INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS QUERÉTARO

PROJECT: LOGISTIC REGRESSION WITH CUDA

SEBASTIÁN GONZÁLEZ TAFOYA
A01233416

CONTENTS

1. Introduction 3
2. Logistic regression 3
3. Machine learning and CUDA 5
4. The CUDA paradigm: Parallel Computing 6
5. Solution
6. Results
7. Conclusion

INTRODUCTION

This project implements logistic regression regression solver with CUDA. The solution is able to take any csv dataset with a label and multiple feature columns, perform normalization on the feature columns and apply multiple logistic regression to fit the Y variable.

LOGISTIC REGRESSION

Logistic regression is a classification algorithm that seeks to model the probability of an event occurring depending on the values of the independent variables, which can be categorical or numerical. It is used to predict a binary outcome based on a set of independent variables.

A binary outcome is one where there are only two possible scenarios—either the event happens or it does not happen. Independent variables are those variables or factors which may influence the outcome (or dependent variable). However, the independent variables can be continuous, discrete ordinal (1, 2, 3) or discrete nominal (blue, brown, green).

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the Sigmoid function, was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform.

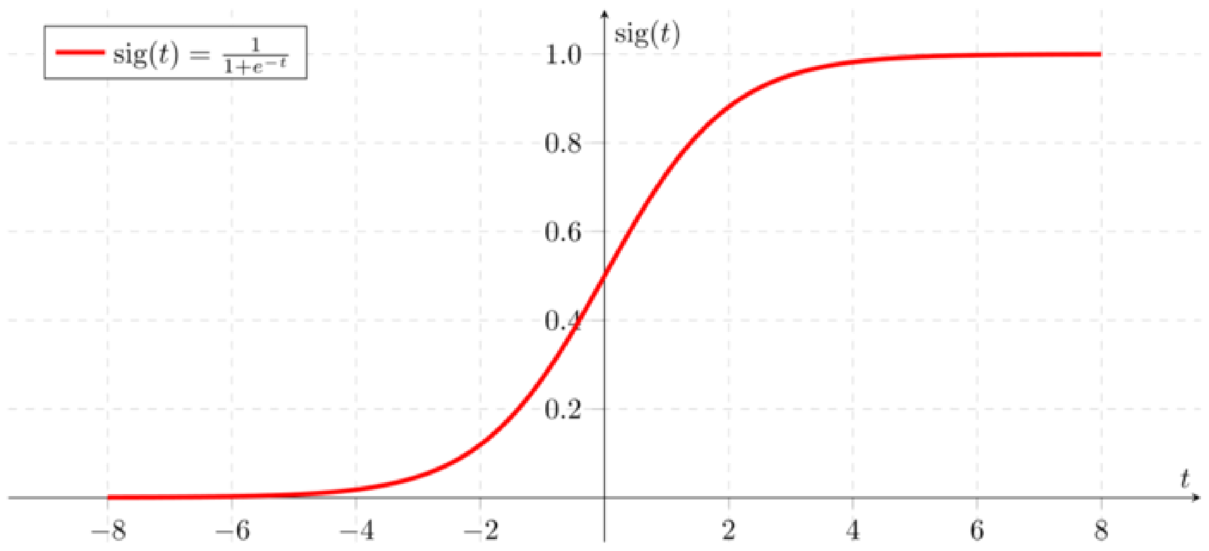


Figure 1: Sigmoid Activation Function

In the case of multiple logistic regression, numerous variables can be considered in the formula:

$$P(y(i)=1)=\frac{1}{1+\exp(-(\beta 0+\beta 1\,x(i)\,1+\ldots+\beta\,p\,x(i)\,p))}$$

To predict which class a data belongs to, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.
Say, if predicted_value ≥ 0.5, then classify a tumor as malignant else as not malignant.

