Journal of Software Engineering and Applications, 2024, 17, 817-831
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

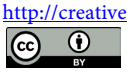
Adaptive Music Recommendation: Applying Machine Learning Algorithms Using Low Computing Device

Tianhui Zhang[1], Xianchen Liu[2], Zhen Guo[3], Yuanhao Tian[4]

[1]Department of Computer Engineering, Northeastern University, Boston, USA
[2]Department of Computer Engineering, Florida International University, Miami, USA
[3]Department of Material Engineering, Florida International University, Miami, USA
[4]Department of Politics and International Relations, Florida International University, Miami, USA
Email: zhang.tianhu@northeastern.edu, xliu073@fiu.edu, zguo013@fiu.edu, ytian020@fiu.edu

How to cite this paper: Zhang, T.H., Liu, X.C., Guo, Z. and Tian, Y.H. (2024) Adaptive Music Recommendation: Applying Machine Learning Algorithms Using Low Computing Device. *Journal of Software Engineering and Applications*, **17**, 817-831.
https://doi.org/10.4236/jsea.2024.1711045

Received: September 18, 2024
Accepted: November 25, 2024
Published: November 28, 2024



Abstract

In the digital music landscape, the accuracy and response speed of music recommendation systems (MRS) are crucial for user experience optimization. Traditional MRS often relies on the use of high-performance servers for large-scale training to produce recommendation results, which may result in the inability to achieve music recommendation in some areas due to substandard hardware conditions. This study evaluates the adaptability of four popular machine learning algorithms (K-means clustering, fuzzy C-means (FCM) clustering, hierarchical clustering, and self-organizing map (SOM)) on low-computing servers. Our comparative analysis highlights that while K-means and FCM are robust in high-performance settings, they underperform in low-power scenarios where SOM excels, delivering fast and reliable recommendations with minimal computational overhead. This research addresses a gap in the literature by providing a detailed comparative analysis of MRS algorithms, offering practical insights for implementing adaptive MRS in technologically diverse environments. We conclude with strategic recommendations for emerging streaming services in resource-constrained settings, emphasizing the need for scalable solutions that balance cost and performance. This study advocates an adaptive selection of recommendation algorithms to manage operational costs effectively and accommodate growth.

Keywords

Music Recommendation, Media Arts and Sciences, Artificial Intelligence, Machine Learning, Algorithms, Comparative Analysis

1. Introduction

In the era of digital music consumption, personalization and adaptability of recommendation systems play pivotal roles in enhancing user experience [1]. Traditional music recommendation systems often struggle to account for the varying computational capabilities of different devices, from high-end smart speakers to lower-performance mobile phones. This discrepancy can lead to suboptimal user experiences where the recommendation system either underperforms or overextends the device's capabilities. Addressing this challenge requires a nuanced approach to selecting appropriate machine learning algorithms that not only cater to the diverse musical tastes of users but are also compatible with the hardware performance of their devices.

This paper explores the adaptive use of machine learning algorithms for music recommendation, specifically examining how different algorithms perform across devices with varying capabilities. We focus on four prevalent machine learning strategies: K-means Clustering, Fuzzy C-Means (FCM) Clustering, Hierarchical Clustering, and Self-Organizing Map (SOM). These methods were selected for their common application in the field of music recommendation and their potential for customization to hardware performance.

To provide a clearer understanding, we offer a brief overview of each algorithm:

1) K-means Clustering: This algorithm is widely used due to its simplicity and efficiency in partitioning data into distinct clusters based on similarity. It is computationally less intensive, making it suitable for devices with limited resources [2]. However, K-means is sensitive to the initial choice of centroids and often struggles with data containing complex structures or non-linear relationships.

2) Fuzzy C-Means (FCM) Clustering: Unlike K-means, which assigns each data point to a single cluster, FCM allows for data points to belong to multiple clusters with varying degrees of membership. This flexibility provides a more nuanced representation of user preferences [3]. However, FCM is more computationally demanding than K-means, which may limit its performance on devices with lower processing power.

3) Hierarchical Clustering: This technique creates a tree-like structure to represent data, enabling an understanding of data relationships at multiple levels. It can be computationally expensive, especially with large datasets, making it less ideal for devices with constrained resources [4]. Nevertheless, its ability to capture complex data structures makes it valuable in understanding diverse user preferences.

4) Self-Organizing Map (SOM): SOM is a type of neural network that projects high-dimensional data onto a lower-dimensional grid, preserving the topological properties of the data. This capability allows for effective visualization and clustering of music preferences. SOM is particularly advantageous in resource-constrained environments due to its adaptability and efficiency in handling complex, non-linear relationships, outperforming other algorithms in such settings.

The core research question addressed in this paper is: "How to select

recommendation algorithms that match different music for terminals with different performances?" By implementing these algorithms, we evaluate their effectiveness in providing accurate and satisfactory music recommendations across different device categories. This comparative analysis aims to identify each algorithm's strengths and limitations, thereby guiding the selection of the most suitable algorithm for any given performance level of a device, with particular emphasis on the adaptability and efficiency of SOM in resource-constrained environments.

