Linear Regression Models

Formulation, Objectives, and Optimization

Cedric Kiplimo

Centre for Data Science and Artificial Intelligence (DSAIL)

July 6, 2026

Goal of Regression

- Predict continuous target variable t given D-dimensional input $\mathbf{x}$.

Goal of Regression

- Predict continuous target variable t given D-dimensional input $\mathbf{x}$.
- Key property: Models are linear functions of adjustable parameters.

Goal of Regression

- Predict continuous target variable t given D-dimensional input $\mathbf{x}$.
- Key property: Models are linear functions of adjustable parameters.
- Generalization: Moving from straight lines to flexible feature combinations.

The Linear Basis Function Model

- Richer class of functions via nonlinear transformations (**basis functions** ϕ).

The Linear Basis Function Model

- Richer class of functions via nonlinear transformations (**basis functions** ϕ).
- General Model:

$$\hat{y}(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

The Linear Basis Function Model

- Richer class of functions via nonlinear transformations (**basis functions** ϕ).
- General Model:

$$\hat{y}(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

- **The Golden Rule:** Still "Linear" because it is linear in the **parameters w**.

The Linear Basis Function Model

- Richer class of functions via nonlinear transformations (**basis functions** ϕ).
- General Model:

$$\hat{y}(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

- **The Golden Rule:** Still "Linear" because it is linear in the **parameters w**.
- Equation in vector form:

$$\hat{y}(x, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(x)$$

Common Basis Functions

- **Identity:** $\phi_j(x) = x_j$ (Standard multiple regression).

Common Basis Functions

- **Identity:** $\phi_j(x) = x_j$ (Standard multiple regression).
- **Polynomial:** $\phi_j(x) = x^j$ (Global curve fitting).

Common Basis Functions

- **Identity:** $\phi_j(x) = x_j$ (Standard multiple regression).
- **Polynomial:** $\phi_j(x) = x^j$ (Global curve fitting).
- **Gaussian:** Localized distance mapping:

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$

The Design Matrix (Φ)

- How to handle N observations?

The Design Matrix (Φ)

- How to handle N observations?
- **Design Matrix:** $N \times M$ matrix stacking all samples.

The Design Matrix (Φ)

- How to handle N observations?
- **Design Matrix:** $N \times M$ matrix stacking all samples.
- First column is always 1s (for the bias w_0).

The Design Matrix (Φ)

- How to handle N observations?
- **Design Matrix:** $N \times M$ matrix stacking all samples.
- First column is always 1s (for the bias w_0).
- Compact prediction for the entire dataset:

$$\hat{\mathbf{y}} = \Phi \mathbf{w}$$

The Probabilistic Framework

- Real world noise: $y = \mathbf{w}^T \phi(x) + \epsilon$

The Probabilistic Framework

- Real world noise: $y = \mathbf{w}^T \phi(x) + \epsilon$
- **Noise (ϵ):** Assumed Gaussian centered at zero.

The Probabilistic Framework

- Real world noise: $y = \mathbf{w}^T \phi(x) + \epsilon$
- **Noise (ϵ):** Assumed Gaussian centered at zero.
- **Precision (β):** Inverse variance $1/\sigma^2$.

The Probabilistic Framework

- Real world noise: $y = \mathbf{w}^T \phi(x) + \epsilon$
- **Noise (ϵ):** Assumed Gaussian centered at zero.
- **Precision (β):** Inverse variance $1/\sigma^2$.
- High $\beta \rightarrow$ Noise is tiny.

The Probabilistic Framework

- Real world noise: $y = \mathbf{w}^T \phi(x) + \epsilon$
- **Noise (ϵ):** Assumed Gaussian centered at zero.
- **Precision (β):** Inverse variance $1/\sigma^2$.
- High $\beta \rightarrow$ Noise is tiny.
- Note: Include Figure 1.16 here showing Gaussian distribution around the line.

Maximum Likelihood Intuition

- Find parameters that make observed data "most probable."

Maximum Likelihood Intuition

- Find parameters that make observed data "most probable."
- Total Likelihood $= p(y_1) \times p(y_2) \cdots \times p(y_N)$.

Maximum Likelihood Intuition

- Find parameters that make observed data "most probable."
- Total Likelihood $= p(y_1) \times p(y_2) \cdots \times p(y_N)$.
- **Log-Likelihood:** Converts product to summation.

