Handwritten Digit Recognition

Group Members:-
Dhruvi Desai
Grishika Sharma,
Aayushi Chauhan,
Priyanka Patel

Abstract

Handwritten digit recognition is one of the significant issues in pattern recognition applications. Perusing postal locations, Bank check processing, recognizing numeric entries in forms filled up by hand (say -- tax forms)are some of the applications of handwritten character recognition systems.The purpose of this project is to recognize handwritten digits as a computer program same as human apprehend digits. The core of the issue exists in the capacity to build up a productive algorithm that can perceive manually written digits. Here we have developed a handwritten digit recognition operating procedure which recognises digit 0 to 9 using statistical techniques Logistic Regression and Linear Regression. The main objective of this project is to ensure an effective and reliable approach for recognition of handwritten digits with the help of concepts of linear algebra and machine learning.

Key Words

- *Logistic Regression*
- *Linear Regression*
- *Machine learning algorithm*
- *Classification algorithm*
- *Linear correlation*
- *Least square estimation*
- *Maximum likelihood estimation*
- *Gradient ascent*
- *Gradient descent*

Introduction

Today Machine Learning can be used to identify and convert paper data into digital information and multiple copies of this information can be easily made without any manual work . Digit recognition system is simply the working of a machine to prepare itself or perceive the digits from various sources like messages, bank checks, papers, images, and so forth. Handwritten character recognition can be divided into two categories, namely on-line and off-line handwriting recognition. On-line recognition includes a live change of character composed by a client on a tablet or a smartphone. In contrast, off-line recognition requires automatic conversion of a scanned image into a computer-readable text format. Logistic Regression is basically a Machine Learning based classification algorithm. It is a predictive analysis algorithm which is based on the concept of probability. Linear Regression is also a supervised Machine Learning algorithm. In this method, we train a model to predict the behavior of the data based on certain variables. In the case of linear regression, the two variables which are on the x-axis and y-axis should be linearly correlated. Linear regression is based on ***least square estimation*** which says regression coefficients should be chosen in such a way that it minimizes the sum of the squared distances of each observed response to its fitted value. While logistic regression is based on ***Maximum Likelihood Estimation*** which says coefficients should be chosen in such a way that it maximizes the Probability of Y given X (likelihood). With ML, the computer uses different "iterations" in which it tries different solutions until it gets the maximum likelihood estimates.Here Gradient descent (an optimization algorithm) is used to find the values of parameters (coefficients) of a function that minimizes a cost function and gradient ascent is used to maximize the objective function. In simple words, Linear regression is utilized to estimate the dependent variable in case of a change in independent variables. Whereas logistic regression is used to calculate the probability of an event. For example, an event can be whether the head will come or not on tossing a coin.

Approach:

1. Training the model using Logistic Regression:

Logistic Regression was originally developed by Joseph Berkson in 1944. Logistic Regression is a statistical model that predicts the probability that a random variable belongs to a certain category or class. It uses a logistic function to model a binary dependent variable. We had used the sigmoid function to form the hypothesis. For a data set, we find the likelihood function as Bernoulli Distribution and using the maximum likelihood estimation method, we had estimated the model's parameters using the gradient ascent algorithm.

