Machine Learning

Himaja Arcbati
700772489

① Given

Dataset : $(X,Y) = \{(1,1),(2,2),(3,2),(4,5)\}$

Model form : $\hat{y} = \theta_1 x + \theta_2$

MSE : $J(\theta) = 1/N \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$, with $N = 4$

i) Model $\theta = (1,0) \longrightarrow \hat{y} = 1 \cdot x + 0 = x$

x	Y	$\hat{y}$	residual $e = y - \hat{y}$	e^2
1	1	1	1-1=0	0
2	2	2	2-2=0	0
3	2	3	2-3=-1	1
4	5	4	5-4=1	1

Sum of Squared residuals = 0+0+1+1 = 2

MSE = 2/4 = 0.5

ii) Model $\theta = (0.5,1) \longrightarrow \hat{y} = 0.5x + 1$

x	Y	$\hat{y}$	residual $e = y - \hat{y}$	e^2
1	1	0.5+1+1 = 1.5	1-1.5 = -0.5	0.25
2	2	0.5.2+1 = 2	2-2 = 0	0
3	2	0.5.3+1 = 2.5	2-2.5 = -0.5	0.25
4	5	0.5+4+1 = 3	5-3 = 2	4

Sum of Squared residuals = 0.25+0+0.25+4 = 4.5

MSE = 4.5/4 = 1.125

iii) Compared MSEs : 0.5 vs 1.125
Lower is better ⇒ $\theta = (1,0)$ fits the data better.

② Cost function :

i) $J(\theta_1, \theta_2) = 8 + 4$

Assume, $J(\theta_1, \theta_2) = 8 + 4[(\theta_1 - 0.3)^2 + (\theta_2 - 0.7)^2]$

At $(0.1, 0.2)$:

$J = 8 + 4[(0.1 - 0.3)^2 + (0.2 - 0.7)^2] = 8 + 4[(0.04 + 0.25)]$

$= 8 + 4(0.29) = 9.16.$

At $(0.5, 0.9)$:

$J = 8 + 4[(0.5 - 0.3)^2 + (0.9 - 0.7)^2]$

$= 8 + 4[(0.04 + 0.04)]$

$= 8 + 0.32 = 8.32$

ii) closer guess : $(0.5, 0.9)$

iii) Random guessing wastes Computation because it does not use feedback from the Cost function to improve guesses. Gradient descent, by Contrast, follows the slope of the error Surface to iteratively approach the minimum. As datasets grow more Complex, random guessing becomes impractical.

③ First Gradient Descent Iteration

Dataset : (1,3), (2,4), (3,6), (4,5)

Start : $\theta = (0,0)$, learning $\alpha = 0.01$

$\hat{y} = 0$ for all point

Step 2 : Residuals ($r = \hat{y} - y$) : **

$r = [-3, -4, -6, -5]$

$\Sigma r = -18$, $\Sigma (x \cdot r) = -3(1) - 4(2) - 6(3) - 5(4) = -51$

Step 3 :

Gradient (for MSE) :

$$\nabla J = 2/m \begin{bmatrix} \Sigma r \\ \Sigma (x \cdot r) \end{bmatrix}, \quad m = 4$$

$$\nabla J = 2/4 [-18, -51] = [-9, -25.5]$$

Step 4 :

Update :

$$\theta = (0,0) - 0.01 [-9, -25.5] = [0.09, 0.255]$$

Step 5 : Cost Values :

- $J(0,0) = 1/4 (9 + 16 + 36 + 25) = 21.5$
- $J(0.09, 0.255)$
- $(1)(0.09) + 0.255 = 0.345$, error $= 2.655$ → squared $= 7.05$
- 0.435, error $= -3.565$ → squared $= 12.5$
- 0.525, error $= -5.475$ → squared $= 29.96$
- 0.615, error $= -4.385$ → squared $= 19.22$
- Total $= 68.93$ → divide by $4 = 17.23$

Cost went down : $21.5 \rightarrow 17.23$

④ Compare Random Guessing Vs Gradient Descent

Dataset : (1,2), (2,2), (3,4), (4,6)

Try guesses :

(0.2, 0.5) : prediction = [0.7, 0.9, 1.1, 1.3]

Errors = [-1.3, -1.1, -2.9, -4.7]

Squares = [1.69 + 1.21 + 8.41 + 22.09] = 33.4 → /4 = 8.35

(0.9, 0.1) : predictions = [1.0, 1.9, 2.8, 3.7]

Errors = [-1, -0.1, -1.2, -2.3]

Squares = 1 + 0.01 + 1.44 + 5.29 = 7.74 → /4 = 1.935

Gradient descent at step 1

Predictions = [0,0,0,0] . Residuals = [-2, -2, -4, -6]

$\Sigma r = -14$, $\Sigma xr = -40$

Gradient = (2/4)[-14, -40] = [-7, 20]

update → $\theta = (0.07, 0.2)$

Cost = Compute : predictions = [0.27, 0.34, 0.41, 0.48]

Errors = [-1.73, -1.66, -3.59, -5.52]

Squared Sum = 50.1 → /4 = 12.52

Random guess (0.9, 0.1) did better for this case.

⑤ Recognizing underfitting vs overfitting

Training error high, Test error high → underfi

Happens because model is too simple to capture patterns (high bias)

Fixes:

1. Use a more complex model (add features, non linear terms)
2. Train longer or reduce regularization.

⑥ Comparing Models:

Model A - low train error, high test e
overfitting (high Variance)

Model B - high train error, high test error
underfitting (high bias).

Bias Variance tradeoff

Overfitting = low bias, high Variance

Underfitting = high bias, low Variance

Improvements:

for A (overfitting): add regularization, get mor
data, prune Complexity

for B (underfitting): Use more features,
stronger model, reduce regularization.