Foundations of Machine Learning

Subject Code: CS5590

Assignment-4

Submitted by:

Antala Aviraj
Roll No: CS24MTECH14011

Department of Computer Science and Engineering
IIT-Hyderabad

Date: November 28, 2024

November 28, 2024

Answer 1) VC-Dimension of H:

The VC-dimension of the hypothesis space H is **2**. let's see how?

Hypothesis Space H

The hypothesis space H consists of intervals defined by two parameters $\{p, q\}$, where a point $x \in R$ is classified as **1** if $p < x < q$, and **0** otherwise. The VC-dimension is the maximum number of points that can be *shattered* by H, meaning all possible binary labelings can be realized for those points.

1 Point

- A single point x_1 can be classified as either 1 or 0 by choosing appropriate values of p and q:

- **Label as 1**: Choose $p < x_1 < q$.
- **Label as 0**: Choose $x_1 < p$ or $q < x_1$.

So, H can shatter 1 point.

2 Points

- Consider two points $x_1 < x_2$. H can realize all possible binary labelings:

- **Label as 00**: Choose $q < x_1$.
- **Label as 01**: Choose $p < x_1 < q < x_2$.
- **Label as 10**: Choose $p < x_2 < q < x_1$.
- **Label as 11**: Choose $x_1 < p < x_2 < q$.

Since all four possible labelings can be realized, H can shatter 2 points.

3 Points

- Consider three points $x_1 < x_2 < x_3$. H cannot realize all $2^3 = 8$ possible labelings. For example:

- **Labeling 101** ($x_1 = 1, x_2 = 0, x_3 = 1$): This requires $p < x_1 < q < x_2 < p < x_3$, which is impossible.
- Similarly, other discontinuous patterns like 010 or 101 cannot be achieved because intervals defined by $\{p, q\}$ only allow contiguous sequences of 1s.

So, H cannot shatter 3 points.

Answer 2) Regularizer:

Suppose Gaussian noise $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$ is added independently to each feature x_k. This means the noisy version of each input x_k becomes:

$$\tilde{x}_k = x_k + \epsilon_k$$

where ϵ_k has a mean of 0 and a variance of σ^2.

With noisy inputs $\tilde{x}_i = [\tilde{x}_{i,1}, \tilde{x}_{i,2}, \ldots, \tilde{x}_{i,D}]$, our model's prediction becomes:

$$y(\tilde{x}_i, w) = w_0 + \sum_{k=1}^{D} w_k \tilde{x}_{i,k}.$$

Substituting $\tilde{x}_{i,k} = x_{i,k} + \epsilon_{i,k}$, we expand the prediction as:

$$y(\tilde{x}_i, w) = w_0 + \sum_{k=1}^{D} w_k (x_{i,k} + \epsilon_{i,k}).$$

we can further simplified as:

$$y(\tilde{x}_i, w) = \left(w_0 + \sum_{k=1}^{D} w_k x_{i,k} \right) + \sum_{k=1}^{D} w_k \epsilon_{i,k}.$$

we can observe form the equation that:

- The first part, $w_0 + \sum_{k=1}^{D} w_k x_{i,k}$, is the prediction $y(x_i, w)$ using the original noise-free input x_i.
- The second part, $\sum_{k=1}^{D} w_k \epsilon_{i,k}$, is an additional noise term introduced by the Gaussian noise on each $x_{i,k}$.

So we can rewrite the prediction with noisy inputs as:

$$y(\tilde{x}_i, w) = y(x_i, w) + \sum_{k=1}^{D} w_k \epsilon_{i,k}.$$

Now we need to find Expected Value and Variance of the Noise Term:

The additional term $\sum_{k=1}^{D} w_k \epsilon_{i,k}$ represents the cumulative effect of noise on each feature, weighted by w_k. Since each $\epsilon_{i,k}$ is drawn independently from $\mathcal{N}(0, \sigma^2)$, let's find its expectation and variance:

1. **Expectation:**

$$E\left[\sum_{k=1}^{D} w_k \epsilon_{i,k}\right] = \sum_{k=1}^{D} w_k E[\epsilon_{i,k}] = \sum_{k=1}^{D} w_k \cdot 0 = 0.$$

So, the expected value of the noise term is zero, meaning the noise does not systematically shift the predictions up or down.