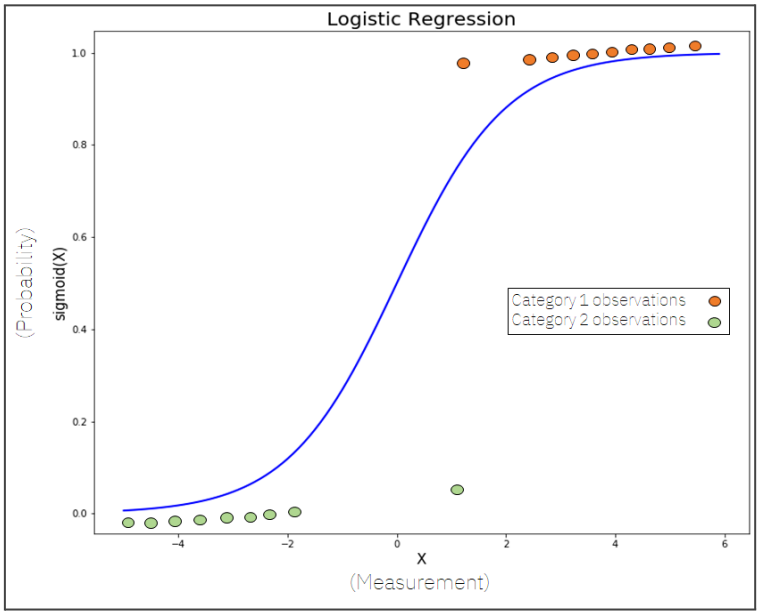


Figure 1: Logistic Regression

Maximum likelihood estimation helps in estimating the parameters from the training data. The best value for parameters in the equation predicts a value almost equal to 1 for the default class and a value almost equal to 0 for the other class. Here, the intuition for maximum likelihood is that a search procedure seeks values for the parameters that minimizes the error in the probabilities predicted by the model.

Initially, the function used for the hypothesis is sigmoid/logistic function which is given by,

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

Reasons for choosing sigmoid function are:

(1)Sigmoid looks like it starts at zero that ramps up to one i.e., it can neither be larger than zero nor less than zero.

(2)Derivative of sigmoid is mathematically convenient.

Then we had to define the probability for the output and the equation for bernoulli distribution. Defining the likelihood cost of theta which represents how plausible our model is.

Equation for **Likelihood** is given by:

$$L(\theta|\, y; x) = P(Y|X; \theta) = \prod_{i=1}^{m} h(x_i)^{\,y_i}(1 - h(x_i))^{(1-y_i)}$$

where, X is the input,

Y is the class we had defined,

h(x_i) represents the predicted response for i^{th}observation i.e.,x_i,

and m is the number of data points given as an input.

By multiplying all these probabilities, we can get the likelihood function. This function will allow us to find the theta parameters. To find the model parameters we have to maximize the likelihood function. But for multiple multiplication of probabilities, maximization turns out to be a complex process. To deal with this issue we find the log likelihood function. Equation to find log likelihood is:

$$L(\theta) = \sum_{i=1}^{m} Y_i \, log(\sigma(\theta^T \overline{x_i})) + \ldots\ldots + (1\text{-yi})(1-\sigma(\theta^T x_i))$$

Gradient Ascent: To find the local maximum of a function, gradient ascent is used to take steps proportional to the positive of the gradient of the function. Machine Learning has a cost function and they can either be concave or convex, if it is concave we use Gradient Ascent.

Gradient Ascent aims at maximizing the objective function.

$$\theta j \leftarrow \theta j + \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Cost function: It is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number. Depending on the problem Cost Function can be formed in many different ways. The purpose of Cost Function is to be either:

1. Minimized - then returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which Cost Function returns a number which is as small as possible.
2. Maximized - then the value it yields is named a reward. The goal is to find values of model parameters for which the returned number is as large as possible.

2. Training the model using Linear Regression:

Linear Regression was developed by Adrien-Marie Legendre in the early 19th Century. Linear Regression is a powerful statistical tool for data analysis and machine learning. It assumes a hypothesis that is a linear combination of the independent variables. It is a linear approach to modeling the relationship between dependent variables and one or more

independent variables. Using gradient descent, the parameters of the model can be found.

