CSCE 633: Machine Learning Homework-1

Arunachalam Venkatachalam

23rd February 2024

1 Gradient Calculation

1. $f(x, y) = x^2 + \ln(y) + xy + y^3$**.**

$$\nabla f(x,y) = \frac{\partial f(x,y)}{\partial x}\hat{i} + \frac{\partial f(x,y)}{\partial y}\hat{j}$$
$$\frac{\partial f(x,y)}{\partial x} = 2x + y$$
$$\frac{\partial f(x,y)}{\partial y} = \frac{1}{y} + x + 3y^2$$
$$\nabla f(x,y) = (2x+y)\hat{i} + (\frac{1}{y} + x + 3y^2)\hat{j}$$
$$\textbf{Substituting point } (x,y) = (10,-10)$$
$$\nabla f(x,y) = 10\hat{i} + 309.9\hat{j}$$

2. $f(x, y, z) = \tanh(x^3y^3) + \sin(z^2)$**.**

$$\nabla f(x,y) = \frac{\partial f(x,y,z)}{\partial x}\hat{i} + \frac{\partial f(x,y,z)}{\partial y}\hat{j} + \frac{\partial f(x,y,z)}{\partial z}\hat{k}$$
$$\frac{\partial f(x,y,z)}{\partial x} = 3x^2y^3 \cdot sech^2(x^3y^3)$$
$$\frac{\partial f(x,y,z)}{\partial y} = 3y^2x^3 \cdot sech^2(x^3y^3)$$
$$\frac{\partial f(x,y,z)}{\partial z} = 2z\cos(z^2)$$
$$\nabla f(x,y) = 3x^2y^3 \cdot sech^2(x^3y^3)\hat{i} + 3y^2x^3 \cdot sech^2(x^3y^3)\hat{j} + (2z\cos(z^2))\hat{k}$$

Substituting point $(x, y, z) = (-1, 0, \pi/2)$

$$\nabla f(x, y) = 0\hat{i} + 0\hat{j} + \pi \cos(\pi^2/4)\hat{k}$$

$$\nabla f(x, y) = 0\hat{i} + 0\hat{j} - 2.4542473833\hat{k}$$

2 Matrix Multiplication

1.

$$\begin{bmatrix} 10 \\ -5 \\ 2 \\ 8 \end{bmatrix} \begin{bmatrix} 0 & 3 & 0 & 1 \end{bmatrix}$$

Ans:

$$\begin{bmatrix} 10*0 & 10*3 & 10*0 & 10*1 \\ -5*0 & -5*3 & -5*0 & -5*1 \\ 2*0 & 2*3 & 2*0 & 2*1 \\ 8*0 & 8*3 & 8*0 & 8*1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 30 & 0 & 10 \\ 0 & -15 & 0 & -5 \\ 0 & 6 & 0 & 2 \\ 0 & 24 & 0 & 8 \end{bmatrix}$$

2.

$$\begin{bmatrix} 1 & -1 & 6 & 7 \\ 9 & 0 & 8 & 1 \\ -8 & 1 & 2 & 3 \\ 10 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 2 & 0 \\ 0 & -1 & 1 \\ -3 & 0 & 4 \\ 3 & 4 & 7 \end{bmatrix}$$

Ans:

$$\begin{bmatrix} 6+0-18+21 & 2+1+0+28 & 0-1+24+49 \\ 54+0-24+3 & 18+0+0+4 & 0+0+32+7 \\ -48+0-6+9 & -16-1+0+12 & 0+1+8+21 \\ 60+0+0+3 & 20-4+0+4 & 0+4+0+7 \end{bmatrix}$$

$$= \begin{bmatrix} 9 & 31 & 72 \\ 33 & 22 & 39 \\ -45 & -5 & 30 \\ 63 & 20 & 11 \end{bmatrix}$$

3 Programming Questions

3.1 Data Processing

Initial shape of the Dataframe: (6250, 13)
Feature, Labels: (6250, 11), (6250, 1)

Dataset Description

The dataset consists of hourly air quality data, likely from a semiconductor manufacturing plant, with each row representing a sample. There are 6250 samples and 13 columns excluding the title row. The columns contain various measurements related to air pollutants and environmental conditions, with data types being signed integers or float values.

