Justin Lam

CIVL 498A HW-3: Logistic Regression, GDA, Naïve Bayes.

Note: For hand calculation problems, you are supposed to write down your answers on a piece of paper and then submit the scanned version of the paper(s). Show your work by writing out derivations if there is any. For coding problems, you can either use Google Colab or Jupyter Notebook (on your own machine) to write and run the code. When submitting, submit only the ipynb file (you should use relative file directory for input files e.g., "./data.csv" in your code, so when I run the file on my local computer, files would load normally), and make sure when "run all" is clicked, all required results from the questions are shown (this is how I will grade your answer. If when I "run all" and your code crashes, it will be deemed as wrong).

1. **(Hand calculation)** Use the data in the following table:

Observations	x_1	x_2	**y**
1	1	0	1
2	2	1	0

Use maximum likelihood method to solve a logistic regression problem from the data above, i.e., find the hypothesis describing the relationship between y and the variables (x_1, x_2), plus another variable representing the intercept (x_0). Note that y has only two classes (i.e., 0 and 1). So, this is a binary classification problem. You are supposed to perform SGD for one epoch on the data, assuming all thetas have an initial value of zero, and the learning rate alpha is one. Hint: Look closely to the notes titled "Notes_LogisticRegression_MLMetrics.pdf" from week 5. Do not just write out the update rule for thetas and then calculate. Follow the derivation from the notes and replicate the derivation for arriving at the update rule.

1. **MLZ:**

$$\text{Argmax } L(\theta) = P(\vec{y}|x;\theta) = \prod_{i=1}^{m} P(y^{(i)} | x^{(i)}; \theta)$$

$$\text{Log } L(\theta) = \ell(\theta)$$

$$g(\theta^T x) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Assume:

$$P(y=1 | x;\theta) = h_\theta(x)$$

$$P(y=0 | x;\theta) = 1 - h_\theta(x)$$

$$\rightarrow p(y|x;\theta) = h_\theta(x)^y \cdot (1-h_\theta(x))^{1-y}$$

where

$$y \{0,1\}$$

$$\text{Arg max } L(\theta) = P(y^{(i)}|x^{(i)};\theta)$$
$$= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \bullet \left(1-h_\theta(x^{(i)})\right)^{1-(y^{(i)})}$$

$$\text{Argmax}_\theta L(\theta) = \text{argmax } \ell(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))$$

For 1 train sample (x_1, x_2, y)

$$\ell(\theta) = y \cdot \log h_\theta(x) + (1-y) \log(1-h_\theta(x))$$

$$\ell(\theta) = y \log g(\theta^T x) + (1-y) \log(1-g(\theta^T x))$$

$$\frac{d\,\ell(\theta)}{d(\theta_j)} = \left[\frac{y}{g(\theta^T x)} - \frac{(1-y)}{1-g(\theta^T x)}\right] \times \frac{d\left(g(\theta^T x)\right)}{d\theta_j}$$

$$\frac{d\left(g(\theta^T x)\right)}{d\theta_j} = g'(z) = g(z)\cdot\left(1-g(z)\right)$$

$$\frac{d\,\ell(\theta)}{d(\theta_j)} = \left[\frac{y}{g(\theta^T x)} - \frac{(1-y)}{1-g(\theta^T x)}\right] \times g(\theta^T x)\left(1-g(\theta^T x)\right) \times \frac{d\,\theta^T x}{d\theta_j}$$

$$= \left[y\left(1-g(\theta^T x)\right) - (1-y)\,g(\theta^T x)\right] x_j$$

$$\frac{d\,\ell(\theta)}{d(\theta_j)} = \left(y - g(\theta^T x)\right) x_j = \left(y - h_\theta(x)\right)\cdot x_j$$

For SGD: $\theta_j = \theta_j + \alpha \times \frac{d\,\ell(\theta)}{d\theta_j}$

$$\theta_j = \theta_j + \alpha \times \left[y - h_\theta(x)\right]\cdot x_j$$

$$\theta_j = \theta_j + \alpha\left[y - \frac{1}{1+e^{-\theta^T x}}\right]\cdot x_j$$

Guess: $\theta_{0,1,2} = 0$, $\alpha = 1$, $x_0 = 1$

$$\theta_j = \theta_j + \alpha\left[y - \frac{1}{1+e^{-\theta^T x}}\right] \cdot x_j$$

