PART I
Logistic Regression

DATASET USED: Iris Flower Dataset

1 Basics of Logistic Regression

Logistic regression is a widely used classification technique in Machine Learning.

Logistic Regression is natively suited to binary classification problems, but certain variaitions have been developed that extend its applicability beyond 2-class problems.

It has a relatively modest number of parameters – just one parameter for every input feature – making for a very simple model; it is not suited to complex classification tasks. However, by using a simple model such as this, the risk of overfitting the data is eliminated. In addition, model simplicity also saves processing power which results in faster development of the model.

In the logistic model, each set of inputs is vectorized into a single input vector and further, all the parameters of the model are also vectorized into another parameter vector. Then, the scalar product of these 2 vectors is calculated. This product is then passed through the logistic function to get the final output of the model – always a real number between 0 and 1. A full mathematical description can be found in Section 2.

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

The Logistic(Sigmoid) Function[6]

2 Mathematical Description and Notation

$X = [X_1, X_2, X_3, \dots]$ *are the inputs to the model*

$\theta = [\theta_0, \theta_1, \theta_2, \dots]$ *are the parameters of the model*

$X = [X_0 = 1 : X_1, X_2, X_3, \dots]$ *bias term is appended*

$Z = X \cdot \theta$ *take the dot product*

$h = \frac{1}{1 + e^{-Z}}$ *output is the sigmoid function of* Z

3 Training the Model

3.1 What is Training?

In classification, the function of any model is to take a set of inputs and predict some output values that decide which class the input example belongs to. In binary classification though, it predicts just a single output. Typically, this output ranges from 0 to 1 and a decision boundary of 0.5 is placed on this range to infer which class is predicted.

To produce an output, the model has some parameters that the input is computed with. Thus, the value of the parameters defines wholly what the output is going to be for every input shown to the model.

The task of training (called fitting the model) is to tune these parameters such that the model can accurately distinguish between inputs of different classes. Hence, labelled data - data for which the correct outputs are known - is fed into the model and the model's predictions on each of these data are calculated. Then a cost function needs to be used, as explained under the following heading.

3.2 Cost Function

The Cost Function measures the error in classifying using a given model. The more the predictions correlate with the actual labels, the lesser should be the value of the cost function.

Thus, the task of fitting the model accurately is accomplished by monitoring the cost function and reducing it's value by adjusting the model's parameters. Therefore, the cost function should also suggest which way to tune each of the parameters of the model.

For logistic regression, the following cost function is used:

$$J(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)), & \text{if y = 1} \\ -log(1 - h_\theta(1 - x)), & \text{if y = 0,} \end{cases}$$

Since the behavior of the function is defined piecewise, its shape is as shown in figures 1 & 2. It is apparent that the function's tendency is towards zero as the prediction tends toward 1, in figure 1 (where the true label is 1), or 0 in figure 2 (where the true label is 0).

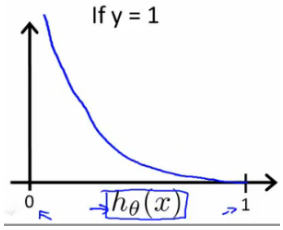

Figure 1: Logistic Cost Function when y=1

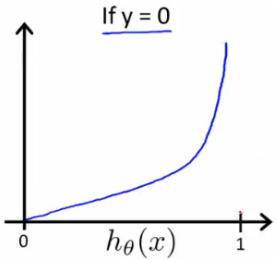

Figure 2: Logistic Cost Function when y=0

The parameters of the model are tuned in a manner that minimizes the cost function. This is achieved with an optimization algorithm...

3.3 Gradient Descent – The Cost Minimization Algorithm

Gradient Descent is a first-order iterative minimization algorithm for finding the minimum of a function. It is based on the following observation: That if a multivariate function F(X) is defined and differentiable in the neighbourhood of a point 'a', then it decreases fastest in the direction opposite the gradient of F at 'a', i.e. - ∇ F(a).

