Proposal GSOC

Implementing QGMM

Profile -

- Name : Aman Pandey
- IRC Name : johnsoncarl
- GitHub : github.com/johnsoncarl
- E-mail : aman0902pandey@gmail.com

- **Interest and hobbies** **:** machine learning, Blockchain, playing computer games, Cricket, Leadership, Dance.
- **University** : Sardar Vallabhbhai National Institute of Technology, Surat, India
- **Field of study** **:** Civil Engineering
- **Date study was started**: July 2017
- **Expected graduation date:** May 2021
- **Time Zone** **:** UTC +5:30
- **Contact No** : +91-9909792126, +91-6352376278

Language	Proficiency	How long I have coded in these
C++	4/5	3 years
Python	3/5	1 year
Solidity	3/5	5 Months
Octave	1.5/5	2 months

Contribution to Open Source Organisations -

Not much, But have worked on a variety of self-projects, and mainly use GitHub for hackathon purposes. I have been working C++ since my schools and have proficient knowledge in it.

But I'm familiar with all the concepts of Git and GitHub.

Projects and Coursework Related to Machine Learning -

I started getting familiar with machine learning by taking

‘Stanford’s Machine learning MOOC on Coursera by - Sir Andrew Ng’
‘CS229 - Instructor Sir Andrew Ng’
‘Stat110X - Instructor Dr Joe Blitzstein’
‘Advanced C++ - Microsoft’
‘Intermediate C++ - Microsoft’

supervised learning- neural networks, linear regression, logistic regression, model selection, a support vector machine

unsupervised learning K-means algorithm, principal component analysis, Gaussian Mixture Models, collaborative filtering, recommender systems

Projects -

- handwriting recognition using neural nets
- building email spam classifiers
- optical character recognition in pics
- movie recommendation systems
- photo size compression using K-means

Presently I'm learning and implementing Deep Learning using TensorFlow and its wonderful applications in Blockchain. The functioning of tensorflow, to develop the translators in future.

-Project implementation-

The goal of my project is to implement the Quantum Gaussian Mixture Models algorithm with its benchmarking with proper Clustering Performance Evaluation techniques and applications in various Unsupervised clustering tasks and preparing a documentation explaining *the advantages of MLPACK(C++ based) library* against another libraries and hence a research paper based on the study with comprehensive details related to the performance. The relevant details are explained below:

-> *Implementing QGMM*

I will be implementing QGMM with required segmentation of the work into proper data feeding, the implementation, the wave functions, its EM process holder, its gradient estimation and updating, with added features, to write tests and finally using

various tasks as benchmarking tool to compare it results with other relevant super popular clustering algorithm, somewhat similar to the GMMs but a little more or less features than that. It will also contain the LOAD/SAVE feature.

My work will be particularly based on a research paper on “Quantum Clustering and Gaussian Mixture” by Mahajabin Rahman and Davi Gieger, which is provided by the organisation itself.

My main work will be implementation and the researching on the advantages of MLPACK doing researches by implementing various evaluation techniques and comparison with other unsupervised clustering techniques, along with preparing the documentation and a research paper showing off their results. The evaluation techniques I am going to implement are explained in the next section.

I will be taking the help of implementation of these techniques from the scikit-learn library, which I came across when trying to make a scikit to mlpack translator.

E.g.
https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/metrics/cluster/tests/test_unsupervised.py

After all, these, If get some free time, I would love to initiate scikit <-> mlpack translator.

The major section of the clustering evaluation techniques can be referenced from here:
E.g
https://www.sciencedirect.com/science/article/pii/S0306437915000733

-> Implementation of the following Clustering Performance Evaluation techniques:

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define separations of the data similar to some ground truth set of classes or satisfying some assumption such that members belong to the same class are more similar that members of different classes according to some similarity metric.
I will take up 3 to 4 evaluation techniques so that I can provide with more complete work at the end of the GSoC period. Most of the techniques are External Index kind Scalar Accuracy Measures.

- **Adjusted Rank Index**
  With the knowledge of the ground truth and the predicted labels, we have Adjusted Rank Index to check the similarity of two assignments by ignoring the labels themselves.
  It is basically the adjusted format of the Rank Index, and also allows the null values like 0.

  Mathematical Formulation:
  If *C* is the ground truth labels and K are the predicted lables then:

  let -

  a - no. of pairs of elements in the same set as in C and as in the same set as in K
  b - no. of pairs of elements in the different sets as in C and in different sets as in K

  The raw index for this will be given as:

$$\mathrm{RI} = \frac{a + b}{C_2^{n_{samples}}}$$

  Where C(n samples)2 is the total possible pairs.

  But it may lead to a value close to 0, in case the number of clusters reaches same order magnitude as the no. of the samples.

  Here, ARI comes into action, and we can discount the expected RI, and define ARI as follows:

$$\mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]}$$

  References:
  - https://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index
  - https://github.com/bjoern-andres/partition-comparison

- **Mutual Information Based Scores**
  Given the same conditions as above, the Mutual Information is a function that measures the assignment agreement, ignoring permutation.
  These are of two types **NMI(**Normalised Mutual Information**)** & **AMI(**Adjusted Mutual Information**).** These are related to each other in a similar way as RI and ARI.

