TARUMT

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Assignment

BMCS3003 DISTRIBUTED SYSTEMS AND PARALLEL COMPUTING

2023/2024

Student's name/ ID Number : Lim Wai Min /2107489

Student's name/ ID Number : Teo Yu Tong /2103717

Student's name/ ID Number : -

Programme : RSW

Tutorial Group : 2

Date of Submission to Tutor :21/8/2023

Title:

Machine Learning for movie gross predictor (K-Means Clustering Algorithm)

1. Introduction

Predicting Total Movie Earnings using the KMeans Clustering Algorithm

In the realm of entertainment, where creativity intertwines with commerce, Predicting a movie's earnings before its release is like guessing the path of a shooting star. The variables at play are multifaceted – encompassing genres, star power, marketing strategies, and audience preferences – all of which coalesce into the elusive realm of box office earnings. Conventional methods of prediction often falter in capturing the intricate interplay of these elements, resulting in forecasts that sometimes miss the mark.

Enter machine learning, a burgeoning field that holds the promise to untangle the intricate web of factors impacting movie earnings. Among the diverse array of machine learning techniques, the KMeans clustering algorithm emerges as a beacon of possibility. With its roots in unsupervised learning, KMeans offers a novel perspective – a lens through which the tapestry of movies and their earnings can be segmented, analyzed, and understood.

As the cinematic landscape evolves, so too does the quest to harness data-driven insights. This essay embarks on a journey into the world of movie earnings prediction, where the marriage of machine learning and the KMeans clustering algorithm seeks to decode the enigma of financial success. By delving into the underlying patterns and groups within the intricate ecosystem of movie attributes, this exploration aims to unearth the nuances that govern a film's trajectory from the silver screen to the box office.

2. Problem Statement

In the dynamic landscape of the entertainment industry, predicting the total earnings of a movie before its release has become a challenging endeavor. This challenge is compounded by the intricate interplay of various factors such as genre, cast, marketing strategies, and audience preferences. Traditional methods of prediction often fall short of capturing the complexity of these relationships, resulting in inaccurate forecasts. To address this issue, we propose a novel approach that leverages the KMeans clustering algorithm, alongside distributed and parallel computing techniques, to predict the total earnings of movies more accurately.

Distributed and Parallel Computing Techniques for KMeans Clustering:

2.1. Multi-threading: Multi-threading plays a key role in accelerating the prediction process through KMeans clustering:

- 2.1.1. Data Partitioning: Dividing the movie dataset into smaller segments enables parallel processing of data points, enhancing efficiency.
- 2.1.2. Cluster Assignment: Concurrently assigning data points to clusters using multiple threads reduces computational time.

2.2. Cluster Computing: For KMeans clustering in movie earnings prediction:

- 2.2.1. Data Distribution: Distribute movie data among cluster nodes, allowing independent processing and clustering.
- 2.2.2. Centroid Computation: Nodes collaboratively compute centroids, enabling faster convergence and more accurate clustering.

2.3. Data Parallelism: In the context of KMeans clustering:

- 2.3.1. Concurrent Cluster Analysis: Parallel analysis of clusters within different movie categories improves prediction accuracy and speed.
- 2.3.2. Load Balancing: Efficiently distributing the computational workload among resources ensures optimal performance.

2.4. Task Parallelism

Task Parallelism enables the concurrent execution of diverse tasks or processes on separate processors or cores. A common example is multi-threaded applications.

2.5. Shared Memory & Message Passing

The main ways parallel tasks communicate are through shared memory and message passing. In shared memory, tasks talk by using common variables. With message passing, tasks send and receive messages to communicate such as MPI (Message Passing Interface) for inter-process communication.

By synergizing the capabilities of the KMeans clustering algorithm with these distributed and parallel computing techniques, we aim to revolutionize the way we predict total movie earnings. The selection of these techniques hinges upon the specific requirements of predicting movie earnings, the computational resources available, and the desired accuracy and efficiency levels. This project not only contributes to the advancement of predictive analytics in the entertainment industry but also addresses the intricate challenges of real-time data analysis. Through this endeavor, we strive to provide stakeholders with more accurate insights into the financial prospects of movies, thus shaping the decision-making processes in the dynamic world of film production and distribution.

