International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Exponential Regression via Linear Modeling using Stochastic Gradient Descent and Optimization

Jogimol Joseph[1], Dr. K. Mathew[2]

[1]Assistant Professor, Mount Zion College Of Engineering, Kadammanitta, India
Jogimolb[at]gmail.com

[2]Principal, Mount Zion College Of Engineering, Kadammanitta, India

Abstract: ***Exponential regression is used to model situations in which growth begins slowly and then accelerates rapidly without bound, or where decay begins rapidly and then slows down to get closer and closer to zero. Sometimes linear regression can be used with relationships which are not inherently linear, but can be made to be linear after a transformation. Exponential problems can be lineaized by Stochastic Gradient Descent as well as optimization in cost function.***

Keywords: Exponential Regression, Stochastic Gradient Descent, Optimization, cost function

1. Introduction

Big data refer to technologies and initiatives that tackle diverse, massive data to address the traditional technologies, skills, and infrastructure efficiently. The volume, velocity, and variety of data are greatly high. Big Data is not a single technology or initiative, but it depends on several domains of business and technology. Usually, machine learning algorithms label the incoming data and recognize patterns in it. Then, the ML model translates patterns into insights for business operations. ML algorithms are also used to automate certain aspects of the decision-making process. The probability for a continuous random variable can be summarized with a continuous probability distribution. Continuous probability distributions are encountered in machine learning, most notably in the distribution of numerical input and output variables for models and in the distribution of errors made by models. Knowledge of the normal continuous probability distribution is also required more generally in the density and parameter estimation performed by many machine learning models. As such, continuous probability distributions play an important role in applied machine learning.

2. Exponential Distribution

The exponential distribution is a continuous probability distribution where a few outcomes are the most likely with a rapid decrease in probability to all other outcomes. It is often used to model the time elapsed between events.

Some examples of domains that have exponential distribution events include:

- The time between clicks on a Geiger counter.
- The time until the failure of a part.
- The time until the default of a loan.

This paper considers the use of various Lagrange multiplier tests in testing linearity of univariate time series models against non-linear alternatives. It is assumed that there is not enough theory available for selecting the correct type of non-linearity if the true model is non-linear. The LM tests may then be regarded as linearity tests against incorrect non-linear models. In particular, we consider the following exponential model:

$$y = \alpha e^{\beta x}$$

Taking the natural of both sides of the equation, we have the following equivalent equation:

$$\ln y = \ln \alpha + \beta x$$

This equation has the form of a linear regression model (where I have added an error term ε):

$$y' = \alpha' + \beta x + \varepsilon$$

Common trend in continuous data patterns is ***exponential growth,*** which is also commonly seen as ***exponential decay.*** In exponential growth, a future value is proportionally related to the current value. The general formula for this type of growth can be written as:

$y = y_0$ (1 + r) x

Where $\boldsymbol{y_0}$ is the quantity's initial value (when $\boldsymbol{x} = 0$), and $\boldsymbol{r}$ is the growth rate of the quantity.

Let us prepare test data and create two related variables *x*,*y*, where *y* is equal to *x* elevated to an exponent *e*, plus some Gaussian noise. For convenience we have set the Gaussian noise variance dependent to the exponent too.

```
#test data setting
e=2.465 #exp
x=np.arange(0,25,0,01)
y=xe +np.random.normal(0,10e,x.shape)
```

If we plot the data with a seaborn regression plot, we can easily spot a non-linear relation.

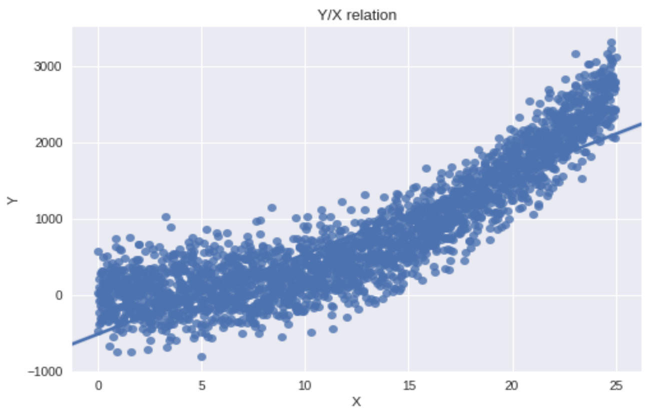

Figure 1: seaborn regression plot

3. Stochastic Gradient Descent (SGD) Method

First, we propose a new Stochastic Gradient Descent (SGD) method for optimizing the sum of a finite set of smooth functions, where the sum is strongly convex. While standard stochastic gradient methods converge at sub-linear rates for this problem, the proposed method incorporates a memory of previous gradient values in order to achieve a linear convergence rate. In a machine learning context, numerical experiments indicate that the new algorithm can dramatically outperform standard algorithms, both in terms of optimizing the training error and reducing the test error quickly.

3.1 Finite sum problems

Consider minimizing an average functions

$$\min_x \; 1/m = \sum_{i=1}^{m} (f(x))$$

Where we have dataset of m points y_i , and each f_i is an error function evaluated at the ith data point.
As $\nabla \sum_{i=1}^{m} fi(x) = \sum_{i=1}^{m} \nabla$ $f_i(x)$, gradient descent updates

are as follows
$x^{(k)}=x^{(k-1)}-t_k\frac{1}{m}\sum_{i=1}^{m} \nabla$ $f_i(x^{k-1})$,k=1,2,3….

