IB Mathematics HL Internal Assessment, 2017 May session

Applying logistic regression to university admissions data

One of the topics that is extremely important to me during the final year of school is university admissions. Since some universities, for example in Europe, have several entrance exams that are used to evaluate student's performance and admit or reject him from the university, I decided to look at how the admittance correlates with the exams results of a particular university admissions data. Being fascinated by the capabilities of machine learning and artificial intelligence, I decided to look into one of its models application in this topic. Although university decision to admit a student includes many factors, like personal statements and recommendation letters, it was decided to investigate how the percentage scores in two admittance exams correlate with the chances of being admitted.

Logistic regression is "a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome"[1] and "<…> the outcome variable in logistic regression is binary or dichotomous"[2]. In the case of initial problem, the outcome variable is the admission result of the student to the university, based on two different exam results.

1. Description of the problem

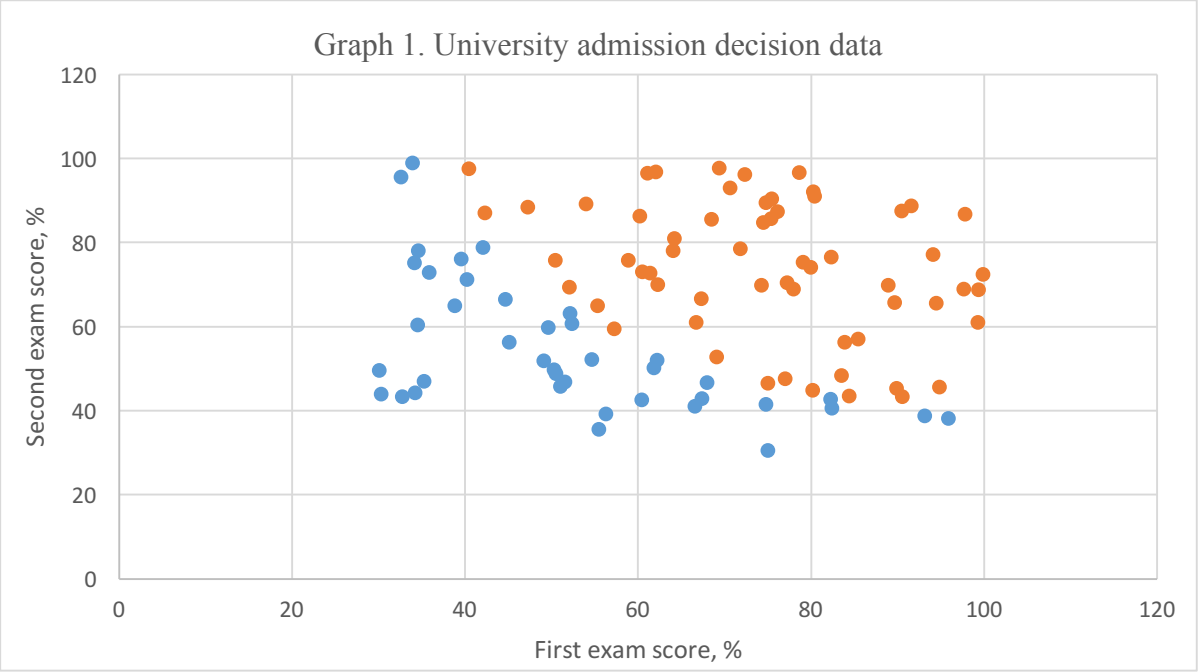
Table 1 shows several examples of the data of the student, his/her score of the first exam, score of the second exam and the final university decision, that is, whether the student is accepted or not. The data is taken from Machine Learning Course by Stanford University.[3]

[1] MedCalc Software bvba, Logistic Regression
[2] Hosmer D. W., Lemeshow S., Applied Logistic Regression. Second Edition, p. 1
[3] Ng. A., Machine Learning Course by Stanford University, 2011

Table 1. First exam score, second exam score and the university admission decision

Student number	First exam score, %	Second exam score, %	Admission decision
1	34.62	78.02	Rejected
2	30.29	43.89	Rejected
3	35.85	72.90	Rejected
4	39.54	76.04	Rejected
5	60.18	86.31	Accepted
6	79.03	75.34	Accepted
7	61.11	96.51	Accepted
…	…	…	…
100	74.78	89.53	Accepted

The data in *Table 1* contains a number p, $p = 100$, of different student scores and admission decision. The data from *Table 1* is represented in *Graph 1*. Orange dots show acceptation and blue dots show rejection.



Initially, it can be seen from the graph that the first exam score weights much more than the second exam score, since students that score high on the first exam (for example around 80%), but score only 50% on the second exam are still admitted, whereas those that score nearly 100% on second exam, but only around 35% on the first exam are not admitted.

The aim of this investigation *is to determine the function that divides the students into the accepted and rejected types (decision boundary) and use it to predict whether a new student is accepted or rejected*. Predicting a new student acceptation or rejection means that given some first exam score, x_1, and some second exam score, x_2, the probability $\boldsymbol{P(x_1,x_2)}$ that the student is accepted to the university should be found.