3. Literature review

3.1. K-Means Clustering

K-means Clustering is a type of unsupervised machine learning method used to divide an unlabeled dataset into distinct groups. It involves training a computer to use unlabeled, unclassified data giving the algorithm the freedom to work on such data without supervision. This technique allows the system to organize data based on similarities, patterns, and differences without any prior guidance or labeling. Essentially, it lets the machine find natural groupings in the data without external supervision. In order to make the data points within each group more similar to one another and distinct from the data points within the other groups, while points in different groups are distinct. Essentially, it sorts objects based on their similarities and differences. (geeksforgeeks.org, 2019)

The K-means clustering method supports a variety of distance measures, including Euclidean, Manhattan, squared Euclidean, and cosine distances. The K-means clustering method supports a variety of distance measures, including Euclidean, Manhattan, squared Euclidean, and cosine distances. The Euclidean distance represents the straight-line distance between two points, P and Q. Commonly used to measure the gap between two locations, it determines the separation in Euclidean space between these points. Squared Euclidean Distance Measure The only difference between this and the Euclidean distance measurement is that it does not include the square root. The Manhattan distance, also known as the distance measured along straight lines that are parallel to the axes, is the combined total of the horizontal and vertical distances. It's important to note that use the absolute value to neutralize any impact from negative numbers. Cosine Distance Measure the angle formed by joining the two vectors from the origin is utilized.(Mayank Banoula, 2023)

4. Methodology:

4.1. Data Collection & Pre-processing

4.1.1. Data Collection

Collecting data from Kaggle was a platform for machine learning and data analysis to provide the dataset relevant to our objective. We will choose the movie dataset available on Kaggle that provides needed information about the movie.

4.1.2. Data Cleansing

Handling duplication entries identify and remove any duplicated entries, and handle missing values.

4.1.3. Model Training with Validation

Uses the K-Means clustering method to group movies based on their characteristics or entries. Each cluster represents movies with similar gross potential. For example, movies that will fit with a specific cluster may be used to predict the gross for new movies.

4.2. Parallel Computing Techniques

4.2.1. CUDA

Compute Unified Device Architecture is referred to as CUDA, it is a parallel computing platform and an Application Programming Interface (API) model that was developed by NVIDIA. It is an extension of C++/C programming. The developers can use the CUDA to perform common computing tasks speed up such as processing matrices and other linear algebra operations. (amitverma2d, 2021)

In GPU-accelerated applications, the tasks are divided into two parts: sequential and compute-intensive. The sequential part of the workload is handled by the CPU which is designed for single-threaded performance, while the program's compute-intensive component is executed in parallel over thousands of GPU cores. When utilizing CUDA, the developers can write code languages like C, C++, Fortran, Python, and MATLAB to express parallelism using extensions in the form of a few simple keywords. (“CUDA Zone - Library of Resources,” 2017) These are some common uses of GPU Parallel Computing including High-performance computing, Deep learning, Machine learning, and so on. (“Understanding NVIDIA CUDA: The Basics of GPU Parallel Computing,” n.d.)

CUDA offers advantages like being able to quickly read information from any address in memory. It also has a shared space where CUDA tasks can quickly access and store

information. This space is faster than other methods and can handle more data at once. One of the main limitations of CUDA is its support only to NVIDIA hardware. While CUDA source code operates based on C++ syntax rules on the host machine or GPU, older versions were reliant on C syntax. This implies that modern CUDA source code might not always function as intended. (amitverma2d, 2021)

4.2.2. OpenMP

OpenMP serves as a powerful tool for parallelization, particularly in shared memory environments, allowing code sections to be executed concurrently, enhancing performance and resource utilization.

Utilizing OpenMP Directives for Parallelization

The key to OpenMP's effectiveness is found in its directives, which act as a way to construct code segments that may be executed in parallel. (passlab.github.io, n.d.) OpenMP directives, distinguished by their simple syntax, are strategically positioned to instruct the compiler in producing parallel code. These directives provide programmers the ability to specify designate loops that can be parallelized, such as "for," "while," or "do" loops.(people.math.sc.edu, 2011) Another example is through directives like omp_set_thread_num(), the number of threads can be specified to align with the system's processing resources.(Dubovyk, 2019) The programmer defines the sections of the code that can run simultaneously using these directives, essentially dividing the burden across the threads.