So in every step, a chosen point 'a' is moved a distance proportional to the local gradient, in a direction opposite to the gradient. This logic is summarized as an update step:

$$a_{n+1} = a_n - \alpha \nabla F(a_n) \tag{2}$$

α is called the Learning Rate. It scales the length covered in every step and hence, it defines how fast the optimization proceeds.

The reasoning behind this update step is as follows:

Since the cost decreases in every step, and since the updates converge only when the gradient = 0, the point 'a' would have to converge on the minimum.

However, there are 2 main caveats to this reasoning:

- The minimum that is converged on may be just a local minimum.
- If α is too large, an update might skip over a minimum and cause a cost increase

3.4 Gradient Descent Applied to Logistic Regression

The cost function, J(θ) from Section 3.2 is rewritten as:

$$J(\theta) = -y \log(h_\theta(X)) - (1-y) \log(1 - h_\theta(X)) \tag{3}$$

This form makes it readily differentiable.

Note: J is a function of θ here since θ are the variables in optimizing J

Hence,

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial[-y \log(h_\theta(X)) - (1-y) \log(1 - h_\theta(X))]}{\partial \theta_j}$$

Replacing $h_\theta(X)$ using Equation 1,

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial[-y\log(\frac{1}{1+e^{-\theta^T x}}) - (1-y)\log(1-\frac{1}{1+e^{-\theta^T x}})]}{\partial \theta_j}$$

On simplifying and differentiating,

$$\frac{\partial J}{\partial \theta_j} = (h_\theta(X) - y)x_j$$

Averaged over m examples,

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(X) - y)x_j$$

Thus, from (2),

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(X) - y)x_j \tag{4}$$

Equation (4) is the final update step used in Logistic Regression.

4 The One vs. Rest Approach to Classification

The dataset chosen, the Iris Dataset [3], contains examples belonging to 3 classes. However, the logistic regression model is best suited for binary classification problems. Therefore, the data labels must be changed to make a binary classification problem out of the Iris Dataset. This is achieved using one-vs-rest classification, where each example is classified as belonging to one particular class (label 'ONE') vs not belonging to that class (label 'REST'). This class needs to be chosen beforehand and typically, the choice of this class influences the ease of classification. The One-vs-Rest approach can be used to easily convert an n-class problem to a 2-class one, so that Logistic Regression becomes possible.

Further, the One-vs-Rest approach also suggests a method to run logistic regression on a fully multi-class problem. This method is commonly used for the logistic technique; however, it will not be explored in this project. If there are C classes, C classifiers can be trained, each of which determines whether the input example belongs to one of the C classes or not, a la One-vs-Rest. Then, by comparing the classification scores of the C classifiers, the best class that the example would fall into may be selected.

Since the Iris Dataset has three classes of inputs, there are three ways to implement One-vs-Rest classification – each one classifying a specific class vs the rest.

5 Experiment 1

5.1 Objective

To train logistic regression classifiers on the Iris dataset using the One-vs-Rest approach

5.2 Procedure

* 3 Classifiers were trained to respectively distinguish between *Iris setos*a and the rest, *Iris virginica* and the rest, *Iris versicolor* and the rest.
* Data was loaded from Iris.csv and no pre-processing was done apart from that required for One-vs-Rest. The data was divided, in each case, into an 80:20 ratio of training:testing data.
* The model's parameters were initialized to all 0's and trained for 10k iterations using Logistic Regression
* Hyperparameters: $\alpha = 0.1$; regularization was not used.
* The Cost was monitored after every iteration and the graph of Cost-vs-Iterations was plotted every time.

5.3 *Iris setosa* vs. Rest Classifier

5.3.1 Observed Data

From Training

Initial Cost = 66.23 Final Cost = 0.066

From Testing

Correct Examples: 30/30 (100%)
Incorrect Examples 0/30 (0%)

		True Class		
		One	Rest	Total
Predicted Class	One	10	0	10
	Rest	0	20	20
	Total	10	20	30

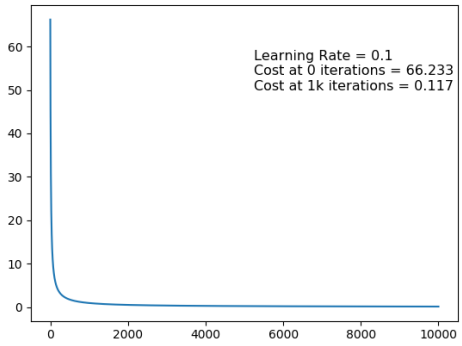

Figure 3: Graph of first 1k iterations of training

5.3.2 Conclusions

* The cost reduces sharply for the first few iterations but slowly for later iterations.
* The Cost plateaus out after a few hundred iterations, and it does not tend to zero.
* The Cost consistently falls after every iteration, although the fall is less later on.

