Machine Learning in Medicine and Biology - Homework 2

Kamal

September 2025

Implement Gradient Descent for Linear Regression

(R **or** Python)

Rubric

- Use the design-matrix form $y = X\beta + \varepsilon$ with intercept β_0 and slopes $\beta_1, \ldots, \beta_d$.
- Define and work with the **loss** $\mathcal{L}$, the **sum of squared errors (SSE)**, and the **mean squared error (MSE)**. (2 pts)
- Learn about chain-rule, gradients, and Jacobians. (
- Implement batch gradient descent and verify against the normal equations. (4 pts)
- Compare different clustering methods in google colab. (2 pts)

Set-up and notation

We observe n examples (rows) and d features (columns). For each example $i \in \{1, \ldots, n\}$, the feature row is $x_i \in \mathbb{R}^d$ with target $y_i \in \mathbb{R}$. Define the *design matrix* $X \in \mathbb{R}^{n \times (d+1)}$, parameter vector $\beta \in \mathbb{R}^{d+1}$, and response vector $y \in \mathbb{R}^n$ as

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

Predictions for all n samples are $\hat{y} = X\beta$.

Notice, the matrix X will have all of the values collected from your samples simultaneously. These values do not change, they are fixed in space. The β values are the things that change in this formulation. This is inverted from how you think about equations normally, where x is the variable but the coefficients are constant. Now the variables are held constant by their sample values and the slopes (coefficients) associated with those variables x are changed to establish best fit.

Terminology (in words).

- **Loss** (denoted by the calligraphic letter $\mathcal{L}$): a single scalar that measures how well the coefficients β fit the data (X, y). This is calculated by subtracting the values y from the computed values $\hat{y}$. Squaring allows us to sum the magnitude difference, and the mean of that accounts for the number of sample points. Smaller $\mathcal{L}$ means a better fit.
- **Sum of Squared Errors (SSE):** total squared discrepancy,

$$\mathrm{SSE}(\beta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \|X\beta - y\|^2 .$$

- **Mean Squared Error (MSE):** average squared discrepancy,

$$\mathrm{MSE}(\beta) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \frac{1}{n} \|X\beta - y\|^2 .$$

- **Scaled loss (programming convenience):**

$$\mathcal{L}(\beta) = \frac{1}{2n} \|X\beta - y\|^2$$

  This has the *same* minimizer as MSE: multiplying by a positive constant does not change the arg min. When you implement this, you should see why this factor of 2 makes intuitive sense.

- **Gradient:** Let $\beta = (\beta_0, \beta_1, \ldots, \beta_d)^\top$. The **gradient of a scalar loss** $\mathcal{L}(\beta)$ *with respect to* β is the column of partial derivatives:

$$\boxed{\frac{\partial \mathcal{L}}{\partial \beta} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \beta_0} \\ \frac{\partial \mathcal{L}}{\partial \beta_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \beta_d} \end{bmatrix}} \qquad \text{(we will also write this as } \nabla_\beta \mathcal{L}(\beta)\text{)}.$$

- **Jacobian (def and example):** For a vector-valued function $u(\beta) \in \mathbb{R}^m$, the **Jacobian** is the matrix of all first-order partial derivatives: entry (i, j) is $\partial u_i / \partial \beta_j$.

  Concrete special case $n = 2, d = 1$*:*

$$X = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{21} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad u(\beta) = X\beta - y = \begin{bmatrix} \beta_0 + x_{11}\beta_1 - y_1 \\ \beta_0 + x_{21}\beta_1 - y_2 \end{bmatrix}.$$

  The Jacobian is

$$\frac{\partial u}{\partial \beta} = \begin{bmatrix} \frac{\partial u_1}{\partial \beta_0} & \frac{\partial u_1}{\partial \beta_1} \\ \frac{\partial u_2}{\partial \beta_0} & \frac{\partial u_2}{\partial \beta_1} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{21} \end{bmatrix} = X.$$

- **Chain rule (using the 2×2 template).** Let $u(\beta) = X\beta - y$ and $g(u) = \frac{1}{2n} u^\top u$ (a scalar). Then $\mathcal{L}(\beta) = g(u(\beta))$ and

$$\frac{\partial \mathcal{L}}{\partial \beta} = \left(\frac{\partial u}{\partial \beta}\right)^\top \frac{\partial g}{\partial u} = X^\top \left(\frac{1}{n} u\right) = \frac{1}{n} X^\top (X\beta - y).$$

  This generalizes directly to any n, d.

- $1/(2n)$ **vs.** $1/n$ **in the gradient (equivalent choices).** Let $e = X\beta - y$. Different but equivalent conventions:

$$\begin{aligned}
&\text{SSE:} \quad \text{SSE} = \|e\|^2 && \Rightarrow \quad \frac{\partial \text{SSE}}{\partial \beta} = 2\,X^\top e \\
&\text{Half-SSE:} \quad \tfrac{1}{2}\|e\|^2 && \Rightarrow \quad \frac{\partial}{\partial \beta}\Big(\tfrac{1}{2}\|e\|^2\Big) = X^\top e \\
&\text{MSE:} \quad \tfrac{1}{n}\|e\|^2 && \Rightarrow \quad \frac{\partial \text{MSE}}{\partial \beta} = \frac{2}{n}\,X^\top e \\
&\boxed{\text{Scaled loss:} \quad \tfrac{1}{2n}\|e\|^2} && \Rightarrow \quad \boxed{\frac{\partial \mathcal{L}}{\partial \beta} = \frac{1}{n}\,X^\top e}
\end{aligned}$$

  All have the *same minimizer*; we use $\frac{1}{2n}$ to cancel the 2 in the derivative.

