Introduction to Parallel and Scientific Computing

Classification of handwritten digits through SVD and K-Means (CUDA)

Submitted By :

Ajinkya Rawankar (2018201020)
Pranav Verma (2018201047)
Amrit Kataria (2018201067)

INTRODUCTION

Hand written digit recognition is one of the classical classification problem in the field of machine learning. Developing a system for the aforementioned includes a machine to understand and classify the images of handwritten digits as 10 digits (0–9).

Over a period of time, several approaches have been developed towards solving this problem. Some of them are :

- Neural Network based approach
- Deep Network based approach
- RNNs
- Naive means based approach
- SVD

In this project, we have focussed on the last two approaches ie., Naive means and SVD approach.

Typically, finding SVD and means are compute intensive and would take a lot of time when run serially on CPU since the computation involves a lot of matrix multiplication. Moreover, when large matrices are involved (like in MNIST dataset), the computations might take a very long time.

Since a lot of these computations can be parallelised, we can take advantage of this and using the appropriate framework and resources, decrease the time dramatically. In this project, we have used CUDA framework to write parallelised C++ code and ran the executable on NVIDIA GPU to get the results quickly. This can be achieved because the GPU architecture is massively parallelised and facilitates the matrix operations.

An example for how the parallelism can be achieved can be seen in the reduction of an array, where we calculate the sum of all the elements in the array. The usual approach takes O(n) time, while the parallel approach can achieve the same in O(log n) time using the tournament algorithm.

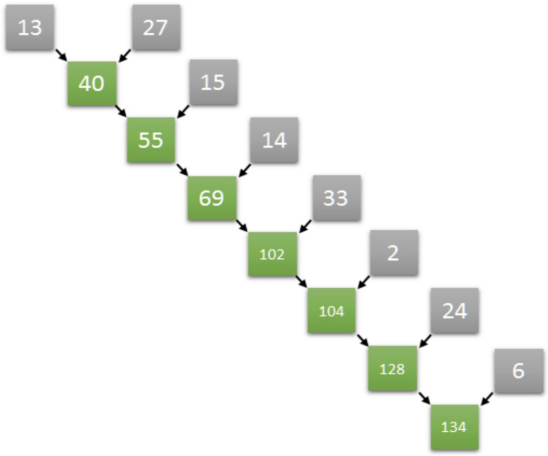

Fig 1 : Sequential sum of an array

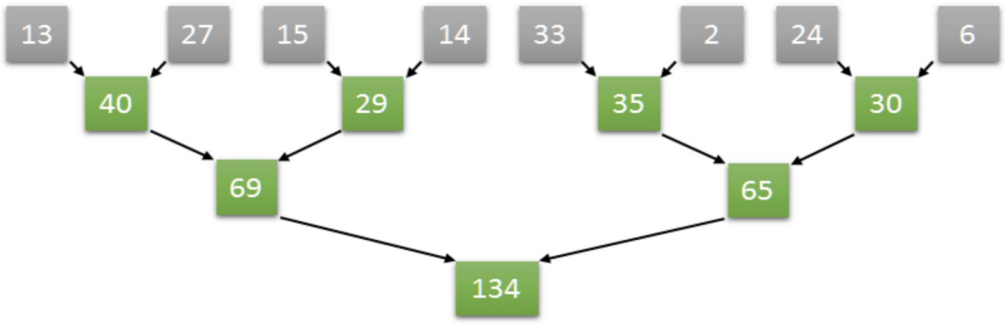

Fig 2 : Parallel sum of an array

SVD (Single value decomposition) Algorithm

SVD is the factorisation of the matrix A (m x n) in three matrices such that :

$$A = U \Sigma V^T$$

where,

U **:** m x m unitary matrix

Σ : m x n rectangular diagonal matrix with non-negative real numbers on the diagonal

V **:** n x n unitary matrix

This approach is based on the modelling of the variation within each class using orthogonal basis vectors computed using the SVD algorithm.

Let A be an m × n matrix. The Singular Value Decomposition (SVD) of A, $A = U\Sigma V^T$, where U is m × m and orthogonal, V is n × n and orthogonal, and Σ is an m × n diagonal matrix with nonnegative diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$, $p = \min\{m, n\}$, known as the singular values of A, is an extremely useful decomposition that yields much information about A, including its range, null space, rank, and 2-norm condition number.