5.4 *Iris virginica* vs. Rest Classifier

5.4.1 Observed Data

From Training

Initial Cost = 78.147; Final Cost = 8.77

From Testing

Correct Examples: 29/30 (97%)
Incorrect Examples 1/30 (3%)

		True Class		
		One	Rest	Total
Predicted Class	One	10	0	10
	Rest	1	19	20
	Total	11	19	30

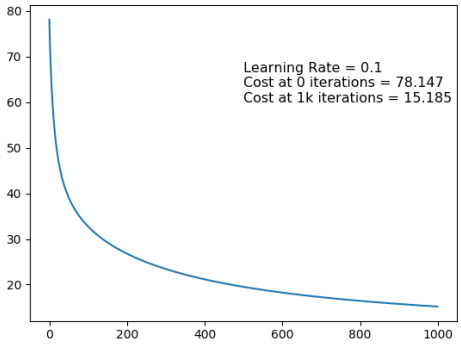

Figure 4: Graph of first 1k iterations of training

5.4.2 Conclusions

* The cost falls quickly for the first iterations, but not as quickly as for *Iris setosa*.
* The Cost plateaus out after a few hundred iterations, and it tends to a value higher than for *Iris setosa*.
* The Cost consistently falls after every iteration, although the fall is less later on.

5.5 *Iris versicolor* vs. Rest Classifier

5.5.1 Observed Data

From Training

Initial Cost = 78.278; Final Cost= 60.6

From Testing

Correct Examples: 22/30 (73%)
Incorrect Examples 8/30 (27%)

		True Class		
		One	Rest	Total
Predicted Class	One	3	7	10
	Rest	1	19	20
	Total	4	26	30

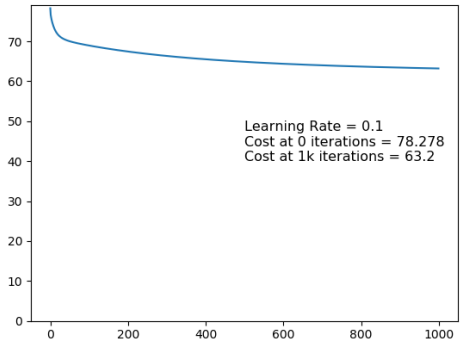

Figure 5: Graph of first 1k iterations of training

5.5.2 Conclusions

* The cost reduces appreciably only for a few starting iterations.
* The cost plateaus out far earlier than for 1. Iris setosa or 2. Iris virginica
* The total reduction in cost is minimal overall and does not tend to improve with more iterations.

6 Experiment 2

6.1 Ideation

* In Gradient Descent, the size of the update step is proportional to the magnitude of the gradient of the Cost Function, i.e. $\Delta\theta = -\alpha\nabla\theta$.
* Therefore, as the magnitude of Cost and that of it's gradient fell in the previous experiment, the size of the update step became smaller and smaller until finally, towards the end of the training, the update step size was negligible.
* How can this effect be counteracted? How to ensure that the update step size does not become negligible?

6.2 Approach

For the following experiment, the learning rate was made adjustable. The value of α was increased as the value of Cost fell. The objective was to see how far and how fast the cost would fall as compared to Experiment 1, where α was a constant.

A scaling factor(S.F.) was used in the updation of α: every time the cost fell by a factor of S.F., α would be increased by a factor of S.F.

