimizes Error (contd.)

Key Insight:

- The optimal $\hat{y}$ is found by projecting y perpendicular onto col(X)
- This minimizes the error length: $||r|| = ||y - \hat{y}||$ is smallest
- Compare distances from y to different points on the line - perpendicular is shortest

Mathematical statement: $X^T r = X^T(y - \hat{y}) = 0$

Deriving the Optimal Solution via Projection

Step 1: State the Orthogonality Condition

From geometry: The error $r = y - \hat{y}$ must be perpendicular to $\text{col}(X)$

$$r \perp \text{col}(X) \quad \Longrightarrow \quad X^T r = 0$$

This is **the fundamental condition** for least squares optimality.

Deriving via Projection: Step 2

Express in Terms of

Since optimal $\hat{y}^* = X\beta^*$ and $r^* = y - \hat{y}^*$:

$$X^T r^* = 0$$

$$X^T(y - \hat{y}^*) = 0$$

$$X^T(y - X\beta^*) = 0$$

Now we have an equation for β^* that we can solve!

Deriving via Projection: Step 3

Derive the Normal Equations

Expand the orthogonality condition:

$$X^T(y - X\beta^*) = 0$$

$$X^T y - X^T X\beta^* = 0$$

$$X^T X\beta^* = X^T y$$

These are the **Normal Equations** – a linear system for β^*.

Deriving via Projection: Step 4

Solve for β^*

Starting from the normal equations:

$$X^T X\beta^* = X^T y$$

Assuming $X^T X$ is invertible:

$$\beta^* = (X^T X)^{-1} X^T y$$

This is the **closed-form solution** for ordinary least squares!

Key insight: *We derived this using* ***projection geometry*** *(r col(X)) instead of* ***calculus*** *($J/$ $=$ 0). Both paths lead to the same solution!*

The Projection Matrix

Mathematical Form

From the normal equations, we get:

$$\beta^* = (X^T X)^{-1} X^T y$$

Predicted values:

$$\hat{y} = X\beta^* = X(X^T X)^{-1} X^T y = Py$$

where $P = X(X^T X)^{-1} X^T$ is the **projection matrix**

Projection Matrix: Key Properties

1. **Idempotent**: $P^2 = P$ (projecting twice = projecting once)
2. **Symmetric**: $P^T = P$
3. **Projects onto col**(X): $PX = X$
4. **Residual matrix**: $I - P$ projects onto orthogonal complement
   Aerospace insight: *The projection matrix* P *extracts the component of observed drag that can be explained by our aerodynamic features, leaving unexplained variance in the residuals*

When Direct Solution Fails

Challenges with $(X^T X)^{-1}$

Problem 1: Singular Matrix

- Occurs when $n < d + 1$ (more features than samples)
- Multicollinearity: highly correlated features

Problem 2: Computational Cost

- Matrix inversion: $O(d^3)$ operations
- For large d (high-dimensional features), impractical

Problem 3: Numerical Stability

- Ill-conditioned matrices (high condition number)
- Small perturbations $\rightarrow$ large changes in solution

Solutions When Direct Method Fails

1. **Regularization**: Ridge, Lasso
2. **Gradient descent**: Iterative optimization
3. **QR decomposition**: Numerically stable direct method
4. **SVD**: Most stable, handles rank deficiency

Gradient Descent: Iterative Approach

Algorithm

Initialize: $\beta^{(0)}$ randomly or to zeros

Iterate: For $t = 0, 1, 2, \dots$ until convergence:

$$\beta^{(t+1)} = \beta^{(t)} - \eta \nabla_\beta \text{RSS}(\beta^{(t)})$$

where $\eta > 0$ is the **learning rate**

Gradient Descent: Gradient Computation

Gradient Computation

$$\nabla_\beta \text{RSS} = -2X^T(y - X\beta)$$

Update rule:

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

Gradient Descent Variants

Batch Gradient Descent

Use all n samples in each iteration:

$$\beta^{(t+1)} = \beta^{(t)} - \eta \nabla_\beta \text{RSS}(\beta^{(t)})$$

- Stable convergence
- Slow for large n

Stochastic Gradient Descent (SGD)

