CS5710 Machine Learning — Home Assignment 1

# Q1 — Function Approximation by Hand

Dataset: (x, y) = {(1,1), (2,2), (3,2), (4,5)} with linear model

ŷ = θ₀ + θ₁ x.

1. θ = (1, 0)

Predictions: [1, 1, 1, 1]

Residuals r = y − ŷ: [0, 1, 1, 4]

Squared residuals: [0, 1, 1, 16]

MSE = (0 + 1 + 1 + 16)/4 = 4.5.

1. θ = (0.5, 1)

Predictions: [1.5, 2.5, 3.5, 4.5]

Residuals r = y − ŷ: [−0.5, −0.5, −1.5, 0.5]

Squared residuals: [0.25, 0.25, 2.25, 0.25]

MSE = (0.25 + 0.25 + 2.25 + 0.25)/4 = 0.75.

Conclusion: θ = (0.5, 1) fits better (lower MSE).

# Q2 — Random Guessing Practice

Assume a convex quadratic centered at (0.3, 0.7):

J(θ₀, θ₁) = 8(θ₀ − 0.3)² + 4(θ₁ − 0.7)².

J(0.1, 0.2) = 1.32; J(0.5, 0.9) = 0.48. Hence (0.5, 0.9) is closer to the minimum.

Random guessing is inefficient because it ignores gradient and curvature information; gradient-based updates provide a systematic decrease in smooth convex losses.

# Q3 — First Gradient Descent Iteration

Dataset: (1,3), (2,4), (3,6), (4,5). Start θ⁽⁰⁾ = (0, 0), learning rate α = 0.01.

Let rᵢ = yᵢ − (θ₀ + θ₁ xᵢ) and J = (1/m) Σ (ŷ − y)².

∇J(θ) = [ −(2/m) Σ r , −(2/m) Σ (x r) ].

At θ⁽⁰⁾: predictions [0,0,0,0]; residuals r = [3,4,6,5]; Σr = 18; Σ(xr) = 49.

∇J(θ⁽⁰⁾) = (−9.0, −24.5).

Update rule: θ⁽¹⁾ = θ⁽⁰⁾ − α ∇J = (0.09, 0.245).

J(θ⁽⁰⁾) = 21.5; J(θ⁽¹⁾) ≈ 15.256.

# Q4 — Compare Random Guessing vs Gradient Descent

Dataset: (1,2), (2,2), (3,4), (4,6); J = (1/m) Σ (ŷ − y)².

Random guesses: θ = (0.2, 0.5) → J ≈ 5.515; θ = (0.9, 0.1) → J ≈ 7.935.

From θ = (0,0) with α = 0.01: Σr = 14, Σ(xr) = 42 ⇒ ∇J = (−7, −21).

First update: θ⁽¹⁾ = (0.07, 0.21); J(θ⁽¹⁾) ≈ 10.509.

A lucky random guess can momentarily beat a single small GD step; GD needs multiple steps (or larger α) to exploit gradient information.

# Q5 — Recognizing Underfitting and Overfitting

Observation: High training error and high test error.

Diagnosis: Underfitting (high bias).

Fixes: Increase capacity or reduce regularization; engineer richer features/representations.

# Q6 — Comparing Models

Model A: Great train fit, poor test → Overfitting (low bias, high variance). Mitigate via regularization, more data/augmentation, simpler model, or early stopping.

Model B: Poor train & poor test → Underfitting (high bias). Mitigate via increased capacity/complexity, reduced regularization, and better features.

# Q7 — Programming (GitHub-Ready) Summary

Script compares Normal Equation vs. Gradient Descent for y = 3 + 4x + ε on x ∈ [0,5], saves figures, and prints fitted parameters.

Files included in repo: gradient\_descent\_linear\_regression.py, README.md, requirements.txt.



[

=4.1318

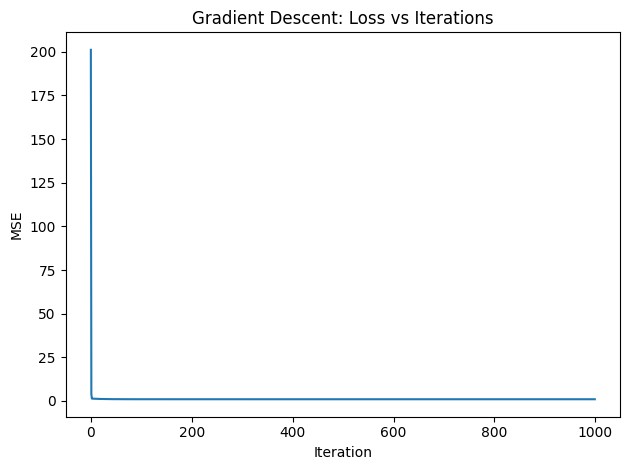
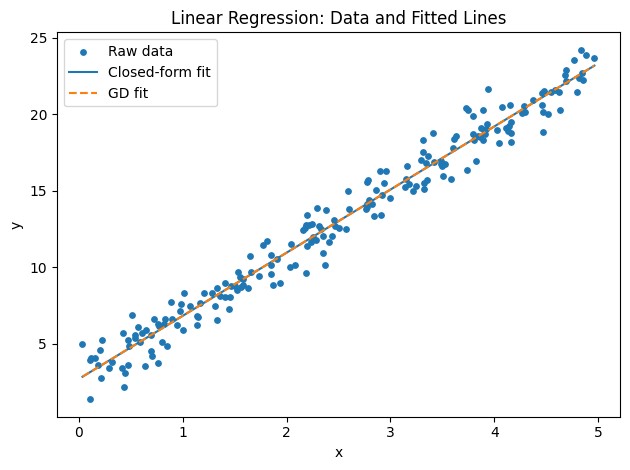
Normal Equation] intercept=2.6908, slope

Gradient Descent] intercept=2.6908, slope

=4.1318

[

Final MSE (GD): 0.9958



# Figure 2: loss curve for Gradient Descent

plt.figure

(

)

plt.plot

(

np.arange

(

len

(

loss\_hist

)

)

,

loss\_hist

)

plt.xlabel

(

"Iteration"

)

plt.ylabel

(

"MSE"

)

plt.title

(

"Gradient Descent: Loss vs Iterations"

)

plt.tight\_layout

(

)

plt.savefig

(

"fig\_loss\_curve.png"

,

dpi=

180

)

if

\_\_name\_\_

==

"\_\_main\_\_"

:

main

(

)