1 Abbreviations

—ABRE—Description— ——-————— —QK-Means—Quantum K-Means —qubit—Quantum bit —PCA—Principal Component Analysis —PC—Principal Component —SVD—Singular Value Decomposition —GPGPU—General-Purpose Computing on Graphics Processing Units

Contents

2 Introduction

2.1 Motivation

The scope of the thesis is Big Data and Cluster Ensembles.

Success of EAC clustering on hard data sets.

Interesting problems from big data.

Combining the two.

2.2 Objectives

The main objectives for this work are:

- Aplication of Evidence Accumulation Clustering in Big Data (main goal)
- Exploration of methods that may be included in the EAC paradigm under Big Data constraints (literature review and testing)
- Study of quantum inspired clustering algorithms and evaluation of integration in EAC
- Study of parallel computation techniques and evaluation of integration in EAC
- Validation of Big Data EAC on real data (ECG for emotional state discovery and/or heart disease diagnostic)

2.3 Outline

Explain briefly the work done

The scope of the thesis is Big Data and Cluster Ensembles. A main requirement in this context is to have fast clustering techniques. This may be accomplished in two ways: algorithmically or with parallelization techniques. The former deals with finding faster solutions while the later takes existing solutions and optimizes them with execution speed in mind.

The initial research was under the algorithmic path. More specifically, exploring quantum clustering algorithms. The findings of this exploration were revealed this algorithms to be a poor match for EAC and turned the focus of the research to parallelization techniques. Two main paradigms of parallelization were found appropriate: GPGPU and distributed (among several machines). While the first is a readily available resource in common machines, the second is able to address problems dealing with larger datasets.

3 Context

3.1 Clustering with EAC

3.1.1 Clustering

Advances in technology allow for the collection and storage unprecedented amount and variety of data. Since data is mostly stored electronically, it presents a potential for automatic analysis and thus creation of information and knowledge. A growing body of statistical methods aiming to model, structure and/or classify data already exist, e.g. linear regression, principal component analysis, cluster analysis, support vector machines, neural networks. Many of these methods fall into the realm of machine learning, which is usually divided into 2 major groups: *supervised* and *unsupervised* learning. Supervised learning deals with labelled data, i.e. data for which ground truth is known, and tries to solve the problem of classification. Unsupervised learning deals with unlabelled data and tries to solve the problem of clustering.

Cluster analysis is the backbone of the present work. The goal of data clustering, as defined by [2], is the discovery of the *natural grouping(s)* of a set of patterns, points or objects. In other words, the goal of data clustering is to discover structure on data, structured or not. And the methodology used is to group patterns that are similar by some metric (e.g. euclidean distance, Pearson correlation) and separate those that are dissimilar.

Clustering is used in a wide variety of fields to solve numerous problems, e.g.:

- image segmentation in the field of image processing;
- generation of hierarchical structure for easy access and retrieval of information systems;
- recommender systems by grouping a users by their behavious and/or preferences;
- clustering customers for targeted marketing in
- clustering gene expression data in biology;
- grouping of

3.1.2 Ensemble Clustering

Ensemble clustering Data from real world problems appear in different configurations regarding shape, size, sparsity, etc. Different clustering algorithms are appropriate for different data configurations, e.g. K-Means using euclidean distance as metric tends to group patterns in hyperspheres so it is more appropriate for data whose structure is formed by hypershere like clusters.If the true structure of the data at hand is heterogeneous in configuration, a single clustering algorithm might perform well for some part of the data while other performs better for some other part. The underlying idea behind ensemble clustering is to use multiple clusterings from one or more clustering algorithms and combine them in such a way that the final clustering is better than any of the individual ones.

Formulation Some notation and nomenclature, adopted from [1], should be defined since it will be used throughout the remainder of the present work. The term *data* refers to a set X of n objects or patterns $X = \{\mathrm{x}_1, ..., \mathrm{x}_n\}$, and may be represented by $\chi = \{x_1, ..., x_n\}$, such that $x_i \in \mathbb{R}^d$. A clustering algorithm takes χ as input and returns k groups or *clusters* C of some part of the data, which form a *partition* P. A clustering *ensemble* $\mathbb{P}$ is group of such partitions. This means that:

$$\mathbb{P} = \left\{P^1, P^2, ...P^N\right\} \; P^j = \left\{\mathrm{x}_1^j, \mathrm{x}_2^j, ...\mathrm{x}_{k_j}^j\right\} \; C_k^j = \{x_a, x_b, ..., x_z\}$$

overview of EAC The Evidence Accumulation Clustering (EAC) makes no assumption on the number of clusters in each data partition. Its approach is divided in 3 steps:

1. Produce a clustering ensemble $\mathbb{P}$ (the evidence)
2. Combine the evidence
3. Recover natural clusters

A clustering ensemble, according to [1]., can be produced from (1) different data representations, e.g. choice of preprocessing, feature extraction, sampling; or (2) different partitions of the data, e.g. output of different algorithms, varying the initialization parameters.

