Parallel Implementation of Neighbourhood Repulsed Correlation Metric Learning

Saurav Rai[#1], Pallav Kumar Baruah[#2]

[#]*Department of Mathematics and Computer Science*
Sri Sathya Sai Institute of Higher Learning, Puttaparthi, India
[1]sauravraiee@gmail.com
[2]pkbaruah@sssihl.edu.in

Abstract

Metric learning has attracted a lot of attention in recent times due to its usefulness and benefits in the domains of Machine learning, Natural Language Processing, and Big data. It is used as a pre-processing step in many Machine learning algorithms. However, as the size of the input data increases, the amount of time used for learning the metric which captures the semantic nature of the data also increases. In this paper, we have implemented parallel (CUDA) version of one such metric learning algorithm called as Neighbourhood Repulsed Correlation Metric Learning (NRCML). To illustrate the applicability of this parallel implementation, we derive motivation from the experimental evaluation that shows our implementation has greatly reduced the training time of the original sequential implementation. The proposed method has shown improvement in the performance of about 100x and above.

Keywords — *Metric Learning, CUDA, NRCML*.

I. INTRODUCTION

Metric learning is the task of learning a distance function for the space of input data. The types of objects being compared differ based on what function is used. From the perspective of feature learning, metric learning essentially learns a new feature space that must preserve the similarity and dissimilarity relation among the objects. Many learning and data mining algorithm's performance depend critically on their being given a good metric over the input space. There has been a lot of interest in the research scholars in the field of metric learning due to its wide range of applications in Computer vision, Natural language processing and Data visualization as it improves the overall performance of the algorithms by maintaining the semantic nature of the data.

Metric learning has been developed as one of the essential methods for various application such as Face recognition, Computer vision tasks and got the attention of many researchers. The existence of the training examples is a key factor in choosing metric learning algorithm supervised or unsupervised. Class labels are present for most of the supervised learning algorithms. In the case of unsupervised learning, such labels are not provided leaving it on its own to find groups of data points that belong to the same class in its input.

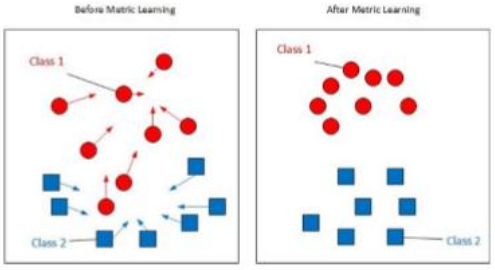

Fig. 1: Metric Learning

There are various types of Metric Learning techniques that have been published over the last decade as discussed in [1]. These techniques have been classified under five categories of 1) Ensemble approach. 2) The non-linear methods whose objective is to optimize a function in non-linear space. 3) The regularized techniques which require regularization conditions to meet the learning condition. 4) The probabilistic designs and methods whose objective is to optimize the likelihood criterion. 5) Cost-variant algorithms which improve the cost function. The NRCML[2] falls under the cost-variant category of metric learning techniques. NRCML[2] is a metric learning technique which learns a distance metric under which examples from a similar class are mapped closer as possible and pushing away examples from other classes.

The remainder of the paper is organized as follows. Section II describes metric learning and the motivation for this paper. Section III describes the Neighbourhood Repulsed Correlation Metric Learning technique [2]. Section IV introduces the design of our parallel implementation of NRMCL algorithm with performance evaluation, experiment-al results along with software tools utilized to achieve the same. Section V concludes the paper

with the future work for further improvement in this study.

