Linear Regression

Thien Phuc, Tuan Ngoc, Khoa Nguyen, Phuc Nguyen

Faculty of Computer Science
University of Information Technology (UIT)
Vietnam National University - Ho Chi Minh City (VNU-HCM)

October, 2025

Contents

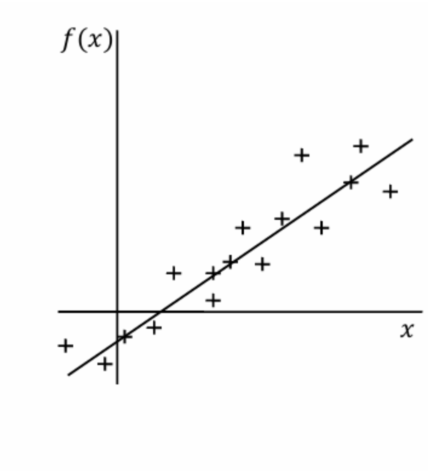
Linear regression: Introduction

- **Regression problem:** learn a function $y = f(\mathbf{x})$ from a given training data
$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$$
such that $y_i \approx f(\mathbf{x}_i)$ for every i.
  - Each observation of $\mathbf{x}$ is represented by a vector in an n-dimensional space, e.g.,
  $$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$$
  Each dimension represents an *attribute* / *feature* / *variate*.
  - Bold characters denote vectors.
- **Linear model:** if $f(\mathbf{x})$ is assumed to be of linear form
$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_n x_n$$
  - $w_1, \ldots, w_n$ are the regression coefficients / weights. w_0 is called the "bias".
- **Note:** learning a linear function is equivalent to learning the coefficient vector
$$\mathbf{w} = (w_0, w_1, \ldots, w_n)^T$$

Linear regression: Example

- What is the best function?

x	y
0.13	-0.91
1.02	-0.17
3.17	1.61
-2.76	-3.31
1.44	0.18
5.28	3.36
-1.74	-2.46
7.93	5.56
...	...



Prediction

- For each observation

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$$

  - The true output: $c_{\mathbf{x}}$
  - Prediction by our system:

$$y_{\mathbf{x}} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

  - We often expect $y_{\mathbf{x}} \approx c_{\mathbf{x}}$.

- Prediction for a future observation:

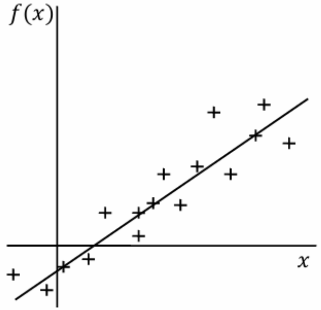
$$\mathbf{z} = (z_1, z_2, \ldots, z_n)^T$$

  - Use the learned function to make the prediction:

$$f(\mathbf{z}) = w_0 + w_1 z_1 + w_2 z_2 + \cdots + w_n z_n$$

Learning a regression function

- Learning goal: learn a function f^* such that its prediction in the future is the best.
  - Its generalization is the best.
- Difficulty: infinite number of functions.
  - How can we learn?
  - Is function f better than g?
- Use a measure.
  - Loss function is often used to guide learning.



Loss function

- **Definition:**
  - The *error/loss* of the prediction for an observation $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$:
  $$r(\mathbf{x}) = [c_x - f(\mathbf{x})]^2 = (c_x - w_0 - w_1 x_1 - \ldots - w_n x_n)^2$$
  - The *expected loss* of f over the whole space:
  $$E = E_{\mathbf{x}}[r(\mathbf{x})] = E_{\mathbf{x}}[c_x - f(\mathbf{x})]^2$$
  where $E_{\mathbf{x}}$ is the expectation over $\mathbf{x}$.
- The goal of learning is to find f^* that minimizes the expected loss:
$$f^* = \arg\min_{f \in \mathcal{H}} E_{\mathbf{x}}[r(\mathbf{x})]$$
where $\mathcal{H}$ is the space of functions of linear form.
- **But, we cannot work directly with this problem during the learning phase.** *(Why?)*

Cost, risk

Empirical loss

- We can only observe a set of training data

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\},$$

  and have to learn f from D.

