Introduction to Machine Learning - Practicum 4 - Logistic Regression

Topics covered: Logistic Regression, Stochastic Gradient Descent Algorithm

Objectives:

- To get familiarized implementing a machine learning algorithm, which is Logistic Regression from the scratch (i.e., without using any machine learning API).
- To implement Stochastic Gradient Descent algorithm to learn the parameters of logistic regression function from the training data.
- To apply logistic regression algorithm to classify whether the bank notes is genuine or fake.

1. Logistic Regression

In statistics, logistic regression, or logit regression, or logit model is a regression model where the dependent variable is categorical. In this practicum, we will focus on the binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as fake/original. In this practicum, you will design the logistic regression algorithm to classify the bank notes as genuine or fake using the dataset provided by the UCI Machine Learning repository https://archive.ics.uci.edu/ml/datasets/banknote+authentication#.

2. Stochastic Gradient Descent

Stochastic gradient descent, also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. In other words, this algorithm tries to find minima or maxima by iteration. When using this method, you must select a learning rate (alpha) parameter that determines the size of the improvement step to take on each iteration of the procedure. In this practicum, you will implement the stochastic gradient descent algorithm from the scratch to learn the parameters for your logistic regression model.

3. Dataset

This dataset is about distinguishing genuine and forged banknotes. Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400 x 400 pixels. Due to the object lens and distance to the investigated object, gray-scale pictures with a resolution of about 660 dpi were gained. A Wavelet Transform tool was used to extract features from these images. The input and the output attributes are as follows:

1. Variance of Wavelet Transformed image (continuous)
2. Skewness of Wavelet Transformed image (continuous)
3. Kurtosis of Wavelet Transformed image (continuous)
4. Entropy of image (continuous)
5. Class (target): Presumably 0 for genuine and 1 for forged

Note: The dataset need to be randomly shuffled before you split the data.

4. Your tasks

a. Use the given function `loadData(filename)` (in *LogisticR-start.py*) to load banknotes data into an array `X`. `X.shape` should return `(1372,5)`.

b. Write a function `dataNorm(X)` for feature normalization. Don't forget to insert a dummy feature X_0 as the first column. The shape of the returned `X_norm` is `(1372,6)`. The mean and sum for each feature:

		Mean	Sum
Col0	1	1	1372
Col1	Variance	0.5391	739.664
Col2	Skewness	0.5873	805.78
Col3	Kurtosis	0.2879	395.03
Col4	Entropy	0.6689	917.75
Col5	Class(output)	0.4446	610

c. Construct the logistic regression equation to classify whether the bank note is genuine or fake. Write down your equation on a paper.

d. Construct the error function(entropy loss) using the logistic regression equation that you have done in task c.

e. Use the error function (task d) to write a function `errCompute()` for monitoring the convergence. This function takes in dataset `X_norm` and parameters `theta` (with shape (5,1)), returns the error `J`. Run `errCompute(X_norm, np.zeros((X_norm.shape[1]-1,1)))`, you should get 0.6931. You may find that the function `scipy.special.expit()` would be helping in implementing sigmoid.

f. Implement function `stochasticGD()`. This function takes in dataset `X_norm` (should be shuffled), `theta`, learning rate `alpha`, and maximal iterations `num_iters`. It returns the learned theta. To test your code, a shuffled data set (***shuffled.data***, but it is a non_normalized data set) is provided. Use the normalized shuffled data set to run `theta = stochasticGD(X_shufnorm, np.zeros((X_shufnorm.shape[1]-1,1)), 0.01, 1372*20)` , you'll get the cost(return by `errCompute()`) 0.3151.
Use your learned theta to predict each sample in `X_shufnorm`. The predict value $\hat{y}$ for each sample is in file ***predict.data***. The accuracy should be 90.6%.

g. Plot a graph on the error function against iteration number.