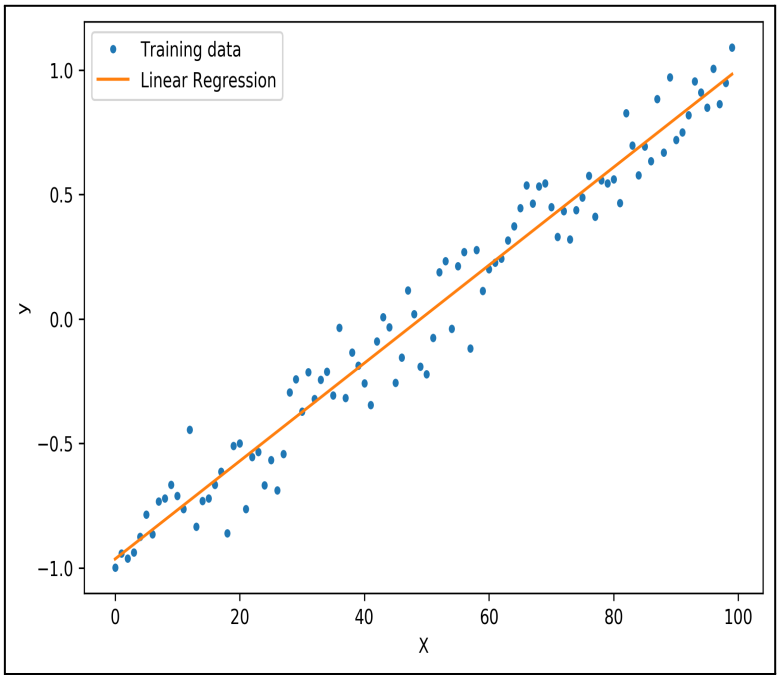


Figure 2: Linear Regression

Here, we are measuring the accuracy of our hypothesis using the formula of cost function which is as follows:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

This cost function(J) of linear regression is the Root Mean Squared Error (RMSE) between predicted value and the true value.

Gradient Descent: To find the local minimum of a function, gradient descent is used to take steps proportional to the negative of the gradient of the function. It is a way to minimize an adjective function J(θ) parameterized by a model's parameters by updating the parameters in the opposite direction of the gradient of the objective function. Cost function can either be concave or convex, if it is convex we use Gradient Descent.

$$\theta j \leftarrow \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Gradient Descent for linear Regression is:

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\frac{\partial}{\partial\theta_0}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

Reason for doing partial derivative is that Partial Derivative represents the rate of change of the functions as the variable changes. In this case, we change values for θ_0 and θ_1 and identifies the rate of change. To apply rate of change values for θ_0 and θ_1, following are the equations:

$$\theta_0: \quad \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$$

$$\theta_1: \quad \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x^{(i)}$$

Now, to find the best minimum, we repeat the above steps until convergence (the property of approaching a limit more and more closely as the number of terms increases of the function) is achieved.
Algorithm for above procedure is:

repeat until convergence{

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x^{(i)}$$

}

We can compute the model parameters using two types of Gradient Descents: (1) Batch Gradient Descent and (2) Stochastic Gradient Descent.

(1) **Batch Gradient Descent:** This method computes the gradient of the cost function for the entire training data set:

$$\theta = \theta - \eta \cdot \nabla\theta\, J(\theta)$$

Pseudocode for Batch Gradient Descent is:

```
for i in range (nb_epochs):
    params_grad = evaluate_gradient(loss_function, data, params)
    params = params - learning_rate * params_grad
```

Here, epochs is a hyperparameter that defines the number of times the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. Then we first compute the gradient vector params_grad of the loss function for the whole dataset w.r.t. parameters. Then we update the parameters in the direction of the gradients with the learning rate determining how big of an update we perform.

(2)Stochastic Gradient Descent: Stochastic gradient descent (SGD) performs updating of parameter for each training example x^i and label y^i :

