Stochastic Gradient Descent using Linear Regression with Python

J V N Lakshmi
Research Scholar
Department of Computer Science and Application
SCSVMV University, Kanchipuram, India

Abstract: Information is mounting exponentially and hungry for knowledge. As data is increasing, world is moving in hunting knowledge with the help of analytics in this age of Big Data. The flooding of data emerging from diverse domains is labelled for automated learning methods for data analysis is meant as machine learning. Linear Regression is a statistical method for plotting the line and is used for predictive analysis. Gradient Descent is the process which uses cost function on gradients for minimizing the complexity in computing mean square error. These methods are implemented in this paper using python programming tool for analysing the datasets.

Keywords: Big Data, Machine Learning, Linear Regression, Predictive Analysis, Gradient Descent, Mean Square Error

I. INTRODUCTION

Machine Learning is learning technique consisting of certain algorithms that identify patterns to expect the potential data, or to execute crucial decision making under uncertainty situations. Gaining Knowledge, understanding the patterns, image detection and face recognition are a few phrases which machine learning defines precisely. This refers to transformation for completing a job related to artificial intelligence (AI). A model designed should preserve the features of the algorithm adopting to the modifications in its environment and figuring necessary actions by possibly predicting the effects.

A. ***Some of the reasons why machine learning algorithms are applied on datasets are illustrated as below:***

- With certain pairs of input data tasks cannot be defined by training examples. In these situations, machines are trained to adjust their internal structures accordingly to achieve desired outputs.
- Machine learning techniques extract hidden relationships from massive piles of data in correlating and predicting further forecasts.
- Machine learning based algorithms are also used for designing humanoids where some added features are required to improve the internal properties to think, look, understand and act.
- Job up gradation of existing machine plans is employed by using techniques of Machine learning.
- Knowledge in encoding certain tasks is complex by human but machines learn gradually by capturing the information and encoding was done.
- Environments transform over periods. Constant redesigning will assist in adapting the changes per the environment. Machines that can adapt to a changing environment would reduce the need for constant redesign.
- Natural Language processing is discovered in identifying and understanding the upcoming languages to adapt for constant growing changes in vocabulary.

- Enduring by restore of AI systems to conform to innovative knowledge is impractical, but these methods made it possible.

This paper is organized in as drafted below in section 1 features of python programming. Section 2 stochastic Gradient Descent is discussed. In section 3 Linear Regression concepts. Section 4 describes the predictions using stochastic gradient descent with linear regression. And finally the results are discussed in section 5 and section 6 concludes.

II. PYTHON PROGRAMMING

Python is simple, innovative and cognitive programming platform for research and higher education. It is efficient in maintaining high level data structures, classes and object oriented approaches. Python has an elegant IPython console for mathematical and statistical computations. This platform uses interpreter for translation and is ultimate for language scripting and is also best for web applications such as JANGO. This is best platform for mastering the machine learning algorithms.

A language excels at string processing–that is, the manipulation of strings lists a few languages with good string processing capabilities and compares them in terms of the degree to which they are still being actively developed by a community of developers and users and whether they are object-oriented [1].

III. STOCHASTIC GRADIENT DESCENT

The aim of the learning process is to optimize the objective function. Stochastic Gradient Descent is one of the supervised machine learning technique optimizes the cost function in the learning process. The objective is to minimize the cost function and is defined as J to understand the weights by using sum of squared errors between trained set and real outcomes.

$$J(w) = \frac{1}{2}\sum_{i}\left(y^{(i)} - \emptyset\left(Z^{(i)}\right)\right)^2$$

The cost function defined is convex and powerful called gradient descent (incremental) to determine the minimum cost to classify the samples in the dataset. Partial derivation is computed on the cost function with respect the each weight w_j $\frac{\partial J}{\partial w_j} = -\sum_i (y^{(i)} - \emptyset(z^{(i)}))x_j^{(i)}$

Batch Gradient descent takes the entire batch as training set is a costly operation if m is large. The incremental algorithm is preferred over batch gradient descent.

Executed code of Stochastic Gradient Descent using python language is drafted in figure 1 as given below.