II. BACKGROUND

A. Metric Learning

Recently, several algorithms which are based on metric learning have been introduced. Some of these methods have been favourably utilized as a pre-processing step to address the face recognition problem. Due to its pervasive nature, it has been applied to many computer vision tasks such as human age estimation [3], object recognition [4], human activity recognition [5], visual tracking [6], and face recognition [7], person re-identification [8], text analysis and bioinformatics. The basic objective of all these metric learning algorithms is to discover a suitable distance metric so that the distance between the examples of the similar group is decreased and for other groups is increased as large as possible. Metric Learning can be classified into unsupervised, semi-supervised and supervised. The unsupervised approach usually relies on distributions of raw data and its main objective is to preserve the geometrical structure of data points by projecting in low-dimensional manifold. Semi-supervised learning approach tries to acquire and study distance metric with pairwise constraints. These constraints will indicate whether the examples are related or not. The third category is the supervised approach which suffers from heavy manual labelling labour and deals with finding a metric that captures the discriminative nature of the samples. This paper deals with one such supervised method called Neighbourhood Repulsed Correlation Metric Learning[2], which captures the semantic similarity of the input data.

B. Motivation

The Neighbourhood Repulsed Correlation Metric Learning (NRCML) algorithm [2] has shown good results for datasets with wild facial images. It is used for kinship verification via face analysis and has successfully been tested for robustness with images having improper lighting conditions and highly varying poses. The training of the (NRCML) model described in the paper[2] was performed on smaller datasets. The robustness of the model was not tested on larger datasets. One of the possible reason may be due to exponential growth in training time with respect to the corresponding increase in the dataset size. To claim that a model is robust it should also be tested with large datasets with more images. We found that the training part of the model is taking a large amount of time and there is a scope for parallelizing it. It increases the performance of the overall algorithm as well as gives scope for testing robustness on larger datasets. The model trained on large datasets also gives accurate results. Thus overall it provides us with a quick and accurate result.

II. NRCML

Consider $S = \{(s_1, s_2, \ldots, s_N)| \in R^{dxN}\}$ as the training set of N samples, where $s_i \in R^d$. All the terms are as discussed in the paper [2]. The Mahalanobis distance metric learning learns a matrix $M \subset R^{dxd}$ using the training set S, where the distance between s_i and s_j is calculated as:

$$d_M(s_i, s_j) = \sqrt{(s_i - s_j)^T M (s_i - s_j)} \qquad (1)$$

The correlation similarity measure between two vectors a and b is defined as:

$$Corr(a, b) = \frac{a^T b}{||a|||b||} \qquad (2)$$

Because of the correlation of two vectors always lies in the range of -1 and 1, the correlation similarity measure becomes useful for distance metric learning. Let S = $\{(x_i, y_i)| i = 1,2,\ldots,N\}$ be the set of training examples consisting of N face pairs. The pairs x_i and y_i have the kin relationship, where x_i and y_i are the feature vectors of the i^{th} parent and i^{th} child, respectively. The objective of the NRMCL algorithm is to learn a useful metric d so that the distance between x_i and y_i is as small as possible if they belong to the same group and as large as possible if they don't belong to the same group. The correlation similarity of each face pair x and y using A can be calculated as follows

$$Corr(x, y, A) = \frac{Ax^T Ay}{||Ax|||Ay||} \qquad (3)$$

where A is a linear projection that is to be learned by the NRCML algorithm.

Since d is a metric, $d(x_i, y_j)$ should satisfy the symmetry, non-negativity and triangle inequality. Hence A must be symmetric and positive semidefinite. The NRCML algorithm is formulated as the following optimization problem:

$$\begin{aligned} \min_A H(A) = {} & H_1(A) + H_2(A) - H_3(A) + H_4(A) \\ = {} & \frac{1}{NK}\sum_{i=1}^{N}\sum_{t_1}^{K} Corr(x_i, yit_1, A) \\ & + \frac{1}{NK}\sum_{i=1}^{N}\sum_{t_2}^{K} Corr(xit_2, y_i, A) \\ & - \frac{1}{N}\sum_{i=1}^{N} Corr(x_i, y_i, A) \\ & + \alpha||A - A0|| \end{aligned} \qquad (4)$$

where, y_{it}, represents the t_1^{th} k- nearest neighbour of y_i and x_{it_2} represents the t_2^{th} k-nearest neighbour of x_i respectively. The objective function of H1 in Eq. 4 is to enforce that if y_{it_1} and y_i are near, they are to be separated as far as from x_i in the learned distance metric space. Similarly, the objective function of H2 in Eq. 4 is enforce a constraint that if x_{it_2} and x_i are close, they should be separated as far as possible with y_i in the learned distance metric space. On the other hand, H3 in Eq. 4 ensures that x and y are pushed as close as possible in the learned distance metric space because they are from same class. H4 acts as a regularizer to enforce that the learned projection A is as close to the predefined projection A0 which is a parameter which balances contribution of different terms. The NRCML algorithm is summarized in following Algorithm 1.