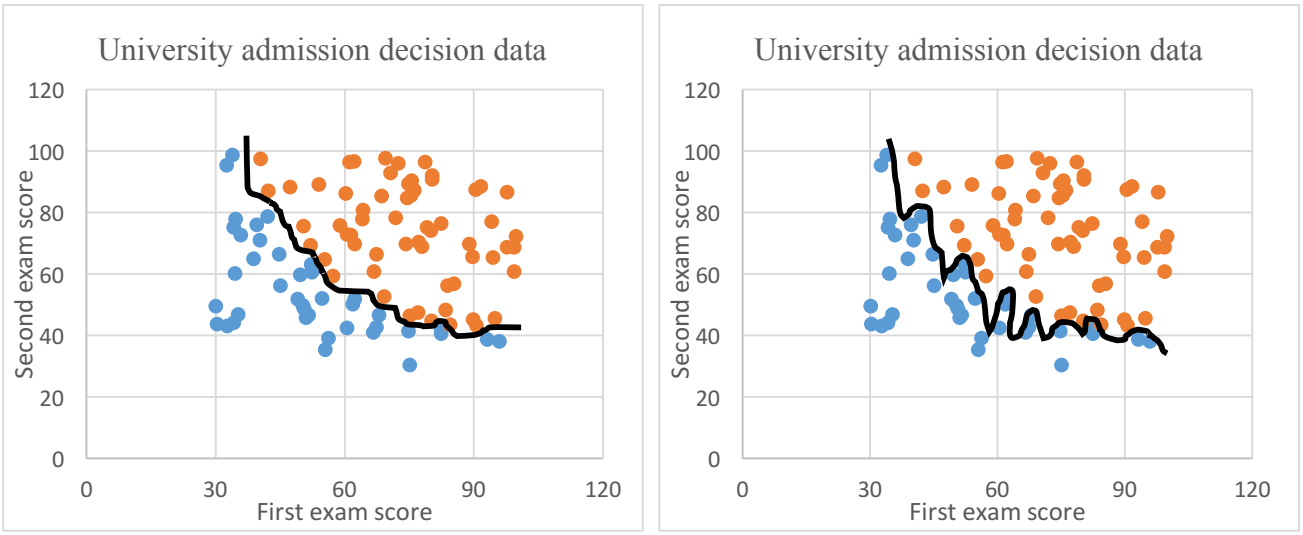
2. Hypothesis function

In this investigation, the exam scores can be noted as inputs x_0, x_1 and x_2. x_1 is the score of the first exam and x_2 is the score of the second exam. $x_1 \in [0\%, 100\%]$, $x_2 \in [0\%, 100\%]$. x_0 has no meaning and is used solely for function transformation purposes as it will be seen later. $x_0 = 1$.

y is the output, a discrete value. As the student is either accepted or rejected, $y \in [0, 1], y \in N$. In this investigation, $y = 0$ will mean the rejection, and $y = 1$ will mean being accepted.

As stated, one of the aims of this investigation is to find a *hypothesis function* (name from this[4] source), $h(x)$, that can be later used to separate the two types of students and is used to predict y using hitherto unseen x_1 and x_2. Two wild guesses of how $h(x)$ might look like in this particular case can be made from the data in *Table 1* and are plotted in *Graph 2.*

Graph 2. Possible hypothesis functions coloured in black



[4] Ng. A., CS299 Lecture notes, Stanford, p. 2

Such functions may not be the most accurate to describe the decision boundary and different approaches to finding $h(x)$ might give more accurate results. In general, it is clear that this hypothesis function can take various forms (linear, polynomial, exponential, logarithmical etc.), but for this case I decided to start off simply with the following (simple linear function):

$$h(x_0, x_1, x_2) = k_0 x_0 + k_1 x_1 + k_2 x_2$$

k_0, k_1, k_2 are coefficients (in some literature also called *weights*) that can simply transform the hypothesis function. Since $x_0 = 1$, it can be substituted with 1. Furthermore, $h(1, x_1, x_2)$ can be written as a product of matrixes:

$$h(X) = k^T X$$

k^T – a transpose of a matrix of coefficients (to simplify further matrix calculations);
X - a matrix of inputs;

$$k^T = \begin{pmatrix} k_0 \\ k_1 \\ k_2 \end{pmatrix}^T = (k_0 \quad k_1 \quad k_2);\ X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

3. Sigmoid function and mapping input as a probability

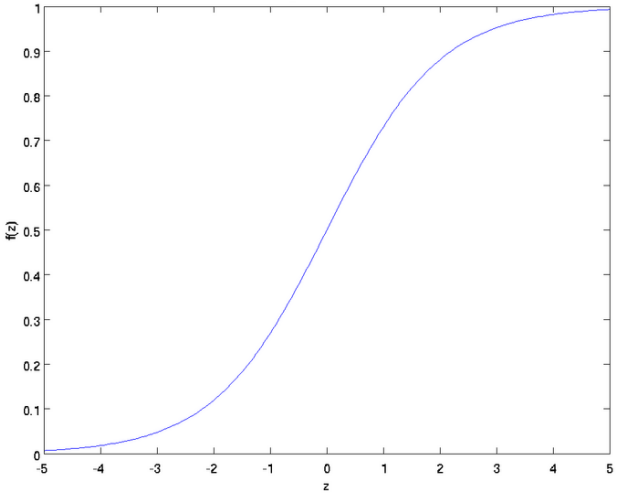
The hypothesis function $h(X)$ can produce a wide range of values given any inputs at different coefficients, yet the output *y* should be a discrete value (either 0 or 1). Therefore, Sigmoid function can be used on the hypothesis function to limit its range to $0 \leq h(X) \leq 1$ and this allows to interpret input as a probability of *y* being 0 or 1[5]. The function is written as follows:[6]

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

An important note made in *Applied Logistic Regression* book is that "many distribution functions have been proposed for use in the analysis of a dichotomous outcome variable"[7], meaning one could choose different types of logistic functions. However, for the sake of the familiarity with the function, Sigmoid function was decided to be used in this exploration. It is plotted in *Graph 3*.