Pollutants

- NMHC(GT): Hourly averaged overall Non Metanic HydroCarbons concentration in microgram per cubic meter (mi-crog/m^3).
- C6H6(GT): Hourly averaged Benzene concentration in microgram per cubic meter (microg/m^3).
- NOx(GT): Hourly averaged NOx concentration in parts per billion (ppb).
- NO2(GT): Hourly averaged NO_2 concentration in microgram per cubic meter (microg/m^3).

Sensor Responses

- PT08.S1(CO): Hourly averaged sensor response for CO (Target) in microgram per cubic meter (microg/m^3).

- PT08.S2(NMHC): Hourly averaged sensor response to NMHC.
- PT08.S3(NOx): Hourly averaged sensor response for NOx.
- PT08.S4(NO2): Hourly averaged sensor response for NO_2.
- PT08.S5(O3): Hourly averaged sensor response for O_3.

Environmental Conditions

- T: Temperature in Celsius.
- RH: Relative Humidity.
- AH: Absolute Humidity.

Missing values in each column : 77
Total missing values : $77 * 12 = 924$

Shape of Dataframe after dropping rows with missing values : $(6173, 12)$
Feature, Labels : $(6173, 11), (6173, 1)$

3.2 Exploratory Data Analysis

Normal Distribution?
By visual inspection, we could infer that all features does not have a normal distribution. Moreover fitting a bell curve on top the plot confirms that the distribution is not normal and the data has multiple peaks.

Outliers?
There are outliers present in the distribution which could be inferred from the histogram plots. Peaks/bars that are distant from the major/central peak signify the presence of outliers. A few sensor values which are quite far from the rest of the data distribution are the major outliers and make the distribution non-normal. Given a sample, we could not say if they are actual outliers or some rare values that needs to be considered for training the model. For this sample of data, they act as outliers skewing the data.

Require Normalization?
Since the data has outliers and is non-normally distributed, it could bias the model(slopes and intercept) if not normalized. Also, the given data has features with diverse ranges (different range of values). If the ranges are different, it would be difficult to compare the corresponding feature's importance. And also if not scaled or normalized, the predictions would be poor and misinterpreted. The above factors lead to sub-optimal performance of the model. Thus normalization is required for the given dataset.

Normalization Technique used
I employed Z-score standardization to normalize the data. It transforms the data of all features such that it has a mean of 0 and a standard deviation of 1, making the distribution close to normal distribution. This ensures that all features are in a common scale and contribute equally to the analysis. Since the skewed points might be outliers or might not, the points are not removed and are thus included while training the model. Standardization does not change the shape of the distribution, meaning if the original data is skewed the standardized data will retain those, which is what we wanted. The formula for standardization is:

$$X_{\text{norm}} = \frac{X - \mu}{\sigma}$$

- X_{norm} is the standardized value of X,
- X is the original value of the feature considered.,
- μ is the mean of the feature considered.,
- σ is the standard deviation of the feature considered.

Pearson's Correlation Coefficient
Pearson's coefficient tell you how strongly two variables are related along with the proportionality (inverse or direct). The coefficient ranges from -1 to 1.

+1: **Perfect positive correlation** - both variables increase or decrease together perfectly.
−1: **Perfect negative correlation** - one variable increases as the other decreases perfectly.
0: **No linear correlation** - there's no straight-line relationship between the variables.

Correlation between two variables
Correlation coefficient between PT08.S2(NMHC) and PT08.S5(O3): 0.91, indicating a very strong positive correlation. This means that as the values of PT08.S2(NMHC) increase, the values of PT08.S5(O3) tend to increase as well, and vice versa. They co-vary closely.

Correlation association between variables(features):
Positive Correlation

- The correlation of C_6H_6 (GT) with AH, RH, and T are 0.984, 0.924, 0.971 respectively. This signifies that if there is a change in AH, RH, or T, C_6H_6 will change. If AH, RH, or T decreases, C_6H_6 will also decrease and vice versa.
- T, RH, and AH: These three variables have a strong positive correlation (>0.9) with each other. This means that they tend to increase or decrease

together. For example, if the temperature increases, the relative humidity and absolute humidity are likely to increase. Similarly, if relative humidity decreases, temperature, and absolute humidity are likely to decrease.

- $PT08.S_2$(NMHC) and $PT08.S_4$(NO2) are highly correlated with a positive correlation coefficient of 0.875. //
- $PT08.S_5$(O3) and NO2 have a moderate positive correlation coefficient of 0.262, indicating a weak tendency for these two variables to increase or decrease together. Higher NO2 levels might be associated with slightly higher ozone levels, but these variables are affected by other factors largely.

