9.3 SOLUTION - REGULARIZATION

12 September 2025 02:35 AM Avinash Yadav

- **Applied to Model Weights:**
  - Regularization is applied to the weights of the model to penalize large values and encourage smaller, more generalizable weights.

- **Introduced via Loss Function or Optimizer:**
  - Adds a penalty term $\lambda\sum w_i^2$ to the loss function in L2 regularization.

$$Loss_{reg} = Loss_{original} + \lambda\Sigma w_i^2$$

  - In weight decay, directly modifies the gradient update rule to include λw_i, effectively shrinking weights during training.

$$w \leftarrow w - \eta(\Delta Loss + \lambda w)$$

- **Penalizes Large Weights:**
  - Encourages the network to distribute learning across multiple parameters, avoiding reliance on a few large weights.

- **Reduces Overfitting:**
  - Helps the model generalize better to unseen data by discouraging overly complex representations.

- **Controlled by a Hyperparameter:**
  - A regularization coefficient (λ, often set via weight_decay in optimizers controls the strength of the penalty. Larger values lead to stronger regularization.

- **No Effect on Bias Terms:**
  - Regularization is typically applied only to weights, not biases, as biases don't directly affect model complexity.

- **Active During Training:**
  - Regularization affects weight updates only during training. It does not explicitly influence the model during inference.

* APPLYING REGULARIZATION IN OPTIMISATION STEP THROUGH WEIGHT DECAY :-

During gradient descent, we can directly add loss to the gradient.

```python
optimizer = optim.SGD(
    model.parameters(),
    lr=0.1,
    weight_decay=1e-4
)
```

Python

λ decides the amount of regularization we want to apply