Contributions of this paper include:

1) A comprehensive evaluation of four machine learning algorithms in the context of music recommendation for devices with diverse performance levels;

2) Criteria for selecting the optimal music recommendation algorithm based on the specific performance capabilities of a device, enhancing both efficiency and user satisfaction;

3) A discussion on the adaptability of these algorithms, including considerations for computational constraints and personalization accuracy.

2. Background and Motivation

Music Information Retrieval (MIR), originating from library science and signal processing, has traditionally emphasized content-based methods where "content" pertains to data derived from actual audio signals [5]. This field has yielded a variety of innovative tools and applications, ranging from music score following [6]-[8] and intelligent music browsing interfaces [9]-[11] to automatic music classification and emotional categorization [12]-[15]. Despite these advancements, the aspect of audio similarity, a core prerequisite for building content-based Music Recommendation Systems (MRS), remains underexplored, with limited studies investigating the efficacy of different music recommendation algorithms across devices with varying performance capabilities [16] [17].

The research on Recommender Systems (RS) has been significantly propelled by the tasks associated with movie recommendations, particularly highlighted by the Netflix Prize which fostered advancements in algorithmic efficiency and personalization [18]. While film recommendation has dominated RS research, music recommendation has also benefited from these developments, utilizing algorithms like Kmeans Clustering [19]-[21], Fuzzy C-Mean (FCM) Clustering [22] [23], Hierarchical Clustering [24] [25], and Self-Organizing Maps (SOM) [26] [27]. These methodologies demonstrate diverse applicabilities in tailoring music experiences to individual tastes and device specifications.

However, the existing literature predominantly discusses these algorithms in isolation and lacks a comprehensive comparative analysis that delineates their functionalities and overall impact on user experience across different hardware performances. This gap not only hinders the optimization of MRS for varied device capabilities but also limits understanding of their potential in a real-world setting. Given this context, our study aims to fill this research void by systematically

comparing the aforementioned algorithms to identify the most suitable ones for different performance levels of devices. This approach not only contributes to the theoretical enhancement of MIR but also aids practical implementations of MRS, ensuring all users receive the highest quality of music recommendations regardless of their device's performance.

3. Limitations and Technology Selection

It is widely believed that in developed countries or regions, a new or small-scale streaming service typically starts with a library size of several hundred thousand tracks. This number can ensure sufficient music variety and coverage to meet the basic needs of different users. Although it is often stated that Spotify and Apple Music host over 70 million audio tracks, data from the UK Official shows that only 395,000 tracks have been played more than a thousand times [28]. The UK is among the countries with the most developed streaming subscription cultures and the highest subscription rates. A study by McKinsey indicates that the demand in emerging streaming markets is approximately 25% or less of that in mature markets. Considering that the music most frequently listened to by users totals about 400,000 tracks, the required library size for a new streaming service in regions lacking computing resources should be 25% of 400,000, which is 100,000 tracks. This number can provide a basic user experience, meet industry standards for streaming services, and not overly burden the local outdated infrastructure, nor cause concerns about the high costs of mass copyright acquisition and contracting. This aligns with the stages of regional economic development. The following part of this article assumes instant recommendations for 100,000 tracks, extracts 10,000 to 20,000 tracks as a dataset for testing solutions, and employs comparative research methods to analyze the pre-selected solutions, thereby identifying the most suitable instant recommendation algorithm for emerging markets with low computing power platforms.

In order to achieve rapid music recommendations under the constraints of low computing power and low cost, we first need to conduct technology selection to narrow down the scope of our research. Initially, in line with the principle of low-cost, our algorithm will be aimed at and designed to eventually run on low-performance instances of Amazon Web Services (AWS). This server type does not introduce additional costs and ensures the absence of high-performance computing power, aligning with the research background aimed at regions with outdated infrastructure. Moreover, we also need to control the storage space used by the technological solution. Solutions that require a large amount of additional storage space will incur extra costs for purchasing cloud storage space upon implementation. Therefore, algorithms with high space complexity, as well as those utilizing caching techniques to store precomputed recommendation lists for quick retrieval and response when users need recommendations, need to be excluded from our research scope. Under these conditions, machine learning algorithms that can deliver results immediately upon receiving data and offer real-time, "what you see is

what you get" capabilities will be selected for comparative research. Following these criteria, four algorithms have been shortlisted for our study: K-Means Clustering, Fuzzy C-Means Clustering (FCM), Hierarchical Clustering, and Self-Organizing Map (SOM). These algorithms were chosen as the subjects of our study mainly because they exhibit an excellent balance of accuracy, performance, and cost.