Creating Parallel Executions:

Utilizing Shared Memory:

OpenMP's primary strength lies in its suitability for shared memory parallelism. In situations where there is a single memory space shared by multiple processors, OpenMP empowers developers to parallelize operations that can be effectively distributed across these processors.(people.math.sc.edu, 2011)

Optimizing Resource Utilization:

By utilizing OpenMP directives, the program can effectively request and utilize a specified number of threads, aligning with the available processors.(passlab.github.io, n.d.) These threads can collaboratively execute code segments, improving execution time and system resource utilization.

Flexibility of Execution Modes:

Seamless Transition:

OpenMP can seamlessly transit between serial and parallel execution modes. The program can be compiled and executed in serial mode while retaining the potential for parallel execution when specified.(iq.opengenus.org, 2019)

Compatibility Across Systems:

OpenMP enables consistent execution across different systems. The same program can be executed in serial or parallel mode, even on computers without OpenMP support. For example, It seamlessly integrates with diverse environments including Linux, macOS, and Windows. (passlab.github.io, n.d.) This versatility ensures that the program's functionality remains intact across various environments.

The methodology adds a dynamic component to parallel computing techniques by using OpenMP. Computational performance is improved by the ability to specify parallelizable parts using directives and to manage the number of threads. OpenMP maximises the use of the resources at its disposal, leading to even greater performance gains. The smooth experience that is made possible by system compatibility while maintaining programme functionality.

Tasks:

Name:	Parallel Computing Techniques
Lim Wai Min	CUDA
Teo Yu Tong	OpenMP

References

1. amitverma2d. (2021, October 13). Introduction to CUDA Programming. Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/introduction-to-cuda-programming/

2. CUDA Zone - Library of Resources. (2017, July 18). Retrieved from NVIDIA Developer website:https://developer.nvidia.com/cuda-zone#:~:text=CUDA%C2%AE%20is%20a%20parallel

3. Understanding NVIDIA CUDA: The Basics of GPU Parallel Computing. (n.d.). Retrieved August 21, 2023, from www.turing.com website: https://www.turing.com/kb/understanding-nvidia-cuda#what-is-cuda

4. geeksforgeeks.org. (2019, May 30). K means Clustering - Introduction - GeeksforGeeks. Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/k-means-clustering-introduction/

5. Mayank Banoula. (2023, April 13). K-Means Clustering Algorithm. Retrieved from Simplilearn.com website: https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm

6. people.math.sc.edu. (2011, May 14). OPENMP - C++ Examples of Parallel Programming with OpenMP. Retrieved August 21, 2023, from people.math.sc.edu website: https://people.math.sc.edu/Burkardt/cpp_src/openmp/openmp.html

7. iq.opengenus.org. (2019, August 20). Get introduced to using OpenMP to parallelize a C++ code. Retrieved from OpenGenus IQ: Computing Expertise & Legacy website: https://iq.opengenus.org/introduction-to-openmp-to-parallelize-a-cpp-code/
8. Dubovyk, S. (2019, September 2). Introduction to the OpenMP with C++ and some integrals approximation. Retrieved from The Startup website: https://medium.com/swlh/introduction-to-the-openmp-with-c-and-some-integrals-approximation-a7f03e9ebb65
9. passlab.github.io. (n.d.). 1.1. Introduction of OpenMP — Parallel Programming and Performance Optimization With OpenMP. Retrieved from passlab.github.io website: https://passlab.github.io/OpenMPProgrammingBook/openmp_c/1_IntroductionOfOpenMP.html
10.

https://github.com/sakshi170920/Movie-Recommendation-System
https://github.com/msk610/MovieGrossPredictor
https://github.com/ztizzlegaming/movie-recommendation-system

Method:
https://github.com/vinayak1998/Parallel-K-Means-Clustering/tree/master

https://github.com/Defvyb/k_means