[5] Rosaen K., The Sigmoid function in Logistic Regression
[6] Weisstein, Eric W. "Sigmoid Function."
[7] Hosmer D. W., Lemeshow S., Applied Logistic Regression. Second Edition, p. 6

Graph 3. Sigmoid function[8]



From the graph it can be seen that $f(z) \in (0,1); z \in R$. From its domain and range it is quite obvious to say that Sigmoid function takes any input value z and produces some value between 0 and 1. The horizontal asymptotes of this function are $\lim_{z\to\infty} f(z) = 1$ and $\lim_{z\to-\infty} f(z) = 0$. Thus, the greater the value of the input, z, the greater the probability of the output, y, being a 1, and vice-versa. When $z \geq 0$, $f(z) \geq 0.5$, which indicates that $y = 1$ (since the probability is greater than 0.5). Moreover, when $z < 0$, $f(z) < 0.5$ and the output $y = 0$.

Hence, the new hypothesis function can be now written as follows:

$$h(X) = f(k^T X) = \frac{1}{1 + e^{-k^T X}}$$

4. Error function of hypothesis function

Clearly, coefficients k_0, k_1, k_2 must be found with which the hypothesis function has the greatest accuracy and the students are classified in the most correctly way possible. This means that hypothesis function value for every student result gives the lowest error possible according to the known data.

The accuracy of the hypothesis function can be determined by the error function (in some literature called *cost function*). It has the following expression[9]:

[8] Plot of the Sigmoid function, UFLDL Tutorial
[9] Ng. A., Machine Learning Course by Stanford University, 2011

$$J(k) = \frac{\sum_{i=1}^{p} Error(f(k^T X^i)),\ y_i)}{p}$$

$f(k^T X^i)$ – hypothesis function for *i*th student;

y_i – the output of the *i*th student.

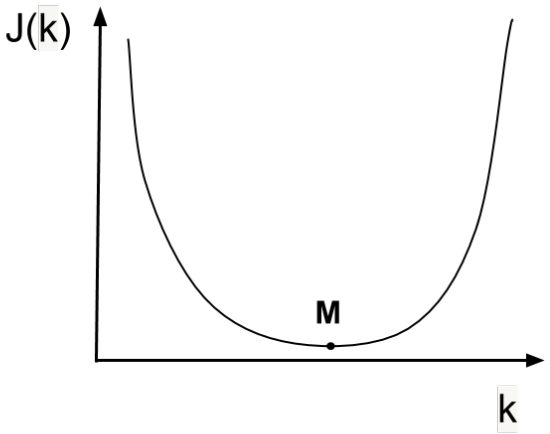
$$X^i = \begin{pmatrix} 1 \\ x_1^i \\ x_2^i \end{pmatrix}$$

x_1^i, x_2^i are the inputs, in this case exam scores, of the *i*th student. For example, from data in *Table 1*, it can be seen that for the student $i = 1$, $x_1^1 = 34.62\%$ and $x_2^1 = 78.02\%$

The $Error\left(f(k^T X^i)\right)$ term defines the error for the *i*th student that the hypothesis function makes, given its output is y_i.[10] For example, if $h(X^1) = f(k^T X^1) = 0.7$, but the output $y_1 = 0$, the $Error(f(k^T X^1),\ y_1)$ term will display some value that corresponds to this "error" between the expected and the real value. Thus, a conclusion can be made that $J(k)$ defines the mean of individual errors (the sum of individual errors for every student, $Error(f(k^T X^i),\ y_i)$, is divided by the total number of students, *p*).

Graph 4 shows typical $J(k)$ function graph:[11]

Graph 4. $J(k)$ function



From the graph, it can be seen that $J(k)$ must have one global minimum at which $\frac{dJ(k)}{dk} = 0$ and the point *M* is a vertex. At point *M*, $\frac{dJ(k)}{dk} = 0$ and $J(k)$ has the lowest value, hence hypothesis function $h(X)$ is the most accurate. It follows that it is important to define the $Error(f(k^T X^i),\ y_i)$ term, so

[10] Ng. A., Machine Learning Course by Stanford University, 2011
[11] Ng. A., Machine Learning Course by Stanford University, 2011

that $J(k)$ has this shape, since it only has one point at which the value of it is lowest. Although there may be several types of $J(k)$ that would give a function of this form, $J(k)$ can be of this form when the term $Error(f(k^T X^i),\ y_i)$ is of the following form[12]:

$$Error(f(k^T X^i),\ y_i) = y_i \times \log(f(k^T X^i)) - (1 - y_i) \times \log(1 - f(k^T X^i))$$

$J(k)$ can be then written as follows:

$$J(k) = -\frac{\sum_{i=1}^{p}(y_i \times \log(f(k^T X^i)) + (1 - y_i) \times \log(1 - f(k^T X^i)))}{p}$$

