Foundations of Machine Learning

Subject Code: CS5590

Assignment-3

Submitted by:

Antala Aviraj
Roll No: CS24MTECH14011

Department of Computer Science and Engineering
IIT-Hyderabad

Date: October 31, 2024

October 31, 2024

Answer 1) Non-Uniform Weights in Linear Regression:

Given a dataset with data points (x_n, y_n) where $n = 1, \ldots, N$, each data point has:

- x_n: an input vector,
- y_n: the target value,
- $\sigma_n > 0$: a weighting factor indicating the importance of each data point,
- $\Phi(x_n)$: a transformation of x_n

We want to find a weight vector w that minimizes the error function:

$$E_D(w) = \frac{1}{2} \sum_{n=1}^{N} \sigma_n \left(y_n - w^T \Phi(x_n) \right)^2$$

where $w^T \Phi(x_n)$ is the prediction for each x_n.

Let's first expand the Error Function

$$E_D(w) = \frac{1}{2} \sum_{n=1}^{N} \sigma_n \left(y_n^2 - 2 y_n w^T \Phi(x_n) + (w^T \Phi(x_n))^2 \right)$$

For finding minimum expression we need to differentiate with Respect to w

Taking the derivative with respect to w, term by term:

- The derivative of $\frac{1}{2} \sum_{n=1}^{N} \sigma_n y_n^2$ is zero because it does not depend on w.
- The derivative of $-\sum_{n=1}^{N} \sigma_n y_n w^T \Phi(x_n)$ with respect to w is:

$$-\sum_{n=1}^{N} \sigma_n y_n \Phi(x_n)$$

- The derivative of $\frac{1}{2} \sum_{n=1}^{N} \sigma_n (w^T \Phi(x_n))^2$ with respect to w is:

$$\sum_{n=1}^{N} \sigma_n (w^T \Phi(x_n)) \Phi(x_n)$$

So we can write this equation as:

$$\frac{dE_D(w)}{dw} = -\sum_{n=1}^{N} \sigma_n y_n \Phi(x_n) + \sum_{n=1}^{N} \sigma_n (w^T \Phi(x_n)) \Phi(x_n)$$

To find the minimum, set $\frac{dE_D(w)}{dw} = 0$:

$$-\sum_{n=1}^{N} \sigma_n y_n \Phi(x_n) + \sum_{n=1}^{N} \sigma_n (w^T \Phi(x_n)) \Phi(x_n) = 0$$

Rearranging terms:

$$\sum_{n=1}^{N} \sigma_n (w^T \Phi(x_n)) \Phi(x_n) = \sum_{n=1}^{N} \sigma_n y_n \Phi(x_n)$$

Solve for w

Notice that $w^T \Phi(x_n)$ is a scalar, so we can rewrite this as:

$$w^T \sum_{n=1}^{N} \sigma_n \Phi(x_n) \Phi(x_n)^T = \sum_{n=1}^{N} \sigma_n y_n \Phi(x_n)$$

Now we have an equation where w is multiplied by a sum of terms:

$$w = \left(\sum_{n=1}^{N} \sigma_n \Phi(x_n) \Phi(x_n)^T \right)^{-1} \sum_{n=1}^{N} \sigma_n y_n \Phi(x_n)$$

Provided that $\sum_{n=1}^{N} \sigma_n \Phi(x_n) \Phi(x_n)^T$ is pseudo-invertible (or invertible) to ensure that the inverse exists.

The optimal weight vector w^* that minimizes the weighted error function is:

$$w^* = \left(\sum_{n=1}^{N} \sigma_n \Phi(x_n) \Phi(x_n)^T \right)^{-1} \sum_{n=1}^{N} \sigma_n y_n \Phi(x_n)$$

Answer 2) Neural Networks:

1. **Define the Cross-Entropy Error Function** For a multi-class classification problem, the cross-entropy error for a single sample is defined as:

$$E = -\sum_{k=1}^{K} t_k \ln(y_k)$$

where:

- K is the total number of classes,
- t_k is the one-hot encoded target for class k, and
- y_k is the predicted probability for class k.

