Mohammad Asim Iqbal

Hot Dog / Not Hot Dog Classifier- Logistic Regression

Abstract

I have designed a hot dog classifier using Logistic Regression and Gradient Descent with the help of sygmoid function. The main aim of this project was to examine the efficiency of logistic regression when working with images. Another important aspect is the manipulation of hyperparameters like the learning rate and number of iterations. During the manipulation of learning rate, I learned that the best accuracy on the testing data was 56% using eeta=0.01. The number of iterations for the best case was 1500. The classifier's accuracy can be increased with the help of Neural Networks. PCA (Principal Component Analysis) can also be used for this classification.

Background

Hot Dog or not Hot dog is a popular problem on Kaggle, and many people have attempted it. Some of the best solutions for this problem were achieved using CNN and Tensors but there were no solutions involving Logistic Regression and Gradient Descent using the Sigmoid Function. I wanted to attempt the classification with this method to analyze Logic Regression's efficiency while dealing with images.

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. **Note:** Logistic Regression can also be used for Multi Class Classification using the one v/s all technique. **Source:** Statistics Solution

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. **Source:** ML Glossary

The sigmoid takes a value and returns the squashed value between -1 and 1.

Related Work

Kaggle is an online platform where people can solve different challenges which includes image classification problems. There are many people who have tried to solve the hot dog problem in particular but there are many other problems like the Titanic Classification- Survived or Deceased, Dog or Cat Classification and the Garbage Classification: cardboard, glass, metal, paper, plastic or trash and Cat/Non-Cat classification. I chose this specific problem because the solutions only used strategies like Tensor Flow, Edge Detection, CNN, there was no solution pertaining to Logistic Regression.

For this research a person needs to know about the mechanics of Logistic Regression and Gradient Descent using the sigmoid function. I have explained all this in detail later in the report.

Methodology

There are several steps in the methodology used in the program. With the knowledge of Logistic Regression, you can easily understand the project

1. Image Processing

I have used the image from the PIL Library to get the image from the dataset. The image can be any dimension but it's a 3D image.

The pillow converts the image into a dimension of 64x64x3 and then the flatten method converts the image into a row of 12288x1

Then this row is added to the matrix to form the training and testing data for our logistic Regression