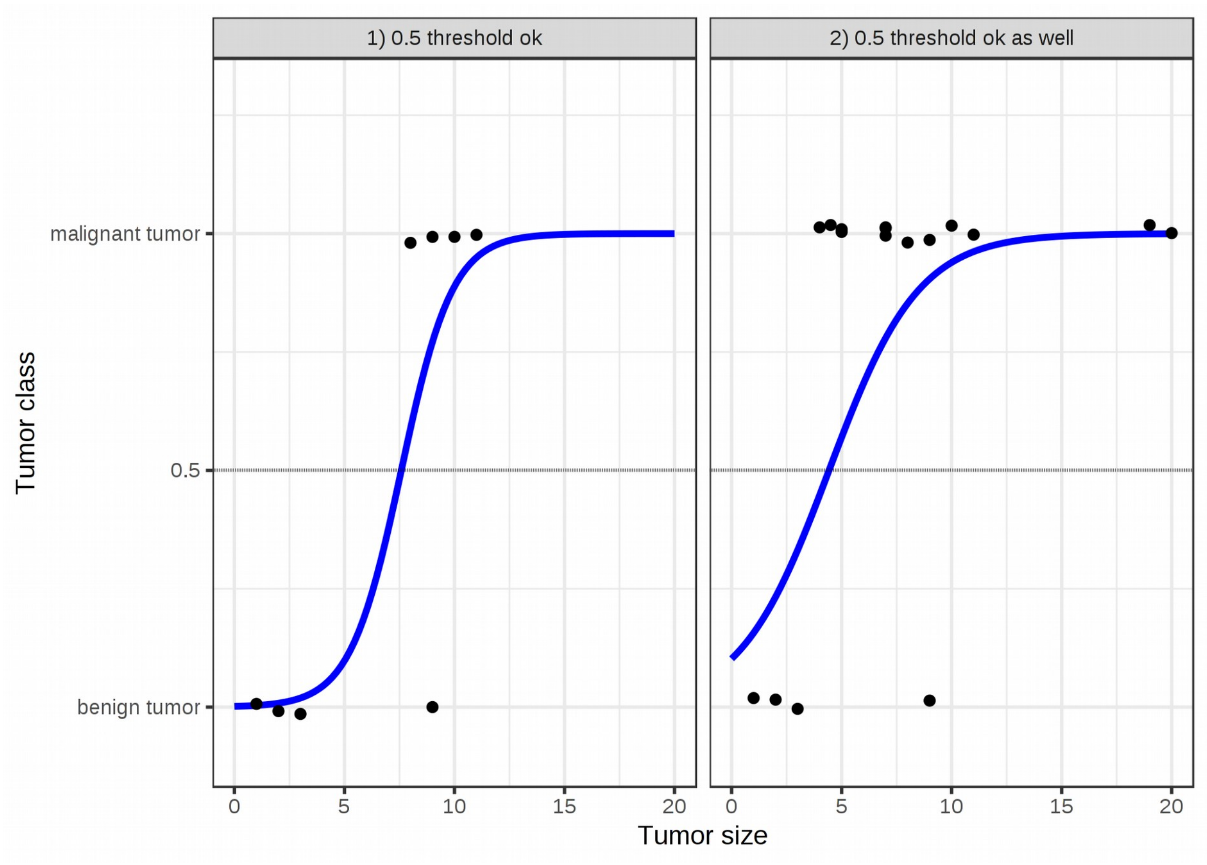


Figure 2: The logistic regression model finds the correct decision boundary between malignant and benign depending on tumor size. The line is the logistic function shifted and squeezed to fit the data.

MACHINE LEARNING AND CUDA

The training phase of machine learning algorithms can become very resource intensive, because there are very large and repetitive multiplications of matrices involved.

Fortunately, these types of operations can be described as “embarrassingly parallel”, meaning that little or no effort is needed to separate the overall task into a set of smaller tasks to be computed in parallel. Tasks that are embarrassingly parallel are ones where it's easy to see that the set of smaller tasks are independent with respect to each other.