The ensemble of partitions is combined in the second step, where a non-linear transformation turns the ensemble into a co-association matrix, i.e. a matrix C where each of its elements n_{ij} is the association value between the object pair (i, j). The association between any pair of patterns is given by the number of times those two patterns appear clustered together in any cluster of any partition of the ensemble. The rationale is that pairs that are frequently clustered together are more likely to be representative of a true link between the patterns [1], revealing the underlying structure of the data. The construction of this matrix is at the very core of this method.

The co-association matrix itself doesn't output a clustering partition. Instead, it is used as input to other methods to obtain the final partition. Since this matrix is a similarity matrix it's appropriate to use in algorithms take this type of matrices as input, e.g. hierarchical algorithms such as Single-Link, K-Medoids. Typically, algorithms use a distance as the similarity, which means that they minimize the values of similarity to obtain the highest similarity between objects. However, a low value on the co-association matrix translates in a low similarity between a pair of objects, which means that the co-association matrix requires prior transformation for accurate clustering results, e.g. replace every similarity value n_{ij} between every pair of object (i, j) by $max\{C\} - n_{ij}$.

examples of applications

advantages

disadvantages quadratic space and time complexities because of the nxn co-association matrix

3.2 The Big Data paradigm

examples of success application

characteristics and challenges

4 State of the art

4.1 Big data clustering

4.2 Quantum clustering

The field of quantum computing has shown promising results regarding potential speedups in several tasks over their classical counterparts. There are two major paths for the problem of quantum clustering. The first is the quantization of clustering methods to work in quantum computers. This translates in converting algorithms to work partially or totally on a different computing paradigm, with support of quantum circuits or quantum computers. Literature suggests that quadratic (and even exponential in some cases) speedup may be achieved. Most of the approaches for such conversions make use of Groover's search algorithm, or a variant of it, e.g. [1]. Most literature on this path is also mostly theoretical since quantum circuits are not easily available and a working feasible quantum computer has yet to be invented. This path can be seen as part of the bigger problem of quantum computing and quantum information processing.

An alternative to using real quantum systems would be to simulate them. However, simulating quantum systems is a very hard task by itself and literature suggest is not feasible. Given that the scope of the thesis is to accelerate clustering, having the extra overhead of simulating the systems would allow speedups.

The second path is the computational intelligence approach, i.e. to use quantum inspired algorithms that muster inspiration from quantum analogies. A study of the literature will reveal that this path typically further divides itself into two other branches. One comprehends the algorithms based on the concept of the quantum bit, the quantum analogue of a classical bit with interesting properties found in quantum objects. The other branch models data as a quantum system and uses the Schrödinger equation to evolve it.

In the following two sections these approaches for quantum inspired computational intelligence are explored.

[1] N. Wiebe, A. Kapoor, and K. Svore, "Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning," p. 31, 2014.

4.2.1 Quantum bit

The quantum bit is a quantum object that has the properties of quantum superposition, entanglement and ...

A qubit can have any value between 0 and 1 (superposition property) until it is observed, which is when the system collapses to either state. However, the probability with which the system collapses to either state may be different. The superposition property or linear combination of states can be expressed as

$$[\psi] = \alpha[0] + \beta[1]$$

where ψ is an arbitrary state vector and α, β are the the probability amplitude coefficients of basis states [0] and [1], respectevely. The basis states correspond to the spin of the modeled particle (in this case, a ferminion, e.g. electron). The coefficients are subjected to the following normalization:

$$|\alpha|^2 + |\beta|^2 = 1$$

where $|\alpha|^2$, $|\beta|^2$ are the probabilities of observing states [0] and [1], respectevely. α and β are complex quantities and represent a qubit:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Moreover, a qubit string may be represented by:

$$\begin{bmatrix} \alpha_1 \\ \beta_1 \end{vmatrix} \begin{vmatrix} \alpha_2 \\ \beta_2 \end{vmatrix} \begin{bmatrix} \alpha_3 \\ \beta_3 \end{bmatrix}$$

The probability of observing the state [000] will be $|\alpha_1|^2 \times |\alpha_2|^2 \times |\alpha_3|^2$

To use this model for computing purposes, black-box objects called *oracles* are used.