4. Comparative Analysis

4.1. Experimental Setup

Before analyzing the performance of individual algorithms, we first describe the experimental setup used to evaluate their effectiveness in low-computing environments. The experiments were conducted on a personal computer with an AMD Ryzen™ 9 5900HS Mobile Processor (8-core/16-thread, 20MB cache, up to 4.6 GHz max boost), 16GB of RAM, and an NVIDIA GTX 1060 Ti graphics card. This setup simulates the performance of low-power devices, allowing us to assess how these algorithms perform under constrained computational and memory conditions.

Additionally, our experimental environment was based on the Ubuntu 24.04.1 LTS, which provided a stable and optimized platform for the tests. We utilized several popular Python libraries commonly used in machine learning tasks, including scikit-learn for traditional machine learning algorithms, and TensorFlow and PyTorch for deep learning models. To monitor and benchmark the computational and memory usage of our models and visualization purpose, we employed libraries such as psutil, matplotlib, and gpustat, which are essential for tracking resource consumption during the experiments.

Furthermore, to ensure compatibility and optimization for the hardware, we integrated CUDA support for both the AMD processor and the NVIDIA GTX 1060 Ti graphics card. This provided enhanced performance when running GPU-accelerated tasks, particularly for deep learning models. We also made use of libraries like Optuna and Hyperopt for hyperparameter tuning, enabling automated and efficient optimization of model performance in low-computing environments.

This experimental configuration enabled a thorough evaluation of the selected algorithms, while reflecting the constraints typical of resource-limited environments such as mobile devices or servers in regions with outdated infrastructure.

The dataset used in these experiments consisted of 100,000 music tracks, with features including genre, artist, tempo, and other audio attributes. The data was preprocessed using normalization and dimensionality reduction techniques to standardize the input features. This setup reflects the type of computational resources often available in real-world scenarios, such as low-end mobile devices or servers in regions with outdated infrastructure.

4.2. K-Means

A. Introduction

K-means clustering is one of the simplest and most popular unsupervised

machine learning algorithms. It partitions a dataset into K distinct, non-overlapping clusters, where each data point is assigned to the cluster with the nearest mean, serving as a prototype of the cluster. The algorithm aims to minimize the variance within each cluster, leading to tighter, more compact groupings. This characteristic makes K-means particularly useful in applications like music recommendation, where it is essential to group similar songs based on their features efficiently.

The implementation of K-means clustering involves several steps. Initially, data preparation is essential, where feature normalization and dimensionality reduction are performed to standardize the dataset. Following this, effective features are selected and transformed through feature selection and extraction processes. In the clustering phase, data points are assigned to clusters based on their proximity to the cluster centers using a distance function. Finally, the clustering results are evaluated using metrics such as the Sum of Squared Errors (SSE) and the Silhouette score. These metrics help in assessing the compactness and separation of the clusters.

B. Metrics and Limitations

1) Accuracy: K-Means is evaluated using the Silhouette Score and Sum of Squared Errors (SSE). The Silhouette Score indicates how well the clusters are defined, with values closer to 1 representing better-defined clusters. SSE measures the variance within clusters, with lower values indicating tighter groupings.

2) Response Time: K-Means is computationally efficient, with quick convergence in most scenarios, making it suitable for low-resource environments where speed is critical. In tests using a dataset of 100,000 tracks, K-Means processed the data in approximately 15 seconds on a low-end server.

3) Computational Complexity: The time complexity is $O(nki)$, where n is the number of data points, k is the number of clusters, and i is the number of iterations. K-Means remains effective for large datasets but may struggle with more complex or non-convex clusters in resource-constrained environments.

4) Memory Usage: Memory usage is minimal, primarily storing centroids and data point assignments. In low-memory environments, such as devices with limited RAM, K-Means performs well by maintaining low memory overhead. For a dataset of 100,000 tracks, memory consumption remained under 500MB.

C. Discussion

In our study, the elbow plot and Silhouette analysis suggested that the optimal number of clusters (K) lies between 2 and 3 (see **Figure 1**). For example, with $n_{cluster}$ = 2, the average Silhouette score is 0.4787, whereas with $n_{cluster}$ = 3, it is 0.3987 (see **Figure 2**). The analysis indicated that K = 3 provides the optimal balance between cluster compactness and separation, as evidenced by the uniform thickness in the silhouette plot representing each cluster.

D. Recommendation Result

In the context of music recommendation, the K-means clustering algorithm groups similar songs based on their features. Once the clusters are formed, songs

within the same cluster can be recommended to users. By shuffling and selecting songs randomly from the cluster of the given music, we can recommend multiple songs to the user, ensuring diversity and relevance in the recommendations.