- **Empirical loss** (residual sum of squares):

$$RSS(f) = \sum_{i=1}^{M} (y_i - f(\mathbf{x}_i))^2 = \sum_{i=1}^{M} (y_i - w_0 - w_1 x_{i1} - \cdots - w_n x_{in})^2$$

- $\frac{1}{M} RSS(f)$ is an approximation to $E_{\mathbf{x}}[r(\mathbf{x})]$.
- $\left| \frac{1}{M} RSS(f) - E_{\mathbf{x}}[r(\mathbf{x})] \right|$ is often known as generalization error of f.
- Many learning algorithms base on this RSS and its variants.

Convergence in Linear Regression (1)

Convergence refers to the point where the model's parameters stabilize at optimal values. Linear regression typically uses the **Gradient Descent** algorithm to minimize the loss function by iteratively updating weights.

1. Loss Function:

$$L(\mathbf{w}) = \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

where:

- $\mathbf{X} \in \mathbb{R}^{N \times d}$: normalized data matrix
- $\mathbf{y} \in \mathbb{R}^{N}$: true label vector
- $\mathbf{w} \in \mathbb{R}^{d}$: weight vector

Convergence in Linear Regression (2)

2. **Gradient:**

$$\nabla L(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{N}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{1}{N}\mathbf{X}^\top\mathbf{y}$$

3. **Hessian:**

$$\mathbf{H} := \nabla^2 L(\mathbf{w}) = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$$

Properties:

- $\mathbf{H}$ is symmetric and positive semidefinite.
- If columns of $\mathbf{X}$ are linearly independent $\Rightarrow$ $\mathbf{H}$ is positive definite.

Convergence in Linear Regression (3)

4. Optimal Solution:

$$\nabla L(\mathbf{w}^*) = 0 \Rightarrow \mathbf{X}^\top \mathbf{X}\mathbf{w}^* = \mathbf{X}^\top \mathbf{y}$$

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

5. Gradient Descent Update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \left(\frac{1}{N}\mathbf{X}^\top \mathbf{X}\mathbf{w}_t - \frac{1}{N}\mathbf{X}^\top \mathbf{y} \right)$$

Convergence in Linear Regression (4)

6. Parameter Error:

$$\mathbf{e}_t := \mathbf{w}_t - \mathbf{w}^* \quad \Rightarrow \quad \mathbf{e}_{t+1} = (\mathbf{I} - \eta \mathbf{H})\mathbf{e}_t$$

Recursive Form:

$$\mathbf{e}_t = (\mathbf{I} - \eta \mathbf{H})^t \mathbf{e}_0$$

Spectral Decomposition:

$$\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top, \quad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$$

$$(\mathbf{I} - \eta \mathbf{H})^t = \mathbf{Q} \cdot \mathrm{diag}((1 - \eta\lambda_i)^t) \cdot \mathbf{Q}^\top$$

$$\Rightarrow \mathbf{e}_t = \mathbf{Q} \cdot \mathrm{diag}((1 - \eta\lambda_i)^t) \cdot \mathbf{Q}^\top \mathbf{e}_0$$

Convergence in Linear Regression (5)

7. Convergence Condition:

$$|1 - \eta\lambda_i| < 1 \quad \forall\lambda_i > 0 \quad \Rightarrow \quad 0 < \eta < \frac{2}{\lambda_{\max}(\mathbf{H})}$$

Since:

$$\lambda_{\max}(\mathbf{H}) = \frac{1}{N}\sigma^2_{\max}(\mathbf{X}) \quad \Rightarrow \quad 0 < \eta < \frac{2N}{\sigma^2_{\max}(\mathbf{X})}$$

Note: If $\mathbf{H}$ is only positive semidefinite (some $\lambda_i = 0$), error components in those directions remain constant, while others converge to zero. $\Rightarrow$ $\mathbf{w}_t$ converges to a minimizer depending on initialization.

Methods: Ordinary Least Squares (OLS)

- Given dataset $\mathbf{D}$, we find f^* that minimizes the residual sum of squares (RSS):

$$f^* = \arg\min_{f \in \mathcal{H}} RSS(f)$$

$$\Leftrightarrow \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - w_0 - w_1 x_{i1} - \cdots - w_n x_{in})^2 \tag{1}$$

- This method is often known as ordinary least squares (OLS).
- Find $\mathbf{w}^*$ by taking the gradient of RSS and solving the equation $\nabla_{\mathbf{w}} RSS = 0$. Then we have:

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

- Where:
  - $\mathbf{X}$ is the data matrix of size $M \times (n+1)$, whose ith row is:

$\mathbf{X}_i = (1, x_{i1}, x_{i2}, \ldots, x_{im})$; $\mathbf{X}^{-1}$ denotes the inversion of matrix $\mathbf{X}$; $\mathbf{y} = (y_1, y_2, \ldots, y_M)^T$

- Note: we assume that $\mathbf{X}^T\mathbf{X}$ is invertible.