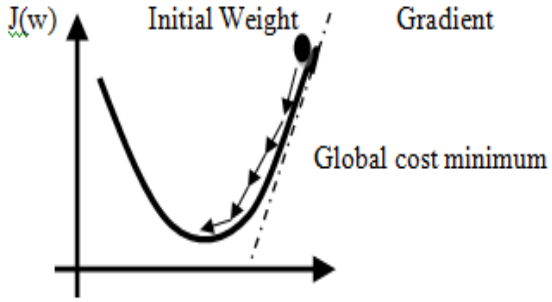

Figure 1: Cost Function

```
def rmse_metric(actual,predicted):
    sum_error = 0.0
    for i in range(len(actual)):
        prediction_error = predicted[i]-actual[i]
        sum_error+=(prediction_error**2)
    mean_error = sum_error/float(len(actual))
    return sqrt(mean_error)
def evaluate_algorithm(dataset, algorithm, nfolds, *args):
    folds = cross_validation(dataset, nfolds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set=sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        rmse = rmse_metric(actual, predicted)
        scores.append(rmse)
    return scores
```

Figure 2: Stochastic Gradient Descent

IV. LINEAR REGRESSION

A straight line is assumed between the input variables (x) and the output variables (y) showing the relationship between the values. Statistics on the training data is required to estimate the coefficients. These estimates are used in model for prediction for further data processing. The line of simple linear regression model is y= a1+a2*x where a1 and a2 are the coefficients of the linear equation.

Estimating the coefficients is given as follows:

$$a1 = \frac{sum\left(\left(x(i) - mean(x)\right) * \left(y(i) - mean(y)\right)\right)}{sum\left(\left(x(i) - mean(x)\right)^2\right)}$$

$$a0 = mean(y) - a1 * mean(x)$$

The following is the code written in python for calculating stochastic gradient descent using linear regression.

Results of the linear regression using stochastic gradient descent are drafted as following scores**:**

[0.1337053475936872,

1.0,

0.5093111214436483,

0.1337053475936872, and

0.1337053475936872]

MEAN RMSE: 0.382

The graph below shows the linear regression coefficients that are plotted in the scatter plot from the dataset in figure (4).

```
from random import seed
from random import randrange
from math import sqrt
def dataset_minmax(dataset):
    minmax = list()
    for i in range(len(dataset[0])):
        col_values = [row[i] for row in dataset]
        value_min = min(col_values)
        value_max = max(col_values)
        minmax.append([value_min,value_max])
    return minmax
def normalize(dataset, minmax):
    for row in dataset:
        for i in range(len(row)):
            row[i]= (row[i]- minmax[i][0])/(minmax[i][1]-minmax[i][0])
def cross_validation(dataset,nfolds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = len(dataset)/nfolds
    for i in range(nfolds):
        fold = list()
        while len(fold)<fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
def rmse_metric(actual,predicted):
```

Figure 3: Linear Regression embedded in stochastic Gradient Descent

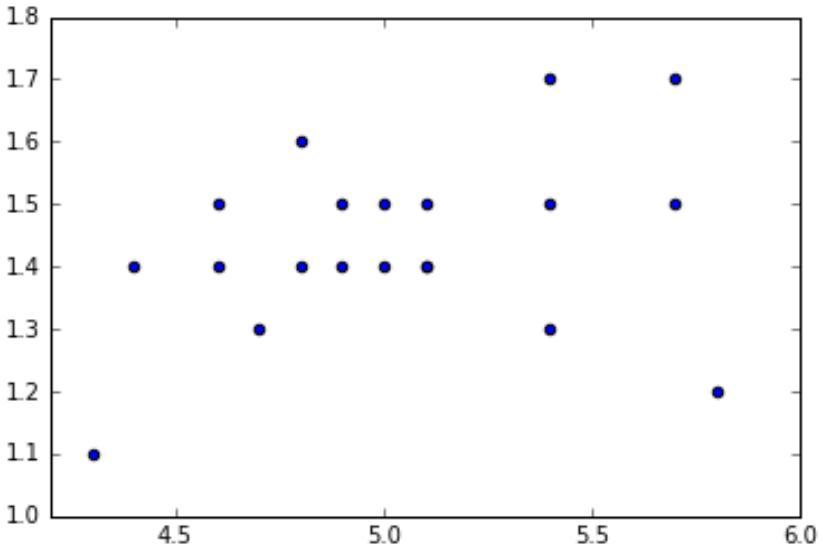

Figure 4: Linear Regression coefficients

The r-square: 0.100337464181

V. ALGORITHM LINEAR REGRESSION WITH STOCHASTIC GRADIENT DESCENT

K – Fold Cross validation technique is realized to approximation the performance of the learned model on the hidden data. Root mean squared error will be used to evaluating the behavior such as split of cross validation, metrics, evaluation of algorithm, prediction of coefficients functions to train the model.

