UC Riverside

Final Project Report

**Implementing a Neural Network**

**in**

**CUDA C**

Report by,

Rajat M. Jain

rjain032@ucr.edu

December 3rd, 2021

Aim: To implement a single hidden layer perceptron in CUDA C to design a binary classifier.

1. Overview:

In this project, I parallelize the serial approach of implementing Neural Networks by breaking down the steps into fundamental matrix operations.

1. Technical Description:

Implementing a neural network on the very low level is just operating functions like multiplication and subtraction on matrices. Taking my understanding from Lab 3 I define several kernels which perform different kinds of matrix operations like multiplication, dot product, subtraction and even performing the Sigmoid activation function for binary classification. Since it is a single hidden layer perceptron my code even facilitates backpropagation and updating of weights. In my approach I do not use any biases.

I operate on a few samples of data from the Iris dataset to implement a binary classifier on the basis of the sepal length, sepal width, petal length and the petal width. The binary classifier predicts 0 for “Setosa” species and 1 for “Versicolor” species.

I use supervise learning to where “Setosa” is labeled 0 and “Versicolor” is labeled 1. To train my model the input is multiplied by a randomly initialized weight matrix and is fed into the perceptron along without a bias and all other weighted inputs. Inside the perceptron back propagation takes place and its final value is then fed as an argument into the activation function. The value of the activation function produces the output y of perceptron. This predicted y is then subtracted from the actual y to calculate the error.

The “main.cu” contains all the host side code while “nn.cu” contains all the device side code.

Functions and their description:

main(): Handles initializing all the parameters and calling the kernels.

dmatbymat(): Multiplies 2 matrices.

dmatsub(): Subtracts 2 matrices.

dsigmoid(): Applies the Sigmoid activation function for binary classification - f(x) = 1/(1 + e^-x).

ddersigmoid(): Applies the Sigmoid activation function for the derivative of the function - f'(x) = f(x)(1 - f(x).

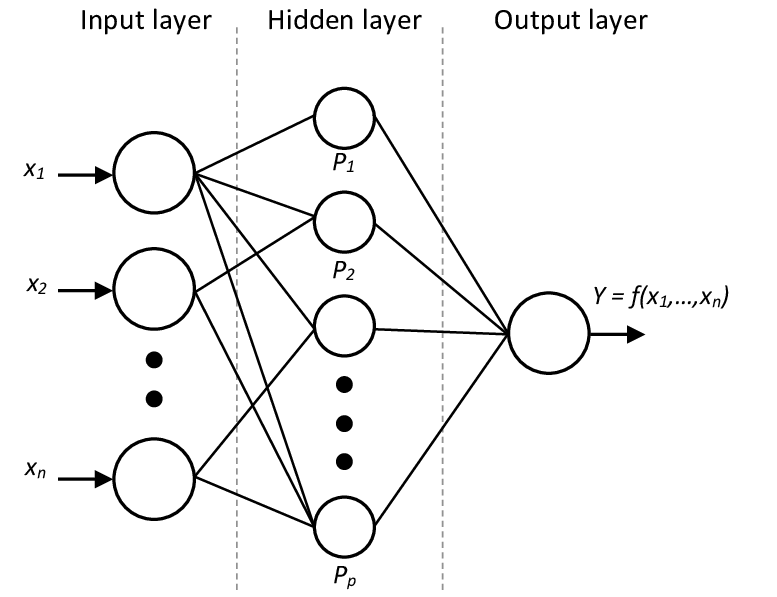
ddotprod(): Computes the dot product of 2 matrices.

dm1dotm2T(): Computes the dot product of 2 matrices

dm1Tdotm12(): Computes the dot product of 2 matrices

kdispmat(): Printing the input array dimensions (h,w)

kfit(): Running the algorithms for 50 epochs.



1. Status:

The project currently is implemented on 8 data samples from the Iris dataset and works very well giving very less error for its predictions.

1. Technical Challenges:

The same approach can be applied to the entire large data set but by using streams or tiled memory to make it run efficiently on large data. Due to the unavailability of time, the scope of this project covers the implementation on 8 data samples only. Unified memory can also be used given a lot of time is consumed by sending data back and forth the host to the device.

1. Results:

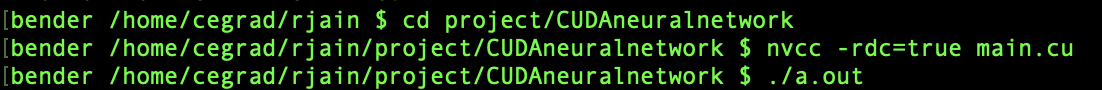
Text

Description automatically generated

1. Compiling:

Text

Description automatically generated



References:

1. <https://docs.nvidia.com/cuda/>
2. https://archive.ics.uci.edu/ml/datasets/iris
3. <https://stackoverflow.com/questions/27590166/how-to-compile-multiple-files-in-cuda>
4. <https://www.youtube.com/watch?v=gAgZkdTF4KQ&t=1s>
5. <https://developer.nvidia.com/blog/even-easier-introduction-cuda/>
6. https://github.com/UCR-CSEE217/lab3-matrixmultiply-rajatjain007