Methods: OLS

- **Input:**

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$$

- **Output: $\mathbf{w}^*$**
- **Learning: compute**

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

  - Where $\mathbf{X}$ is the data matrix of size $M \times (n+1)$, whose i^{th} row is

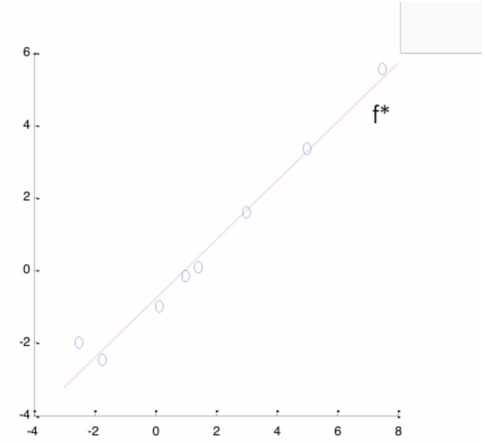
$$\mathbf{X}_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$$

  - $\mathbf{B}^{-1}$ denotes the inversion of matrix $\mathbf{B}$.
  - $\mathbf{y} = (y_1, y_2, \ldots, y_M)^T$.
  - Note: we assume that $\mathbf{X}^T\mathbf{X}$ is invertible.
- **Prediction for a new $\mathbf{x}$**:

$$y_x = w_0^* + w_1^* x_1 + \cdots + w_n^* x_n$$

Methods: OLS example

x	y
0.13	-0.91
1.02	-0.17
3.17	1.61
-2.76	-3.31
1.44	0.18
5.28	3.36
-1.74	-2.46
7.93	5.56
...	...



$$f^*(x) = 0.81x - 0.78$$

Methods: Limitations of OLS

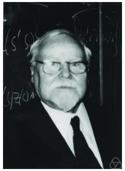
- OLS cannot work if $\mathbf{X}^\top \mathbf{X}$ is not invertible
  - If some columns (attributes/features) of $\mathbf{X}$ are dependent, then $\mathbf{X}$ will be singular and therefore $\mathbf{X}^\top \mathbf{X}$ is not invertible.
- OLS requires considerable computation due to the need of computing a matrix inversion.
  - Intractable for the very high dimensional problems.
- OLS very likely tends to overfitting, because the learning phase just focuses on minimizing errors on the training data.

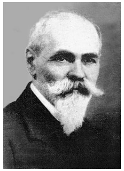
Methods: Ridge Regression (1)

- Given $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$, we solve for:

$$f^* = \arg\min_{f \in \mathcal{H}} \mathrm{RSS}(f) + \lambda \|\mathbf{w}\|^2$$
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - \mathbf{A}_i \mathbf{w})^2 + \lambda \sum_{j=1}^{n} w_j^2 \tag{2}$$

- Where $\mathbf{A}_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$ is composed from $\mathbf{x}_i$; and λ is a regularization constant $(\lambda > 0)$. $\|\mathbf{w}\|_2$ is the L_2 norm.


Tikhonov
smoothing an ill-posed problem


Zaremba
model complexity minimization

Andrew Ng
need no maths, but it prevents overfitting!

Methods: Ridge Regression (2)

- Problem (2) is equivalent to the following:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - \mathbf{A}_i\mathbf{w})^2 \quad \text{subject to} \quad \sum_{j=1}^{n} w_j^2 \leq t \quad (t \text{ is constant}) \tag{3}$$

- **Regularization/Penalty term:** $\lambda\|\mathbf{w}\|_2^2$
  - Limits the magnitude/size of $\mathbf{w}^*$ (i.e., reduces the search space for f^*).
  - Helps us to trade off between the fitting of f on $\mathbf{D}$ and its generalization on future observations.