Negative Correlation

- NOX(GT) and $PT08.S_3$(NOx) have a negative correlation coefficient of -0.450. This means that they tend to move in opposite directions. If the concentration of NOX(GT) increases, the concentration of $PT08.S_3$(NOx) is likely to decrease.
- $PT08.S_3$(NOx) and $PT08.S_5$(O3) have a moderate negative correlation coefficient of -0.216, indicating that there is a weak tendency for these two variables to move in opposite directions. When NOx values increase, they are associated with low O3 levels. The relationship is not strong; rather, they are affected by other factors more significantly.

3.3 Linear Regression Implementation

The manual_linear_regression function implements a linear regression from scratch. The specifics of the function are given below:

Inputs

- x_{train_array}: A 2D NumPy array representing the training data features. Each row represents a data point, and each column represents a feature.
- y_{train_array}: A 1D NumPy array representing the target values for the training data x_{train_array}.
- η: learning rate
- $num\_iterations$: number of times the model will iterate and get updated through gradient descent.

Outputs

- $cost\_list$: A list that stores the cost (mean squared error) of the model after each iteration.
- $\hat{\beta}$: A 1D NumPy array representing the estimated slope coefficients for each feature.

- y_{pred}: A 1D NumPy array representing the predicted target values for the training data using the estimated model.

Steps

1. Initialization:
   - Sample Size $m = x_{train\_array}[0]$
   - Slope coefficients initialized to zeros $\hat{\beta} = \mathbf{0}_{n_{features} \times 1}$
2. Iteration loop:
   (a)

$$y_{pred} = x_{train\_array} * \hat{\beta}$$
$$cost = \frac{1}{m} \sum_{i=1}^{m} (y_{pred[i]} - y_{train\_array[i]})^2$$

   Mean squared error (MSE) is employed as the cost function:

$$MSE_i = (y_{pred[i]} - y_{train\_array[i]})^2$$
$$cost = \frac{1}{m} \sum_{i=1}^{m} MSE_i$$

   (b) Calculate gradient

$$\nabla_{cost} \hat{\beta} = \frac{1}{m} [[x^T_{train\_array}][(y_{pred} - y_{train\_array})]]$$

   (c) Gradient Descent is used as the optimization algorithm to find the best model coefficients. Slope coefficients are updated based on the below equation

$$\hat{\beta} \leftarrow \hat{\beta} - \eta(\nabla_{cost} \hat{\beta})$$

   where η is the learning rate and $\nabla_{cost} \hat{\beta}$ is the gradient of the cost function with respect to the slope coefficients.
3. Return: $(cost\_list, \hat{\beta}, y_{pred})$

3.4 Logistic Regression Implementation

The manual_logistic_regression function implements a logistic regression from scratch using the binary cross-entropy loss function and gradient descent optimization. The specifics of the function are given below:

Inputs

- x_{train_array}: A 2D NumPy array containing the training data features.
- y_{train_array}: A 1D NumPy array containing the binary target labels (0 or 1) for the training data x_{train_array}.
- η: learning rate
- $num\_iterations$: number of times the model will iterate and get updated through gradient descent.

Outputs

- $\hat{\beta\_logistic}$: A 1D NumPy array representing the estimated slope coefficients for each feature.
- $y_{prediction\_logistic}$: A 1D NumPy array containing the predicted binary labels (0 or 1) for the training data x_{train_array}.

Steps

1. Initialization:
   - Sample Size $m = x_{train\_array}[0]$
   - Slope coefficients initialized to zeros.

$$\hat{\beta\_logistic} = \mathbf{0}_{n_{features} \times 1}$$

   - Epsilon ϵ, a small value added to prevent sigmoid(log) function from divide by zero error.
2. Iteration loop:
   (a) Sigmoid Function: Sigmoid function is used to map the output of the model to a certain binary class, either 0 or 1 based on the threshold. In our case, the threshold was (label >1000)

$$\sigma(z) = \frac{1}{1 + \exp(-z_i)}$$

   where z_i is the input to the sigmoid function.

(b) The binary cross-entropy (or log loss) cost function for logistic regression is given by:

$$\text{Cost}(\hat{y}, y) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i + \epsilon) + (1 - y_i)\log(1 - \hat{y}_i + \epsilon)]$$

where:

- $\hat{y}_i$ is the predicted probability after the processing the raw prediction through the sigmoid function.
- y_i is the actual label (0 or 1).