Let U and V have column partitions,

$$U = [u_1 \ \cdots \ u_m] \ , \ V = [v_1 \ \cdots \ v_n] \ .$$

From the relations

$$Av_j = \sigma_j u_j \ , \ A^T u_j = \sigma_j v_j \ , \qquad j = 1, \ldots, p,$$

it follows that

$$A^T A v_j = \sigma_j^2 \, v_j \ .$$

We can implicitly apply the symmetric QR algorithm to A^TA. As the first step of the symmetric QR algorithm is to use Householder reflections to reduce the matrix to tridiagonal form, we can use Householder reflections to instead reduce A to upper bidiagonal form

$$U_1^T A V_1 = B = \begin{bmatrix} d_1 & f_1 & & & \\ & d_2 & f_2 & & \\ & & \ddots & \ddots & \\ & & & d_{n-1} & f_{n-1} \\ & & & & d_n \end{bmatrix}$$

The following steps are followed for predicting the digit from the test set:

- We calculate the SVD for each digit set (0-9).
- For each digit set, we consider the 'U' part of SVD and find the largest twenty eigen vectors. These eigen vectors form a set of vectors for each digit class.
- We then find the product of U and U^T for each class.
- We then find : min $|| I – U_kU_k^T|| Z$. The class for which this value stands the lowest, we assign that class to the test image.

Naive Means Algorithm

In the classification of an unknown digit we need to compute the distance to known digits. Different distance measures can be used, and perhaps the most natural one to use is the Euclidean distance: stack the columns of the image in a vector and identify each digit as a vector in R^{784} (for MNIST dataset). Then define the distance function,

$$(x , y) = || x – y ||_2$$

Training: Given the manually classified training set, compute the means (centroids) m_i, i = 0, . . . , 9, of all the 10 classes.

Classification: For each digit in the test set, classify it as 'k' if m_k is the closest mean.

Results

Various comparisons have been made among running times, number of threads and accuracy.

- Sequential vs parallel implementation of naive means classification
  (Training set : 1000 images , Test set : 200 images)

