Student Name:_____Roll No:_____Section:_____

CS334 - Machine Learning

Lab 12 - Error Based Learning - Part - 2

Instructor: Ms. Maham Ashraf
E-mail: mashraf@uit.edu

Semester: Fall, 2023

Objective

The purpose of this lab session is to introduce Error based learning models, such as Gradient Descent algorithm, Regression based classifier for categorical and continues features, and Support Vector Machine (SVM).

Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

How to Submit

- Submit lab work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19a-001-SE_LW01.py)
- Submit home work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Home work file name should be saved with your roll number (e.g. 19a-001-SE_HW01.py)

1 Learning Rate Schedule using Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.
Strictly speaking, SGD is merely an optimization technique and does not correspond to a specific family of machine learning models. It is only a way to train a model. Often, an instance of *SGDClassifier* or *SGDRegressor* will have an equivalent estimator in the *scikit-learn* API, potentially using a different optimization technique. For example, using SGDClassifier(loss='log_loss') results in logistic regression, i.e. a model equivalent to *LogisticRegression* which is fitted via SGD instead of being fitted by one of the other solvers in *LogisticRegression*. Similarly, *SGDRegressor* (loss='squared_error', penalty='l2') and Ridge solve the same optimization problem, via different means.
The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

The class *SGDRegressor* implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties to fit linear regression models. *SGDRegressor* is well suited for regression problems with a large number of training samples (> 10.000), for other problems we recommend Ridge, Lasso, or ElasticNet.

Students Tasks: The following piece of code is given below that explains how you can use SGD regressor on datasets. Use `food.csv` data set and implement SGD regressor, print the learning graph and accuracy of the model.

```
import numpy as np
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
n_samples, n_features = 10, 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
# Always scale the input. The most convenient way is to use a pipeline.
reg = make_pipeline(StandardScaler(),
                    SGDRegressor(max_iter=1000, tol=1e-3))
reg.fit(X, y)
```

2 Handling Categorical Descriptive variable

Categorical features can only take on a limited, and usually fixed, number of possible values. For example, if a dataset is about information related to users, then you will typically find features like country, gender, age group, etc. Alternatively, if the data you're working with is related to products, you will find features like product type, manufacturer, seller and so on.

These are all categorical features in your dataset. These features are typically stored as text values which represent various traits of the observations. For example, gender is described as Male (M) or Female (F), product type could be described as electronics, apparels, food etc.

Note that these type of features where the categories are only labeled without any order of precedence are called nominal features.

Features which have some order associated with them are called ordinal features. For example, a feature like economic status, with three categories: low, medium and high, which have an order associated with them.

There are also continuous features. These are numeric variables that have an infinite number of values between any two values. A continuous variable can be numeric or a date/time.

Regardless of what the value is used for, the challenge is determining how to use this data in the analysis because of the following constraints:

- Categorical features may have a very large number of levels, known as high cardinality, (for example, cities or URLs), where most of the levels appear in a relatively small number of instances.
- Many machine learning models, such as regression or SVM, are algebraic. This means that their input must be numerical. To use these models, categories must be transformed into numbers first, before you can apply the learning algorithm on them.
- While some ML packages or libraries might transform categorical data to numeric automatically based on some default embedding method, many other ML packages don't support such inputs.

- For the machine, categorical data doesn't contain the same context or information that humans can easily associate and understand. For example, when looking at a feature called City with three cities New York, New Jersey and New Delhi, humans can infer that New York is closely related to New Jersey as they are from same country, while New York and New Delhi are much different. But for the model, New York, New Jersey and New Delhi, are just three different levels (possible values) of the same feature City. If you don't specify the additional contextual information, it will be impossible for the model to differentiate between highly different levels.

You therefore are faced with the challenge of figuring out how to turn these text values into numerical values for further processing and unmask lots of interesting information which these features might hide. Typically, any standard work-flow in feature engineering involves some form of transformation of these categorical values into numeric labels and then applying some encoding scheme on these values.