Sample 1: (1, 1, 0, 1)

$$\theta_0 = 0 + 1\left[1 - \left(\frac{1}{1+e^{-(0\cdot 1 + 0\cdot 1 + 0\cdot 0)}}\right] \times 1 = 0.5$$

$$\theta_1 = 0 + 1\left[1 - \left(\frac{1}{1+e^{-0}}\right] \times 1 = 0.5$$

$$\theta_2 = 0 + 1\left[1 - \left(\frac{1}{1+e^{-0}}\right] \times 0 = 0$$

Sample 2: (1, 2, 1, 0)

$$\theta_0 = 0.5 + 1\left[0 - \left(\frac{1}{1+e^{-(0.5\times 1 + 0.5\times 2 + 0\times 1)}}\right] \times 1$$
$$= -0.318$$

$$\theta_1 = 0.5 + 1\left[0 - \left(\frac{1}{1+e^{-(1.5)}}\right] \times 2 = -1.135$$

$$\theta_2 = 0 + 1\left[0 - \left(\frac{1}{1+e^{-(1.5)}}\right] \times 1 = -0.818$$

$$y = -0.318 - 1.135x_1 - 0.818x_2$$

2. **(Coding Problem)** Use the dataset named "waterQuality.csv" for the following questions. The data pertains to the water quality of more than 3000 water bodies. The data is synthesized for machine learning use only, and does not represent the true water quality of these water bodies.

Column description:

- ph: pH of water (0 to 14)
- Hardness: capacity of water to precipitate soap in mg/L
- Solids: total dissolved solids in ppm
- Chloramines: amount of Chloramines in ppm
- Sulfate: amount of Sulfates dissolved in mg/L
- Conductivity: electrical conductivity of water in μS/cm
- Organic_carbon: amount of organic carbon in ppm
- Trihalomethanes: amount of Trihalomethanes in μg/L
- Turbidity: measure of light emitting property of water in NTU
- Potability: if water is safe for consumption i.e., Potable -1, Not potable -0

Unit used:

- ppm: parts per million
- μg/L: microgram per liter
- mg/L: milligram per liter
- μS/cm: microSiemens per centimeter
- NTU: Nephelometric Turbidity Units

2.1. Read in the water quality data as a dataframe, and perform data exploration, including: showing the first 5 rows of the data using head(), showing the data type of columns using info(), showing stats of the numerical columns using describe(). Draw histograms of all numerical columns to show the distributions of the data.

2.2. Compute the correlation matrix for all numerical columns of the dataset. Use seaborn's heatmap() class to plot the correlation matrix with annotation on the side. Use seaborn's pairplot() class, with no additional parameters (i.e., call it like this sns.pairplot(df)) to show the scatter plot version of the correlations among features.

2.3. Fill the missing values in every column using the mean value of that column, and then perform outlier removal. Outliers are defined as the rows with any one of the ten columns that has a feature value falls out of 3 standard deviations from the mean (look at the code example we have in StarterCode_Pandas_Plotting.ipynb regarding outlier detection).

2.4. Build a logistic regression classifier using the linear_model.LogisticRegression class, then perform 10-fold cross validation (using the model_selection.cross_validate class) for the logistic regression classifier using the following metrics (i.e., test scores in the cross_validate class): precision_score, recall_score, f1_score, log_loss. Print the test scores for every cross validation split, and then calculate the average test scores.

2.5. Same as 2.4., but use the classifier discriminant_analysis.LinearDiscriminantAnalysis instead.

3. **(Coding Problem)** Use the dataset named "twitterData.csv" for the following questions. I collected this data using twitter API from code inspired in this page. The data collected include 126 tweets from the week of Feb.8th 2022 to Feb.14th 2022. The tweets are about the housing market in Canada. The last column named "sentiment" was created and labeled by me and has two sentiments as "negative" and "positive". The labels may not be correct, since the housing market is complicated, but they are good enough labels for this assignment.

3.1. Read in the tweet data as a dataframe, and print the number of positive sentiment tweets and the number of negative sentiment tweets.

3.2. Use the class TfidfVectorizer from sklearn.feature_extraction.text, with the following parameters: max_df=0.95, min_df=2, stop_words="english", to transform the original tweets, which are in the format of strings, to a matrix of numerical values for each tweet. This method is called ("term-frequency, inverse document frequency"), which is widely used in natural language processing. For this question, you are supposed to first apply the TfidfVectorizer to the "tweet" column of the original dataframe, and then create a new dataframe called "dfTFIDF" using the output of TfidfVectorizer. This new dataframe should have 126 rows, and 286 columns, with the column names being the words from tweets retrieved using: vectorizer.get_feature_names_out().

3.3. Perform an 80-20 train/test split of the tweet data using the train_test_split class from sklearn.model_selection, with the X being dfTFIDF, and the y being sentiment of the tweet, use 42 as the random_state. Then, train a naïve bayes model using the class MultinomialNB from sklearn.naive_bayes using the training data, and report the score() of the model for the training data.

3.4. Use the trained model from 3.4. to predict the sentiment of the testing data, and then draw a confusion matrix of the classification results using the confusion_matrix class from sklearn.metrics, and the heatmap class from seaborn.

3.5. Use your trained model from 3.4. to classify the sentiment for the following two statements: "vancouver housing price too high", and "canada\'s housing market had the best year ever with price growth continues across canada". Print out the classification result (i.e., positive or negative).