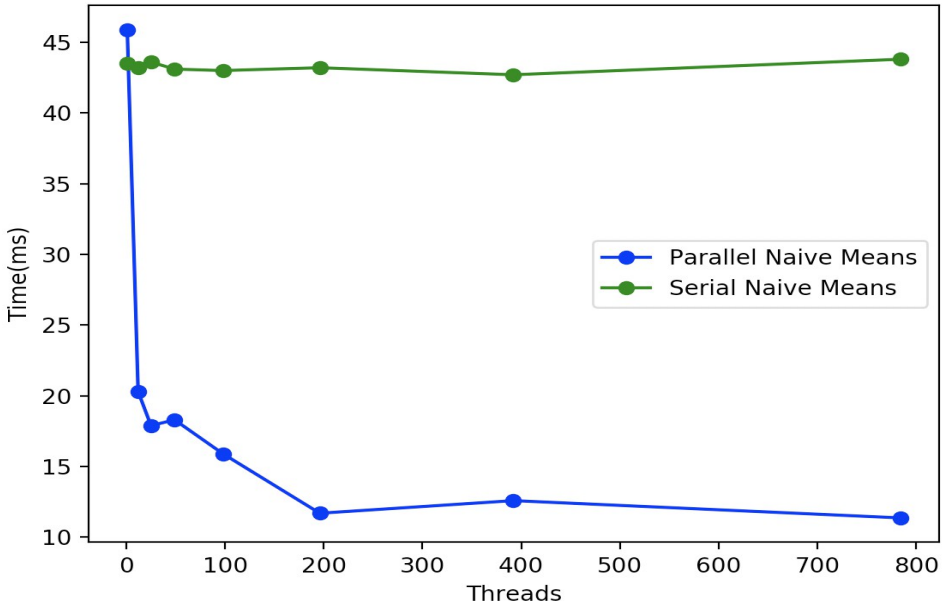

Fig 3 : Serial vs Parallel comparison of Naive Means Classifier

- Accuracy of Naive means vs SVD for different dataset size and split of 80:20

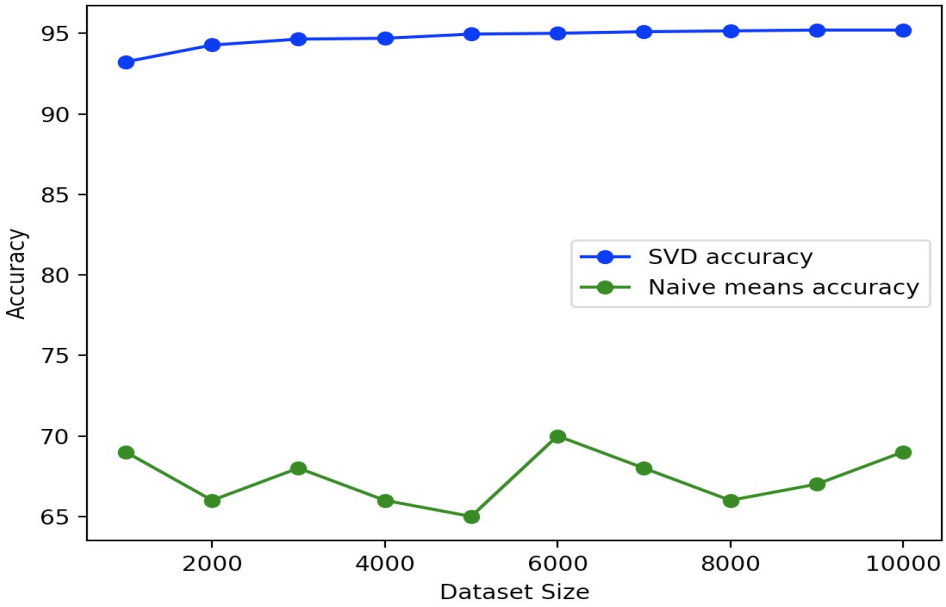

Fig 4 : SVD vs Naive means accuracy

- Number of threads vs running time for parallel SVD
  (Training set : 1000, Test set : 200)

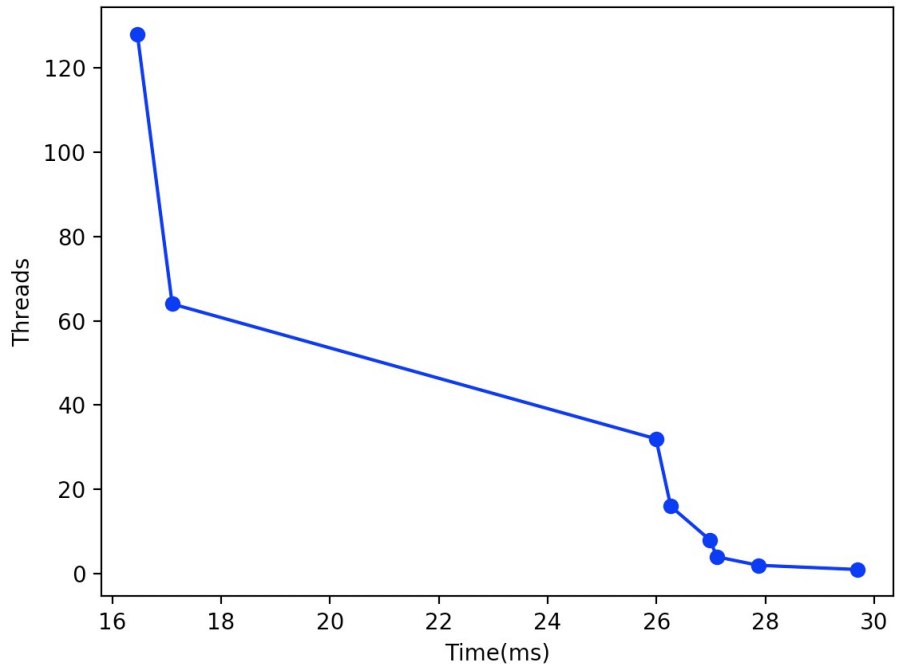

Fig 5 : Threads vs Running time for parallel SVD

Conclusions

- For lower number of threads, parallel implementation takes more time due to overhead involved in creation of threads as observed in Fig. 3. But, as number of threads are increased expected behaviour is observed as parallel implementation executes faster than sequential.
- The classification power of SVD is better than Naive means classification. This can be observed from Fig. 4 above.

References

- https://docs.nvidia.com/cuda/ (Cuda documentation)
- Matrix methods in Data Mining and Pattern Recognition
- https://algowiki-project.org/en/The_serial-parallel_summation_method (Parallel summation)
- https://www.youtube.com/watch?v=91hR9W_nzCU (Programming with CUDA)