Purpose: To calculate the probability of the correctness of the data obtained from sensors using logistic regression.

Theory: Logistic regression is a learning algorithm for classification of data, meaning by it maps the data obtained as input to it, into some class depending upon what it has learned from the training examples that were provided to it. Here I would like to focus only on the binary classification although this algorithm can be easily extended to multiclass classification with minor changes.

Scope: The scope of the document is restricted to obtaining the probability of correctness of data obtained from sensors.

What does the algorithm do: In general the algorithm does this it maps the training set into an n + 1 dimensional space for n real world features of the sensor, computes the equation of the region approximately encompassing all the positive instances. When a new instance occurs it maps this new instance into this n + 1 dimensional space. Obtains a real number corresponding to the equation of the region and maps this real number onto (0,1).

Input: The input to this algorithm is a set of training examples. Each training example consist of values corresponding to all its features at specific instance and also an additional value classifying that training example in one of the output classes(in our case whether the data was dependable or not). The features of the training examples can be any parameter related to the sensors on which the output of the data obtained from sensors depends(for example time for which the pin of ultrasonic sensors goes high). Note that if the output from sensors depends on more than one powers of a parameter (eg. X^m and X^n) then these are considered as distinct features of the training example. So in general a training example can be

represented as a vector X = $\left[\begin{matrix} X0 \\ X1 \\ . \\ . \\ Xn \end{matrix}\right]$ where *X0,X1,X2…..Xn* € R. Note that only the features *X1,X2,…..Xn* are real world features where as the feature X0 is a constant feature used to model the constant term in the equation of the region that we intend to find for these training examples. Usually X0 = 1.

Process: Given a training set the algorithm outputs a (n + 1) X 1 vector corresponding to the coefficients of the equation of the n + 1 space dimensional region. Let's say this vector is θ = $\left[\begin{matrix} \theta 0 \\ \theta 1 \\ . \\ . \\ \theta n \end{matrix}\right]$ where every θi denotes the coefficient corresponding to the feature Xi.

Now once we have this vector when we get a new reading (apart from those in the training set) we feed the value $\theta^T X$ to the sigmoid function which maps the real value obtained from $\theta^T X$ to (0,1).

So all remains to be known is how to obtain the vector θ.

In order to find out the vector θ we optimize a cost function which reduces the error of our hypothesis (output of the sigmoid function) in our training example the cost function for logistic regression is defined as follows.

Cost Function: $-(1/m)\sum(y\log(h(x)) + (1 - y)\log(1 - h(x)))$

The function might seem a bit recondite to begin with but if we dissect it into smaller parts it becomes clear.

Intuition for choosing the cost function: The cost function is a function which models the error of our hypothesis. The expected value of the training example (sensor reading good = 1 and sensor reading bad = 0) is denoted by 'y' and the hypothesis of the sigmoid function is given by $h(\theta^T X)$ for some θ and X. The cost function comprises of two terms one corresponding to when expected output y = 1 ($y\log(h(x))$)then at h(x) = 1 error should be zero and it should monotonically decrease for the range (0,1) with value at zero being infinity. It turns out that the –log(x) function perfectly serves the purpose and the other term ($(1-y)\log(1 - h(x))$) corresponds to when the expected value is 0 and similar trick of the –log function attached to model the error. The cost function sums the error for all the training examples in the training set and takes the average.

The algorithm optimizes on this cost function by varying the values of θ by using algorithms like Gradient Descent or other advanced optimization algorithms to obtain optimum θ.

- Aditya Joshi.