  I will be majorly focusing on the NMI.

  In order to make a trade-off between the quality of the clustering against the number of clusters, NMI is utilized as a quality measure.in various studies. Moreover, NMI can be used to compare clustering approaches with different numbers of clusters, because this measure is normalized.

  Mathematical Formulation:
  Following link explains best the NMIs
  - https://course.ccs.neu.edu/cs6140sp15/7_locality_cluster/Assignment-6/NMI.pdf

  References:
  - https://github.com/robince/gcmi/blob/master/python/gcmi.py
  - https://arxiv.org/abs/1110.2515

- **Homogeneity, Completeness & V-Measure**
  It is kind of Intuitive metric using conditional entropy analysis.
  - **homogeneity**: each cluster contains only members of a single class.
  - **completeness**: all members of a given class are assigned to the same cluster.

  Mathematical Formulation:

The Homogeneity and Completeness measures are given as the following:

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

where *H(C|K) is the* ***conditional entropy of the classes given the cluster assignments*** *and is given by:*

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right)$$

and *H(C) is the* ***entropy of the classes*** *and is given by:*

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot\log\left(\frac{n_c}{n}\right)$$

n is the total number of samples, n_c *is the no belonging to C,* n_k *belonging to k, and finally* $n_{c,k}$ *belonging to samples from class C assigned to cluster K.*

The ***conditional entropy of clusters given class*** *H(K|C) and the* ***entropy of clusters*** *H(K) are defined in a symmetric manner.*

V-measure *as the* ***harmonic mean of homogeneity and completeness****:*

$$v = 2\cdot\frac{h\cdot c}{h+c}$$

References:

- https://www.aclweb.org/anthology/D07-1043

Few more possible and plausible implementation are:

- **Fowlkes-Mallows scores**
- **Silhouette coefficient**
- **Calinski-Harabaz Index**
- **Davis-BouldIn Index**

Whichever seems to be more doable & of more use to mlpack, will be chosen.

-> <u>Documentation:</u>

A proper doc explaining the use cases and advantages of *QGMM,* focusing mainly on the

Speed and reliability of this algorithm written in C++. With a proper visualisation of outcome from each evaluation of the algorithm, against other algorithms will be displayed.

Will try to make a web interface for the testing the algorithm with different parameters.

-> Research Paper:
After performing an exhaustive number of benchmarking tests and come up write a research paper. I'm familiar with LaTeX and the strict rules of following the specified formats.

-> C++ API based command line Interface:
I am familiar with writing bash scripts and how they work. And as, making a C++ file, including the algorithm class and calling various class methods takes considerable amount of time and doesn't takes care of MLPack's off the shelf nature of using state of the art algorithms directly from command line itself without caring much for the algorithm implementation making these algorithms to be used by anyone who is not familiar with C++.
My task would be to get from the user -

- The algorithm to implement
- Parameter file for the algorithm with defaults available if not given.
- Automated output the predictions related to selected classes in a CSV file(optional)
- Filter functions such as displaying a particular evaluation results

Writing tests for the algorithms on various tasks such as -

- Colour Segmentation 3d data example as given in the Research paper provided by the organisation.

After which lastly -

- writing MAN pages
- Weekly reports of my progress
- Youtube Video tutorials explaining the concepts and how to use the feature efficiently
- Exploring more of MLPACK and finding its application in *WEBASSEMBLY* and *BLOCKCHAIN*.

The paper will present a comprehensive evaluation of the algorithms. Our objective is to produce results that how QGMM is able to consistently produce accurate results and generalize well. It will also explain it's possible to use cases.

Timeline -

I'll be working on the project about 42-48 hours per week and will be sharing my progress with my mentors and will regularly write blogs.

Pre-GSoC period - In this period I'll be discussing implementation with mentors and going through research papers related to evaluation techniques and will discuss other implementation methods. In the start of this period, I'll be having my End Semester exams of University (29th April - 5th May) so I will not be active on the project. After that, I will be travelling to my HomeTown that will take 2 more days. Finally, I'll be having my 2.5 months long summer vacations, and I am good to go with the project, and start reading research papers and start with its elemental execution in python and C++.

30th May - 25th June - Implemeting parts of QGMM, out of MLPACK.

25th June - 15th July - Attaching the parts to the MLPACK library, and start working with the tests. Plus the Loading and saving of model and proper functioning of EM algorithm will be checked independently.

16th July - 26th July - Start Documenting the algorithm, and analysing it with the tests.

27th July - 10th August - Writing code for API type CLI interface. Documenting the whole code, writing MAN pages, clean code and improve readability.

10th August - 20th August - Perform an exhaustive number of benchmarking tests and come up write a research paper.

20th August - 23rd August - Doing the leftovers and making youtube videos and improving and submitting final and final submission of a paper, and contacting mentors for there view.

23rd August - 26th August - I'll make a final report and submit the work for end-term evaluation.