CS5710 — Home Assignment 1

Student Name: Manikanth Reddy Devarapalli

Course: Machine Learning, Fall 2025

Department of Computer Science & Cybersecurity, University of Central Missouri

1 — Function Approximation by Hand

Dataset: (x,y) = {(1,1), (2,2), (3,2), (4,5)}
Model: $\hat{y} = \theta1 * x + \theta2$
Mean Squared Error (MSE): $J(\theta) = (1/N) * \text{sum } (y - \hat{y})^2$, N = 4

Model A: $\theta = (1, 0) \rightarrow \hat{y} = x$

Predictions and residuals:
x=1, y=1 → $\hat{y}$=1; residual r = y - $\hat{y}$ = 0; r^2 = 0
x=2, y=2 → $\hat{y}$=2; residual r = y - $\hat{y}$ = 0; r^2 = 0
x=3, y=2 → $\hat{y}$=3; residual r = y - $\hat{y}$ = -1; r^2 = 1
x=4, y=5 → $\hat{y}$=4; residual r = y - $\hat{y}$ = 1; r^2 = 1
Sum of squared residuals = 2; MSE = 0.5 = 2.0/4

Model B: $\theta = (0.5, 1) \rightarrow \hat{y} = 0.5 * x + 1$

Predictions and residuals:
x=1, y=1 → $\hat{y}$=1.5; residual r = y - $\hat{y}$ = -0.5; r^2 = 0.25
x=2, y=2 → $\hat{y}$=2; residual r = y - $\hat{y}$ = 0; r^2 = 0
x=3, y=2 → $\hat{y}$=2.5; residual r = y - $\hat{y}$ = -0.5; r^2 = 0.25
x=4, y=5 → $\hat{y}$=3; residual r = y - $\hat{y}$ = 2; r^2 = 4
Sum of squared residuals = 4.5; MSE = 1.125 = 4.5/4

Conclusion: Model A has lower MSE and fits the dataset better.

2 — Random Guessing Practice

Cost function: $J(\theta1, \theta2) = 8*(\theta1 - 0.3)^2 + 4*(\theta2 - 0.7)^2$
Guess (0.1, 0.2):
$(\theta1 - 0.3) = 0.1 - 0.3 = -0.2 \rightarrow (-0.2)^2 = 0.04 \rightarrow 8 * 0.04 = 0.32$
$(\theta2 - 0.7) = 0.2 - 0.7 = -0.5 \rightarrow (-0.5)^2 = 0.25 \rightarrow 4 * 0.25 = 1.0$
J(0.1,0.2) = 0.32 + 1.0 = 1.32

Guess (0.5, 0.9):
 $(\theta1 - 0.3) = 0.5 - 0.3 = 0.2 \rightarrow 0.2^2 = 0.04 \rightarrow 8 * 0.04 = 0.32$
 $(\theta2 - 0.7) = 0.9 - 0.7 = 0.2 \rightarrow 0.2^2 = 0.04 \rightarrow 4 * 0.04 = 0.16$
 J(0.5,0.9) = 0.32 + 0.16 = 0.48

Which guess is closer to the minimum (0.3, 0.7)?

- J(0.1,0.2) = 1.32; J(0.5,0.9) = 0.48. The smaller value (0.48) corresponds to (0.5,0.9), so that guess is closer.

Why random guessing is inefficient?

- Random guessing wastes computation and time because it does not follow information about the gradient or the structure of the cost surface.
- Instead, optimization methods (e.g., gradient descent) use local derivative information to move systematically toward minima and reach solutions much faster.