The following matrixes can be defined to transform $J(k)$ into a simpler form:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{100} \end{pmatrix};\ X_n^i = (X^1 \quad X^2 \quad \dots \quad X^{100}) = \begin{pmatrix} 1 & 1 & & 1 \\ x_1^1 & x_1^2 & \dots & x_1^{100} \\ x_2^1 & x_2^2 & & x_2^{100} \end{pmatrix}$$

n refers to the input x_1 or x_2. For example, $X_2^1 = 78.02\%$. $J(k)$ can be therefore written as follows using matrix multiplication:

$$k^T X_n^i = (k_0 \quad k_1 \quad k_2) \times \begin{pmatrix} 1 & 1 & & 1 \\ x_1^1 & x_1^2 & \dots & x_1^{100} \\ x_2^1 & x_2^2 & & x_2^{100} \end{pmatrix} = (k^T X^1 \quad k^T X^2 \quad \dots \quad k^T X^{100})$$

$$\log(1 - f(k^T X_n^i)) = \log((1 \quad 1 \quad \cdots \quad 1) - (f(k^T X^1) \quad f(k^T X^2) \quad \cdots \quad f(k^T X^{100}))) =$$

$$= (\log(1 - f(k^T X^1)) \quad \log(1 - f(k^T X^2)) \quad \cdots \quad \log(1 - f(k^T X^{100})))$$

$$1 - Y = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{100} \end{pmatrix} = \begin{pmatrix} 1 - y_1 \\ 1 - y_2 \\ \vdots \\ 1 - y_{100} \end{pmatrix}$$

$$\log(f(k^T X_n^i)) \times Y = \log(f(k^T X^1 \quad k^T X^2 \quad \dots \quad k^T X^{100})) \times \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{100} \end{pmatrix} =$$

$$= \log(f(k^T X^1))y_1 + \log(f(k^T X^2))y_2 + \cdots + \log(f(k^T X^{100}))y_{100}$$

$$J(k) = -\frac{1}{100}\left(\log\left(f(k^T X_n^i)\right) \times Y + \log(1 - f(k^T X_n^i)) \times (1 - Y)\right)$$

[12] Ng. A., CS299 Lecture notes, Stanford, p. 18

If $f\left(k^T X_n^i\right) = h$, then this can be rewritten (to make it look less complex for the following derivations):

$$J(k) = -\frac{1}{100}(\log h \times Y + \log(1-h) \times (1-Y))$$

5. Gradient decent

Gradient decent technique is used to for updating coefficients *k* using these formulas until $J(k)$ reaches its minimum value[13,14]:

$$k_{new\,0} = k_0 - \alpha \frac{\partial J(k)}{\partial k_0}$$

$$k_{new\,1} = k_1 - \alpha \frac{\partial J(k)}{\partial k_1}$$

$$k_{new\,2} = k_2 - \alpha \frac{\partial J(k)}{\partial k_2}$$

$k_{new0}, k_{new1}, k_{new2}$ are updated coefficients of hypothesis function.

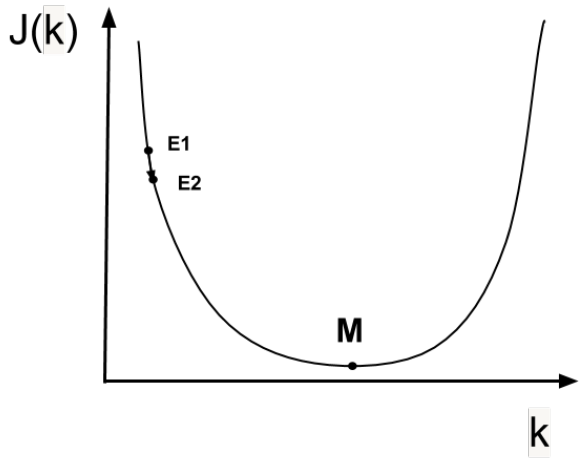
$\frac{\partial J(k)}{\partial k_0}, \frac{\partial J(k)}{\partial k_1}, \frac{\partial J(k)}{\partial k_2}$ are the partial derivatives of the cost function with respect to different coefficients. Partial derivative is a "derivative of a function of multiple variables when all but the variable of interest are held fixed during the differentiation"[15]. Basically, it follows that every partial derivative term minimizes the error of the hypothesis function with respect to every possible coefficient, meaning that the coefficients becomes more and more accurate.

α is the learning rate (set manually);

Using basic intuition, $\frac{\partial J(k)}{\partial k_n}$ calculates the slope of $J(k)$ with the respect to the coefficient k_n and is then multiplied by a manually selected learning rate α, which is just some value. Thus, the product can be thought of as a "step" that must be taken from any point in $J(k)$ towards the point M to minimize $J(k)$.

[13] Ng. A., CS299 Lecture notes, Stanford, p. 4
[14] Magdon-Ismail M., Learning From Data. Lecture 9. Logistic Regression and Gradient Descent
[15] Weisstein, Eric W. "Partial Derivative."