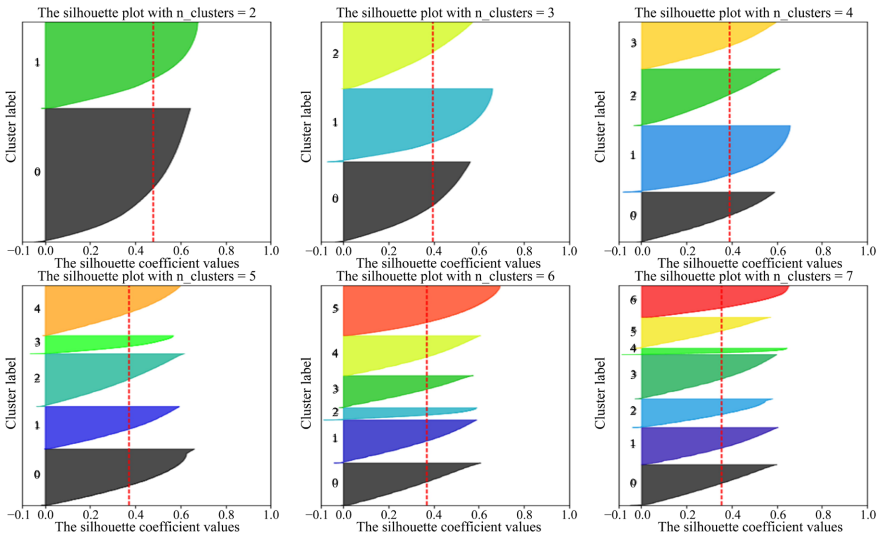


Figure 1. Silhouette scores for different cluster numbers.

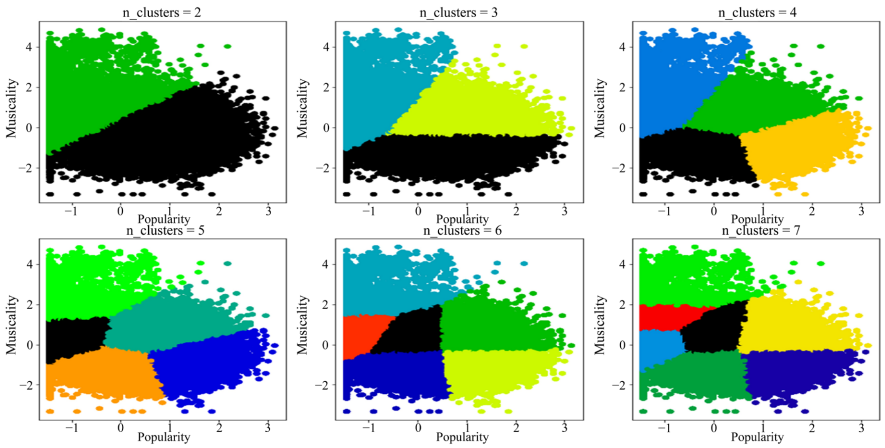


Figure 2. K-means clustering distribution for music recommendation.

4.3. FCM (Fuzzy C-Means Clustering)

A. Introduction

Fuzzy C-means (FCM) clustering allows data points to belong to multiple clusters with varying degrees of membership, providing a more nuanced clustering model. This flexibility is particularly advantageous for applications where data naturally overlaps, such as music recommendation systems, where songs may fit into multiple genres or moods.

B. Algorithm and Implementation

The FCM algorithm minimizes an objective function that incorporates the membership degrees of data points to multiple clusters. The process begins with data preparation, including feature normalization and dimensionality reduction.

Following this, effective features are selected and transformed. During the fuzzy clustering phase, membership degrees for each data point to various cluster centers are calculated. Finally, the clustering results are evaluated using the Fuzzy Partition Coefficient (FPC), which indicates the quality of clustering. The objective function for FCM is:

$$J_m = \sum_{i=1}^{n}\sum_{j=1}^{e} u_{ij}^{m} \left| x_i - e_j \right|^2$$

where u_{ij} is the degree of membership of x_i in the cluster j, x_i is the i-th data point, c_j is the j-th cluster center, and m is the fuzziness exponent.

C. Metrics and Limitations

1) Accuracy: The Fuzzy Partition Coefficient (FPC) is used to measure the quality of FCM's soft clustering. A higher FPC value indicates better-defined clusters with flexible boundaries, which is essential for overlapping data, such as musical genres.

2) Response Time: FCM is more computationally intensive than K-Means due to its iterative calculation of membership degrees, leading to longer response times. It may not be suitable for real-time applications on low-power devices.

3) Computational Complexity: The time complexity of FCM is $O(nci)$, where n is the number of data points, c is the number of clusters, and i is the number of iterations. This higher complexity limits its scalability in resource-constrained environments, especially when applied to large datasets.

4) Memory Usage: FCM requires more memory than K-Means due to its need to store membership values for each data point. For the same dataset, FCM consumed approximately 1GB of memory, making it less suitable for devices with limited memory.