2. **Softmax Activation Function** The predicted probability y_k is obtained by applying the softmax function to the logits a_k:

$$y_k = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}$$

Derivative of E with Respect to y_k To use the chain rule, we first find the derivative of the error E with respect to y_k:

$$\frac{\partial E}{\partial y_k} = -\frac{t_k}{y_k}$$

Now Apply the Chain Rule Using the chain rule, we now compute $\frac{\partial E}{\partial a_k}$ as follows:

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial a_k}$$

Substituting $\frac{\partial E}{\partial y_k} = -\frac{t_k}{y_k}$:

$$\frac{\partial E}{\partial a_k} = -\frac{t_k}{y_k} \cdot \frac{\partial y_k}{\partial a_k}$$

Now we need to Compute $\frac{\partial y_k}{\partial a_k}$ (Derivative of Softmax) Since y_k is defined by the softmax function, we compute its derivative with respect to logits a_k and a_j (for $j \neq k$):

- **Case 1**: $j = k$

$$\frac{\partial y_k}{\partial a_k} = y_k(1 - y_k)$$

- **Case 2**: $j \neq k$

$$\frac{\partial y_k}{\partial a_j} = -y_k y_j$$

Substitute back for $\frac{\partial y_k}{\partial a_k}$:

- For $j = k$:

$$\frac{\partial E}{\partial a_k} = -\frac{t_k}{y_k} \cdot y_k(1 - y_k)$$

which simplifies to:

$$\frac{\partial E}{\partial a_k} = -t_k(1 - y_k) = y_k - t_k$$

Answer 3) Ensemble Methods:

Consider an ensemble model comprising M individual models. For a given input x, each model m (where $m = 1, 2, \ldots, M$) makes a prediction denoted by $y_m(x)$. The true value of the function is represented by $f(x)$, where $f(x) = x^2$.

We are interested in comparing two types of errors:

1. Average Expected Sum-of-Squares Error (E_{AV}): This measures the average squared error across all individual models in the ensemble:

$$E_{AV} = \frac{1}{M} \sum_{m=1}^{M} E_x \left[(y_m(x) - f(x))^2 \right]$$

where E_x denotes the expectation over the distribution of x.

2. Ensemble Expected Error (E_{ENS}): This measures the squared error of the average prediction across the ensemble:

$$E_{ENS} = E_x \left[\left(\frac{1}{M} \sum_{m=1}^{M} y_m(x) - f(x) \right)^2 \right]$$

Our goal is to show that:

$$E_{ENS} \leq E_{AV}$$

To achieve this, we will utilize Jensen's inequality.

Jensen's Inequality

Jensen's Inequality states that for any convex function $\phi(y)$ and any random variable Y,

$$\phi(E[Y]) \leq E[\phi(Y)]$$

In simpler terms, applying a convex function ϕ to the expectation of a random variable Y gives a result that is less than or equal to the expectation of $\phi(Y)$.

Now Apply Jensen's Inequality to the Ensemble Problem.

In our case, we're using the squared error function and also this is convex function:

$$\phi(y) = (y - f(x))^2$$

This function measures the squared difference between a prediction y and the true value $f(x)$, and it is convex in y, so we can apply Jensen's inequality.

Here, Y represents the prediction made by an individual model for a given input x. Specifically, we consider the average prediction of the M models in the ensemble, denoted as $\bar{y}(x)$:

$$\bar{y}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x)$$

where $y_m(x)$ is the prediction made by the m-th model. Thus, $\bar{y}(x)$ is the average prediction of all models in the ensemble.

As we know that $\bar{y}(x) = \frac{1}{M}\sum_{m=1}^{M} y_m(x)$ is the average prediction, we can treat it as $E[Y]$ for the ensemble. By Jensen's inequality:

$$\phi\left(\bar{y}(x)\right) \leq \frac{1}{M}\sum_{m=1}^{M} \phi(y_m(x))$$

or specifically:

$$(\bar{y}(x) - f(x))^2 \leq \frac{1}{M}\sum_{m=1}^{M} (y_m(x) - f(x))^2$$

This inequality shows that the squared error of the average prediction $\bar{y}(x)$ is less than or equal to the average of the squared errors of the individual predictions $y_m(x)$.