Following example give you an introduction to handle categorical variables during Machine Learning process.

```
# Importing the dataset
dataset = pd.read_csv('startup.csv')
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, 4]
# Convert the column into categorical columns
states=pd.get_dummies(X['State'], drop_first=True)
# Drop the state coulmn
X=X.drop('State',axis=1)
# concat the dummy variables
X=pd.concat([X,states],axis=1)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
    random_state = 0)
# Fitting Multiple Linear Regression to the Training set
from sklearn . linear_model import Linear Regression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
# Predicting the Test set results
y_pred = regressor . predict ( X_test )
from sklearn.metrics import r2_score
score = r 2 _score ( y_test , y_pred )
print(score)
```

Students Tasks:

1. Load `flight.csv` data set.
2. Apply pre-processing on data set.
3. Encode categorical features into numerical values.
4. Apply linear regression model from `sklearn` library.
5. Predict the delay of flight departure.

3 Handling Categorical target variable - Logistic Regression

Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multi-class problems.

You'll need an understanding of the sigmoid function and the natural logarithm function to understand what logistic regression is and how it works.
The Figure-1 shows the sigmoid function (or S-shaped curve) of some variable x:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

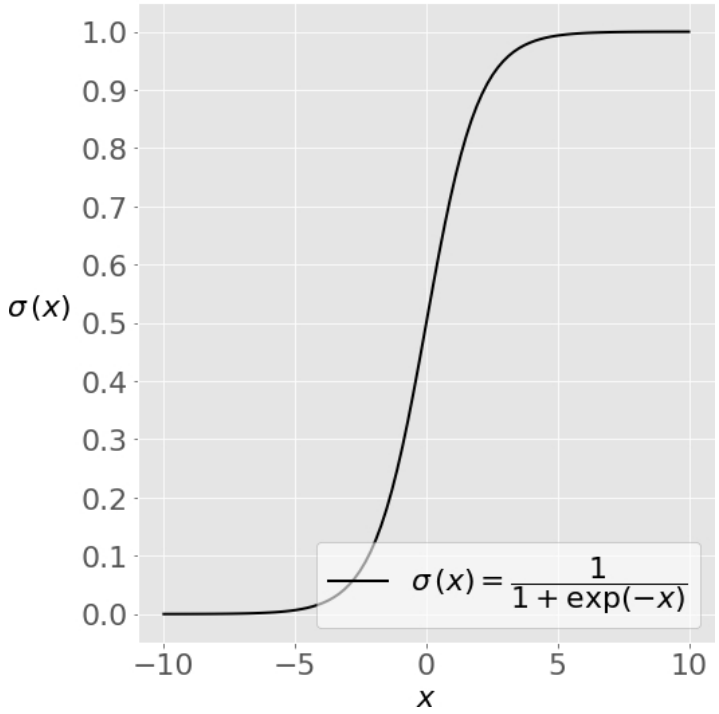

Figure 1: Sigmoid function

The sigmoid function has values very close to either 0 or 1 across most of its domain. This fact makes it suitable for application in classification methods.
This image depicts the natural logarithm log(x) of some variable x, for values of x between 0 and 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

As x approaches zero, the natural logarithm of x drops towards negative infinity. When $x = 1$, log(x) is 0. The opposite is true for log(1 − x).

3.1 Implementation

Step 1: Import Packages, Functions, and Classes First, you have to import Matplotlib for visualization and NumPy for array operations. You'll also need LogisticRegression, classification_report(), label encoder, and confusion_matrix() from scikit-learn: **Step 2: Get Data**: Load `RPMVibration2.csv` data into data frame.

```
data = pd.read_csv('RPMVibration2.csv')
df = pd.DataFrame(data)
le = LabelEncoder()
X= np.array(df.loc[:,['RPM','Vibration']])
df['Status'] = le.fit_transform(df['Status'])
y = np.array(df['Status'])
```

Step 3: Fit Model and analyze parameters,

```
# fit Model
model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(X, y)
print('Model Information:')
print('Model classes:', model.classes_,'\n Model Intercept w[0]: ',
      model.intercept_,'\n Model coefficients w[1] and w[2]: ',model.coef_)
```