Thus, with $\alpha_{start} = 0.1$, the following update of α was run after every iteration:

$$\alpha = \alpha_{start} \times (S.F)^{\lfloor \log(\frac{J_{init}}{J}} \rfloor \tag{5}$$

Three values of S.F were used: $S.F = \{2,\ 5,\ 10\}$

6.3 Observed Data

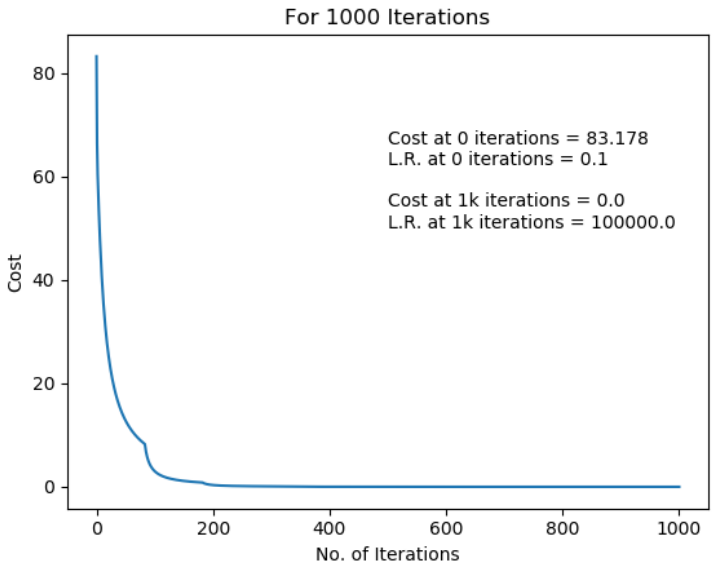

Figure 6: Falling Cost (with S.F = 10)

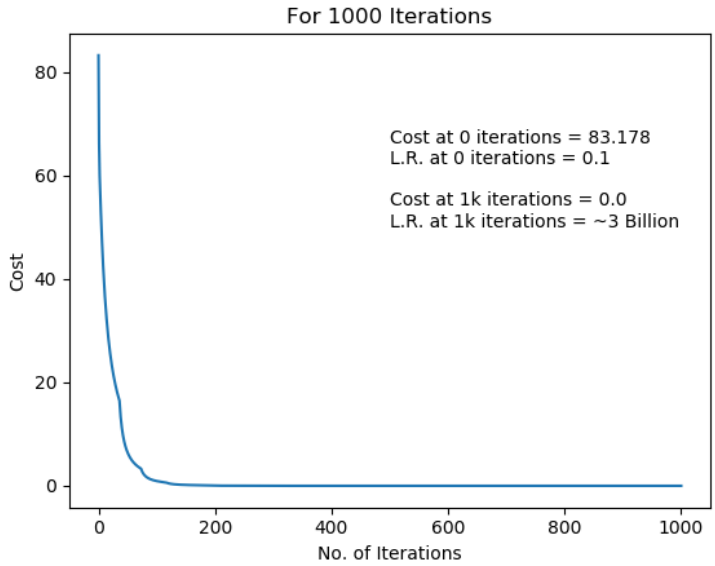

Figure 7: Falling Cost (with S.F = 5)

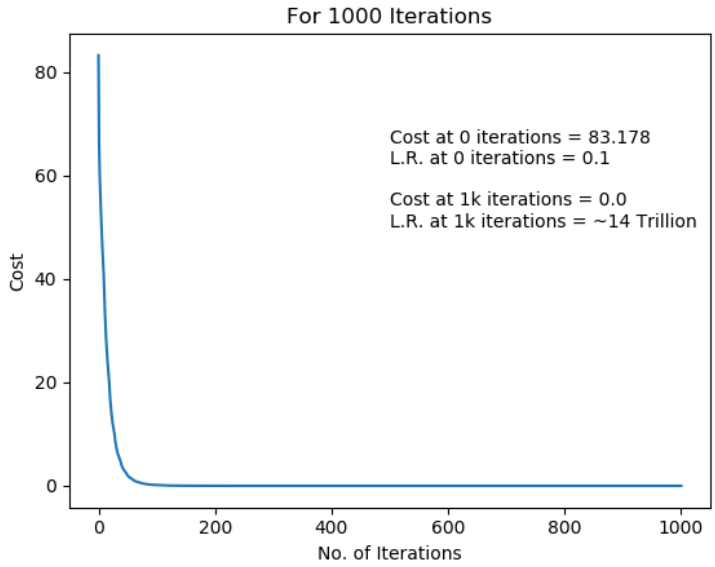

Figure 8: Falling Cost (with S.F = 2)

The following tables compare the three values of S.F with each other.