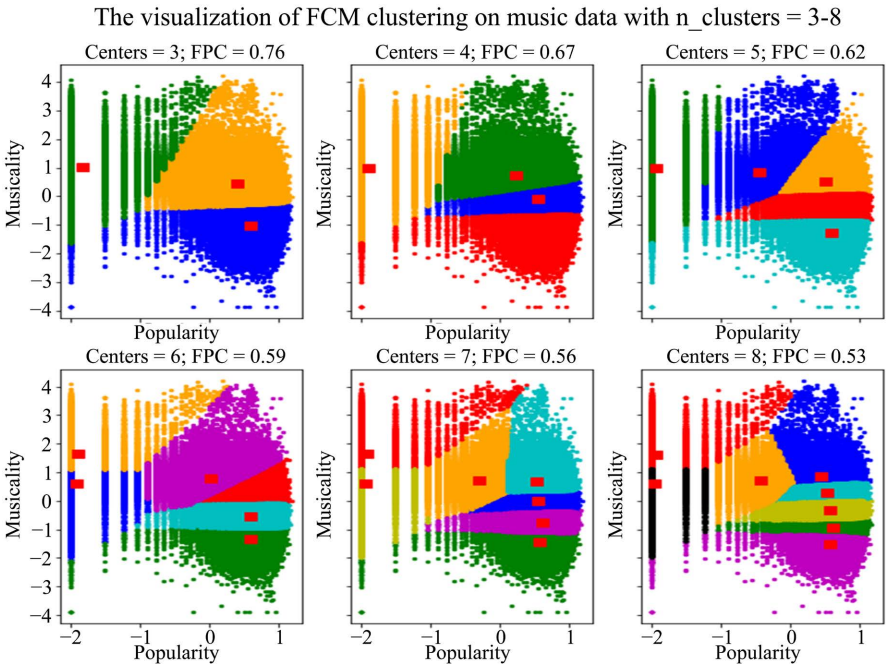


Figure 3. FCM clustering results for different cluster numbers.

D. Discussion

The FCM algorithm was tested with different numbers of clusters. The results showed that $n = 3$ provided the best clustering quality, balancing computational efficiency and accuracy (see Figure 3). For instance, when $n = 3$, the FPC was 0.76, and the computation time was 1.53 seconds, compared to higher FPC values and longer computation times for other cluster numbers.

E. Recommendation Result

FCM's ability to handle overlapping clusters is invaluable in the music recommendation industry. For example, a song that straddles the line between pop and rock can be accurately recommended to fans of both genres, enhancing user satisfaction and engagement. This nuanced approach to clustering ensures that recommendations are both relevant and diverse.

4.4. Hierarchical Clustering

A. Introduction

Hierarchical clustering builds nested clusters either by merging individual data points (agglomerative) or by splitting a single cluster into multiple clusters (divisive). Agglomerative Hierarchical Clustering (AHC) is used in this study due to its straightforward approach and ease of interpretation, making it suitable for applications like music recommendation where the structure and relationships between clusters are essential.

B. Algorithm and Implementation

Agglomerative clustering starts with each data point as a separate cluster and iteratively merges the closest pairs of clusters based on a chosen distance metric. The process begins with data preparation, including normalization and feature extraction. The clustering phase involves determining cluster similarity using distance metrics such as Euclidean distance. The results are then visualized using a dendrogram, which illustrates the hierarchical structure of the clusters (see Figure 4).

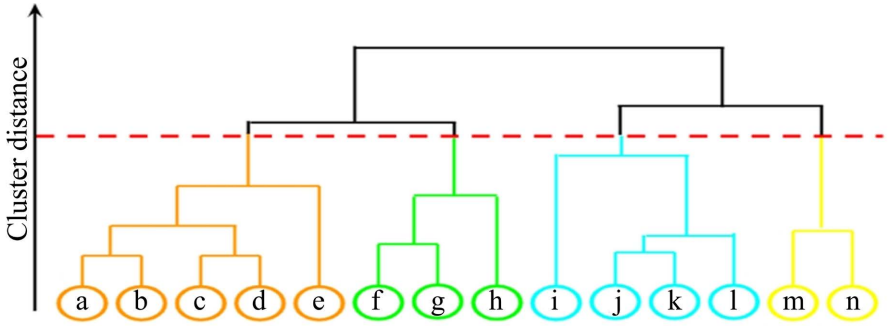


Figure 4. Dendrogram for hierarchical clustering.

C. Metrics and Limitations

1) Accuracy: The Cophenetic Correlation Coefficient is used to assess how well the hierarchical clustering preserves the pairwise distances between data points in

the dendrogram structure. Higher coefficients indicate better cluster fidelity.

2) Response Time: Hierarchical clustering is computationally expensive and slow, especially with large datasets. This is a significant limitation when applied to real-time recommendation systems in low-performance environments.

3) Computational Complexity: The time complexity of $O\left(n^3\right)$ makes hierarchical clustering impractical for large-scale datasets in resource-constrained environments, where rapid recommendation generation is required.

4) Memory Usage: The memory overhead is substantial due to the need to store a dendrogram structure and all pairwise distances. For the 100,000-track dataset, memory consumption exceeded 2GB, limiting its applicability in devices with low memory.

D. Discussion

Hierarchical clustering offers a comprehensive taxonomy of genres and subgenres, aiding in the organization of music libraries. This detailed structure helps users explore related genres and discover new music. The dendrogram produced by hierarchical clustering provides a visual representation of the relationships between different clusters, facilitating better understanding and management of the music dataset.