Graph 5. Gradient decent "step" on a cost function[16]



The "step" is shown as the movement from the cost function from point *E1* to *E2*, which is closer to *M*. Its value is described by the product of partial derivative term and α. The greater α, the greater the value. Thus, the value of this "step" is subtracted from the current value of k_n and the result is the new value $k_{new\,n}$, which would give a lower value of $J(k)$, meaning that the point is closer to the global minimum and, thus, $h(X)$ would have higher accuracy. The new coefficients *k* are then updated at the same time after every $k_{new\,n}$ is calculated and the process is repeated until $\frac{\partial J(k)}{\partial k_n} = 0$ and the point M is reached (since at this point $k_{new\,n} = k_n$). To use gradient decent, $\frac{\partial J(k)}{\partial k_0}$ must be differentiated:

$$\mathbf{1)}\ \frac{\partial J(k)}{\partial k_0} = = -\frac{1}{100}\left(\frac{\partial \log(h) Y}{\partial k_0} + \frac{\partial \log(1-h)(1-Y)}{\partial k_0}\right)$$

$$\mathbf{2)}\ \frac{\partial \log(h) Y}{\partial k_0} = \frac{\partial \log(h)}{\partial k_0} Y + \frac{\partial Y}{\partial k_0}\log(h) = \frac{\partial \log(h)}{\partial k_0} Y = \frac{1}{h}\frac{\partial h}{\partial k_0} Y$$

$$\mathbf{3)}\ \frac{\partial h}{\partial k_0} = \frac{\partial f\left(k^T X_n^i\right)}{\partial k_0} = \frac{-\frac{\partial\left(1+e^{-k^T X_n^i}\right)}{\partial k_0}}{\left(1+e^{-k^T X_n^i}\right)^2} = \frac{\frac{\partial\left(k^T X^0 \quad k^T X^1 \quad \cdots \quad k^T X^{100}\right)}{\partial k_0} e^{-k^T X_n^i}}{\left(1+e^{-k^T X_n^i}\right)^2} =$$

$$= \frac{\left(x_0^1 \quad x_0^2 \quad \cdots \quad x_0^{100}\right) e^{-k^T X_n^i}}{(1+e^{-k^T X_n^i})^2} = \frac{X_0^i e^{-k^T X_n^i}}{(1+e^{-k^T X_n^i})^2}$$

$$\mathbf{4)}\ \frac{\partial \log(h) Y}{\partial k_0} = \frac{1}{h}\frac{\partial h}{\partial k_0} Y = (1+\ e^{-k^T X_n^i})\frac{X_0^i e^{-k^T X_n^i}}{(1+e^{-k^T X_n^i})^2} Y = \frac{X_0^i e^{-k^T X_n^i}}{(1+e^{-k^T X_n^i})} Y$$

$$\mathbf{5)}\ \frac{\partial \log(1-h)(1-Y)}{\partial k_0} = \frac{\partial \log(1-h)}{\partial k_0}(1-Y) + \frac{\partial(1-Y)}{\partial k_0}\log(1-h) = -\frac{1}{1-h}\frac{\partial h}{\partial k_0}(1-Y) =$$

[16] Ng. A., Machine Learning Course by Stanford University, 2011

$$= -\frac{1}{1-h}\frac{X_0^i e^{-k^T X_n^i}}{\left(1+e^{-k^T X_n^i}\right)^2}(1-Y) = -\frac{1+e^{-k^T X_n^i}}{e^{-k^T X_n^i}}\frac{X_0^i e^{-k^T X_n^i}}{\left(1+e^{-k^T X_n^i}\right)^2}(1-Y) =$$

$$= -\frac{X_0^i}{\left(1+e^{-k^T X_n^i}\right)}(1-Y)$$

$$\textbf{6)}\ \frac{\partial J(k)}{\partial k_0} = -\frac{1}{100}\left(\frac{X_0 e^{-k^T X_n^i}}{\left(1+e^{-k^T X_n^i}\right)}Y - \frac{X_0^i}{\left(1+e^{-k^T X_n^i}\right)}(1-Y)\right) =$$

$$= -\frac{1}{100}\left(\frac{X_0^i e^{-k^T X_n^i}}{\left(1+e^{-k^T X_n^i}\right)}Y - \frac{X_0^i}{\left(1+e^{-k^T X_n^i}\right)} + \frac{X_0^i}{\left(1+e^{-k^T X_n^i}\right)}Y)\right) =$$

$$= -\frac{1}{100}X_0^i\left(Y\frac{e^{-k^T X_n^i}+1}{\left(1+e^{-k^T X_n^i}\right)} - \frac{1}{\left(1+e^{-k^T X_n^i}\right)}\right) = \frac{1}{100}\left(h\left(X_n^i\right) - Y^T\right)X_0^i$$

The following shows the technical correctness regarding matrixes of the $\frac{\partial J(k)}{\partial k_0}$ formula:

$$h\left(X_n^i\right) - Y^T = \begin{pmatrix} f(k^T X^1) & f(k^T X^2) & \cdots & f(k^T X^{100}) \end{pmatrix} - \begin{pmatrix} y_1 & y_2 & \cdots & y_{100} \end{pmatrix}$$

$$\left(h\left(X_n^i\right) - Y^T\right)\times X_0^i = \begin{pmatrix} f(k^T X^1) - y_1 & f(k^T X^2) - y_2 & \cdots & f(k^T X^{100}) - y_{100} \end{pmatrix}\times\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} =$$

$$= f(k^T X^1) - y_1)x_0^1 + (f(k^T X^2) - y_2)x_0^2 + \cdots + (f(k^T X^{100}) - y_{100}$$