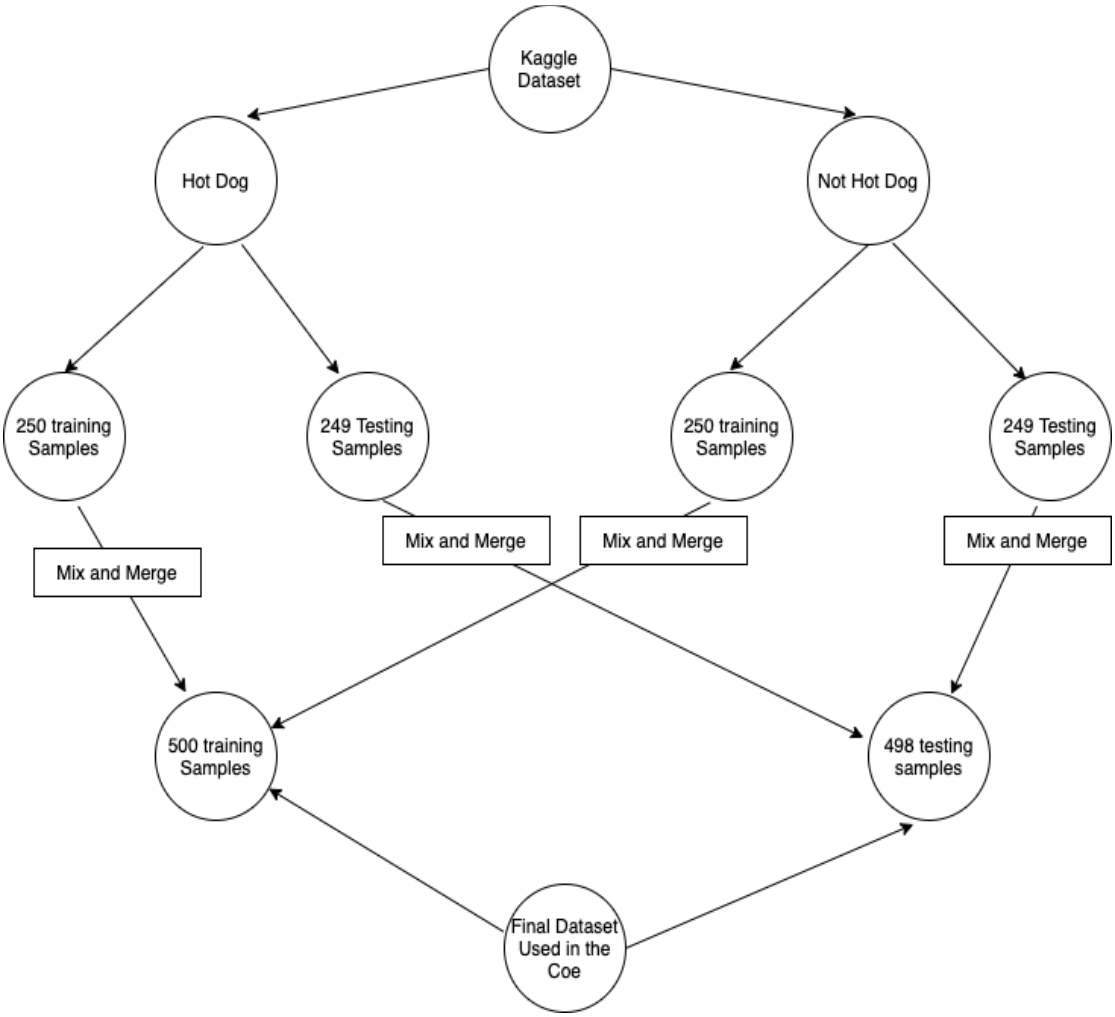
2. Dataset Propagation

We get a row matrix for each image in the dataset. There are 250 images for hot dog and 250 images for not hot dog in the testing set, and there are 249 images of hot dog and 249 images of not hot dog in the training set.

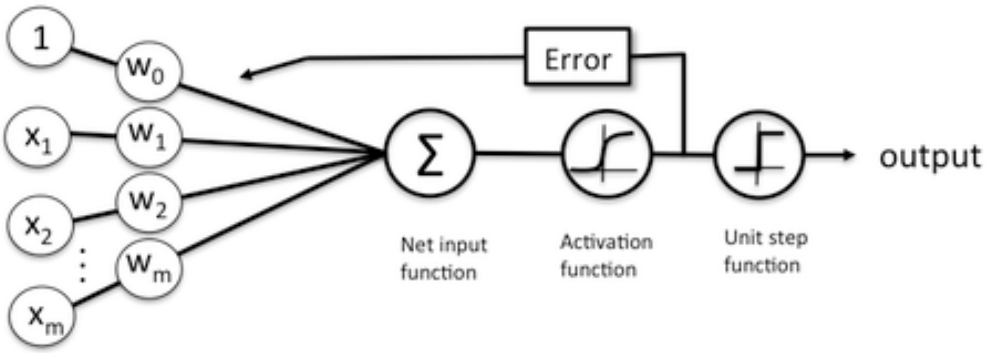
The algorithm takes the matrix after image processing and then combines all the images for hot dog in the training and testing set and then randomize them. Then it selects 250 samples for the training set and 249 for the testing set. It follows the same strategy with not hot dog images.

In Future, if anyone wants to extend this work they can use sklearn's method to divide the data between training and testing. It randomizes the data and then divides it such that 2/3 of the data is for the training set and 1/3 is for the testing set

The Diagram given below better explains the formation of our dataset used for Logistic Regression



3. Logistic Regression



Schematic of a logistic regression classifier.

In the above diagram:

a. **1 through x_m:** is the training matrix we obtained

b. **w_0 through w_m:** are the weight we have optimized in the code

c. **Net Input Function**

$$\sum y_i lng(x,\Theta) + (1-y_i)ln(1-(g(x,\Theta)))$$

d. **Activation Function:** is the Sigmoid Function S(x)

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

e. **Final Equation used for the Classification:**

$$\Theta = \Theta + \frac{eeta}{N}(x^T)(y - g(x,\Theta))$$

Here, we are actually trying to find the probability if a data point belongs to the hot dog class or not. In Logistic Regression we try to maximize the boundary between the line of best fit and the data points. The line is achieved through the cost function and then we try to find optimal values of theta to maximize this distance.