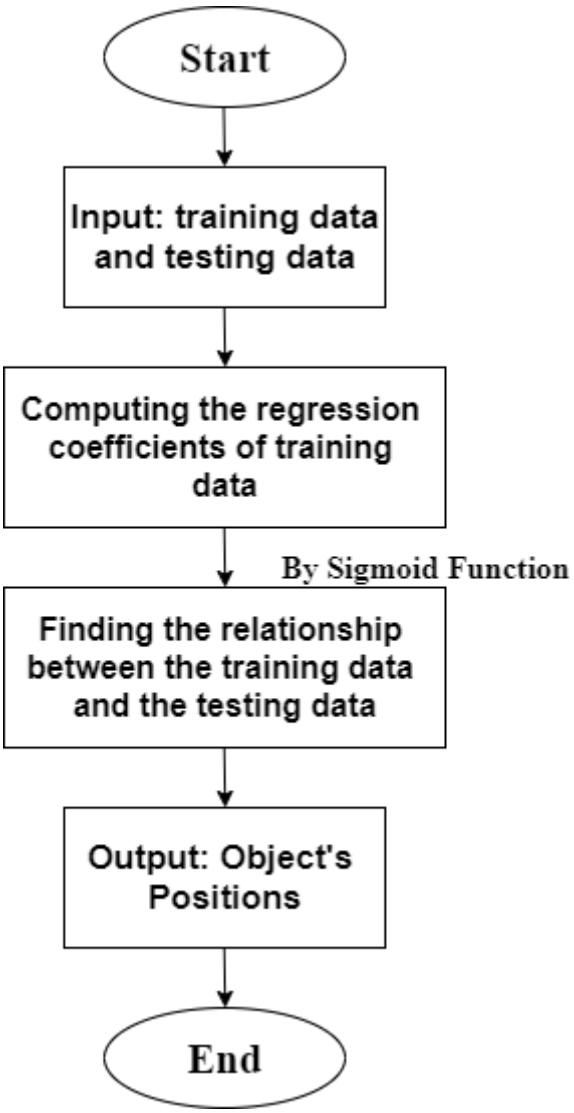
$$\theta = \theta - \eta \cdot \nabla\theta\, J(\theta;\ x^i\ ;\ y^i))$$

Pseudocode for Stochastic Gradient Descent is:

```
for i in range (nb_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function,
    example, params)
        params = params - learning_rate * params_grad
```

Flowcharts:

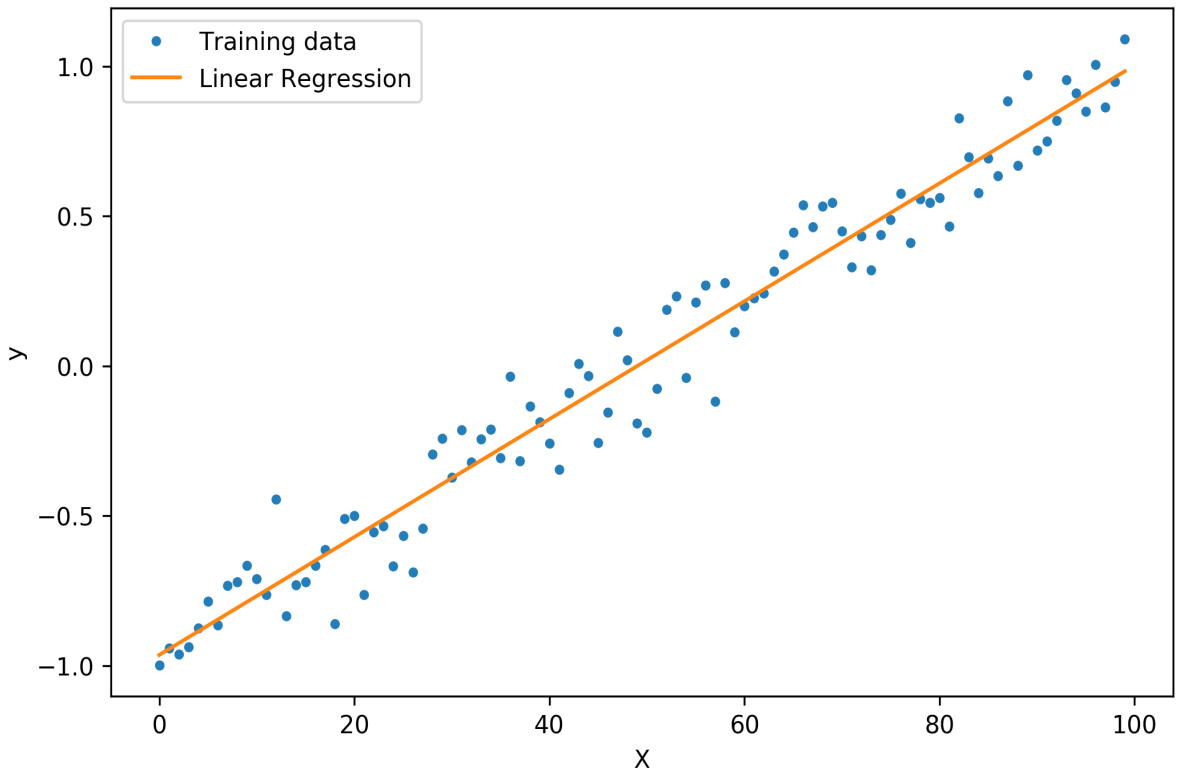
(1) Logistic Regression:

