Chapter 2

Least-squares and Linear Regression

Goal:
- The goal of the least squares (LS) method is to minimize MSE (or RMSE) between the given data and the parametric model.
- Define and analyze a model that is based on a linear relation between data and the outcome.
- Find the linear model parameters by LS.

2.1 Uni-variate Linear LS

2.1.1 Definition

The simplest sub-case is the (random) experiment that produces a set of M points (or measurements), $\{x_k, y_k\}_{k=1}^{M}$ [7]. The **linear model** is

$$y = f(x; w_0, w_1) = w_0 + w_1 x, \tag{2.1}$$

where w_0 and w_1 are the model weights (or parameters). The model outcomes (predictions) are

$$\hat{y}_k = f(x_k; w_0, w_1) = w_0 + w_1 x_k, \tag{2.2}$$

where $\hat{y}_k$ is the prediction outcome of x_k.
The performance **metric** is mean-square error (MSE) that is given by

$$\begin{aligned} J_{mse}(w_0, w_1) &= \frac{1}{M}\sum_{k=1}^{M}(y_k - \hat{y}_k)^2 \\ &= \frac{1}{M}\sum_{k=1}^{M} e_k^2 \end{aligned} \tag{2.3}$$

or root-MSE (RMSE)

$$J_{rmse}(w_0, w_1) = \sqrt{J_{mse}(w_0, w_1)}. \tag{2.4}$$

Note, sometimes MSE is termed as sum of squared errors (SSE).
For both of these metrics, the corresponding **loss** (or cost) function to minimize is

$$\begin{aligned} \mathcal{L}(w_0, w_1) &= \sum_{k=1}^{M}(y_k - \hat{y}_k)^2 \\ &= \sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)^2 \end{aligned} \tag{2.5}$$

since either root and/or constant multiplication does not change the desired minimum,

$$\begin{aligned} w_0, w_1 &= \arg\min_{w_0, w_1} J_{mse}(w_0, w_1) \\ &= \arg\min_{w_0, w_1} J_{rmse}(w_0, w_1) \\ &= \arg\min_{w_0, w_1} \mathcal{L}(w_0, w_1) \end{aligned} \tag{2.6}$$

Note that loss function and performance metrics does not have to be the same.

2.1.2 Minimization

This minimum is given by a solution of the set of equations,

$$\begin{cases} \dfrac{\partial}{\partial w_0}\mathcal{L}(w_0, w_1) = 0 \\ \dfrac{\partial}{\partial w_1}\mathcal{L}(w_0, w_1) = 0 \end{cases} \tag{2.7}$$

The resulting equations are

$$\begin{cases} 2\sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)\cdot(-1) = 0 \\ 2\sum_{k=1}^{M}(y_k - w_0 - w_1 x_k)\cdot(-x_k) = 0 \end{cases} \tag{2.8}$$

After some basic algebraic manipulations, the resulting set of equations is

$$\begin{cases} w_0 M & + w_1 \sum_{k=1}^{M} x_k = \sum_{k=1}^{M} y_k \\ w_0 \sum_{k=1}^{M} x_k & + w_1 \sum_{k=1}^{M} x_k^2 = \sum_{k=1}^{M} x_k y_k \end{cases} \tag{2.9}$$

This set of equations is termed *normal equation*.
The interesting and numerically stable form of the numerical solution is by usage of average estimation by mean,

$$E[\mathbf{z}] = \bar{\mathbf{z}} = \frac{1}{N}\sum_{k=1}^{N} z_k \tag{2.10}$$

$$\mathrm{Var}[\mathbf{z}] = \overline{\mathbf{z}^2} - \bar{\mathbf{z}}^2 \tag{2.11}$$

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = \overline{\mathbf{x}\mathbf{y}} - \bar{\mathbf{x}}\bar{\mathbf{y}} \tag{2.12}$$

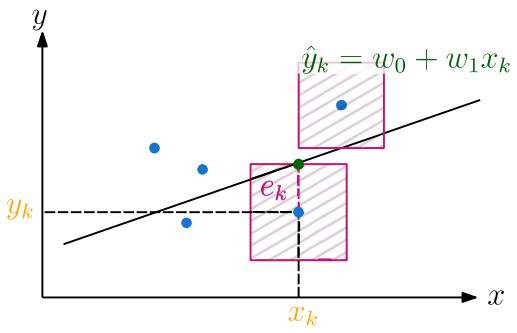

Figure 2.1: Linear regression visualization. The goal is to minimize the total area $\sum_k e_k^2$ of the rectangles.