2. **Variance:**

$$\mathrm{Var}\left(\sum_{k=1}^{D} w_k \epsilon_{i,k}\right) = \sum_{k=1}^{D} w_k^2 \cdot \mathrm{Var}(\epsilon_{i,k}) = \sum_{k=1}^{D} w_k^2 \cdot \sigma^2 = \sigma^2 \sum_{k=1}^{D} w_k^2.$$

This variance term reflects the aggregate noise effect on the prediction due to the weights and the noise variance σ^2.

Now we need to find Expected Squared Error with Noisy Data:

Now, let's calculate the expected squared error between the prediction with noisy inputs and the true target t_i:

$$E_\epsilon\left[(y(\tilde{x}_i, w) - t_i)^2\right].$$

Substitute $y(\tilde{x}_i, w) = y(x_i, w) + \sum_{k=1}^{D} w_k \epsilon_{i,k}$:

$$E_\epsilon\left[\left(y(x_i, w) + \sum_{k=1}^{D} w_k \epsilon_{i,k} - t_i\right)^2\right].$$

Expanding the square:

$$= E_\epsilon\left[(y(x_i, w) - t_i)^2 + 2\left(y(x_i, w) - t_i\right)\sum_{k=1}^{D} w_k \epsilon_{i,k} + \left(\sum_{k=1}^{D} w_k \epsilon_{i,k}\right)^2\right].$$

Since the expectation of the noise term $\sum_{k=1}^{D} w_k \epsilon_{i,k}$ is zero, the middle term become 0:

$$= (y(x_i, w) - t_i)^2 + E_\epsilon\left[\left(\sum_{k=1}^{D} w_k \epsilon_{i,k}\right)^2\right].$$

We know the variance of $\sum_{k=1}^{D} w_k \epsilon_{i,k}$ is $\sigma^2 \sum_{k=1}^{D} w_k^2$, so:

$$E_\epsilon\left[(y(\tilde{x}_i, w) - t_i)^2\right] = (y(x_i, w) - t_i)^2 + \sigma^2 \sum_{k=1}^{D} w_k^2.$$

Expected Sum-of-Squares Error with Noise Averaged Out:

The expected total sum-of-squares error over all samples becomes:

$$E_\epsilon\left[E(w)\right] = \frac{1}{2N} \sum_{i=1}^{N} \left((y(x_i, w) - t_i)^2 + \sigma^2 \sum_{k=1}^{D} w_k^2\right).$$

Separating the terms, we get:

$$E_\epsilon\left[E(w)\right] = \frac{1}{2N}\sum_{i=1}^{N}\left(y(x_i,w)-t_i\right)^2 + \frac{\sigma^2}{2}\sum_{k=1}^{D} w_k^2.$$

The first term is the regular MSE for the original data, and the second term is an added regularization term proportional to $\sum_{k=1}^{D} w_k^2$, which is $L2$ regularization.

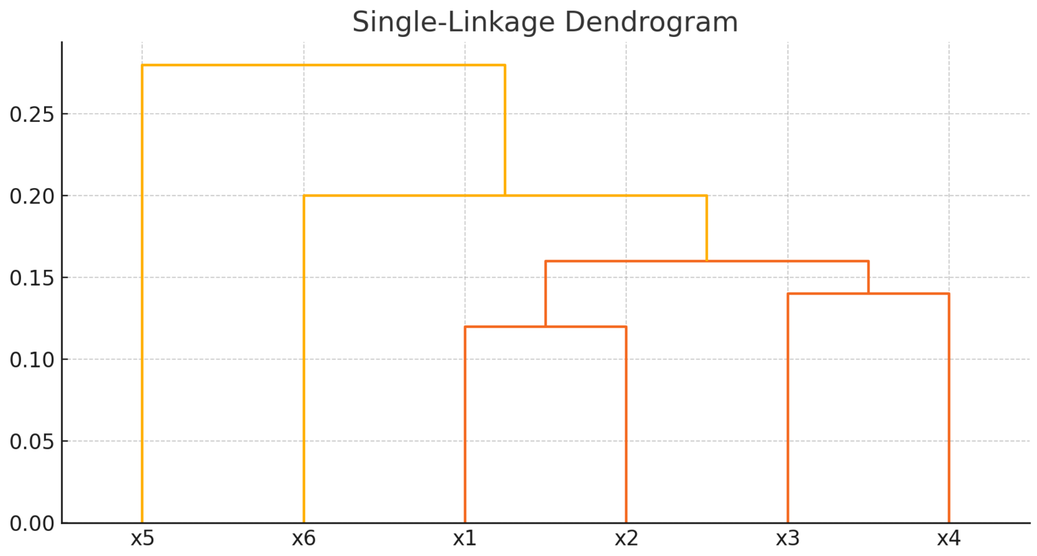
So we can say that, minimizing the sum-of-squares error averaged over noisy data is equivalent to minimizing the standard sum-of-squares error with an $L2$ weight-decay regularization term, where the regularization strength λ equals the noise variance σ^2.

