Regularization and cross-validation

How to control the Underfit and Overfit tradeoffs to find the perfect model?

Ans: Regularization in the loss function → adds a term $\sum_{j=1}^{d} w^{2}_{j}$ → making weight small → for insignificant features

How does Regularization make weights small for insignificant features?

Ans: With the optimization algorithm, → minimizes the values of w_j

$$TotalLoss \ = \ \min_{w_j} \ Lossfunction \ + \ \lambda \sum_{j=1}^{d} w^{2}_{j}$$

How to control Regularization?

Ans: By using regularization parameter λ:

- since too much regularization → makes the model underfit the data
- Too little regularization → makes the model overfit.

Thus ⇒ λ becomes hyperparameter → on tuning gives the overfit-underfit tradeoff

Is squaring of weights the only way for Regularization?

Ans: No, Regularization is majorly of three types:

A. **L1 / Lasso Regularization:** Uses the term $\sum_{j=1}^{d} |w_j|$ → has $\frac{d|w_j|}{w_j} = 0$, when $w_j = 0$ → making the **weight vector sparse.**

B. **L2/ Ridge Regularization**: Uses the term $\sum_{j=1}^{d} w_j^{2}$ → have close to 0 values → for insignificant features.

C. **ElasticNet Regularization:** Combination of both L1 and L2 Regularization → with λ_1 and λ_2 as regularization parameters respectively.

$$TotalLoss \;=\; \min_{w_j} \; Lossfunction \;+\; \lambda_1 \sum_{j=1}^{d} w^2_{\;j} + \lambda_2 \sum_{j=1}^{d} |w_j|$$

How is data split?

Ans: Training, Validation, and testing dataset.

Why split data into a Validation dataset?

Ans: hyperparameter tuning → done only on Validation data → test data solely used for evaluating the model on unseen data.

What are the steps for a model building?

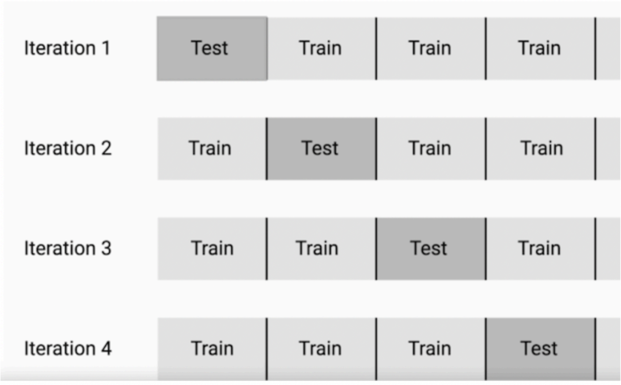
Ans: The steps are :

- Train model → with some regularization parameter λ → on training data
- Measure the model performance → with different values of hyperparameters → on the Validation dataset
- Pick the hyperparameters of the best-performing model
- Measure the performance of the Best-performing model on Test data.

If the data is too small to have a validation dataset, what to do then?

Ans: use **k-Fold CV algorithm** since**:**

- splits data into k smaller sets
- for each iteration, the model trained on k-1 folds
- validated on 1 fold
- performance is averaged over all the iterations.



Iteration 1	Test	Train	Train	Train
Iteration 2	Train	Test	Train	Train
Iteration 3	Train	Train	Test	Train
Iteration 4	Train	Train	Train	Test

Note: Though k-fold is a computationally expensive algorithm, it is useful when the dataset is small.