- **Batch gradient descent: update, learning rate, and stopping.**

$$\boxed{\beta \leftarrow \beta - \alpha \cdot \frac{1}{n} X^\top (X\beta - y)} \quad \text{where} \;\; \alpha > 0 \textbf{ is the learning rate (step size).}$$

- **How to pick** α **in practice.** Try a small grid (e.g., $10^{-3}, 10^{-2}, 10^{-1}, 1$) and pick the largest value that *decreases* $\mathcal{L}$ each step. Two principled options often used:

  a) *Backtracking line search:* start with t=1; shrink $t \leftarrow \beta t$ (e.g. $\beta = 0.5$) until $\mathcal{L}(\beta - t\nabla\mathcal{L}) \le \mathcal{L}(\beta) - c\,t\|\nabla\mathcal{L}\|^2$ (e.g. $c = 0.5$); then take t as the step.

  b) *Least Squares):* for $\mathcal{L} = \frac{1}{2n}\|X\beta - y\|^2$, the Hessian (matrix of second-order partials) is $(1/n)X^\top X$. A safe constant step is $0 < \alpha \le 1/L$ with $L = \lambda_{\max}\big((1/n)X^\top X\big)$.

- **Stopping / "threshold near zero".** Because noise makes $\mathcal{L}$ rarely reach 0, stop when *two* conditions hold for some small tolerances, e.g.

$$\|\nabla\mathcal{L}(\beta)\|_2 \le \varepsilon_{\text{grad}} \quad \text{and} \quad \frac{|\mathcal{L}_t - \mathcal{L}_{t-1}|}{\max(1, \mathcal{L}_{t-1})} \le \varepsilon_{\text{rel}}.$$

  Typical classroom choices: $\varepsilon_{\text{grad}} = 10^{-6}$ and $\varepsilon_{\text{rel}} = 10^{-8}$.

  HOWEVER, for this homework, we are just taking a fixed number of steps.

Part A — Math (show your work)

A.1 Write SSE(β), MSE(β), and $\mathcal{L}(\beta) = \frac{1}{2n}\|X\beta - y\|^2$ in both summation and matrix forms.

A.2 Explain in one paragraph why scaling the loss by a positive constant does not change the minimizer. Think in terms of the chain-rule. You don't have to formally derive it.

A.3 Look closely at the Jacobian definition. $\dfrac{\partial \mathcal{L}}{\partial \beta} = \frac{1}{n}X^\top(X\beta - y)$ The rate of change of all of our β's is what?

A.4 Show that setting the gradient $\dfrac{\partial \mathcal{L}}{\partial \beta} = \frac{1}{n}X^\top(X\beta - y)$ to zero yields $X^\top X\,\beta = X^\top y$.

Part B — Implementation (R or Python)

Data. I have included the wine-quality dataset (WineQT.csv). Read it into your programming language of choice. Your y column is *quality*, the rest constitute your parameters X. Always add the intercept column of ones.

Steps.

B.1 Prepare X (add intercept); optionally standardize features.

B.2 Initialize β (zeros or small random).

B.3 For $T = 2000$ iterations: $\hat{y} = X\beta$; $e = \hat{y} - y$; $g = \frac{1}{n} X^\top e$; $\beta \leftarrow \beta - \alpha g$; record $\mathcal{L}_t = \frac{1}{2n} \|e\|^2$.

B.4 Plot $\mathcal{L}_t$ vs. iteration for several α; discuss convergence.

B.5 Compare β_{grad} to $\beta_{\text{normal}} = (X^\top X)^{-1} X^\top y$ after your descent loop has run. Print $\|\beta_{\text{grad}} - \beta_{\text{normal}}\|_2$.

$\beta_{\text{grad}} = \beta \leftarrow \beta - \alpha \frac{1}{n} X^\top (X\beta - y)$

$\beta_{\text{normal}} = (X^\top X)^{-1} X^\top y$

This is comparing the analytical least squares solutions of the gradient to the gradients derived by gradient descent.

Part C — Classification Methods Comparison Exercise

For this section, all you have to do is run the attached code. It is in python. There is an attached jupyter notebook in google colab to execute. Save the colab notebook to your google drive. You have to set up a huggingface account `https://huggingface.co/`, navigate into the dataset `https://huggingface.co/datasets/Falah/skin-cancer` and click "use this dataset" button on the right. Click on your user avatar on the top-right and click "Access Tokens". Create a new token with one check-mark for read access to gated repositories. Return to your colab session and add the key you just generated to your colab secrets by clicking the key icon on the left of the screen. Give it the name `huggingface`. Run all the colab cells and read the contents. If you are curious, to run the ViT activate the T4 TPU in colab or pay for the A100, it's not necessary to do so for the assignment.

There are 5 questions at the bottom of the colab notebook. Please answer them in your homework submission.

`https://colab.research.google.com/drive/1s-sPmFH2JBPgv_IKXjpwlZP3bMsknA8Q?usp=sharing`

Deliverables

- PDF with Part A derivations and explanations, answers for Part B.
- Code file (R or Python) and plots of the loss curve(s).
- A short paragraph on learning-rate behavior and agreement with the normal equations.