Algorithm for linear regression with stochastic Gradient Descent for dataset

Step 1: Loading the dataset with .csv extension

Step 2: converting the strings in the file to float values for calculation

Step 3: Finding the minimum and maximum values for each column

Step 4: Splitting the data set into k –folds using cross validation technique

Step 5: calculate the root mean squared error

Step 6: Evaluate the algorithm for each cross validation split

Step 7: Make prediction with coefficients

Step 8: Estimate the linear regression coefficients using stochastic gradient descent.

Step 9: Normalize the coefficients for the dataset

This algorithm describes the features in estimating the coefficients of linear regression for normalizing in the dataset.

Calculation of stochastic gradient descent requires two properties Learning Rate and epochs as parameters.

A. Learning Rate

This is applied to adjust the coefficients at each loop which is updated.

B. Epochs

This acts as a counter run through the training data while updating coefficients.

```
def predict(row, coeff):
    yhat = coeff[0]
    for i in range(len(row)-1):
        yhat+=coeff[i+1]*row[i]
    return yhat
def coeff_sgd(train,lrate,nepoch):
    coef = [0.0 for i in range(len(train[0]))]
    for epoch in range(nepoch):
        sum_error = 0
        for row in train:
            yhat = predict(row,coef)
            error = yhat - row[-1]
            sum_error+=error**2
            coef[0]= coef[0]-lrate*error
            for i in range(len(row)-1):
                coef[i+1] = coef[i+1]-lrate*error*row[i]
            print('>epoch = %d lrate = %.3f error = %.3f' %(epoch,lrate,sum_error))
    return coef
dataset = [[1,1],[2,3],[4,3],[3,2],[5,5]]
lrate = 0.001
nepoch = 50
coef = coeff_sgd(dataset, lrate,nepoch)
print coef
```

Figure 5 Estimating Coefficients

Output:

>epoch = 48 lrate = 0.001 error = 1.571

>epoch = 48 lrate = 0.001 error = 1.955

>epoch = 48 lrate = 0.001 error = 2.589

>epoch = 49 lrate = 0.001 error = 0.001

>epoch = 49 lrate = 0.001 error = 1.376

>epoch = 49 lrate = 0.001 error = 1.566

>epoch = 49 lrate = 0.001 error = 1.962

>epoch = 49 lrate = 0.001 error = 2.573

[0.22998234937311363, 0.8017220304137576]

VI. CONCLUSION

In fixed learning rate η is implemented in stochastic gradient descent by an adaptive learning rule which decreases over period. This is especially applicable when large data is processed which has to adapt to situations by transforming spontaneously. This paper concludes by giving python code for linear regression with stochastic gradient descent. The results and output is also being furnished in this paper for the code provided. The graph gives the cost function and the scatter plot drafts the dataset point in the plot. 'r' value is given for the correlated data.

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical Statement: The authors declare that they have followed ethical responsibilities

REFERENCES

[1] Stuart R. and Harald B.: Beginning Python for language Research, pp. 44 – 47, (2007).
[2] Haroshi T., Shinji N. and Takuyu A.:Optimizing multiple machine learning jobs on map reduce. In IEEE – ICCCTS conference at Japan, pp. 59 – 66, (2011).
[3] C.-T. Chu, Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," in NIPS, Eds. MIT Press, pp. 281–288 (2006).
[4] Jason Brownlee: Master Machine learning – How it works, pp. 1-5, (2016).
[5] Walisa and Wichan: An Adaptive ML on Map Reduce for improving performance of large scale data analysis, in EC2 IEEE, pp. 234 – 236, (2013).
[6] Asha and Sravanthi: Building Machine learning Algorithms on Hadoop for Big Data, in IJET Journal Vol 3, No 2, pp. 484 – 489, (2013).
[7] G. Schwarz: Estimating the dimension of a model, The annals of statistics, pp. 461–464, (1978).
[8] Spyder – python IDE
[9] A. Pavlo: A Comparison of Approaches to Large-Scale Data Analysis. Proc.ACM SIGMOD, (2009).
[10] Manar and Stephane : Machine Learning with Python. SIMUREX Oct 2015.
[11] Wiley: Machine learning in Python, Predictive analysis 2015
[12] Caruana, Rich, Nikos Karampatziakis, and Ainur Yessenalina. "An Empirical Evaluation of Supervised Learning in High Dimensions." Proceedings ofthe 25th International Conference on Machine Learning. ACM, 2008.