Thus, the result is indeed a single value that is then adjusted using learning rate. Moreover, when $\frac{\partial J(k)}{\partial k_1}$ is calculated, only part **4)** will differ, since $\frac{\partial h}{\partial k_1} = \frac{X_1^i e^{-k^T X_n^i}}{(1+e^{-k^T X_n^i})^2}$. Thus, for k_1 and k_2:

$$\frac{\partial J(k)}{\partial k_1} = \frac{1}{100}\left(h\left(X_n^i\right) - Y^T\right)X_1^i$$

$$\frac{\partial J(k)}{\partial k_2} = \frac{1}{100}\left(h\left(X_n^i\right) - Y^T\right)X_2^i$$

6. Optimising hypothesis function coefficients

An example of manual calculation of optimisation of hypothesis function coefficients is shown. Firstly, the initialisation of value of the coefficients is needed (for example with range of values of $-20 \leq k_n \leq 20$ because the coefficients for such linear function in this specific data set should not, logically thinking, have values around this order of magnitude). Thus, coefficients are generated using random number generator: $k_0 = 6.353$, $k_1 = -17.781$, $k_2 = -2.743$. $\alpha = 1$ for test purposes.

Starting with the data of the first student:

$$X^1 = \begin{pmatrix} 1 \\ 34.62 \\ 78.02 \end{pmatrix}; h(k^T X^1) = \frac{1}{1 + e^{-(6.353 - 17.781 * 34.62 - 2.743 * 78.02)}} = \frac{1}{1 + e^{823.23}} = \frac{1}{3.34 \times 10^{357}} \approx 0$$

$$k_{new\,0} = k_0 - \alpha \frac{\partial J(k)}{\partial k_0} = k_0 - \alpha \frac{1}{100} \left(h\left(X_n^i\right) - Y^T\right) X_0^i =$$

$$= 6.353 - (1 * 0.01 * ((f(k^T X^1) - y_1) x_0^1 + (f(k^T X^2) - y_2) x_0^2 + \cdots + (f(k^T X^{100}) - y_{100}) x_0^{100}))$$

$k_{new\,1}$ and $k_{new\,2}$ are then calculated using the same principle;

$$k_0 = k_{new\,0}; k_1 = k_{new\,1}; k_2 = k_{new\,2}$$

Clearly, the numbers (for example the value of $h(k^T X^1)$ are too small to deal with manually and only computers can do it. Thus, to find the most optimal coefficients for the hypothesis function, a program must be written, since the calculations are also repeated 1,000,000 times and include heavy calculations, meaning it is not practically possible to manually calculate all the coefficients. The whole code was written by me in *Java* programming language, and the snippet of the code is shown in *figure 1*. The code has 128 lines in total and is executed for several minutes before finishing.

Figure 1. Snippet of Java code for optimizing the parameters written by me

```
//Random generator
static double randomWithRange(double min, double max) {
    double range = (max - min) + 1;
    return (Math.random() * range) + min;
}

//Sigmoid
static double Sigmoid(double x) {
    x = (1 / (1 + Math.pow(Math.E, (-1 * x))));
    return x;
}

// Calculate hypothesis
static double[] calculateH(double[][] data, double[] weights) {

    double[] h = new double[100];

    //Make all hypothesis functions equal to 0
    for(int i = 0; i < h.length; i++) {
        h[i] = 0;
    }

    for (int i = 0; i < 100; i++) {
        for (int y = 0; y < 3; y++) {
            h[i] = h[i] + data[i][y] * weights[y];
        }
        h[i] = h[i] * 0.01;
    }

    return h;
}
```

```
// Calculate cost of the algorithm
static double calculateCost(double[] h, double[][] data) {
    double sum = 0;

    for (int i = 0; i < h.length; i++) {
        sum = sum + (-1 * data[i][3] * Math.log10(h[i]))
                - ((1 - data[i][3]) * Math.log10(1 - h[i]));
    }

    sum = 0.01 * sum;

    return sum;
}

// Calculates gradient and updates the current weights
static double[] calculateGrad(double[] h, double[][] data, double[] weights, double alpha) {
    double[] sum = {0, 0, 0};

    for (int y = 0; y < weights.length; y++) {
        for (int i = 0; i < h.length; i++) {
            sum[y] = sum[y] + ((h[i] - data[i][3]) * data[i][y]);
        }
        sum[y] = 0.01 * sum[y];
    }

    for (int i = 0; i < weights.length; i++) {
        weights[i] = weights[i] - (alpha * sum[i]);
    }

    return weights;
}
```