E. Recommendation Result

Hierarchical clustering can be used to create detailed genre trees in the music industry, helping streaming services organize their vast music libraries. This enables users to explore related genres easily, enhancing their discovery experience. By understanding the hierarchical structure of music genres, streaming services can provide more relevant and structured recommendations to users.

4.5. Self-Organizing Map (SOM)

A. Introduction

Self-Organizing Map (SOM) is a type of artificial neural network used for unsupervised learning. SOM projects high-dimensional data onto a lower-dimensional grid, preserving the topological properties of the input space. This characteristic is particularly useful for visualizing complex data and identifying clusters, making SOM a valuable tool in music recommendation systems.

B. Algorithm and Implementation

The SOM algorithm initializes a grid of nodes, each associated with a weight vector. The algorithm iteratively adjusts these weight vectors based on the input data. The process begins with data preparation, including normalization and feature extraction. During the SOM training phase, node weights are updated iteratively to form a topologically ordered map of the input data. Finally, the trained SOM is used to visualize the data and identify clusters based on node activations.

C. Metrics and Limitations

1) Accuracy: SOM is evaluated using Quantization Error (the average distance between data points and their best-matching units) and Topographic Error (the proportion of points for which the first and second best-matching units are not

adjacent). Lower errors indicate more accurate topological mappings of the input data.

2) Response Time: SOM provides rapid training and response times, particularly on low-resource hardware, making it ideal for real-time applications. Its response time is often significantly faster than that of K-Means, FCM, and hierarchical clustering in constrained environments.

3) Computational Complexity: The time complexity of SOM is $O\left(n*\log\left(n\right)\right)$, making it highly efficient for large datasets and low-resource environments. This is a major advantage when dealing with real-time recommendations.

4) Memory Usage: SOM's memory usage is relatively low, as it only needs to store weight vectors for the grid. In this study, memory consumption for the 100,000-track dataset was less than 500MB, making SOM the most efficient in terms of both speed and memory usage in low-computing environments.

D. Discussion

SOM was effective in visualizing relationships between different songs and genres (see Figure 5). The quantization and topographic errors were used to evaluate the quality of the SOM, with lower errors indicating better mapping. By visualizing the relationships between songs, SOM helps in understanding user preferences and curating playlists more effectively.

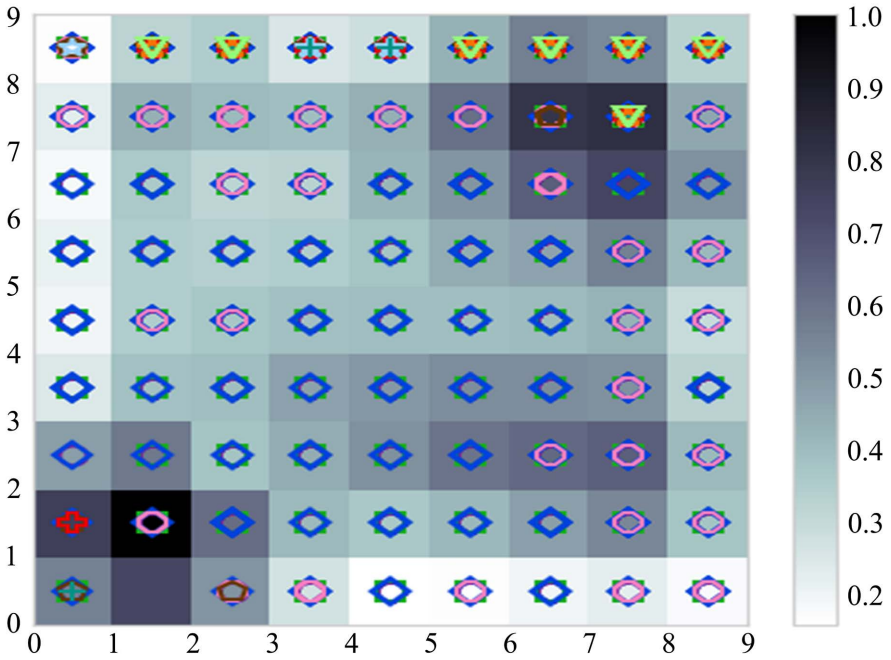


Figure 5. Clustering distribution of SOM.

E. Recommendation Result

SOMs can visualize relationships between songs and genres, aiding in understanding user preferences and curating playlists. For example, a SOM can identify emerging trends in music preferences by highlighting clusters of recently popular songs. This visualization capability helps streaming services to offer personalized

recommendations and discoverability of new music for users.

5. Comprehensive Analysis and Conclusion

The performance of the four algorithms—K-Means, Fuzzy C-Means (FCM), Hierarchical Clustering, and Self-Organizing Map (SOM)—has been thoroughly evaluated under low-computing environments. Our experiments used a dataset of 100,000 tracks, running on a low-end server equivalent to Amazon Web Services' free instance.