Step 4: Evaluate model by analyzing resulted probabilities, predictions, score, precision, recall, and confusion matrix,

```
model.predict_proba(X)
model.predict(X)
model.score(X, y)
```

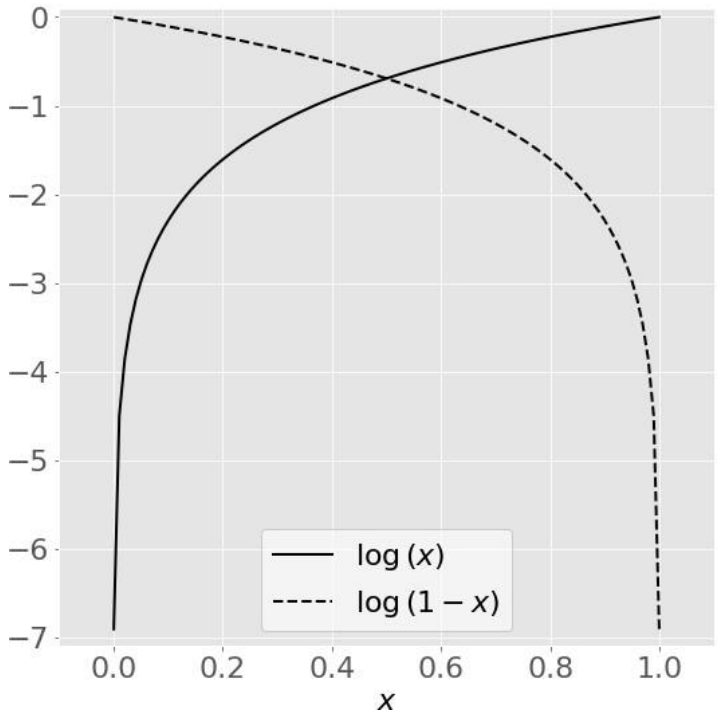

Figure 2: Logarithm function

```
confusion_matrix ( y , model . predict ( X))
classification_report (y, model .predict (X))
```

Students Task: The `ALCustomers.csv` data contains data of airline customer. The main purpose of this dataset is to predict whether a future customer would be satisfied with their service given the details of the other parameters values.

Also the airlines need to know on which aspect of the services offered by them have to be emphasized more to generate more satisfied customers.

1. Develop a prediction model that predict that either customer would be satisfy with the current services or not.
2. Predict the factors (e.g. descriptive variables) that need improvements to increase customer satisfaction rate.

4 Modeling Non-Linear Relationships

All the simple linear regression and logistic regression models that we have looked at so far model a linear relationship between descriptive features and a target feature. Linear models work very well when the underlying relationships in the data are linear. Sometimes, however, the underlying data will exhibit non-linear relationships that we would like to capture in a model.

The Irish farms data during July 2012,shows rainfall (in mm per day), RAIN, and resulting grass growth (in kilograms per acre per day), GROWTH, measured has a strong non-linear relationship between rainfall and grass growth is clearly apparent—grass does not grow well when there is very little rain or too much rain, but hits a sweet spot at rainfall of about 2.5mm per day. It would be useful for farmers to be able to predict grass growth for different amounts of forecasted rainfall so that they could plan the optimal times to harvest their grass for making hay.

To successfully model the relationship between grass growth and rainfall, we need to introduce non-linear elements. A generalized way in which to do this is to introduce basis functions that transform the raw inputs to the model into non-linear representations but still keep the model itself linear in terms of the weights. The advantage of this is that, except for introducing the mechanism of basis functions, we do not need to make any other changes to the approach we have presented so far. Furthermore, basis functions work for both simple multivariable linear regression models that predict