IV. PARALLEL NRCML

We implemented the NRCML algorithm in python language. The packages used for implementing are numpy, scipy and sklearn. We used benchmark dataset called KinfaceW. LBP features which is publicly available. KinfaceW. LBP features is a set of feature vectors for representing images and we got same results as mentioned in [2]. After profiling the whole code which consists of (1) feature extraction, (2) training (metric learning) and (3) classification using SVM on the metric learned, it was observed that the training function of the whole code takes a lot amount of time. We observed that the training phase takes more amount of time possibly due to the following reasons:

1) It has to find the K-nearest neighbours for each sample of child and parent using the cosine similarity metric respectively.

2) It has two for loops, where the outer loop threads over all the samples and the inner loop threads over K-nearest neighbours.

3) Calculation of H1, H2, H3 and H4 requires basic operations such as vector subtraction, matrix matrix multiplication, and matrix element-wise addition and subtraction. Hence we focused on parallelizing the training phase based on the reasons mentioned above, to reduce the training time. We also noted that the basic structure of the algorithm was not altered and only the training part of the algorithm was parallelized. So achieving speedup does not affect the accuracy of the model.

A. Packages Used

1) We used PyCUDA for the implementation of the parallel version of sequential code on NVIDIA GPU as it allows to interact with NVIDIA CUDA parallel computation API in python and provides several python wrappers for the CUDA API. It offers a lot of advantages over other frameworks that expose the same underlying CUDA API. The following are some of the advantages:

• Smart: Easier to write a correct, leak and crash-free code. Dependencies are automatically handled by PyCUDA.

• Automatic Error Translation: CUDA errors are automatically transformed into exeptions in python.

• Speed: An impressive underlying speed. The base layer is written in C++.

• Completeness: It uses the full power of CUDA's driver API.

Algorithm 1: NRCML

Input : $S = \{(x_i, y_i)|i = 1, 2, ..., N\}$,parameters: neighbourhood size k, iteration number T, and convergence error ϵ(0.0001)

Output: Projection matrix A

Step 1: (Initialization):

Search k-nearest neighbours for each x_i and y_i by using the above defined cosine similarity measure

Step 2: (Local Optimization)

for $r = 1, 2, ..., T$, *repeat* **do**

  Compute H_1, H_2, H_3 and H_4, respectively

  Compute $\frac{\partial H(A)}{\partial A}$

  Update A using gradient descent

  $A = A - \omega \frac{\partial H(A)}{\partial A}$

  if $r \geq 2 \cap |A^r - A_{r-1}| < \epsilon$ **then**

    go to Step 3.

  end

end

Step 3: (Output Projection matrix)

Output Projection matrix $A = A^T$

While using PyCUDA we still need the kernel code in C/C++ CUDA. The python wrapper functions of PyCUDA automatically take care of the memory transfer of the parameters and results between host/device.

2) A python package called scikit-cuda (skcuda), is also used which provides python interfaces to many of the functions in the CUDA device/runtime, CUBLAS, CUFFT, and CUSOLVER libraries.

B. Design

1) Given a matrix of size NxD (where N is the number of samples and D is the number of features for each sample) as input. We used PyCUDA in implementing the function that gives us a matrix of size NxN formed by using a cosine similarity measure between samples.

2) We used PyCUDA and skcuda for implementing basic operations like matrix-matrix multiplication,

element-wise matrix addition and subtraction using PyCUDA as these operations are used very frequent used in computing H1, H2, H3, and H4 matrices.
3) We used numpy package for finding norm of the difference of learned projection and predefined projection and calculating the gradient of H1+H2-H3+H4. In this fashion, we designed a model with the help of PyCUDA and skcuda that targeted the above mentioned three reasons and thus improved the performance of the whole code by reducing the training time.