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}\left(y(x_i,w)-t_i\right)^2 + \frac{\sigma^2}{2}\sum_{k=1}^{D} w_k^2.$$

Answer 3) Hierarchical Clustering:

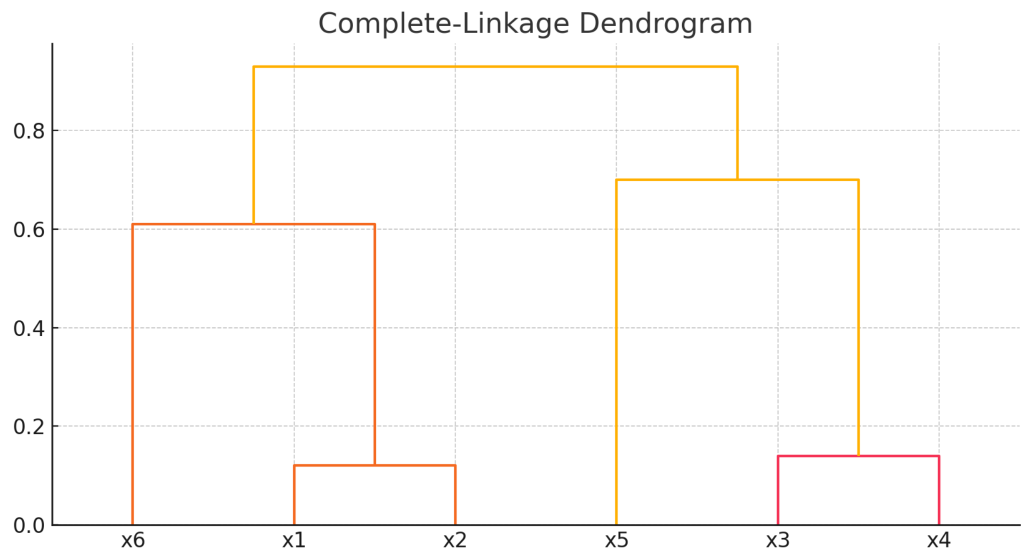
a] Single-Linkage Dendrogram

The single-linkage method clusters points based on the minimum distance between clusters. Below is the dendrogram for the single-linkage clustering:(I generated this graph using the python code)



b] Complete-Linkage Dendrogram

The complete-linkage method clusters points based on the maximum distance between clusters. Below is the dendrogram for the complete-linkage clustering:(I generated this graph using the python code)