4. Gradient Descent

The theta values are achieved through Gradient Descent. Imagine that you are trying to reach the minimum of a line. The Gradient Descent works in a way that we descent on a line until a minimum is achieved. We can take short leaps or long leaps while descending, the leaps depend on the value of the learning rate eeta we use. A major aim of this project is also to find an optimal value of eeta for better classification results.

5. Sigmoid Function

The sigmoid Function is used to squash the distance between the data point and the line of best fit. The function takes in a value and returns a squashed value between 1 and -1. For achieving the value, it uses a function which looks like **s= 1/ 1+ e^-z.** Here, s is the value returned by the sigmoid and z is the entering value, and e is the Euler's Number.

6. Hyperparameters (Learning Rate and Iterations)

For the learning rate, eeta, used in the gradient descent I have tried many different combinations like 0.01, 0.05, 0.005 but the best results were achieved using 0.01 as the value of eeta.

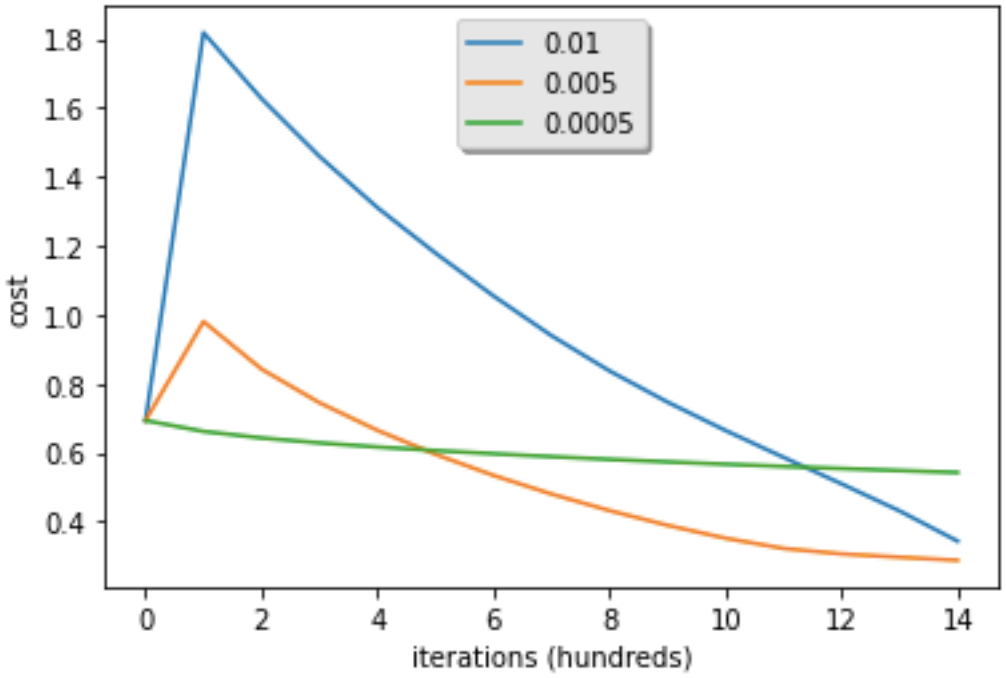
Due to time and space complexity I could not check the algorithm on a lot of iteration, but I tried to work with 1000, 1500, 2000 iterations. The best result was achieved with 1500 iterations.