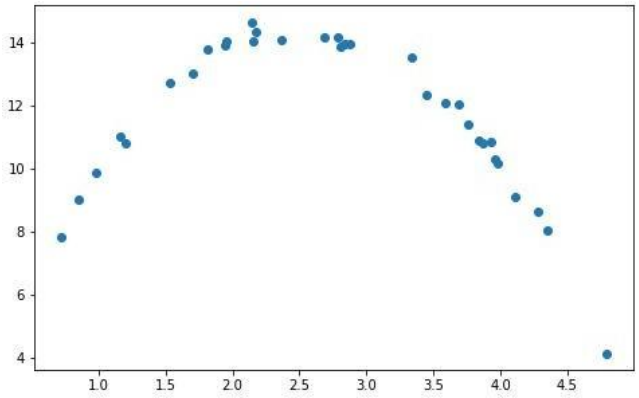

Figure 3: Non-Linear Regression

a continuous target feature and multivariable logistic regression models that predict a categorical target feature.

To use basis functions, we recast the simple linear regression model as follows:

$$M_{\mathbf{w}}(d) = \sum_{k=0}^{b} \mathbf{w}[k] \times \varphi_k(d)$$

where **d** is a set of m descriptive features, **w** is a set of b weights, and φ_0 to φ_b are a series of b basis functions that each transform the input vector **d** in a different way. It is worth noting that there is no reason that b must equal m, and usually b is quite a bit larger than m that is, there are usually more basis functions than there are descriptive features.

One of the most common uses of basis functions in linear regression is to train models to capture **polynomial** relationships. A linear relationship implies that the target is calculated from the descriptive features using only the addition of the descriptive feature values multiplied by weight values. Polynomial relationships allow multiplication of descriptive feature values by each other and raising of descriptive features to exponents. The most common form of polynomial relationship is the **second order polynomial**, also known as the **quadratic function**, which takes the general form $a + bx + cx^2$. The relationship between rainfall and grass growth in the grass growth dataset can be accurately represented as a second order polynomial through the following model:

$$\text{GROWTH} = \mathbf{w}[0] \times \varphi_0(\text{RAIN})^0 + \mathbf{w}[1] \times \varphi_1(\text{RAIN}^1) + \mathbf{w}[2] \times \varphi_2(\text{RAIN})^2$$
$$\text{GROWTH} = 3.707 \times \varphi_0(\text{RAIN})^0 + 8.475 \times \varphi_1(\text{RAIN}^1) - 1.717 \times \varphi_2(\text{RAIN})^2$$

Now we Implement the above model in following steps,

Step 1: Load the data `RainGrowth.csv` into data frame.
Step 2: Assign Rain descriptive feature values as an independent variable in X.
Step 3: Assign Growth target feature values as an dependent variable in Y.
Step 4: Draw scatter plot as shown in Figure - 3.
Step 5: Define objective/basis function, sigmoid functions and fit curve on your model.

```
# define the true objective function
def objective(X, a, b,c):
    return (a * (X**2)) + (b*X)+c
# define the true Sigmoid function
def sigmoid(X, a,b,c):
    y = 1 / (1 + np.exp((a * (X**2)) + (b*X)+c))
    return y
```

Step 6:: Try to fit objective function curve on your data. You can do it by finding Optimized Parameters (popt).

```
xdata =X/max(X)
ydata =Y/max(Y)
```

```
from scipy.optimize import curve_fit
popt, pcov = curve_fit(sigmoid, xdata, ydata)
# Now we plot our resulting regression model.
x = np.linspace(1,7,50)
x = x/max(x)
plt.figure(figsize=(8,5))
y = sigmoid(x, *popt)
plt.plot(xdata, ydata, 'ro', label='data')
plt.plot(x,y, linewidth=3.0, label='fit')
plt.legend(loc='best')
plt.ylabel('Grass Growth')
plt.xlabel('Rain Fall')
plt.show()
```

5 Lab Tasks

1. Use SGD regressor on datasets. Use `food.csv` data set and implement SGD regressor, print the learning graph and accuracy of the model.

2. To learn how to handle descriptive categorical variable in Linear Regression, develop the model as described below,

   (a) Load `flight.csv` data set.

   (b) Apply pre-processing on data set.

   (c) Encode categorical features into numerical values.

   (d) Apply linear regression model from `sklearn` library.