Use one random sample i per iteration:

$$\beta^{(t+1)} = \beta^{(t)} + 2\eta x_i (y_i - x_i^T \beta^{(t)})$$

- Fast updates, scales to large data
- Noisy, oscillates around minimum

Mini-Batch Gradient Descent

Use subset of b samples per iteration (typical: $b = 32, 64, 128$)

- Balance between speed and stability
- Vectorized operations (GPU-friendly)

Learning Rate Selection

Critical Hyperparameter

Too small $(\eta \ll 1)$:

- Slow convergence
- Many iterations needed
- Computationally expensive

Too large $(\eta \gg 1)$:

- Overshooting minimum
- Oscillation or divergence
- Never converges

Adaptive Learning Rates

1. **Learning rate decay**: $\eta_t = \frac{\eta_0}{1+kt}$
2. **Momentum**: Use exponentially weighted moving average of gradients
3. **Adam**: Adaptive moment estimation (modern default)
4. **Line search**: Optimize η at each iteration

Statistical Properties: Assumptions

Classical Linear Regression Assumptions

1. **Linearity**: True relationship is $y = X\beta + \epsilon$
2. **Independence**: Samples (x_i, y_i) are i.i.d.
3. **Homoscedasticity**: Constant error variance $\text{Var}(\epsilon_i) = \sigma^2$
4. **Normality**: $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

Statistical Properties: No Multicollinearity

No multicollinearity: $X^T X$ is full rank

Aerospace context: Features should not be perfectly correlated

Common violations:

- Altitude and air density (directly related via ISA)
- Dynamic pressure and velocity ($q \propto V^2$)
- Mach number and velocity at fixed altitude

Solutions: Regularization, PCA, or remove redundant features

Gauss-Markov Theorem

Under assumptions 1-3

The OLS estimator $\beta^* = (X^T X)^{-1} X^T y$ is:

- **BLUE**: Best Linear Unbiased Estimator
- Minimum variance among all unbiased linear estimators

What Does BLUE Mean?

Best: Minimum variance (most precise estimates)

- Among all linear unbiased estimators, OLS has smallest variance

Linear: Estimator is linear function of y

- Form: $\beta^* = Cy$ for some matrix C

Unbiased: $E[\beta^*] = \beta_{\text{true}}$

- On average, estimates equal true parameter values

Statistical Inference: Understanding Uncertainty

Why Do We Care About Uncertainty?

Engineering Question: After fitting $C_D = \beta_0 + \beta_1 \alpha + \beta_2 \alpha^2$, how confident are we in the coefficients?

Key Concepts:

1. **Standard Error (SE)**: Measures uncertainty in each coefficient
2. **Confidence Intervals**: Range of plausible values for each coefficient
3. **Statistical Significance**: Is a coefficient meaningfully different from zero?

Model Significance: Does Your Model Work?

Simple Check - R^2 Statistic

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = \frac{\text{Variance Explained}}{\text{Total Variance}}$$

- **R^2 0**: Model is useless (no better than average)
- **R^2 1**: Model explains almost all variance (excellent fit)
- **Rule of thumb for aerospace**: $R^2 > 0.7$ often acceptable for preliminary design

Model Significance: F-statistic

Statistical Test

Tests if **any** features are useful (reported by most software)

- **Large F-value** (e.g., $F > 10$): Strong evidence model is useful
- **Small p-value** ($p < 0.05$): Model significantly better than baseline

Bottom line: Check that your model's p-value is small (« 0.05) before using it!

Model Evaluation: R-squared

The "Goodness of Fit" Metric

R^2 tells you: What fraction of the variance is explained by your model?

$$R^2 = 1 - \frac{\text{Prediction errors}}{\text{Total variance}} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Think of it as: *How much better is my model than just using the average?*

Interpreting R^2 Values

$R^2 = 0.99$ (Excellent fit)

- **Example**: Altitude vs atmospheric pressure
- Model captures nearly all variation

$R^2 = 0.75$ (Decent fit)

- **Example**: Fuel consumption vs flight parameters
- Model captures main trends but misses some complexity

$R^2 = 0.30$ (Weak fit)

- **Example**: Turbulence severity vs weather variables
- Many unmeasured factors influence outcome

The R^2 Trap

Problem: R^2 Always Increases with More Features!