The resulting prediction is

$$\hat{\mathbf{y}} = E[\mathbf{y}] + \frac{\mathrm{Cov}[\mathbf{x}, \mathbf{y}]}{\mathrm{Var}[\mathbf{x}]}(\mathbf{x} - E[\mathbf{x}]) \tag{2.13}$$

This is *probabilistic* result.
Notes:

- $\mathrm{Var}[\mathbf{x}] \neq 0$ requirement.
- $E[\mathbf{y}] = E[\mathbf{x}] = 0 \Rightarrow w_0 = 0$.

Concluding notes:

- The resulting model is also termed as linear regression, linear trend-line and linear prediction.
- The straightforward solution may result in ill-conditioned matrix. Reformulation of the solution can result in a better numerical stability, e.g. [7, Ch. 5, Question 5, pp. 260]. There are more accurate algorithms than just multiply my inverse matrix.
- For numerical stability, the variance of x_k samples is required to be non-zero (distinct x_k values).

2.2 Vector/Matrix Notation

2.2.1 Uni-variate model

To improve the mathematical representation, vector notation can be used. This time, the points $\{x_k, y_k\}_{k=1}^M$ are organized into vectors, with a few additional ones, as follows,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \mathbf{1}_M = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^M, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{2.14}$$

The resulting model notation is

$$\hat{\mathbf{y}} = f(\mathbf{X}; \mathbf{w}) = \mathbf{1}_M w_0 + \mathbf{x} w_1 = \mathbf{X}\mathbf{w}, \tag{2.15}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{1}_M & \mathbf{x} \end{bmatrix} \in \mathbb{R}^{M \times 2}$ and $\mathbf{w} = \begin{bmatrix} w_0 & w_1 \end{bmatrix}^T$.
The corresponding loss functions is

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \end{aligned} \tag{2.16}$$

and the corresponding optimal minimum (Eq. (2.6)) results from the solution of normal equation (matrix form)

$$\nabla_{\mathbf{w}} \mathcal{L}(\cdot) = -\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \tag{2.17}$$

and is given by

$$\begin{aligned} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) &= \mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \\ \mathbf{X}^T \mathbf{y} &= \left(\mathbf{X}^T \mathbf{X}\right) \mathbf{w} \\ \mathbf{w}_{opt} &= \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} \end{aligned} \tag{2.18}$$

2.2.2 Multivariate LS

For the multivariate N-dimensional formulation,

$$\mathbf{X} = \begin{bmatrix} \mathbf{1} & \mathbf{x}_1 & \cdots & \mathbf{x}_N \end{bmatrix} \in \mathbb{R}^{M \times (N+1)} \tag{2.19}$$

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_N \end{bmatrix}^T \in \mathbb{R}^{N+1} \tag{2.20}$$

All the LS discussion is the same independent from the number of variables.

Dataset

All the data rows in $(\mathbf{X}, \mathbf{y})$ are called dataset. The matrix $\mathbf{X}$ is assumed full-rank, i.e. columns are linearly independent.

Moore–Penrose inverse (pseudo-inverse)

Moore–Penrose inverse is the extension of an ordinary inverse matrix for none-rectangular matrices,

$$\mathbf{X}^+ = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T, \tag{2.21}$$

such that

$$\mathbf{X}^+ \mathbf{X} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{X} = \mathbf{I}$$

Note, the by-definition implementation of $\mathbf{X}^+$ may have numerical stability problems with $\left(\mathbf{X}^T \mathbf{X}\right)^{-1}$. All the modern programming languages have numerically-stable and efficient implementation of pseudo-inverse calculations.
The common numerical calculation is

$$\mathbf{w}_{opt} = \mathbf{X}^+ \mathbf{y} \tag{2.22}$$

Projection matrix

The model output is given by

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} = \mathbf{X} \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} \\ &= \mathbf{X}\mathbf{X}^+ \mathbf{y} = \mathbf{P}\mathbf{y} \end{aligned} \tag{2.23}$$

where

$$\mathbf{P} = \mathbf{X} \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \tag{2.24}$$

is a *projection* matrix, i.e. projection of $\mathbf{y}$ into a base derived from $\mathbf{X}$.
Important properties of the matrix $\mathbf{P}$:

- Symmetric $\mathbf{P} = \mathbf{P}^T$,
- Idempotent $\mathbf{P} = \mathbf{P}^2$,
- Orthogonality, $\mathbf{P} \perp (\mathbf{I} - \mathbf{P})$
  Proof. $\mathbf{P}(\mathbf{I} - \mathbf{P}) = \mathbf{P} - \mathbf{P}^2 = \mathbf{0}$.
- $\mathbf{I} - \mathbf{P}$ is also projection matrix.