3 — First Gradient Descent Iteration

Dataset: (1,3), (2,4), (3,6), (4,5)
Model: $\hat{y} = \theta1 * x + \theta2$; Start: θ^(0) = (0,0); learning rate $\alpha = 0.01$

Step 1: Predictions at θ^(0) = (0,0)
 For all x, $\hat{y}$ = 0. So predictions vector = [0, 0, 0, 0]
Step 2: Residuals r = y - ŷ = y
 x=1, y=3, $\hat{y}$=0 → residual r = 3.0
 x=2, y=4, $\hat{y}$=0 → residual r = 4.0
 x=3, y=6, $\hat{y}$=0 → residual r = 6.0
 x=4, y=5, $\hat{y}$=0 → residual r = 5.0
 Sum r = 18.0; Sum x * r = 49.0

Step 3: Compute gradient (for MSE J = (1/N) sum (y - ŷ)^2):
 The gradient vector (∂J/∂θ1, ∂J/∂θ2) = -(2/N) * [sum(x * r), sum(r)] where r = y - $\hat{y}$
 Therefore gradient at θ^(0) = (-2/4 * 49.0, -2/4 * 18.0) = (-24.5, -9)

Step 4: Update θ^(1) = θ^(0) - α * gradient
 θ^(1) = (0,0) - 0.01 * (-24.5, -9) = (0.245, 0.09)

Step 5: Compute J(θ^(0)) for reference
 J(θ^(0)) = (1/4) * sum(y^2) = 21.5

Now continue from θ^(1): compute predictions, residuals, gradient, update to θ^(2)
Predictions at θ^(1):
 x=1, $\hat{y}$=0.335, y=3, residual r = y - $\hat{y}$ = 2.665
 x=2, $\hat{y}$=0.58, y=4, residual r = y - $\hat{y}$ = 3.42
 x=3, $\hat{y}$=0.825, y=6, residual r = y - $\hat{y}$ = 5.175

x=4, $\hat{y}$=1.07, y=5, residual r = y - $\hat{y}$ = 3.93
Sum r (at θ^(1)) = 15.19; Sum x*r (at θ^(1)) = 40.75
Gradient at θ^(1) = (-20.375, -7.595)
θ^(2) = θ^(1) - α * gradient = (0.44875, 0.16595)

Compare training loss:
J(θ^(0)) = 21.5
J(θ^(1)) = 15.256
J(θ^(2)) = 10.9223

4 — Compare Random Guessing vs Gradient Descent

Dataset: (1,2), (2,2), (3,4), (4,6)
Two random guesses and one-step gradient descent from θ=(0,0) with α=0.01

Random guess A: θ = (0.2, 0.5)
Compute predictions ŷ_i = 0.2 * x_i + 0.5, residuals and squared residuals; MSE = 8.35
Random guess B: θ = (0.9, 0.1)
MSE = 1.935

Gradient Descent one-step from θ=(0,0):
radient computed from data: -21, -7
θ after one update: (0.21, 0.07)
MSE at this θ = 10.5091

Conclusion: Compare the numeric MSEs above. Gradient descent should produce a θ with lower MSE than most random guesses because it moves in the direction of steepest descent reducing the cost.

5 — Recognizing Underfitting and Overfitting

If training error is very high and test error is also very high → Underfitting.
Why: The model lacks capacity (too simple) to capture patterns in the training data. High training error shows the model can't fit the training set.
Fixes (pick two):
1) Increase model complexity (use more features, polynomial terms, or a more flexible model).
2) Reduce regularization (if using strong regularization like large λ in ridge), or train longer with better hyperparameters.
Other options: perform feature engineering, add more relevant features, or use a model with lower bias.

6 — Comparing Models

Given: Model A fits training data almost perfectly but performs poorly on unseen data.
Model B does not fit training data well and performs poorly on unseen data.

Model A: Overfitting — low bias, high variance. The model memorizes noise/peculiarities in the training set and fails to generalize.

Remedies: add regularization (L2/L1), reduce model complexity, get more training data, or use cross-validation to tune hyperparameters.

Model B: Underfitting — high bias, low variance. The model is too simple to capture true relationships.

Remedies: increase model complexity (e.g., more features, polynomial features), reduce regularization, or try a different model family.

Bias-Variance tradeoff summary: Increasing complexity reduces bias but raises variance; decreasing complexity increases bias but lowers variance. Choose model complexity to balance both.