Model 1: $C_D = \beta_0 + \beta_1 M \rightarrow R^2 = 0.78$

Model 2: Add Reynolds number $\rightarrow R^2 = 0.85$

Model 3: Add random noise feature $\rightarrow R^2 = 0.86 \leftarrow$ Still increased!

The problem: R^2 rewards complexity even when features add no value

Adjusted R^2

Solution: Penalize Adding Useless Features

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

Only increases if new feature improves fit more than expected by chance

Prediction Accuracy: RMSE

Root Mean Squared Error

RMSE = Average prediction error **in the same units as your measurement**

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

Interpretation: "On average, predictions are off by RMSE units"

RMSE: Aerospace Example

Range Prediction

- Model predicts aircraft range for different payloads
- After many flights: **RMSE = 85 km**
- **Interpretation**: "Range predictions are typically off by 85 km"
- **Decision**: Is $\pm$85 km acceptable for mission planning?

MAE vs RMSE

Mean Absolute Error

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

MAE: Treats all errors equally

RMSE: Penalizes large errors heavily (squaring effect)

Aviation rule of thumb:

- **Operational planning**: MAE okay
- **Safety limits**: Use RMSE (penalizes big misses)

The Overfitting Problem

Training vs Real-World Performance

Simple model: Training RMSE $= 0.08$, Test RMSE $= 0.09$ (Similar)

Complex model: Training RMSE $= 0.02$, Test RMSE $= 0.31$ (Much worse!)

What happened? Complex model **overfit** the noise in training data

Cross-Validation: Testing Without a Test Set

The Problem with Single Train-Test Splits

Performance depends heavily on which data you held out

Solution: Cross-validation

- Everyone gets a chance to be test data
- Average performance across all test groups
- More reliable estimate of real-world performance

Cross-Validation: How It Works

5-Fold Cross-Validation Example

1. Divide data into 5 groups
2. Train 5 models, each time holding out a different group
3. Average error across all 5 test groups

Result: Every single data point was used for testing exactly once!

Cross-Validation: How Many Folds?

k = 5 or k = 10 (Most common)

- Good balance between computational cost and reliability

k = 20 or Leave-One-Out (Expensive but thorough)

- When data is very limited

Aerospace best practice:

- Research/development: k=10 standard
- Certification data (expensive): Consider leave-one-out

Model Evaluation Metrics Summary

Metric	Units	When to Use
R^2	Unitless	Initial model assessment
Adj. R^2	Unitless	Model selection (complexity)
RMSE	Same as y	**Primary metric (safety)**
MAE	Same as y	Operational planning
MAPE	Percentage	Comparing different scales

Aerospace Application: Drag Prediction

Problem Setup

Goal: Predict drag coefficient from wind tunnel data

Features:

- Angle of attack: α (deg)
- Mach number: M
- Reynolds number: Re

Target: Drag coefficient C_D

Linear Model with Interaction Terms

$$C_D = \beta_0 + \beta_1\alpha + \beta_2 M + \beta_3 \mathsf{Re} + \beta_4\alpha^2 + \beta_5 M^2 + \beta_6\alpha M + \epsilon$$

Rationale:

- Parabolic drag polar: $C_D \propto \alpha^2$ (induced drag)
- Wave drag: $C_D \propto M^2$ (transonic effects)
- Compressibility: αM interaction

Feature Engineering for Aerodynamics

Polynomial Features

```python
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2, include_bias=True)
X_poly = poly.fit_transform(X)

# [ , M, Re] → [1,  , M, Re,  ², ×M, ×Re, M², M×Re, Re²]
```

Model is still **linear in the coefficients** but **nonlinear in the inputs**

Use Domain Knowledge For Feature Selection

Rather than blindly creating all polynomials, use **aerodynamic theory**:

1. **Dynamic pressure**: $q = \frac{1}{2}\rho V^2 \propto M^2$
2. **Lift-induced drag**: $C_{D_i} = \frac{C_L^2}{\pi e AR}$
3. **Prandtl-Glauert correction**: $C_p = \frac{C_{p,0}}{\sqrt{1-M^2}}$
   Physics-informed feature engineering beats blind polynomial expansion

Scaling and Normalization

Why Scale Features?