Model error

The model error is

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{P}\mathbf{y} = (\mathbf{I} - \mathbf{P})\,\mathbf{y}, \tag{2.25}$$

such that $\mathcal{L}(\mathbf{w}) = \overline{\mathbf{e}^2}$.

Error and data orthogonality

$$\mathbf{e} \perp \mathbf{X} \Rightarrow \mathbf{X}^T\mathbf{e} = \mathbf{0} \tag{2.26}$$

Proof:

$$\begin{aligned} \mathbf{X}^T\mathbf{e} &= \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{y} = \mathbf{0} \end{aligned} \tag{2.27}$$

Error and prediction orthogonality

$$\mathbf{e} \perp \hat{\mathbf{y}} \Rightarrow \hat{\mathbf{y}}^T\mathbf{e} = \mathbf{e}^T\hat{\mathbf{y}} = 0 \tag{2.28}$$

Proof:

$$\begin{aligned} \hat{\mathbf{y}}^T\mathbf{e} &= \mathbf{y}^T\mathbf{P}\,(\mathbf{I} - \mathbf{P})\,\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}\mathbf{P}\mathbf{y} \\ &= \mathbf{y}^T\mathbf{P}\mathbf{y} - \mathbf{y}^T\mathbf{P}\mathbf{y} = 0 \end{aligned} \tag{2.29}$$

Average error

$$\begin{aligned} \bar{\mathbf{e}} &= \frac{1}{M}\sum_{k=1}^{M} e_k \\ &= \sum_{k=1}^{M} e_k = \mathbf{1}^T\mathbf{e} = 0 \end{aligned} \tag{2.30}$$

Proof.

$$\begin{aligned} &\mathbf{X}^T\underbrace{(\mathbf{y} - \mathbf{X}\mathbf{w})}_{\mathbf{e}} = 0 \\ &\begin{bmatrix} \mathbf{1}^T \\ \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}\mathbf{e} = 0 \\ &\mathbf{1}^T\mathbf{e} = 0 \end{aligned} \tag{2.31}$$

MSE

The reduced expression for the resulting minimal MSE is

$$mse_{min} = \sum_{k=1}^{M} y_k^2 - \sum_{j=0}^{N} w_j\mathbf{y}^T\mathbf{x}_j \tag{2.32}$$

Proof.

$$\begin{aligned} mse_{min} &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{y} - \hat{\mathbf{y}})^T\mathbf{e} \\ &= \mathbf{y}^T\mathbf{e} - \cancel{\hat{\mathbf{y}}^T\mathbf{e}} \\ &= \mathbf{y}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - \underbrace{\mathbf{y}^T\begin{bmatrix}\mathbf{1} & \mathbf{x}_1 & \cdots & \mathbf{x}_N\end{bmatrix}}_{\mathscr{R}^{1\times(N+1)}}\mathbf{w} \end{aligned} \tag{2.33}$$

2.3 Loss Function Minimization

Goal: Minimum of the loss function for a given model.

Closed-form solution A closed-form solution for $\mathbf{w}$ is a solution that is based on basic mathematical functions. For example, a "normal equation" is a solution for linear regression/classification.

Local-minimum gradient-based iterative algorithms This family of algorithms is applicable only for convex (preferably strictly convex) loss functions. For example, gradient descent (GD) and its modifications (e.g., stochastic GD) are used to evaluate NN parameters. Another example is the Newton-Raphson algorithm.

- Some advanced algorithms under this category also employ (require) second-order derivative $\frac{\partial^2}{\partial\mathbf{w}}\mathcal{L}$ for faster convergence.
- If either derivative is not available as a closed-form expression, it is evaluated numerically.

Global optimizers The goal of global optimizers is to find a global minimum of non-convex function. These algorithms may be gradient-free, first-derivative or second-derivative. The complexity of these algorithms is significantly higher than the local optimizer and can be prohibitive for more than a few hundred variables in $\boldsymbol{\theta}$.