Methods: Ridge Regression (3)

- We solve for $\mathbf{w}^*$ by taking the gradient of the objective function in (2), and then zeroing it. Therefore we obtain:

$$\mathbf{w}^* = (\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I}_{n+1})^{-1} \mathbf{A}^\top \mathbf{y}$$

Where:

- $\mathbf{A}$ is the data matrix of size $M \times (n+1)$, whose i^{th} row is $\mathbf{A}_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$.
- $\mathbf{B}^{-1}$ denotes the inversion of matrix $\mathbf{B}$.
- $\mathbf{y} = (y_1, y_2, \ldots, y_M)^\top$.
- $\mathbf{I}_{n+1}$ is the identity matrix of size $(n+1) \times (n+1)$.
- Compared with OLS, Ridge can:
  - Avoid the cases of singularity, unlike OLS. Hence Ridge always works.
  - Reduce overfitting.
  - But the error on the training data might be greater than OLS.
- **Note:** The predictiveness of Ridge depends heavily on the choice of the hyperparameter λ.

Methods: Ridge Regression (4)

- Input: $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$ and $\lambda > 0$
- Output: $\mathbf{w}^*$
- Learning: compute

$$\mathbf{w}^* = (\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I}_{n+1})^{-1} \mathbf{A}^\top \mathbf{y}$$

- Prediction for a new $\mathbf{x}$: $y_x = \mathbf{w_0}^* + \mathbf{w_1}^* \mathbf{x_1} + \ldots + \mathbf{w_n}^* \mathbf{x_n}$

An example of using Ridge and OLS

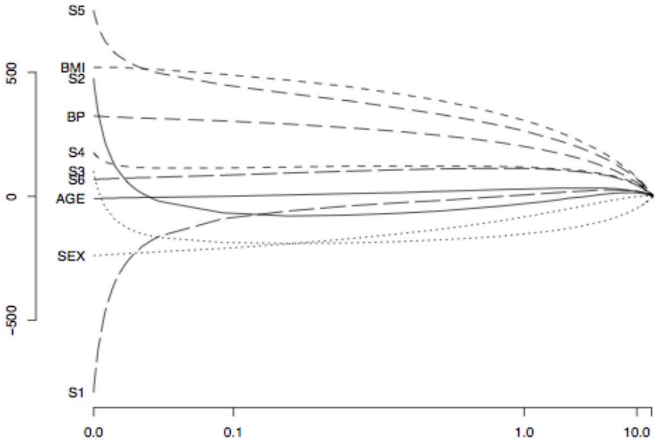
- The training set D contains 67 observations on prostate cancer, each was represented with 8 attributes. Ridge and OLS were learned from D, and then predicted 30 new observations.

w	**Ordinary Least Squares**	**Ridge**
Intercept (0)	2.465	2.452
lcavol	0.680	0.420
lweight	0.263	0.238
age	-0.141	-0.046
lbph	0.210	0.162
svi	0.305	0.227
lcp	-0.288	0.001
gleason	-0.021	0.040
pgg45	0.267	0.133
Test RSS	**0.521**	**0.492**

Table: Comparison of OLS and Ridge

Effects of λ in Ridge regression

- $\mathbf{w}^* = (w_0, S1, S2, S3, S4, S5, S6, AGE, SEX, BMI, BP)$ changes as the regularization constant λ changes.



Methods: Lasso Regression (1)

- Ridge regression uses the L_2 norm for regularization:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - \mathbf{A}_i\mathbf{w})^2 \quad \text{subject to} \quad \sum_{j=1}^{n} w_j^2 \leq t \tag{3}$$

- Replacing L_2 by L_1 norm will result in LASSO:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - \mathbf{A}_i\mathbf{w})^2 \quad \text{subject to} \quad \sum_{j=1}^{n} |w_j| \leq t$$