After *1,000,000* iterations of optimization:

$$\text{J(k)}: 0.08870759;\ k_0 = -22.988, k_1 = 0.1889, k_2 = 0.1839$$

It was noticed that the error is already minimal and cannot be effectively minimized more, since the steps that the gradient decent takes becomes decreasingly small as $\frac{\partial J(k)}{\partial k_n}$ becomes infinitely small. Moreover, the line may not be the best optimized to get the most accurate hypothesis function and $\text{J(k)} \neq 0$. Thus, the optimized hypothesis function can be written as follows:

$$h(1, x_1, x_2) = f(-22.9 + 0.19x_1 + 0.18x_2)$$

As it was presented in *chapter 5*, when $k^T X \geq 0, y = 1$. Thus, to plot the graph, the hypothesis function can be defined as a decision boundary at this specific point when $f(k^T X) = 0.5$

$$0.5 = \frac{1}{1 + e^{22.9 - 0.19x_1 - 0.18x_2}}$$

$$e^{22.9 - 0.19x_1 - 0.18x_2} = 1$$

$$0 = 22.9 - 0.19x_1 - 0.18x_2$$

$$\boldsymbol{x_2 = -1.06x_1 + 127}$$

Another hypothesis function (logarithmic) was decided to be tested, since it may fit the graph better. Thus:

$$h_{lg}(1, x_1, x_2) = f(k_0 + k_1 \log(x_1) + k_2 x_2)$$

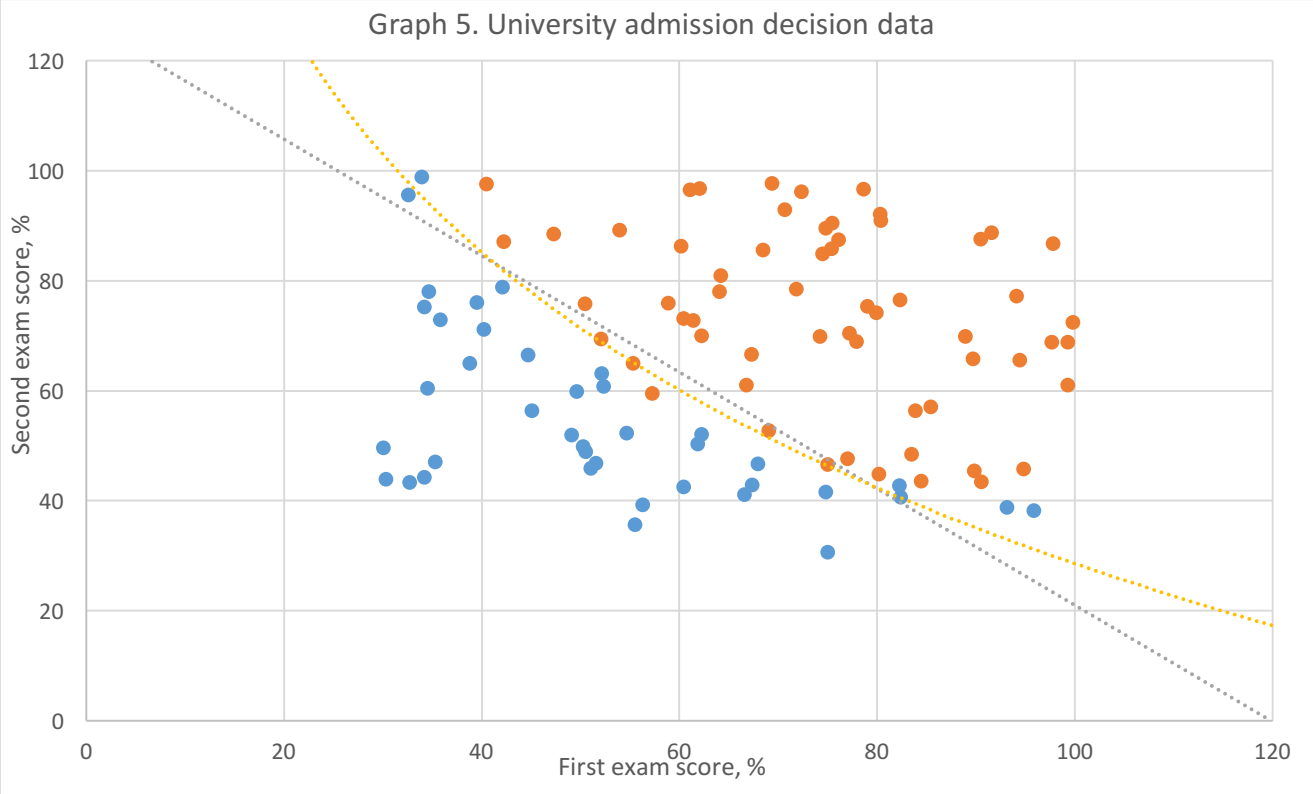
After the optimization with the same formulas were used, the code gave the following results:

$$\mathrm{J(k)}_{logarithmic}: 0.08725983;\ k_0 = -38.4953, k_1 = 17.4928, k_2 = 0.1229$$

$$h_{lg}(1, x_1, x_2) = f(-38.5 + 17.5\log(x_1) + 0.12x_2)$$

$$\boldsymbol{x_2 = -145.8 \log(x_1) + 320.8}$$

Therefore, the two following decision boundaries are shown in *graph 5*:



The grey line shows linear hypothesis function $h(X)$, whereas the yellow line shows the logarithmic hypothesis function $h_{lg}(X)$. Logarithmic hypothesis function is more accurate, since its error, $\mathrm{J(k)}_{logarithmic}$, is smaller than that of linear, J(k) (even though only by a margin).