2.3.1 Iterative Solution - Gradient descent (GD)

Goal: Find the minimum of the function:

- First-order derivative based.
- Local minimum.

Let's assume some function $y = f(x)$, with $x, y \in \mathscr{R}$, differentiable with $\frac{dy}{dx} = f'(x)$.

- $f'(x)$ is a slope of $f(x)$ at a point x.
- By the definition of the derivative, for some small ϵ,

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x)$$

- Given the sign of the derivative,

$$f(x-\epsilon) < f(x), \quad f'(x) > 0$$
$$f(x+\epsilon) < f(x), \quad f'(x) < 0$$

- For sufficiently small ϵ,

$$f(x - \epsilon\,\text{sign}(f'(x))) \leq f(x)$$

The idea of the algorithm is to reduce $f(x)$ by going in direction opposite sign of derivative, $f'(x)$.
Gradient descent (GD) - scalar function: For differentiable function $f(x)$, the iterative algorithm

$$x_{n+1} = x_n - \alpha f'(x_n) \tag{2.34}$$

converges to some local minimum of $f(x)$.
Required parameters are:

- Step-size $\alpha > 0$ is some positive constant or some function of n, α_n.
- x_0 is an initial guess.

Some of the most common stopping conditions are:

- Reaching the point of slow convergence, $|x_{n+1} - x_n| < \epsilon$.
- Limiting the number of iterations, $n \leq n_0$.

Gradient descent (GD) - vector function: For differentiable multivariate and multidimensional function $f(\mathbf{x}) : \mathscr{R}^N \to \mathscr{R}^N$, the iterative algorithm is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla_{\mathbf{x}} f'(\mathbf{x}_n). \tag{2.35}$$

Each dimension is iteratively reduced according to its derivative. Notes:

- Easy to implement.
- Requires analytical or numerical derivative.
- Non-trivial selection of the optimal value of α. In more general case, vector of n-dependent values may be desirable.
- Useful only for the function with single (global) minimum, such as MSE minimization.

GD for MMSE: Optimal values of $\mathbf{w}$ may be found by

$$\begin{aligned}\mathbf{w}_{n+1} &= \mathbf{w}_n - \alpha \nabla_{\mathbf{w}} \mathcal{L} \\ &= \mathbf{w}_n - \frac{\alpha}{M} \mathbf{X}^T (\mathbf{X}\mathbf{w}_n - \mathbf{y})\end{aligned} \tag{2.36}$$

2.4 Takeaways

2.1 This assignment focuses on understanding the interpretation of weights values. For the multivariate vector of the weights, $\mathbf{w}$:

- What is the meaning of the $+$ or $-$ sign of the each weight w_j?
- What is the influence of the magnitude (relative size) of weights w_j?

2.2 This assignment focuses on understanding the effects of sample size on the estimation accuracy of linear regression parameters using the Least Squares (LS) method. The task involves running simulations to generate linear data with varying numbers of data points, followed by fitting a linear regression model to this data and analyzing the resulting mean squared error (MSE).

- Perform linear regression using the LS method, and analyze the MSE across different sample sizes.
- Repeat each MSE evaluation for at least 30 times.
- Summarize the results in `boxplot` as in the plot.
- Does it seem reasonable the the MSE grows with an increase in M?

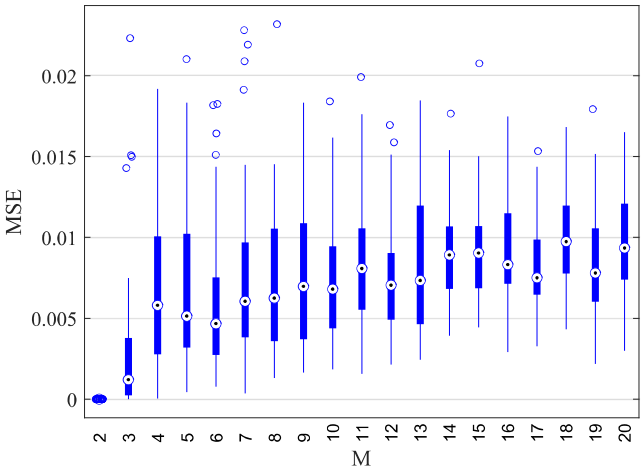

Figure 2.2: MSE as a function of number of data points, M.

These assignments will help you understand the practical implications of linear regression and the influence of data size on model performance.