Applying the expectation E_x over both sides of our inequality, we get:

$$E_x\left[(\bar{y}(x) - f(x))^2\right] \leq E_x\left[\frac{1}{M}\sum_{m=1}^{M} (y_m(x) - f(x))^2\right]$$

Using the linearity of expectation (allowing us to swap the expectation E_x and the summation), we find:

$$E_{ENS} \leq \frac{1}{M}\sum_{m=1}^{M} E_x\left[(y_m(x) - f(x))^2\right]$$

And know that $E_{AV} = \frac{1}{M}\sum_{m=1}^{M} E_x\left[(y_m(x) - f(x))^2\right]$, we conclude:

$$E_{ENS} \leq E_{AV}$$

Answer 4) 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:**

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

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:

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

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.

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

Question 6) Gradient Boosting:

Preprocessing Report for Loan Dataset

This report outlines the data preparation steps applied to the `loan_train.csv` and `loan_test.csv` datasets. The main objective was to clean and standardize the data for use in machine learning. Key steps included filtering data, handling missing values, scaling numerical features, and encoding categorical variables.

1] Data Loading and Filtering

The datasets `loan_train.csv` and `loan_test.csv` were loaded, and only rows with a `loan_status` of "Fully Paid" or "Charged Off" were kept for analysis. This focused the dataset on fully resolved loans, discarding any loans with other statuses.

- **Filter Condition**: The datasets were filtered to include only loans with statuses of "Fully Paid" or "Charged Off".
- **Mapping**: The `loan_status` field was mapped to numerical values:
  - `Fully Paid` $\rightarrow 1$
  - `Charged Off` $\rightarrow -1$

This transformation allowed the `loan_status` column to be used as a target variable in supervised machine learning.

2] Handling Missing Values

- **Dropping Columns with High Missing Rates**: Almost 56 columns were removed since it exceeded the 80% missing threshold.

3] Separating Feature Types

After removing columns with high missing values, the remaining columns were divided into two types for targeted preprocessing:

- **Numerical Columns**:Total 33 numerical columns were detected.
- **Categorical Columns**:Total 22 Categorical columns were detected. Some of the following columns were identified as categorical:
  - `loan_status`
  - `annual_income`
  - `employment_length`

- home_ownership
- loan_amnt
- int_rate
- purpose

Note: The loan_status column was excluded from these lists as it represents the target variable.

4] Preprocessing Pipelines

To ensure that numerical and categorical columns were processed appropriately, two separate preprocessing pipelines were defined and applied using a ColumnTransformer.

1) Numerical Feature Pipeline

The numerical features, comprising 33 columns, were processed with the following steps:

- **Imputation**: Missing values in numerical columns were filled using the **median** of each column. This strategy helps reduce the impact of outliers compared to mean imputation.
- **Standard Scaling**: The numerical columns were standardized using StandardScaler. Each value was scaled so that the resulting feature had a mean of 0 and a standard deviation of 1, which improves performance for models sensitive to feature scale.

2) Categorical Feature Pipeline

The categorical features, comprising 22 columns, were processed with the following steps:

- **Imputation**: Missing values in categorical columns were filled using the **most frequent category** in each column. This minimizes distortions in categorical distributions by preserving the predominant category.
- **One-Hot Encoding**: Categorical variables were encoded using OneHotEncoder, which created binary columns for each unique category. This allows categorical information to be used without assuming any ordinal relationship between categories. The handle_unknown='ignore' parameter was set to avoid errors if the test set contains categories unseen in the training set.
- **Total Numerical Features Processed**: 33
- **Total Categorical Features Processed**: 22
- **Training Data Transformation**: The training data was transformed according to these preprocessing steps, resulting in a processed version of X_train.
- **Testing Data Transformation**: The same transformations were applied to the testing data, producing a processed version of X_test.

5] Transformation Results

- **Training Data**: The training dataset (`X_train`) was transformed according to the preprocessing steps, resulting in `X_train_transformed`.
- **Testing Data**: The same transformations were applied to the testing dataset (`X_test`), producing `X_test_transformed`.

By completing these preprocessing steps, both the training and testing datasets are standardized, allowing for consistent and accurate model training and evaluation.