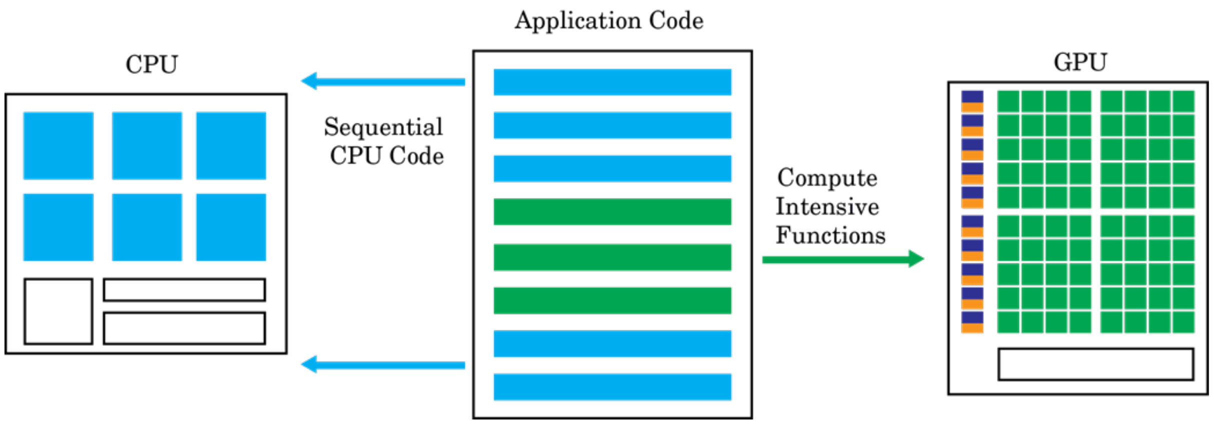
This is why machine learning tends to be a perfect fit for GPUs. This is where CUDA comes into the picture. Nvidia is a technology company that designs GPUs, and they have created CUDA as a software platform that pairs with their GPU hardware making it easier for developers to build software that accelerates computations using the parallel processing power of Nvidia GPUs.

An Nvidia GPU is the hardware that enables parallel computations, while CUDA is a software layer that provides an API for developers.

THE CUDA PARADIGM: PARALLEL COMPUTING

CUDA is a minimal extension of the C and C++ programming languages. The programmer writes a serial program that calls parallel kernels, which may be simple functions or full programs. A kernel executes in parallel across a set of parallel threads. The programmer organizes these threads into a hierarchy of grids of thread blocks. A thread block is a set of concurrent threads that can cooperate among themselves through barrier synchronization and shared access to a memory space private to the block. A grid is a set of thread blocks that may each be executed independently and thus may execute in parallel.

When invoking a kernel, the programmer specifies the number of threads per block and the number of blocks making up the grid. Each thread is given a unique thread ID number within its thread block, and each thread block is given a unique block ID number within its grid. CUDA supports thread blocks containing up to 512 threads. For convenience, thread blocks and grids may have one, two, or three dimensions, accessed via .x, .y, and .z index fields.



LOGISTIC REGRESSION WITH PARALLEL PROGRAMMING

In order to take advantage of the benefits of parallel programming with CUDA, the matrix operations that take place in logistic regression have to be vectorized and splitted betweens multiple subtasks carried out by different threads.

The following diagram shows the process that this solution implements:

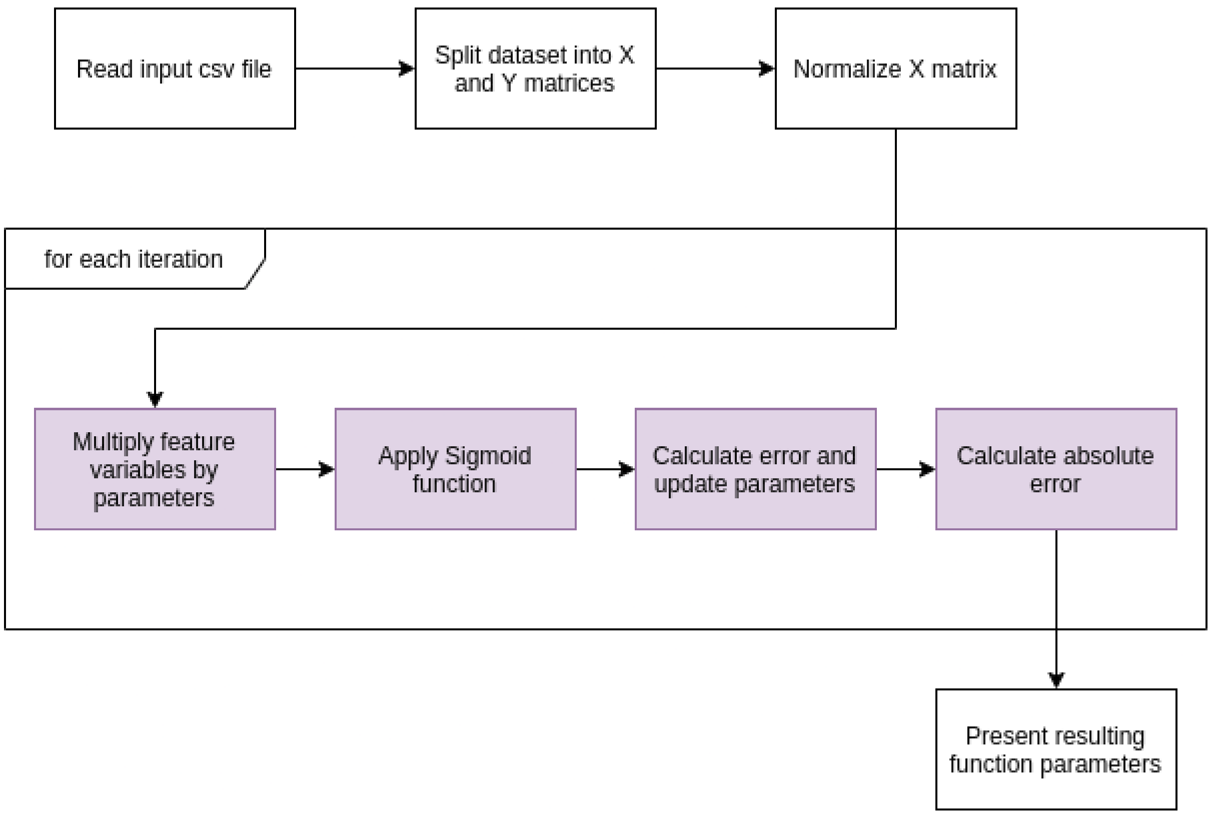

Read input csv file
Split dataset into X and Y matrices
Normalize X matrix
for each iteration
Multiply feature variables by parameters
Apply Sigmoid function
Calculate error and update parameters
Calculate absolute error
Present resulting function parameters

Figure 3: CUDA logistic regression flow diagram.

The steps highlighted in purple are carried out by CUDA kernel functions, executed in a parallel manner for each element in the resulting matrix. The parallelization is possible thanks to the two following principles:

- Vectorized matrix multiplication
  In this technique, each thread is in charge of calculating the dot product between a row from matrix A and a column from matrix B, resulting in one element of matrix C.

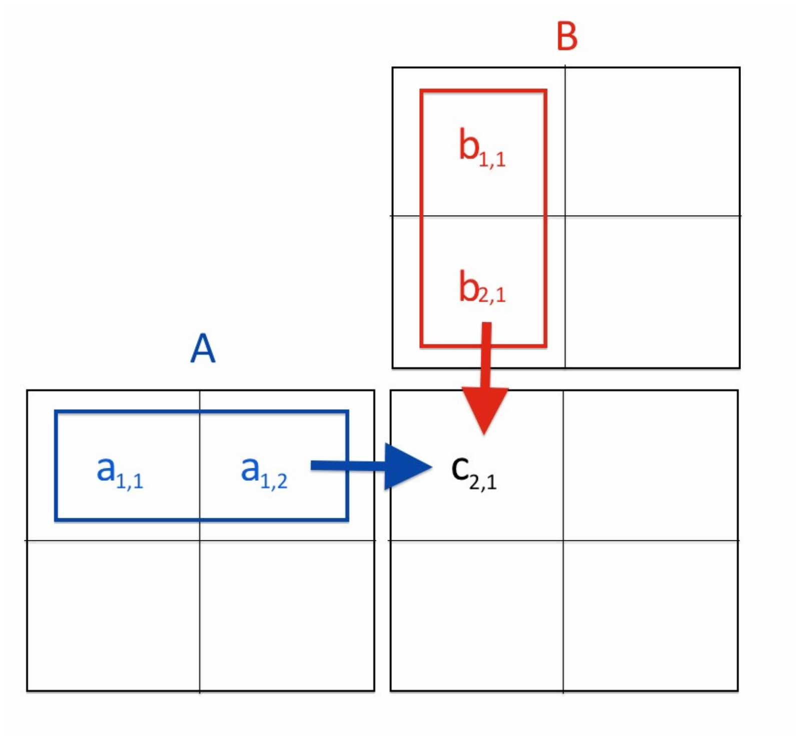


Figure 4: Matrix multiplication illustration.

- Vectorized matrix reduction
  This technique has the objective of calculating the total value of a matrix or vector as a result of the aggregation of its elements in a parallel manner.