Def from wiki: In complexity theory and computability theory, an oracle machine is an abstract machine used to study decision problems. It can be visualized as a Turing machine with a black box, called an oracle, which is able to decide certain decision problems in a single operation. The problem can be of any complexity class. Even undecidable problems, like the halting problem, can be used.

In this context, oracles contain strings of qubits and generate their own input by observing the state of the qubits. After collapsing, the qubit value becomes analogue to a classical bit.

As it stands, oracles aren't quantum systems or even simulate them. The most appropriate description would be a probabilistic Turing machine.

Each string of qubits represents a number, so the number of qubits in each string will define its precision. The number of strings chosen for the oracles depends on the number of clusters and dimensionality of the problem (e.g. for 3 clusters of 2 dimensions, 6 strings will be used since 6 numbers are required). Each oracle will represent a possible solution.

4.2.2 Quantum K-Means

Several clustering algorithms [4-6], as well as optimization problems [7], are modelled after this concept. To test the potential of the algorithms under this paradigm, a quantum variant of the K-Means algorithm based on [5] was chosen as a case study.

4.2.3 Description of the algorithm

The Quantum K-Means (QK-Means) algorithm, as is described in [5], is based on the classical K-Means algorithm. It extends the basic K-Means with concepts from quantum mechanics (the qubit) and genetic algorithms.

The algorithm has the following steps:

1. Initialize population of oracles
2. Collapse oracles
3. K-Means
4. Compute cluster fitness
5. Store
6. Quantum Rotation Gate
7. Collapse oracles
8. Quantum cross-over and mutation
9. Repeat 3-7 until generation (iteration) limit is reached

The algorithm implemented and tested is a variant of the one described in [5]. The genetic operations of cross-over and mutation are both part of the genetic algorithms toolbox, but were not implemented due to the suggestion from [1]. This decision was based on the findings of [8], stating that the use of the angle-distance rotation method in the quantum rotation operation produces enough variability, with a careful choice of the rotation angle.

Initialize population of oracles The oracles are created in this step and all qubit coefficients are initialized with $\frac{1}{\sqrt{2}}$, so that the system will observe either state with equal probability. This value is chosen taken into account the necessary normalization of the coefficients.

Collapse oracles Collapsing the oracles implies making an observation of each qubit of each qubit string in each oracle. This is done by first choosing a coefficient to use (either can be used), e.g. α. Then, a random value r between 0 and 1 is generated. If $\alpha \geq r$ then the system collapses to [0], otherwise to [1].

K-Means In this step we convert the binary representation of the qubit strings to base 10 and use them those values as initial centroids for K-Means. For each oracle, classical K-Means is then executed until it stabilizes or reaches the iteration limit. The solution centroids are returned to the oracles in binary representation.

Compute cluster fitness Cluster fitness is computed using the Davies-Bouldin index for each oracle. The score of each oracle is stored in the oracle itself.

Store The best scoring oracle is stored.

Quantum Rotation Gate So far, we've had classical K-Means with a complex random number generation for the centroids and complicated data structures. This is the step that fundamentally differs from the classical version. In this step a quantum gate (in this case a rotation gate) is applied to all oracles except the best one. The basic idea is to shift the qubit coefficients of the least scoring oracles so they'll have a higher probability of collapsing into initial centroid values closer to the best solution so far. This way, in future generations, we'll not initiate with the best centroids so far (which will not converge further into a better solution) but we'll be closer while still ensuring diversity (which is also a desired property of the genetic computing paradigm). In conclusion, we want to look for better solutions than the one we got before in each oracle while moving in the direction of the best we found so far.

The other approach to clustering that gathers inspiration from quantum mechanical concepts is to use the Schrödinger equation. The algorithm under study was created by Horn and Gottlieb and was later extended by Weisenberg? and Horn.

4.2.4 Horn and Gottlieb's algorithm

The first step in this methodology is to compute a probability density function of the input data. This is done with a Parzen-window estimator in [2,3]. This function will be the wave function in the Schrödinger equation. Having this information we'll compute the potential function that corresponds to the state of minimum energy (ground state = eigenstate with minimum eigenvalue) [2].