This cost function penalizes the model more when it makes confident wrong predictions (i.e., predicting close to 1 when the actual label is 0, or predicting close to 0 when the actual label is 1). The goal of logistic regression is to minimize this cost function, typically using optimization algorithms like gradient descent (explained above in linear regression implementation).

(c) Gradient descent is used to minimize the cost function by iteratively updating the model coefficients:

$$\hat{\beta\_logistic} \leftarrow \hat{\beta\_logistic} - \eta(\nabla_{cost}\hat{\beta\_logistic})$$

3.5 Result Analysis - Linear Regression

No big change was noticed across the average and standard deviation values across all 5-folds, the changes were minimal.

Average RMSE over 5-folds: 72.03575856310333, indicating a reasonable prediction error.

Standard deviation of RMSE over 5-folds: 1.8084361348605946, suggests relatively small variation in performance across folds, implying the model generalizes well to unseen data.

Predicted intercept and slope values:

- β_0 (Intercept): 1053.092069
- $\beta_{\text{NMHC(GT)}}$: 33.220017
- $\beta_{\text{C6H6(GT)}}$: 51.618089
- $\beta_{\text{PT08.S2(NMHC)}}$: 75.191831

- $\beta_{\text{NOx(GT)}}$: 23.214990
- $\beta_{\text{PT08.S3(NOx)}}$: -33.561160
- $\beta_{\text{NO2(GT)}}$: -8.058687
- $\beta_{\text{PT08.S4(NO2)}}$: 28.424360
- $\beta_{\text{PT08.S5(O3)}}$: 97.328165
- β_{T}: 16.124948
- β_{RH}: 54.313082
- β_{AH}: 37.490298

The coefficients indicate the impact of each feature on the target variable. Features with higher absolute coefficients (regardless of the direction - positive or negative) are considered more informative because they have a greater influence on the target variable.

3.6 Result Analysis - Logistic Regression

Accuracy across the folds

- Accuracy Fold-1 : 0.913360
- Accuracy Fold-2 : 0.914980
- Accuracy Fold-3 : 0.900405
- Accuracy Fold-4 : 0.906807
- Accuracy Fold-5 : 0.900324

Average accuracy across 5-folds: 0.9071752439320468 (90.71%)
Standard deviation of accuracy across 5-folds: 0.00619801207511017

Precision across the folds

- Precision Fold-1 : 0.931694
- Precision Fold-2 : 0.939276
- Precision Fold-3 : 0.908108
- Precision Fold-4 : 0.923077
- Precision Fold-5 : 0.920378

Average precision across 5-folds: 0.9245066747580946 (92.45%)
Standard deviation of precision across 5-folds: 0.010562282237093618

Recall across the folds

- Recall Fold-1 : 0.922869
- Recall Fold-2 : 0.926115
- Recall Fold-3 : 0.924347
- Recall Fold-4 : 0.921833
- Recall Fold-5 : 0.914209

Average recall across 5-folds: 0.9218744041177871 (92.18%)
Standard deviation of recall across 5-folds: 0.004094842291364711

F-1 Score across the folds

- F-1 Score Fold-1 : 0.927260
- F-1 Score Fold-2 : 0.932649
- F-1 Score Fold-3 : 0.916155
- F-1 Score Fold-4 : 0.922454
- F-1 Score Fold-5 : 0.917283

Average F-1 Score across 5-folds: 0.9231605049821502 (92.31%)
Standard deviation of F-1 Score across 5-folds: 0.006179907650625113

The changes in accuracy, precision, recall, and F-1 score across different folds are relatively small. The standard deviations for these metrics are also relatively small, indicating that the performance of the model is consistent across the different folds. Overall, there doesn't seem to be a big change in these metrics across the folds, suggesting that the model is stable and performs consistently well across different subsets of the data.

3.7 ROC Curve - Logistic Regression

Higher AUC values indicate better model performance in distinguishing between classes. The model seems to have a consistent AUC curve area ranging between 0.895 to 0.909 approximately.

The results are consistent and model performs well in predicting and classifying the features into the right binary class. This is justified by the consistent metrics and AUC values.