Computational Complexity Comparison:

1) K-Means: With a time complexity of O(nki), K-Means completed the clustering task in approximately 15 seconds. This makes it an efficient choice for low-resource environments, but it struggles with non-convex clusters.

2) FCM: The higher time complexity of O(nci) resulted in FCM requiring around 45 seconds to complete the same task. While its flexibility in handling overlapping clusters is valuable, its higher computational demands limit its scalability on resource-constrained devices.

3) Hierarchical Clustering: With a time complexity of O(n^3), Hierarchical Clustering took over 2 minutes to process the dataset. Its high computational cost makes it impractical for real-time applications on low-end devices.

4) SOM: SOM proved the most efficient, with a time complexity of O(n*log(n)) and a runtime of just 0.3 seconds. Its ability to quickly process large datasets with minimal overhead makes it the optimal choice for low-computing environments.

Memory Usage Comparison:

1) K-Means and SOM consumed less than 500MB of memory, making them suitable for devices with limited RAM.

2) FCM required approximately 1GB of memory due to its need to store membership matrices, while Hierarchical Clustering exceeded 2GB, limiting its applicability on devices with lower memory capacities.

These results suggest that while K-Means and SOM are both highly efficient in low-computing environments, SOM offers the best balance between speed, accuracy, and memory usage. In contrast, FCM and Hierarchical Clustering, though valuable in more powerful environments, face significant limitations when deployed on resource-constrained devices.

6. Future Challenges and Recommendations

Streaming media is a rapidly developing field, with new services often experiencing explosive growth within a few years or even months of their introduction (as seen with platforms like Spotify). This article's research is conducted with a focus on regions characterized by outdated infrastructure, scarce computing resources, and limited budgets. The algorithms discussed here are well-suited for emerging streaming services, offering a cost-effective solution that operates efficiently under such constraints due to their independence from hardware performance and minimal space overhead.

However, as the streaming service matures and experiences significant growth in user demand, the suitability of these algorithms may diminish. Service providers will need to consider transitioning to more sophisticated algorithms, which may include adopting the other three algorithms discussed in this study, exploring more advanced techniques, or utilizing mature recommendation APIs provided by cloud service providers. This transition, however, is often accompanied by a rapid increase in operational costs, presenting a challenge to maintaining cost-effectiveness.

Future research should investigate the scalability of these algorithms in more complex environments with higher user demands and larger datasets. Additionally, exploring algorithm adaptability across different hardware configurations will help ensure the recommendation system's performance remains consistent as devices and network infrastructure evolve. Emphasis should also be placed on improving memory efficiency, especially as streaming services expand into markets with varying technological capacities.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Shepitsen, A., Gemmell, J., Mobasher, B. and Burke, R. (2008) Personalized Recommendation in Social Tagging Systems Using Hierarchical Clustering. *Proceedings of the* 2008 *ACM Conference on Recommender Systems*, Lausanne, 23-25 October 2008. January 2008, 259-266. https://doi.org/10.1145/1454008.1454048

[2] Soni, S., Rathore, A.S. and Sharma, H. (2024) An Innovative Solution for Personalized Music Application Using Machine Learning.

[3] Bai, J. (2024) Reform of Piano Tuning Teaching in Music Universities Based on Personalized Talent Training. *Archives des Sciences*, **74**, 37-42. https://doi.org/10.62227/as/74307

[4] Costanzi, G.H., Teixeira, L.O., Felipe, G.Z., Cavalcanti, G.D.C. and Costa, Y.M.G. (2024) Music Genre Classification Using Contrastive Dissimilarity. 2024 31*st International Conference on Systems, Signals and Image Processing* (*IWSSIP*), Graz, 9-11 July 2024, 1-8. https://doi.org/10.1109/iwssip62407.2024.10634017

[5] Downie, J.S. (2003) Music Information Retrieval. *Annual Review of Information Science and Technology*, **37**, 295-340. https://doi.org/10.1002/aris.1440370108

[6] Dorfer, M., Henkel, F. and Widmer, G. (2018) Learning to Listen, Read, and Follow: Score Following as a Reinforcement Learning Game.

[7] Chou, P., Lin, F., Chang, K. and Chen, H. (2018) A Simple Score Following System for Music Ensembles Using Chroma and Dynamic Time Warping. *Proceedings of the* 2018 *ACM on International Conference on Multimedia Retrieval*, Yokohama, 11-14 June 2018, 529-532. https://doi.org/10.1145/3206025.3206090

[8] Oramas, S., Nieto, O., Barbieri, F. and Serra, X. (2017) Multi-Label Music Genre Classification from Audio, Text, and Images Using Deep Features.