3.2 Stochastic Gradient Descent

When *m* is large in the above update rule, computing all the gradient for all *i* becomes costly, so SGD updates are made the following way
$x^{(k)}= x^{(k-1)}-t_k\nabla fi(x^{(k-1)})$, k=1,2,3….

Where $i_k \in \{1, \ldots, m\}$

The figure shows the behavior of full (batch) gradient descent compared to stochastic gradient descent where n=10, p=2.

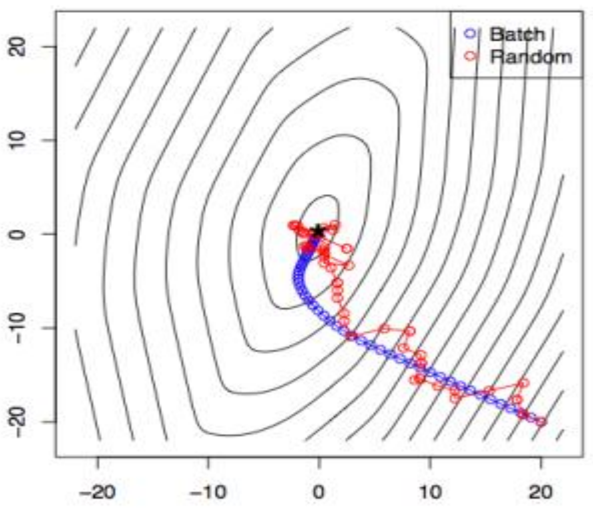

Figure 2: Blue:batch steps, O(np),Red: Stochastic steps, O(p)

Far from the optimum, SGD moves faster where as when we close to optimum, Gradient Descent converges quickly and GSD struggles.

At this point, we would like to perform an experiment: an iterative process that **linearizes** my data by minimizing a cost function.

3.3 The Cost Function and Optimization

The **cost function** is a measure of the 'goodness' of the linear relation that we want to maximize. A good indicator is the **Pearson product-moment correlation coefficient r**, which identifies the strength of the linear correlation between two variables. Pearson r has values between -1 and 1, where 1 is a perfect 1 and 1, where 1 is a perfect positive linear correlation, 0 is no linear correlation, and −1 reveals a perfect negative linear correlation; it means that r =1.Thus to use Pearson r properly, we will take its absolute value and negate it ,because scipy optimize function functions search for minima, whereas we want its maximum.

Let us define the cost function:

```
#define cost function
def cost_function(e):
#y and x are already defined
=np.corrcoef(y,x^e) # returns correlation matrix
#print each iteration
print('r value: {:0.4f} exp:{:.4f}'.format(r[0][1],e)
return –abs(r[0][1])
```

At this point, we have to call one of the Scipy methods. Suitable choice could be the **minimize_scalar** method since our cost function is a scalar function. The algorithm behind this package is **Brent's method**, a root finding algorithm without gradient estimation.

We can also set a **search range**, avoiding the 0 value for the exponent which implies the Pearson r to return an invalid value, even if **numpy.corrcoeff** can handle it. The coefficient is, in fact, defined as:

$$r = \frac{cov(y,x)}{std(y)*std(x)}$$

If x is elevated to 0 the standard deviation is 0, and the ratio returns an invalid value. To perform a **bounded** search let us call:
minimize_scalar(cost_function,bounds=(0.1,10),method='bounded')

The resulting exponent found, in **just 12 iterations**, is 2.482, really close to the exponent we have used to generate the data that is 2.465.

The voice **fun** shows the value of the negative absolute value of Pearson r, which seems to be quite high. Let us plot again *y* and *x* applying the exponent found on *x*, we will notice a **strong linear relationship**:

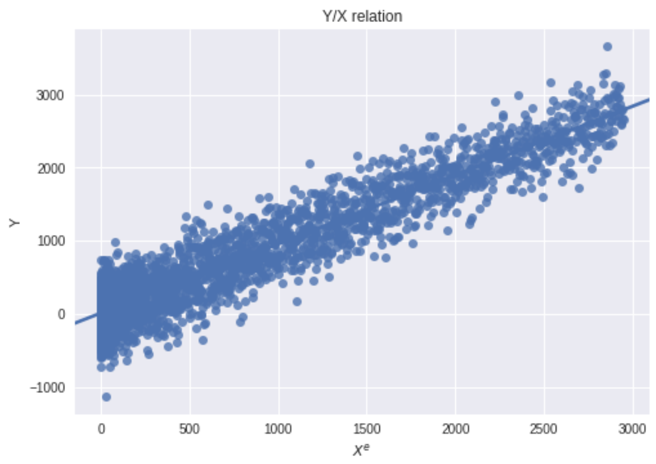

Figure 3: Transformation of exponential to Linear

4. Conclusion

To conclude, it is somewhat possible to transform our exponential problems to linear problems to an extend with the SGD as well as cost function and Optimization method.

References

[1] [NJLS09] A. Nemirovski and A. Juditsky and G. Lan and A. Shapiro (2009), "Robust stochasti optimization approach to stochastic programming
[2] Introduction to Linear Regression Analysis(Wiley Series in probability and statistics) by Douglas C. Montgomery, Elizabeth A Peck and G. Geoferry Vining 2012
[3] Regression Estimators-A comparative study 2E,Marvin H J Gruber
[4] Applied Regression Analysis,Norman R Draper
[5] Data Analysis Using Regression and Multilevel/Hierarchical Models (Analytical Methods for Social Research) – 18 December 2006,Andrew Gelman