This potential function is akin to the inverse of a probability density function. Minima of the potential correspond to intervals in space where points are together. So minima will naturally correspond to cluster centres [2]. The potential of every point in space is a costly computation. One method to address this problem is to compute the potential on the input data and converge this points toward some minima of

the potential function. This is done with the gradient descent method in [2]. Another method [3] is to think of the input data as particles and use the Hamiltonian operator to evolve the quantum system in the time-dependant Schrödinger equation. Given enough time steps, the particles will converge to and oscillate around potential minima.

Both methods take as input parameter the variance σ of the parzen-window estimator.

This method starts off by creating a Parzen-window density estimation of the input data by associating a Gaussian with each point, such that

$$\psi(\mathbf{x}) = \sum_{i=1}^{N} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}$$

where N is the total number of points in the dataset, σ is the variance and ψ is the probability density estimation. ψ is chosen to be the wave function in Schrödinger's equation. The details of why this is are better described in [1-4]. Schrödinger's equation is solved in order of the potential function $V(x)$, whose minima will be the centres of the clusters of our data:

$$V(\mathbf{x}) = E + \frac{\frac{\sigma^2}{2}\nabla^2\psi}{\psi} = E - \frac{d}{2} + \frac{1}{2\sigma^2\psi}\sum_{i=1}^{N}\|\mathbf{x}-\mathbf{x}_i\|^2 e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}$$

And since the energy should be chosen such that ψ is the groundstate (i.e. eigenstate corresponding to minimum eigenvalue) of the Hamiltonian operator associated with Schrödinger's equation (not represented above), the following is true

$$E = -min\frac{\frac{\sigma^2}{2}\nabla^2\psi}{\psi}$$

With all of this, $V(x)$ can be computed. However, it's very computationally intensive to compute V(x) to the whole space, so we only compute the value of this function close to the data points. This should not be problematic since clusters' centres are generally close to the data points themselves. Even so, the minima may not lie on the data points themselves, so what we do is compute the potential at all data points and then apply the gradient descent method to move them to regions in space with lower potential.

There is another method to evolve the system other then by gradient descent which is explained in [4]. Together, this methods make the Dynamic Quantum Clustering algorithm

4.3 Parallel computing

The second direction of the work was given to parallel computing, more spefically to General-Purpose Computation on Graphics Processing Units (GPGPU).

4.3.1 Short Survey of available GPGPU frameworks

4.3.2 Comparison and choice

It basically boils down to OpenCL vs CUDA. OpenCL has the advantage of portability with the issues of performance portability and har d to program. Programming under CUDA, performs well since it was designed alongside with the hardware itself but only works on NVIDIA devices.

4.3.3 Overview of CUDA

5 Methodology

The aim of this thesis is the optimization and scalability of EAC, with a focus for big data. EAC is divided in three steps and each has to be considered for optimization.

The first step is the accumulation of evidence, i.e. generating an ensemble of partitions. The main objective for the optimization of this step is speed. Using fast clustering methods for generating partitions is an obvious solution, as is the optimization of particular algorithms aiming for the same objective. Since each partition is independent from every other partition, parallel computing over a cluster of computing units would result in a fast ensemble generation. Using either or any combination of these strategies will guarantee a speedup.

The second step is mostly bound by memory. The complete co-association matrix has a space complexity of $\mathcal{O}(n^2)$. Such complexity becomes prohibitive for big data, e.g. a two-dimensional data-set of 2×10^6 samples will result in a complete co-association matrix of $14901GB$ if values are stored in single floating-point precision.

The last step has to take into account both memory and speed requirements. The final clustering must be able to produce good results, fast while not exploding the already big space complexity from the co-association matrix.

Initial research was under the field of quantum clustering. After this pursuit proved fruitless regarding one of the main requirements (computational speed), the focus of researched shifted to parallel computing, more specifically GPGPU.

5.1 Quantum Clustering

Research under this paradigm aimed to find a solution for the first and last steps. Two venues were explored: Quantum K-Means and Horn and Gottlieb's quantum clustering algorithm. For both, the experiments that were setup had the goal of evaluating the speed and accuracy performances of the algorithms.

Under the qubit concept, no other algorithms were experimented with since the results for this particular algorithm showed that this kind of approach is unfeasible due to the cost in computational speed. The results highlight that fact.

5.2 GPGPU K-Means

K-Means is an obvious candidate for the generation of partitions since it is simple, fast and partitions don't require big accuracy - partitions generated with K-Means need only a few iterations. For that reason, optimizing this algorithm would ensure that the accumulation of evidence would be performed in a efficient manner.

5.3 Dealing with space complexity of coassocs

6 Discussion?

6.1 References

References

[1] Ana N L Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):835–850, 2005.

[2] Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.