[9] Goto, M. and Dannenberg, R.B. (2019) Music Interfaces Based on Automatic Music Signal Analysis: New Ways to Create and Listen to Music. *IEEE Signal Processing Magazine*, **36**, 74-81. https://doi.org/10.1109/msp.2018.2874360

[10] Schedl, M. (2017) Intelligent User Interfaces for Social Music Discovery and Exploration of Large-Scale Music Repositories. *Proceedings of the* 2017 *ACM Workshop on Theory-Informed User Modeling for Tailoring and Personalizing Interfaces*, Limassol, 13 March 2017, 7-11. https://doi.org/10.1145/3039677.3039678

[11] Schedl, M., Zamani, H., Chen, C., Deldjoo, Y. and Elahi, M. (2018) Current Challenges and Visions in Music Recommender Systems Research. *International Journal of Multimedia Information Retrieval*, **7**, 95-116. https://doi.org/10.1007/s13735-018-0154-2

[12] Mayer, R. and Rauber, A. (2011) Musical Genre Classification by Ensembles of Audio and Lyrics Features. *Proceedings of International Conference on Music Information Retrieval*, Miami, 24-28 October 2011, 675-680.

[13] Sturm, B.L. (2013) Classification Accuracy Is Not Enough: On the Evaluation of Music Genre Recognition Systems. *Journal of Intelligent Information Systems*, **41**, 371-406. https://doi.org/10.1007/s10844-013-0250-y

[14] Huq, A., Bello, J.P. and Rowe, R. (2010) Automated Music Emotion Recognition: A Systematic Evaluation. *Journal of New Music Research*, **39**, 227-244. https://doi.org/10.1080/09298215.2010.513733

[15] Yang, Y. and Chen, H.H. (2012) Machine Recognition of Music Emotion: A Review. *ACM Transactions on Intelligent Systems and Technology*, **3**, 1-30. https://doi.org/10.1145/2168752.2168754

[16] Schedl, M. (2019) Deep Learning in Music Recommendation Systems. *Frontiers in Applied Mathematics and Statistics*, **5**, Article ID: 457883. https://doi.org/10.3389/fams.2019.00044

[17] Deldjoo, Y., Schedl, M. and Knees, P. (2024) Content-Driven Music Recommendation: Evolution, State of the Art, and Challenges. *Computer Science Review*, **51**, Article ID: 100618. https://doi.org/10.1016/j.cosrev.2024.100618

[18] Bell, R.M. and Koren, Y. (2007) Lessons from the Netflix Prize Challenge. *ACM SIGKDD Explorations Newsletter*, **9**, 75-79. https://doi.org/10.1145/1345448.1345465

[19] Logan, B. (2004) Music Recommendation from Song Sets. 5*th International Conference on Music Information Retrieval*, Barcelona, 10-14 October 2004, 425-428.

[20] Yadav, V., Shukla, R., Tripathi, A. and Maurya, A. (2021) A New Approach for Movie Recommender System Using K-Means Clustering and PCA. *Journal of Scientific & Industrial Research*, **80**, 159-165.

[21] Mukhopadhyay, S., Kumar, A., Parashar, D. and Singh, M. (2024) Enhanced Music Recommendation Systems: A Comparative Study of Content-Based Filtering and K-Means Clustering Approaches. *Revue d'Intelligence Artificielle*, **38**, 365-376. https://doi.org/10.18280/ria.380138

[22] Katarya, R. and Verma, O.P. (2017) Effectual Recommendations Using Artificial Algae Algorithm and Fuzzy C-Mean. *Swarm and Evolutionary Computation*, **36**, 52-61. https://doi.org/10.1016/j.swevo.2017.04.004

[23] Jiang, L. (2023) A Fuzzy Clustering Approach for Cloud-Based Personalized Distance Music Education and Resource Management. *Soft Computing*, **28**, 1707-1724. https://doi.org/10.1007/s00500-023-09525-7

[24] Guan, C. and Yuen, K.K.F. (2015) Towards a Hybrid Approach of Primitive Cognitive Network Process and Agglomerative Hierarchical Clustering for Music Recommendation. *Proceedings of the* 11*th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Taipei, 19-20 August 2015, 206-209. https://doi.org/10.4108/eai.19-8-2015.2261344

[25] Chang, H., Huang, S. and Wu, J. (2016) A Personalized Music Recommendation System Based on Electroencephalography Feedback. *Multimedia Tools and Applications*, **76**, 19523-19542. https://doi.org/10.1007/s11042-015-3202-4

[26] Vembu, S. and Baumann, S. (2005) A Self-Organizing Map Based Knowledge Discovery for Music Recommendation Systems. In: Wiil, U.K., Ed., *Computer Music Modeling and Retrieval*, Springer, 119-129. https://doi.org/10.1007/978-3-540-31807-1_9

[27] Hartono, P. and Yoshitake, R. (2013) Automatic Playlist Generation from Self-Organizing Music Map. *Journal of Signal Processing*, **17**, 11-19. https://doi.org/10.2299/jsp.17.11

[28] Hesmondhalgh, D., Campos Valverde, R., Kaye, D. and Li, Z. (2023) The Impact of Algorithmically Driven Recommendation Systems on Music Consumption and Production: A Literature Review. UK Centre for Data Ethics and Innovation Reports.