   (e) Predict the delay of flight departure.

3. To learn how to handle continues target variable in Linear Regression Model, Read the following example and develop the proper model.

   The `ALCustomers.csv` data contains data of airline customer. The main purpose of this dataset is to predict whether a future customer would be satisfied with their service given the details of the other parameters values.

   Also the airlines need to know on which aspect of the services offered by them have to be emphasized more to generate more satisfied customers.

   (a) Develop a prediction model that predict that either customer would be satisfy with the current services or not.

   (b) Predict the factors (e.g. descriptive variables) that need improvements to increase customer satisfaction rate.

4. Develop the Non-Linear Regression model for the EEG dataset (`EEGresponses.csv`) for the following objective/basis function for the EEG problem,

   $\varphi_0(\langle \mathbf{P20}, \mathbf{P45} \rangle) = 1$ $\varphi_4(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P45}^2$
   $\varphi_1(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P20}$ $\varphi_5(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P20}^3$
   $\varphi_2(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P45}$ $\varphi_6(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P45}^3$
   $\varphi_3(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P20}^2$ $\varphi_7(\langle \mathbf{P20}, \mathbf{P45} \rangle) = \mathbf{P20} \times \mathbf{P45}$

$$\sum_{b=0}^{j} w[j]\varphi_j(d) = w[0] \times \varphi_0 + w[1] \times \varphi_1 + w[2] \times \varphi_2 + w[3] \times \varphi_3 + w[4] \times \varphi_4 + w[5] \times \varphi_5 + w[6] \times \varphi_6 + w[7] \times \varphi_7$$

$$M_w(d) = \frac{1}{e^{-\sum_{b=0}^{j} w[j]\varphi_j(d)}}$$

6 Home Work Tasks

1. Submit a complete report on Assignment - 2 & 2.1 in following format,
    - Title of Page with following details,
        - UIT logo
        - Assignment name & and number
        - Student name & roll number
        - program name and section
        - Date of submission
    - Table of contents
    - Introduction
    - Description of data set
    - Detailed discussion on Feature Selection
    - Discussion on selected features and their importance
    - Detailed discussion on models Selection
    - Discussion on selected models and their performance
    - System diagram of complete ML project and discuss each component of the diagram
    - Detailed discussions on results
    - Strong and weak points of the developed model
    - conclusion

7 Submission of Machine Learning Lab Manual

The submission of the Lab manual is the basic requirement of lab based courses. you have to submit lab manual on the day when your lab exam is scheduled. Please follow the guide line for submitting Machine learning lab manual.

- The soft copy of ML lab manual will be submitted on MS-Teams. The submission closing day is your Lab exam day and submission time before 11:55PM.
  Note: To avoid any unfortunate situation, upload your lab manual one day ahead from closing day & time. No excuse will be accepted for non-submission.
- The outline of the format of ML Lab manual is provided below,
    1. Title of Page with following details,
        - USMAN INSTITUTE OF TECHNOLOGY
        - DEPARTMENT OF COMPUTER SCIENCE
        - UIT logo
        - COURSE: MACHINE LEARNING (CS334)
        - Instructor: Dr. Sharaf Hussain
        - program name and section
        - LAB MANUAL
        - Student name & roll number
        - Date of submission
        - Marks Obtained:
        - Signature of Instructor
    2. Table of contents (e.g. S.No:, Lab name, date, page number)
    3. Lab Hand out as provided for in each lab.
    4. Lab Work solution (Task number such as LW-1.1, each lab work solution should start from new page).

5. Home Work solution (Task number such as HW-1.1, each home work solution should start from new page).
6. You will repeat point 3-5 for each lab.
7. Page numbering is mandatory and it should be bottom right corner of each page (e.g Page 1 of n).
8. header contains course name & number on top left of each page, except title page.
9. header should contains semester name (e.g. Spring 2022) on top right corner of the page.
10. header should contains title of the lab on bottom left corner of the page
11. Assignments are not part of your lab manual, therefore, do not include your assignments in your lab manual.