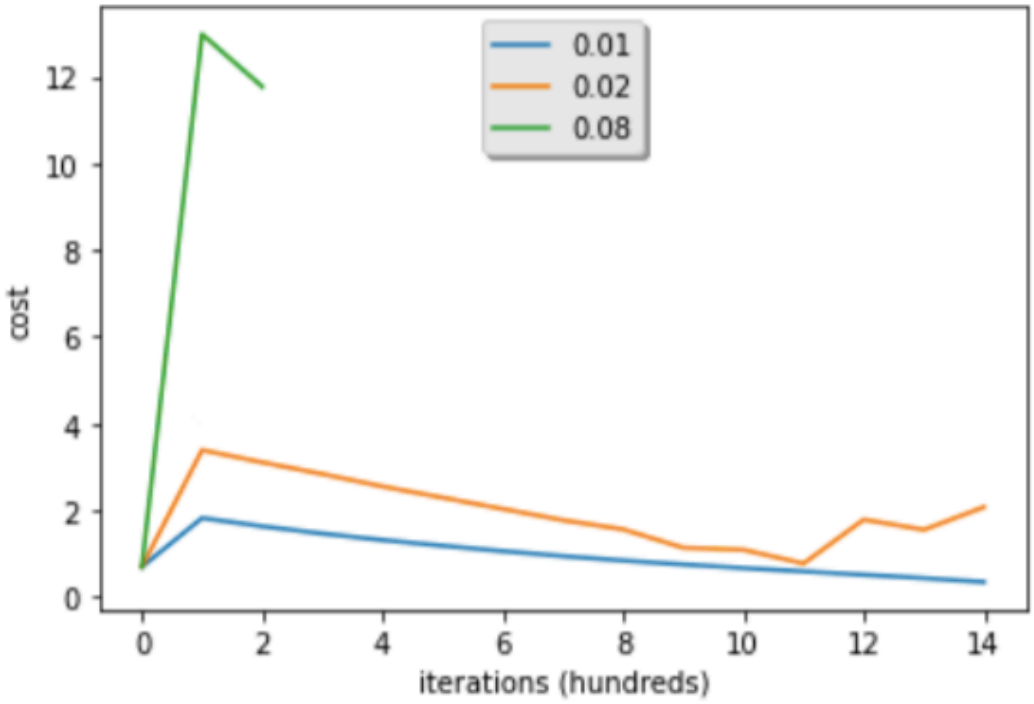
7. Testing

With the theta values I gained with the help of Logistic Regression, I used them to predict the y values for the testing set. The predicted value and the original value of y were used to calculate the accuracy. If the predicted value is same as the original value, then we add 1 to the counter iteratively for 498 rows in the testing set. Then we divide the counter by 498 to find the accuracy.

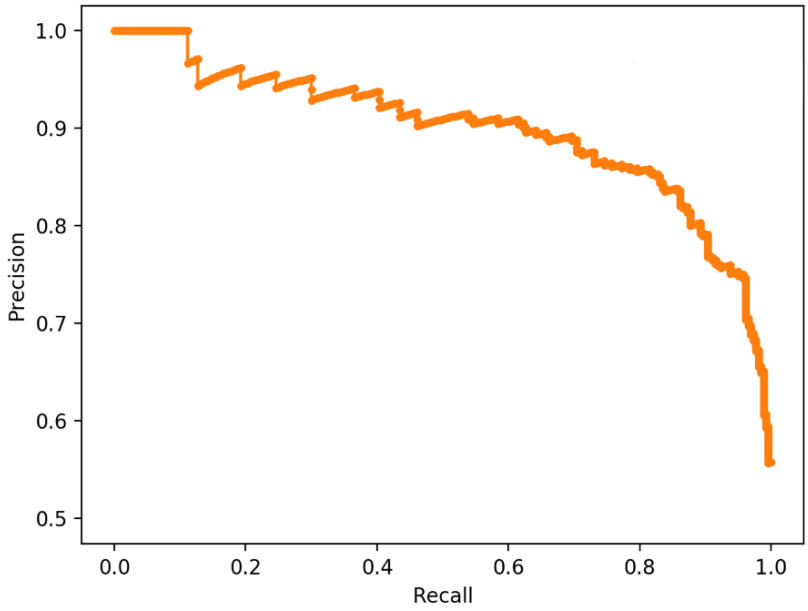
Experiments and Results

1. This is the graph obtained for iteration v/s the cost function





2. This is the graph for P-R Curve:



3. Confusion Matrix

There is a total of 498 classes in the testing set

	Original		
Predicted		**True**	**False**
	True	128	110
	False	108	152

For this confusion matrix eeta=0.01 and no. of iterations= 1500

4. Accuracies

Eeta = 0.01	Eeta = 0.05	Eeta = 0.005
Training Accuracy: 89%	Training Accuracy: 93%	Training Accuracy: 76%
Testing Accuracy: 56%	Testing Accuracy: 53%	Testing Accuracy: 54%

Conclusions

1. Learning Rate

Learning rate defines the leap which we take while descending or ascending down a graph to find minimum or maximum value. The data was tested on 0.01, 0.05, 0.005 as the value of eeta's. The best result was achieved with 0.01 as the value of eeta.