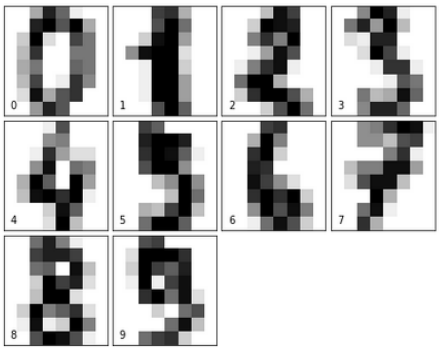


(2) Batch Descent Gradient

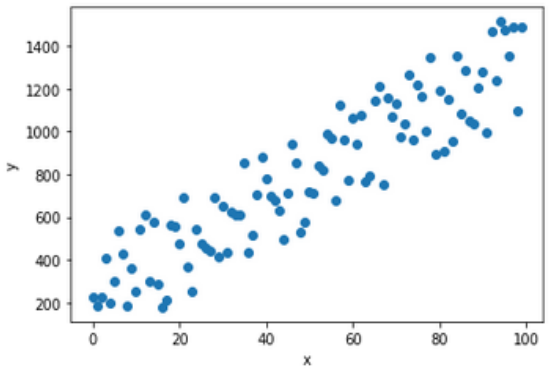
(3) Stochastic Descent Gradient

Screenshots of Results:

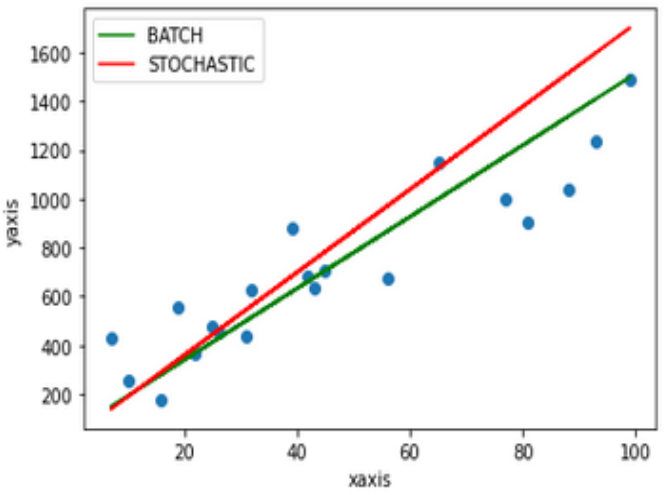

Training data
Linear Regression
1.0
0.5
0.0
−0.5
−1.0
y
0
20
40
60
80
100
X


0
1
2
3
4
5
6
7
8
9

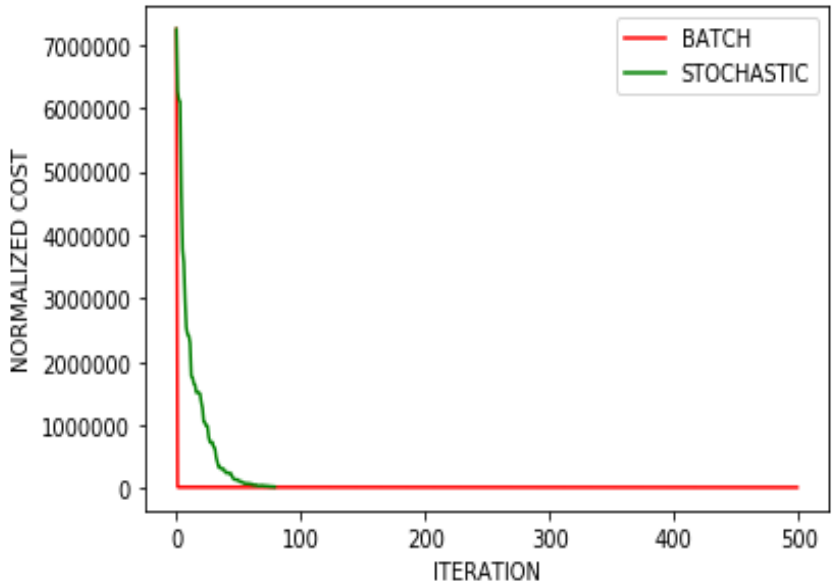
```
The Iteration is: 500
The Cost is: -2.7924499783138748
-----------------------------------------
Accuracy of prediciting a ZERO digit in test set: 0.9944444444444445
```




```
-------------------------
The Iteration is: 0
The Cost is: 7240203.503796968
-------------------------
The Iteration is: 1
The Cost is: 26494.630607239582
-------------------------
The Iteration is: 2
The Cost is: 17711.490518816452
-------------------------
The Iteration is: 3
The Cost is: 17628.987075957317
-------------------------
The Iteration is: 4
The Cost is: 17623.64356797921
```




```
BATCH      T0, T1: 45.78531304947854, 14.669289396227807
STOCHASTIC T0, T1: 19.11423458949738, 16.98609527549524
BATCH rms:      165.9046174016078
STOCHASTIC rms: 229.79484880573838
```




```
Our minimum cost with BGD is: 19071.19095533391
Our minimum cost with SGD is: 27805.016235945233
```

Conclusion:

- In logistic regression we do have a negative cost value it is because of log-likelihood. If we have more number of training sets more our algorithm will be better. If we have less number of iterations it takes less time to run. Well, machine learning projects do take time to run. The accuracy of predicting every digit was more than 0.95.
- As for the linear regression, we know it has two parts batch gradient descent and stochastic gradient descent. As in the output, we plotted two graphs one is the RMS(root mean square) of BGD and SGD. The RMS value of SGD was more than BGD. The second graph was about normalized cost vs iteration. The minimum cost of SGD was more than BGD. The benefit of SGD is that it's computationally a lot faster than BGD.

References:

Meer Zohra, D.Rajeswara Rao paper on "A Comprehensive Data Analysis on Handwritten Digit Recognition using Machine Learning" Approach by. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-6, April 2019

S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair paper on "Handwritten Digit Recognition using Machine Learning Algorithms"

Nelli, Fabio. (2015). Recognizing Handwritten Digits. 10.1007/978-1-4842-0958-5_11.

S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016

Pendharkar, Parag. (2007). A comparison of gradient ascent, gradient descent and genetic-algorithm-based artificial neural networks for the binary classification problem. Expert Systems. 24. 65-86. 10.1111/j.1468-0394.2007.00421.x.

G. David and M. Klein, "Analysis of matched data using logistic regression", In: Proc. of International Conf. on Logistic regression, New York, pp.389-428. 2010.

"Gradient Ascent vs Gradient Descent in logistic Regression", Jan.2017. [Online]. Available:https://stats.stackexchange.com/questions/258721/gradient-ascent-vs-gradient-descent-in-logistic-regression

S. Ruder, "An overview of gradient descent optimization algorithms," NUI Galway Aylien Ltd., June 15, 2017.[Online]. Available: arXiv:1609.04747v2 **[cs.LG]**