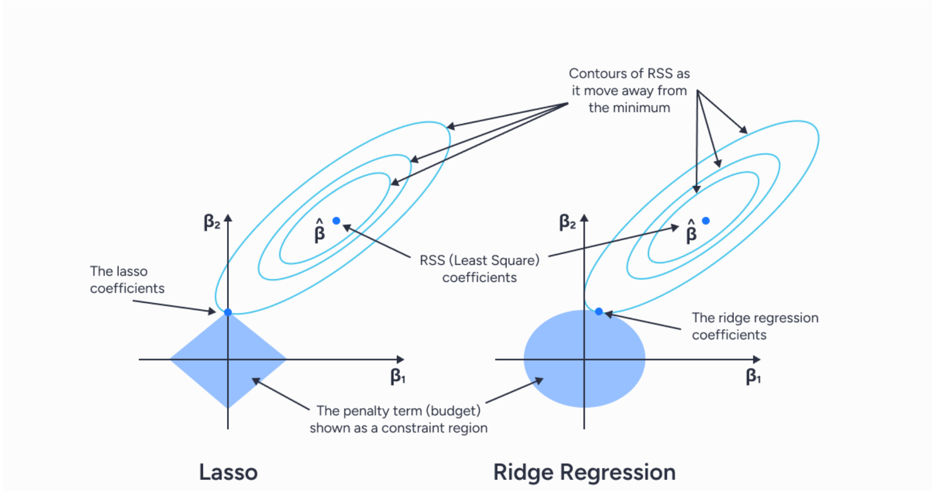
- Equivalently:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{M} (y_i - \mathbf{A}_i\mathbf{w})^2 + \lambda\|\mathbf{w}\|_1 \tag{4}$$

- This problem is non-differentiable $\rightarrow$ the training algorithm should be more complex than Ridge.

Methods: Lasso Regression (2)

- The regularization types lead to different domains for **w**.
- Lasso often produces **sparse** solutions, i.e., many components of **w** are zero.


Contours of RSS as it move away from the minimum
β_2
$\hat{\beta}$
The lasso coefficients
RSS (Least Square) coefficients
β_2
$\hat{\beta}$
The ridge regression coefficients
β_1
β_1
The penalty term (budget) shown as a constraint region
Lasso
Ridge Regression

An example of using OLS, Ridge and Lasso

- The training set **D** contains 67 observations on prostate cancer, each was represented

Table: Comparison of OLS, Ridge, and LASSO Regression Coefficients

w	**Ordinary Least Squares**	**Ridge**	**Lasso**
Intercept (0)	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	-0.141	-0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	-0.288	0.000	
gleason	-0.021	0.040	
pgg45	0.267	0.133	
Test RSS	**0.521**	**0.492**	**0.479**

Methods: Elastic Net (1) - Overview

- **Elastic Net Regression** is an extension of linear regression that incorporates both L1 (Lasso) and L2 (Ridge) regularization penalties into the loss function.
- This blending allows Elastic Net to handle situations where there are a large number of features, and some of them are highly correlated.
- It is a combination of the best of both worlds:

$$\textbf{Lasso (L1)} + \textbf{Ridge (L2)} \implies \textbf{Elastic Net}$$

Methods: Elastic Net (2) - The Loss Function

- The objective function combines both penalties L1(Lasso) and L2(Ridge):

$$\text{Loss} = \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{\text{1.RSS (Residual Sum of Squares)}} + \underbrace{\lambda \left(\sum_{j=1}^{p} |w_j| + \sum_{j=1}^{p} |w_j|^2 \right)}_{\text{Regularization Penalty}}$$

- **Where:**
  1. **RSS**:Residual Sum of Squares.

  λ: **Regularization parameter**, controls the *overall strength* of the penalty.

- **We solve for:**

$$\tilde{\mathbf{w}} = \arg\min_{\mathbf{w}} \left(\sum_{i=1}^{N} (y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j)^2 + \lambda \sum_{j=1}^{p} |w_j| + \lambda \sum_{j=1}^{p} |w_j|^2 \right)$$

Methods: Elastic Net (3) - The Challenge

- There is **no closed-form solution** for Elastic Net.
    - A "closed-form solution" is a direct formula, like in OLS or Ridge
- **Why?** The absolute value term $\|\mathbf{w}\|_1$ from the Lasso penalty is **non-differentiable** at zero.
- **The Solution:** We must use an **iterative optimization algorithm**. The most common algorithm used is called **Coordinate Descent8**

Methods: Elastic Net (4) - Coordinate Descent

- **The Core Idea:** Instead of updating all weights at once, the algorithm updates them **one by one**, cycling through all weights repeatedly until their values converge.
- **The Weight Update Formula:** For each weight w_j, the update rule (derived from the loss function) is:

$$w_j \leftarrow \frac{\overbrace{S\left(\sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i^{(j)}), \lambda\alpha\right)}^{\text{Lasso Part (Soft-Thresholding)}}}{\underbrace{\sum_{i=1}^{n} x_{ij}^2 + \lambda(1-\alpha)}_{\text{RSS Part + Ridge Part}}}$$

- $S(\rho, \gamma)$ is the *Soft-Thresholding* operator, which is what forces weights to be exactly zero.