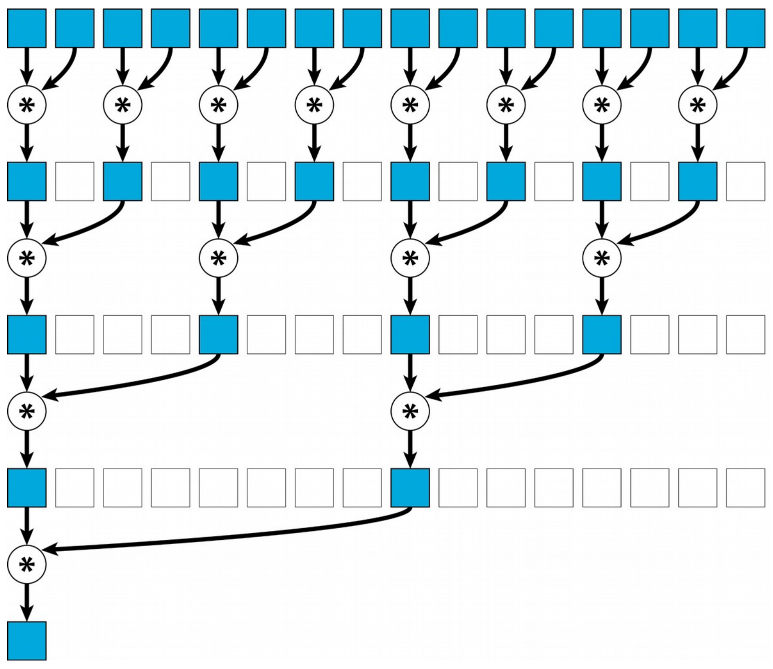


Figure 4: Vector reduction illustration.

INSTRUCTIONS

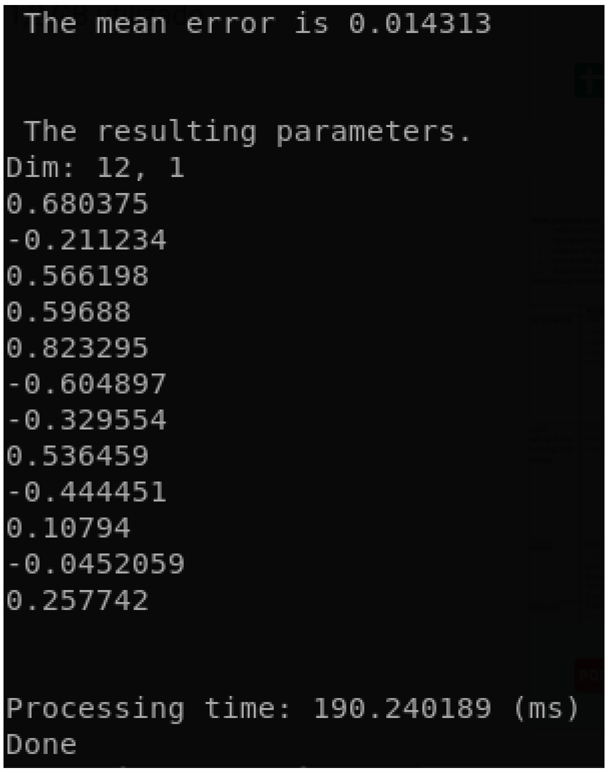
In order to utilize the solution, the following steps should be followed:

1. Place the dataset you want to use as a csv file inside the project folder. NOTE: The label column should be the second one. The first column is considered as the id of the sample, so it is not utilized.
2. Compile main.cu using the following command: *nvcc main.cu*
3. Run the executable file.
4. Enter the name of the csv file you want to use.
5. Observe the results.

NOTE: A successful installation of the CUDA compiler is required to compile the program.

RESULTS

The program is able model a logistic regression function in minimal time, approximating the optimal parameters for the given dataset.



```
 The mean error is 0.014313


 The resulting parameters.
Dim: 12, 1
0.680375
-0.211234
0.566198
0.59688
0.823295
-0.604897
-0.329554
0.536459
-0.444451
0.10794
-0.0452059
0.257742


Processing time: 190.240189 (ms)
Done
```

Figure 4: Output results for sample bening.csv dataset.

CONCLUSIONS

One big component that made possible the rise of machine learning in the last decade was the progress made in GPU computing. Nvidia and CUDA are at the heart of this AI revolution, with CUDA being the state of the art in this area. It is so important that almost all machine learning frameworks are written in it.
Techniques such as logistic regression are just a small part of what is possible with this computing power, but I think it is a good starting point for the implementation of machine learning computing with CUDA.

Artificial Intelligence and Machine Learning are changing the landscape of enterprise IT, so it is important to watch out for these technologies in the years to come.