c] Adjusted Distance Matrix

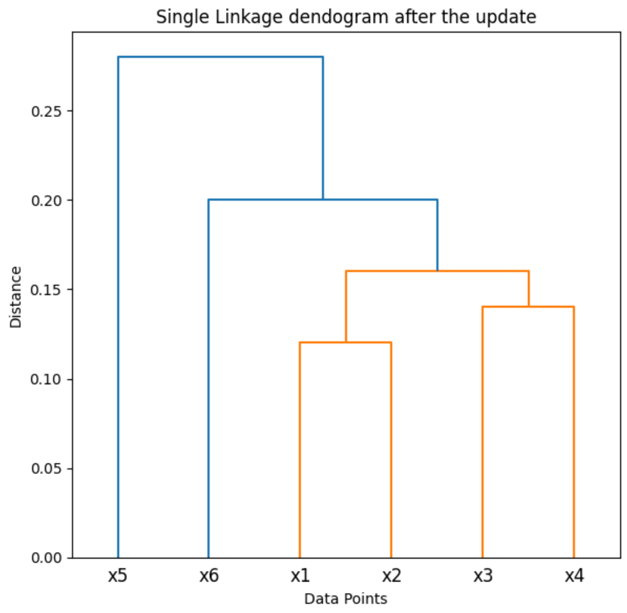
To make the results of single-link and complete-link clustering identical, we modify the distance matrix as follows:

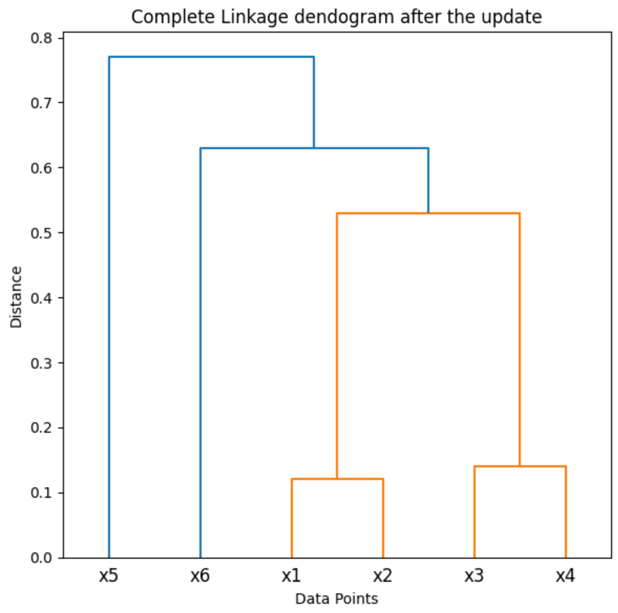
- Change the distance between points x_1 and x_4 ($\text{dist}(x_1, x_4)$) to 0.53. This ensures that the cluster $\{x_1, x_2\}$ is grouped with $\{x_3, x_4\}$ at a smaller distance than $\text{dist}(x_1, x_6) = 0.61$.
- Change the distance between points x_3 and x_6 ($\text{dist}(x_3, x_6)$) to 0.63. This ensures that the cluster $\{x_1, x_2, x_3, x_4\}$ is grouped with $\{x_6\}$ at the final step.

Updated Distance Matrix

$$\begin{bmatrix} 0 & 0.12 & 0.51 & 0.53 & 0.28 & 0.34 \\ 0.12 & 0 & 0.25 & 0.16 & 0.77 & 0.61 \\ 0.51 & 0.25 & 0 & 0.14 & 0.70 & 0.63 \\ 0.53 & 0.16 & 0.14 & 0 & 0.45 & 0.20 \\ 0.28 & 0.77 & 0.70 & 0.45 & 0 & 0.67 \\ 0.34 & 0.61 & 0.63 & 0.20 & 0.67 & 0 \end{bmatrix}$$

Below are the dendrograms for single-linkage and complete-linkage clustering after the matrix modification:





As shown, both dendrograms are now identical.

Answer 4) Principal Component Analysis:

a] Covariance Matrix

The covariance matrix C is calculated as:

$$C = E\left[(x - E[x])(x - E[x])^T\right].$$

For $i \neq j$ (off-diagonal elements):

$$C_{ij} = 0.$$

For $i = j$ (diagonal elements):

$$C_{ii} = \frac{\text{Var}(a)(M+1)}{6} + \frac{\mu^2(M^2-1)}{12} - \mu^2.$$

Thus, the covariance matrix is diagonal:

$$C_{ij} = \begin{cases} \frac{\text{Var}(a)(M+1)}{6} + \frac{\mu^2(M^2-1)}{12} - \mu^2, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

b] Eigenvalues and Eigenvectors

1. The vector $v = (1, 1, \ldots, 1)^T$ is an eigenvector with eigenvalue:

$$\lambda = \frac{\text{Var}(a)(M+1)}{6} + \frac{\mu^2(M^2-1)}{12} - \mu^2.$$

2. Any vector orthogonal to $(1, 1, \ldots, 1)^T$ is also an eigenvector with the same eigenvalue λ.

c] PCA for Feature Selection

PCA is not suitable for feature selection in this problem because:

- All features have the same variance, and there is no meaningful correlation between them.
- The first principal component only captures the overall scale of the data, and the remaining components are indistinguishable.