Problem: Features have different ranges

- $\alpha \in [0°, 20°]$
- $M \in [0.3, 0.9]$
- $\text{Re} \in [10^6, 10^7]$

Issues: Gradient descent dominated by large-scale features

Standardization

$$x_j^{\text{scaled}} = \frac{x_j - \mu_j}{\sigma_j}$$

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Practical Implementation

Scikit-learn Workflow - Setup

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# 1. Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

Practical Implementation (contd.)

Training and Evaluation

```
# 2. Train model
model = LinearRegression()
model.fit(X_train, y_train)

# 3. Evaluate
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

Residual Analysis

Diagnostic Plots

1. Residuals vs. Fitted Values

- Check for patterns (should be random)
- Funnel shape $\rightarrow$ heteroscedasticity

2. Q-Q Plot

- Check normality assumption
- Points should lie on diagonal

3. Residuals vs. Features

- Identify missing nonlinear terms
- Detect outliers

Outliers and Influential Points

Leverage and Cook's Distance

Leverage: How far is x_i from the center of the data?

Cook's Distance: Combined effect of leverage and residual

Rule of thumb: $D_i > 1$ suggests influential point

Aerospace Context

Outliers might be: sensor errors, unusual flight conditions, or model breakdown

Limitations of Linear Regression

When Linear Models Fail

1. **Nonlinear Relationships**: Transonic drag rise, post-stall aerodynamics
2. **Extrapolation Issues**: Dangerous in aerospace (safety-critical)
3. **Model Assumptions**: Homoscedasticity rarely holds in real flight data

Solutions

- Neural networks, tree-based methods, physics-informed ML

Extensions: Weighted Least Squares

When: Heteroscedastic errors (variance varies with x)

$$\beta^* = \arg\min_{\beta} \sum_{i=1}^{n} w_i (y_i - x_i^T \beta)^2$$

Solution: $\beta = (X^T W X)^{-1} X^T W y$

Regularization Preview (Week 3)

Ridge Regression (L2 Regularization)

$$\beta^{*,\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \right\}$$

Solution: $\beta^{*,\text{ridge}} = (X^T X + \lambda I)^{-1} X^T y$

Benefit: Always invertible, even when $X^T X$ is singular

Multicollinearity Detection

Variance Inflation Factor (VIF)

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

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

Rule of thumb:

- VIF < 5: Low correlation
- VIF > 10: Problematic multicollinearity

Practical Tips for Aerospace ML

Data Quality Matters

1. **Sensor calibration**: Check for drift, bias
2. **Data fusion**: Combine multiple sources
3. **Outlier handling**: Physics-based filtering
4. **Missing data**: Interpolation vs. imputation

Practical Tips (contd.)

Feature Selection Strategy

1. Start with **physical model** (drag polar, Breguet range)
2. Add **polynomial terms** based on theory
3. Use **domain expertise** to limit feature space
4. **Cross-validate** to prevent overfitting

Practical Tips (contd.)

Model Validation

- **Train-test split**: 80-20 or 70-30
- **K-fold CV**: For limited data
- **Holdout by flight**: Test on different aircraft/conditions
- **Physics checks**: Verify positive drag, reasonable trends

Key Takeaways

Mathematical Foundations

1. Linear regression minimizes squared error: $\min \|y - X\beta^*\|^2$
2. Closed-form solution: $\beta^* = (X^T X)^{-1} X^T y$
3. Gradient descent for large-scale problems
4. Statistical properties: BLUE under Gauss-Markov

Key Takeaways (contd.)

Practical Implementation

1. Feature scaling essential for numerical stability
2. Cross-validation for generalization assessment
3. Residual analysis for model diagnostics
4. Domain knowledge guides feature engineering

Key Takeaways (contd.)

Aerospace Applications

- Drag prediction, fuel flow modeling, performance estimation
- Physics-informed features improve accuracy
- Always validate against known aerodynamic principles

Next Week: Model Evaluation & Regularization

Preview of Week 3

Topics:

1. Bias-variance tradeoff
2. Ridge regression (L2)
3. Lasso regression (L1)
4. Elastic net
5. Cross-validation for hyperparameter tuning

Aerospace focus: Preventing overfitting in high-dimensional aerodynamic models