For the sake of generalization, the previous experiment can be said to have $S.F{=}\infty$

S.F	α after 100 iter.	Cost after 100 iter.
∞	0.1	6.88
10	1.0	3.002
5	2.5	1.007
2	51.2	0.142

S.F	No of iterations till Cost <0.1	No of iterations till Cost <0.01
∞	N.A	N.A
10	286	444
5	178	270
2	103	162

The graph of Cost vs No. of Iterations was plotted for the first 100 iterations to compare the standard Logistic algorithm with the 3 versions proposed here:

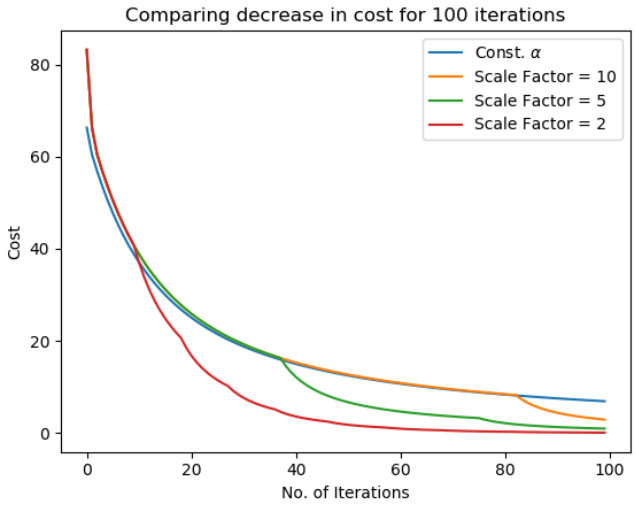

Figure 9: Iterations 1 to 100.

Since the intent behind this experiment was to negate the update step becoming negligible in the later stages, the following graph was plotted which depicts the decrease in cost for 50 update steps after 500 iterations have been completed, comparing these values for the original Logistic algorithm and the 3 others proposed here.

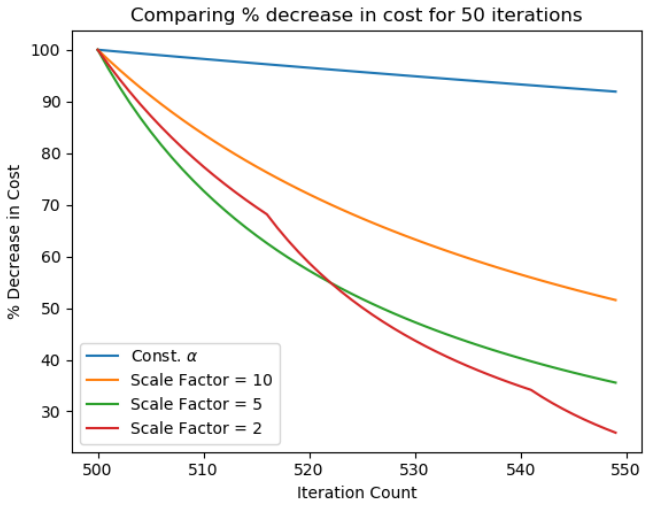

Figure 10: Iterations 50 to 550.

6.4 Conclusions

* In all cases, the cost fell faster than in Experiment 1, where α was a constant. Hence, these changes have led to an improved Logistic Regression algorithm, as was expected in the Ideation section of this experiment.

* Introducing any scaling factor leads to dramatic improvements over the standard Logistic algorithm.

* From Figure 9, the smaller the value of the Scale Factor, the better the performance of the algorithm.

* From Figure 10, there were dramatic improvements to the algorithms performance in later stages of training. Lower S.F performed better here.

****End of Logistic Regression (Part 1)****