C. Performance Evaluation

Our implementation was run on NVIDIA TitanX GPU which has 3072 cores and 12 GB memory and on the host CPU which has 4 cores each with the processing speed of 3.40 GHz. We parallelized the training part of the algorithm focusing on the aforementioned three ways. From the following graph, we can observe that there is a close to the linear improvement of speedup with respect to the increase in dataset size.

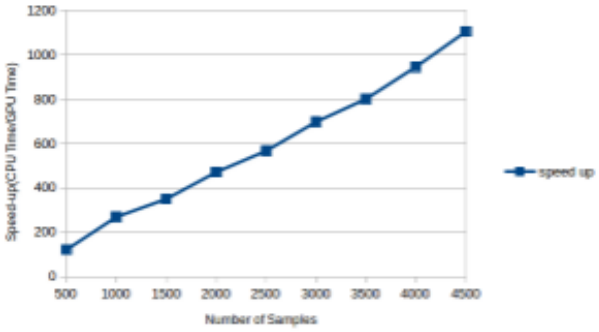

Fig. 2: NRCML Algorithm : Speedup

Samples	CPU Time	GPU Time	Speed-Up
500	124.28	15.27	8.24
1000	501.86	30.67	16.36
1500	1203.54	44.47	27.06
2000	2078.42	59.56	34.89
2500	3091.17	74.23	41.64
3000	4601.74	89.29	51.54
4000	7953.95	119.33	66.66
5000	12708.42	149.17	1105
6000	17618.73	165.23	106.63

V. CONCLUSION AND FUTURE WORK

In this paper, motivated by the fact that the model trained on a large dataset becomes more robust and gives a better result, we have developed a parallel algorithm for neighborhood repulsed correlation metric learning. With the parallelization of the metric learning results in faster learning of the data. In most of the machine learning tasks metric learning is the preliminary task which captures the semantic nature of the given data. Thus developing a faster algorithm for it was the major contribution of the paper. Experimental results also show the performance gained by the model.
In future, we would like to apply this parallel NRCML in areas like computer vision and machine learning to reduce the learning time of the algorithm which will in turn, allow them to learn on larger datasets to give better accurate results.

ACKNOWLEDGMENTS

This work is dedicated to Bhagawan Sri Sathya Sai Baba.

REFERENCES

[1] Panagiotis Moutafis, Mengjun Leng, and Ioannis A Kakadiaris, "An overview and empirical comparision of distance metric learning methods," *IEEE transactions on cybernectics*, vol 47, no. 3, pp 612-625, 2017.
[2] Habin Yan, "Kinship verification using neighborhood repulsed correlation metric learning." *Image Vision Comput.*, vol. 60, pp. 91-97, 2017
[3] Bo Xiao, Xiaokang Yang , Yi Xu, and Hongyuan Zha, " Learning distance metric for regression by semidefinite programming with application to human age estimation." *in Proceedings of the 17th ACM international conference on Multimedia*. ACM ,2009 , pp. 451-460.
[4] Min Tan Zhenfang Hu, Baoyuan Wang, Jieyi Zhao, and Yeuming Wang, " Robust object recognition via weakly supervised metric and template learning", *Neurocomputing*, vol. 181, pp. 96-117, 2016
[5] HM Sajjad Hossain, Md Abdullah Al Hafiz Khan, and Nirmalya Roy, "Active learning enabled activity recogntion," *Pervasive and Mobile Computing*, vol. 38, pp. 312-330 , 2017
[6] Honghong Yang, Shiru Qu, and Zunxin Zheng, "Visual tracking, via online discriminative multiple instance metric learning", *Multimedia Tools and Applications*, pp. 1-19, 2017
[7] Jiwen Lu, Gang Wang, and Jie Zhou," Simulataneous feature and dictionary learning for image set based face recognition." *IEEE Transactions on Image Processing*, 2017.
[8] Rui Zhao, Wanli Oyang, and Xiaogang Wang, " Person reidentification by saliency learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no.2 pp. 356-370, 2017