2. Iterations

These are the number of times we update the value of thetas for the line of best fit that maximizes the distance between the data point and the line. The best result was achieved with 1500 iterations. I could not test the model with higher number of iterations due to time and space complexity.

3. **Overfitting/Underfitting**

   There is possible overfitting of the model in the training data because the training accuracies are really high. This means that our trained model is very much dependent on the training data and has high variance.
   There is possible underfitting in the model while working with training set because the accuracies are really low. The bias is causing the underfitting.

4. **Modifications**

   To deal with Underfitting we can add more features or try a different type of gradient descent. To deal with overfitting we can try to get more images from the internet. There is a Node JS query on stack overflow which can fetch images of Hot Dogs or not Hot Dogs from the internet. We can also use a validation set or a regularization term to prevent overfitting. I tried using S-Cross Validation to overcome overfitting, but I could not achieve optimal results due less availability of time.

Future Work

1. **Improving the model**

   To extend this project further we can begin by improving our model. We can split the training and testing data using sklearn module. We can work with more iterations and then analyze the accuracies. We can use PCA (Principal Component Analysis) to do the classification. Another technique which we can use for image processing is the hdf5 visualizer. This can reduce the run-time of our algorithm.

2. **Using CNN- Convolutional Neural Network**

   We can use CNN to do this classification. It uses a technique called pooling technique and then flattens the data. In the Kaggle exercise many people have tried this technique for classification and they achieved accuracies of more than 80%

3. **Using Tensors**

   Using tensor flow can also prove beneficial as it reduces the dimension using tensor contraction and then we can use several techniques like the gradient descent to classify the data.

4. **Application**

   This classification technique can prove beneficial for an app that wants to help people select best places for hot dogs or different kinds of food. People can upvote on different pictures and places and the algorithm can detect whether the picture is a hot dog or not.

Overall Analysis

The overall project is really interesting. Using Logistic regression for image classification is not preferred widely but it's a good exercise to understand the algorithm's behavior while working with images. After finishing the project, I used sklearn logistic regression model to analyze the efficiency of my algorithm. Sklearn model had an accuracy of 58% which is very close to 56%. If we add a validation set or a penalizing term to the model, I am positive that we can build a better image classifier using logistic regression. We can also use this algorithm to perform multi class evaluation using the one v/s all technique discussed in the presentation. One method which can be really efficient in making this classifier is Naïve Bayes classifier technique. I think it can work better here because here are only two classes and it will have a lesser run-time compared to our model. I wanted to implement the hdf5 visualizer for propagating the dataset from images, but it required a lot of time to understand it better. It would be interesting to see how the algorithm works with this technique. I have processed the images of any dimensions into 64x64 but we can increase the dimensions or even decrease them to see how the algorithm works.

Bibliography

1. https://www.kaggle.com/dansbecker/hot-dog-not-hot-dog/data
   There are 250 samples of hot dog and 250 samples of not hot dog in the training set and 249 samples of hot dog and 249 samples of not hot dog in the testing set . The samples have been collected from Kaggle.com and the link is provided above.

2. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
   Apart from the lecture slides and video I referred to this article for more information on Logistic Regression and Gradient Descent using the sigmoid function

3. https://pythonprogramming.net/convolutional-neural-network-kats-vs-dogs-machine-learning-tutorial/
   This is an article for further research and development in this classifier. We can use neural networks to increase the training accuracy of our algorithm. The project is designed in a way such that it can be used with convolutional neural networks also.

4. https://www.statisticssolutions.com/what-is-logistic-regression/
   This Link provides a proper definition of Logistic Regression

5. https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html#:~:text=Gradient%20descent%20is%20an%20optimization,the%20parameters%20of%20our%20model.
   This link provides a proper definition of Gradient Descent

6. You can check out the problem and solutions to the Hot Dog Problem at "https://www.kaggle.com/dansbecker/hot-dog-not-hot-dog/kernels".

7. https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html
   Image Source for the image used in Logistic Regression Explanation

8. https://www.codecogs.com/latex/eqneditor.php
   Equations have been designed using this website