Maximum Likelihood Intuition

- Find parameters that make observed data "most probable."
- Total Likelihood $= p(y_1) \times p(y_2) \cdots \times p(y_N)$.
- **Log-Likelihood:** Converts product to summation.
- Result: Maximizing log-likelihood is equivalent to minimizing the **Sum-of-Squares Error**.

Geometry of Least Squares

- Target labels vector $\mathbf{y}$ exists in N-dimensional space.

Geometry of Least Squares

- Target labels vector $\mathbf{y}$ exists in N-dimensional space.
- Columns of Φ span an M-dimensional subspace.

Geometry of Least Squares

- Target labels vector $\mathbf{y}$ exists in N-dimensional space.
- Columns of Φ span an M-dimensional subspace.
- Shortest distance is a perpendicular line.

Geometry of Least Squares

- Target labels vector $\mathbf{y}$ exists in N-dimensional space.
- Columns of Φ span an M-dimensional subspace.
- Shortest distance is a perpendicular line.
- **Result:** OLS solution is an **orthogonal projection**.

Geometry of Least Squares

- Target labels vector $\mathbf{y}$ exists in N-dimensional space.
- Columns of Φ span an M-dimensional subspace.
- Shortest distance is a perpendicular line.
- **Result:** OLS solution is an **orthogonal projection**.
- Note: Include Figure 3.2 here for visualization.

The Normal Equation

- Minimize $E_D(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y})$.

The Normal Equation

- Minimize $E_D(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y})$.
- Setting gradient to zero yields:

$$\Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{y}$$

The Normal Equation

- Minimize $E_D(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y})$.
- Setting gradient to zero yields:

$$\Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{y}$$

- **Analytical Solution:**

$$\mathbf{w}_{OLS} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$$

The Normal Equation

- Minimize $E_D(\mathbf{w}) = \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y})$.
- Setting gradient to zero yields:

$$\Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{y}$$

- **Analytical Solution:**

$$\mathbf{w}_{OLS} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$$

- Pseudo-Inverse ($\Phi^\dagger$) generalizes matrix inverse for non-square matrices.

Analytical Failure Cases

- **Multicollinearity:** Perfectly correlated features make matrix singular.

Analytical Failure Cases

- **Multicollinearity:** Perfectly correlated features make matrix singular.
- **Data Scarcity:** $N < M$ leads to extreme overfitting.

Analytical Failure Cases

- **Multicollinearity:** Perfectly correlated features make matrix singular.
- **Data Scarcity:** $N < M$ leads to extreme overfitting.
- **Complexity:** Matrix inversion is $O(M^3)$.

Analytical Failure Cases

- **Multicollinearity:** Perfectly correlated features make matrix singular.
- **Data Scarcity:** $N < M$ leads to extreme overfitting.
- **Complexity:** Matrix inversion is $O(M^3)$.
- If $M = 100,000$, we need iterative approaches.

Gradient Descent

- Start with random weights, step opposite to the gradient.

Gradient Descent

- Start with random weights, step opposite to the gradient.
- **Batch Update:**

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} E_D(\mathbf{w})$$

Gradient Descent

- Start with random weights, step opposite to the gradient.
- **Batch Update:**

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} E_D(\mathbf{w})$$

- η is the **learning rate** (controls step size).

Sequential Learning (SGD)

- Batch GD is slow for large N.

Sequential Learning (SGD)

- Batch GD is slow for large N.
- **Stochastic Gradient Descent (SGD):** Update based on one point at a time.

Sequential Learning (SGD)

- Batch GD is slow for large N.
- **Stochastic Gradient Descent (SGD):** Update based on one point at a time.
- **Comparison:**
  - Batch: Smooth path, expensive steps.
  - SGD: Erratic path, fast/cheap steps.

Analytical vs. Iterative Summary

Property	Analytical	Iterative (GD)
Goal	Solve directly	Iterative tweaks
Cost	$O(M^3)$	$O(MN)$ per iter
Scaling	Small features	Massive features

Analytical vs. Iterative Summary

Property	Analytical	Iterative (GD)
Goal	Solve directly	Iterative tweaks
Cost	$O(M^3)$	$O(MN)$ per iter
Scaling	Small features	Massive features

Closing thought: Error landscape is perfectly convex; you are guaranteed to reach the same global minimum.