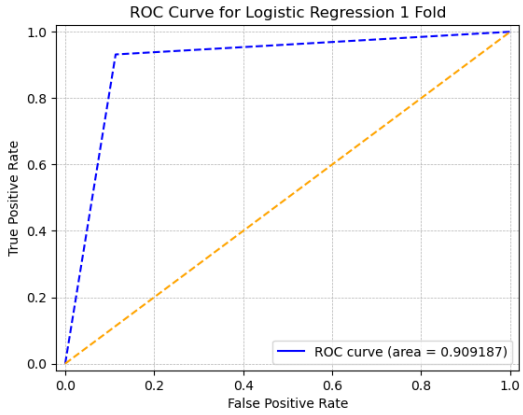
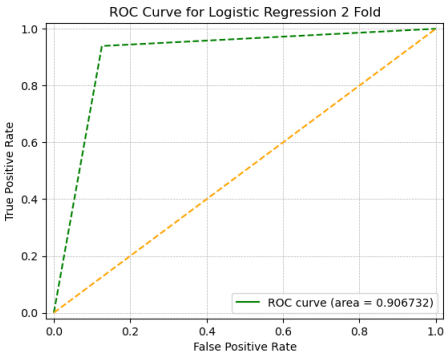
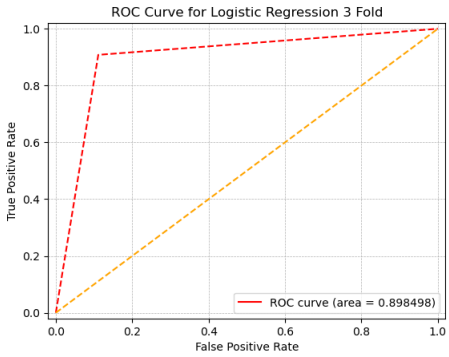


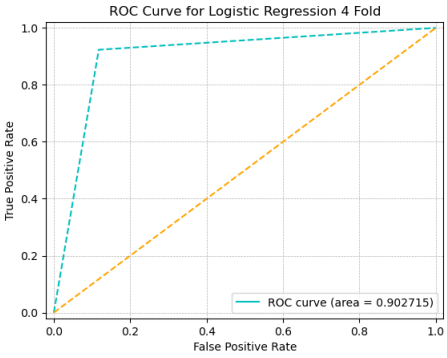
Figure 1: ROC Curve Fold-1



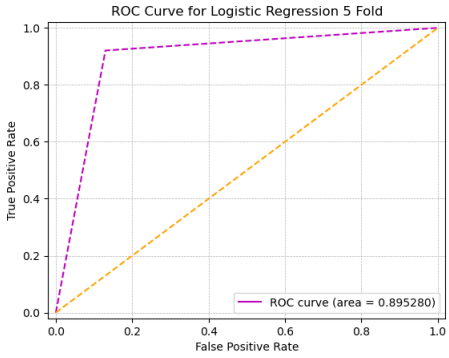
(a) ROC Curve Fold-2



(b) ROC Curve Fold-3



(c) ROC Curve Fold-4



(d) ROC Curve Fold-5

Figure 2: ROC Curves for Folds 2-5

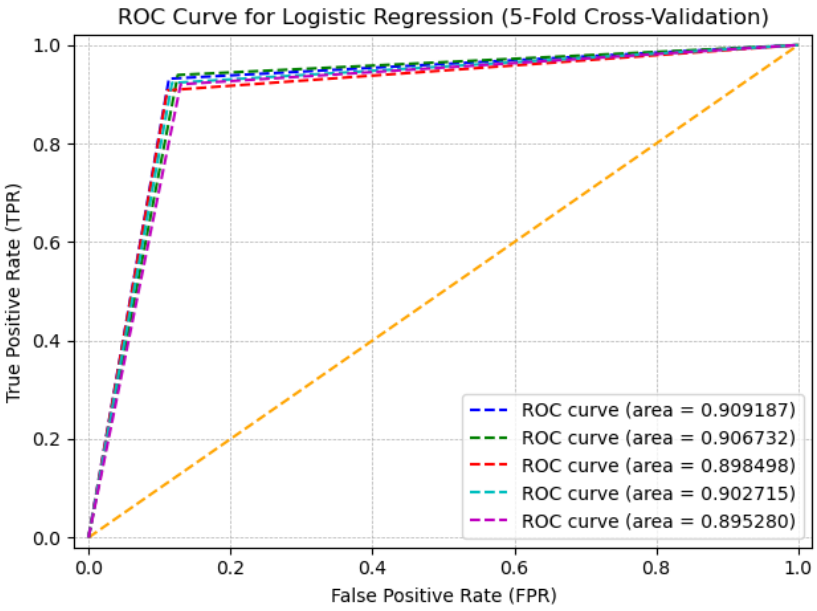


Figure 3: ROC Curve Combined