7. Testing new examples

Let's assume that a new student that wants to be admitted to the university scored 80% from the first exam and 42% from the second exam. $x_0 = 1, x_1 = 80, x_2 = 42$. This point is picked particularly because it is hard to initially assume whether the student is accepted without knowing the exact decision boundary. Thus, the question is whether $y = 0$ or $y = 1$ with these inputs?

$$h(1, 80, 42) = f(-22.9 + 0.19\times80 + 0.18\times42) = \frac{1}{1 + e^{0.14}} = 0.47 \rightarrow y = 0$$

$$h_{lg}(1, 80, 42) = f(-38.5 + 17.5\times \log(80) + 0.12\times42) = \frac{1}{1 + e^{0.156}} = 0.46 \rightarrow y = 0$$

Thus, the student with such exam results wouldn't be admitted to the university regarding both hypothesis functions, yet if we take another hypothetical point ($x_0 = 1, x_1 = 35, x_2 = 92$):

$$h(1, 35, 71) = f(-22.9 + 0.19\times35 + 0.18\times92) = \frac{1}{1 + e^{-0.31}} = 0.58 \rightarrow y = 1\ (accepted)$$

$$h_{lg}(1, 35, 71) = f(-38.5 + 17.5\times \log(35) + 0.12\times92) = \frac{1}{1 + e^{0.44}} = 0.39 \rightarrow y = 0\ (rejected)$$

Point ($x_0 = 1, x_1 = 60, x_2 = 62$):

$$h(1, 60, 62) = f(-22.9 + 0.19\times60 + 0.18\times62) = \frac{1}{1 + e^{-0.31}} = 0.42 \rightarrow y = 0\ (rejected)$$

$$h_{lg}(1, 60, 62) = f(-38.5 + 17.5\times \log(60) + 0.12\times62) = \frac{1}{1 + e^{0.44}} = 0.51 \rightarrow y = 1\ (accepted)$$

Clearly, the results the two functions produce can differ, and one can classify student as accepted, whereas the other as rejected. Moreover, it can be seen in the graph that the two functions will produce the very same probability of the student being accepted or rejected at two particular points where the two functions cross each other. **Using GDC:** coordinates of such point 1 is $(46.42, 77.79)$ and point 2 is $(75.39, 47.09)$. Therefore, at these points both functions will produce the same probability. The output of two functions mostly differs in range $x_1 \in [0; 46.42) \cup (75.39; 100]$, and the output gets closer in range $x_1 \in (46.42; 75.39)$. Yet from the graph it can be seen that not all existing data points are classified correctly. For example, point $(57.24, 59.51)$ would be classified as rejected by both functions, even though it is actually accepted. In general, logarithmic one should be trusted, since, as it was mentioned, $\mathrm{J(k)}_{logarithmic}$ has lower value than linear (at least by a margin).

8. Evaluation

The investigation was carried out successfully and showed that there are several hypothesis function that accurately fits the data and separates accepted and rejected students' data (linear and logarithmic):

$$h(1, x_1, x_2) = f(-22.9 + 0.19x_1 + 0.18x_2)$$

$$h_{lg}(1, x_1, x_2) = f(-38.5 + 17.5\log(x_1) + 0.12x_2)$$

Moreover, it was shown that a new example with unseen inputs can be successfully classified using this function to predict the output; however, the results may differ. In the future, other types of functions could be also used to classify the data (exponential or polynomial), which may have better accuracy.

In general, this model fits well for most types of real-life data that can be classified, since the model can be modified accordingly to the needs and works abstractly. For example, if another entrance exam would be included in the admissions, then the hypothesis function would have another input x_3 and the coefficient k_3, yet the principles and method derived in this investigation would stay the same. New coefficients would be needed to be calculated according to the same formulas derived in this investigation. The decision boundary can be also easily changed to a polynomial or any other type of function according to the given data.

While carrying out the investigation, I showed my personal engagement by learning the concept of partial derivative and its usage, and matrixes, which are out of the syllabus material. Doing the exploration gave me a better insight of the fundamental principles of logistic regression model. This knowledge can be further applied in building neural networks and learning other machine learning models, for example support vector machines that are used to solve similar and more advanced problems using advanced methods.

Bibliography:

- Ng. A., Machine Learning Course by Stanford University, 2011, *Accessed: 2016/01/20*
  Source: https://www.coursera.org/learn/machine-learning
  Source: http://www.holehouse.org/mlclass/06_Logistic_Regression.html
- Ng. A., CS299 Lecture notes, Stanford, p. 2, 4, 18, *Accessed: 2016/01/20*
  Source: http://cs229.stanford.edu/notes/cs229-notes1.pdf
- Plot of the sigmoid function, UFLDL Tutorial, February 2011, *Accessed: 2016/01/27*

Source: http://ufldl.stanford.edu/wiki/index.php/File:Sigmoid_Function.png

- Weisstein, Eric W. "Sigmoid Function." from *MathWorld*--A Wolfram Web Resource, *Accessed: 2016/01/27*

  Source: http://mathworld.wolfram.com/SigmoidFunction.html
- Weisstein, Eric W. "Partial Derivative." From *MathWorld*--A Wolfram Web Resource. *Accessed: 2016/01/27*

  Source: http://mathworld.wolfram.com/PartialDerivative.html
- MedCalc Software bvba, Logistic Regression, *Accessed: 2016/02/15*

  Source: https://www.medcalc.org/manual/logistic_regression.php
- Rosaen K., The Sigmoid function in Logistic Regression, *Accessed: 2016/04/02*

  Source: http://karlrosaen.com/ml/notebooks/logistic-regression-why-sigmoid/
- Hosmer D. W., Lemeshow S., Applied Logistic Regression. Second Edition (2000), p. 1, 6