5) Logistic Regression:

i] Logistic Model and Cross-Entropy Error Function

The logistic regression model is given by:

$$P(\hat{y}=1 \mid x_1, x_2) = \frac{1}{1+\exp\left(-(\theta_0+\theta_1 x_1+\theta_2 x_2)\right)}$$

The cross-entropy error function is:

$$E(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(P(\hat{y}_i)\right) + (1-y_i)\log\left(1-P(\hat{y}_i)\right)\right]$$

ii] Gradient Descent and Updated Logistic Regression Model

Given updated weights after one iteration of gradient descent:

$$\theta_0 = -1.00316626, \quad \theta_1 = 1.50535086, \quad \theta_2 = 0.50196867$$

The updated logistic regression model becomes:

$$P(\hat{y}=1 \mid x_1, x_2) = \frac{1}{1+\exp\left(-(\theta_0+\theta_1 x_1+\theta_2 x_2)\right)}$$

Substituting the updated weights:

$$P(\hat{y}=1 \mid x_1, x_2) = \frac{1}{1+\exp\left(-(-1.00316626+1.50535086\cdot x_1+0.50196867\cdot x_2)\right)}$$

Simplified:

$$P(\hat{y}=1 \mid x_1, x_2) = \frac{1}{1+\exp\left(1.00316626-1.50535086\cdot x_1-0.50196867\cdot x_2\right)}$$

iii] Model Predictions and Evaluation

After convergence of gradient descent, the model is used to make predictions on the test dataset. The evaluation metrics are calculated as follows:

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}}$$

Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Using the test dataset:

- **Accuracy:** 66.67%
- **Precision:** 60.00%
- **Recall:** 100.00%

6) Kaggle - Taxi Fare Price Prediction:

Performance Scores Table:

Model	Training RMSE	Validation RMSE	Private Test RMSE (Score)
LinearRegression	8.33	8.39	7.91451
XGBoost	3.92	4.06	**3.34271**
DecisionTree	0.05	5.48	7.11034
GradientBoosting	4.73	4.79	4.18123
RandomForest	1.41	3.83	**3.30934**

Table 1: Performance of Models on Training, Validation, and Private Test Datasets

Top-2 Scoring Models

1. **RandomForest Regressor**
   - **Private Test Score**: **3.30934**
2. **XGBoost Regressor**
   - **Private Test Score**: **3.34271**

Description of Methods

RandomForest Regressor:

- An ensemble method combining multiple decision trees.
- Reduces overfitting and enhances generalization by averaging outputs of diverse trees.

XGBoost Regressor:

- A gradient boosting framework that builds models sequentially.
- Employs regularization, efficient optimization, and advanced techniques like tree pruning for better performance.

Analysis of Top Performing Methods

Why RandomForest Regressor Performed Well:

- Combines predictions from many trees, reducing the risk of overfitting.
- Excels at handling complex datasets and maintains strong generalization, evident in the lowest private test RMSE.

Why XGBoost Regressor Performed Well:

- Optimizes predictions iteratively, correcting errors from previous models.
- Uses regularization to control model complexity, ensuring balanced performance on training and validation datasets.

Insights on Lower Performing Models

- DecisionTree heavily overfit the training data (RMSE: 0.05), leading to poor generalization.
- LinearRegression and GradientBoosting were outperformed due to simpler architectures (LinearRegression) or less aggressive optimization compared to XGBoost.

In conclusion, **RandomForest** and **XGBoost** were the top models due to their ability to balance complexity, handle non-linear relationships, and generalize effectively across datasets.