On Competitiveness of Nearest-Neighbor-Based Music Classification: A Methodological Critique

Haukur Pálmason[1], Björn Þór Jónsson[1,2], Laurent Amsaleg[3], Markus Schedl[4], and Peter Knees[5]

[1] Reykjavík University, Iceland
[2] IT University of Copenhagen, Denmark
[3] CNRS–IRISA, Rennes, France
[4] Johannes Kepler University Linz, Austria
[5] TU Wien, Austria
bjorn@ru.is

Abstract. The traditional role of nearest-neighbor classification in music classification research is that of a straw man opponent for the learning approach of the hour. Recent work in high-dimensional indexing has shown that approximate nearest-neighbor algorithms are extremely scalable, yielding results of reasonable quality from billions of high-dimensional features. With such efficient large-scale classifiers, the traditional music classification methodology of reducing both feature dimensionality and feature quantity is incorrect; instead the approximate nearest-neighbor classifier should be given an extensive data collection to work with. We present a case study, using a well-known MIR classification benchmark with well-known music features, which shows that a simple nearest-neighbor classifier performs very competitively when given ample data. In this position paper, we therefore argue that nearest-neighbor classification has been treated unfairly in the literature and may be much more competitive than previously thought.

Keywords: Music classification; Approximate nearest-neighbor classifiers; Research methodology.

1 Introduction

The traditional role of nearest neighbor classification in music information retrieval (MIR) research is that of a straw man opponent: after the music features have been fine-tuned to optimize the results of the learning approach of the day, which typically means reducing both the dimensionality and quantity of the features, the k-NN classifier is then applied to those features directly, inevitably losing. We argue, however, that as the strength of k-NN classification lies precisely in the ability to handle a large quantity of data efficiently, this methodology is grossly misleading.

1.1 Trends in High-Dimensional Indexing

In the music domain, Schnitzer [22] developed a content-based retrieval system that operates on a collection of 2.3 million songs and can answer audio queries

in a fraction of a second by using a filter-and-refine strategy. As state-of-the-art content description models build upon high-dimensional Gaussian distributions with costly similarity calculations, an approximate projection into a vector space is used to perform high-speed nearest neighbor candidate search, before ranking candidates using the expensive model. Using the efficiency of k-NN, the combined approach speeds up queries by a factor of 10–40 compared to a linear scan.

In the image retrieval domain, recent work in high-dimensional indexing has shown that approximate nearest-neighbor algorithms are extremely scalable, as several approaches have considered feature collections with up to several billions of features. Jégou et al. [9] proposed an indexing scheme based on the notion of product quantization and evaluated its performance by indexing 2 billion vectors. Lejsek et al. [12] described the NV-tree, a tree index based on projections and partitions, with experiments using 2.5 billion SIFT feature vectors. Babenko and Lempitsky used the inverted multi-index to index 1 billion SIFT features [2] and deep learning features [3] on a single server. Finally, the distributed computing frameworks Hadoop and Spark have been used with collections of up to 43 billion feature vectors [8, 14]. Evaluation of the quality of these approaches shows that approximate nearest neighbor retrieval yields results of reasonable quality. In particular, many studies have shown that while results from individual queries may lack quality, applications with redundancy in the feature generation usually return results of excellent quality (e.g., see [12, 14]). These results indicate that in order to fairly evaluate k-NN classification as an independent methodology, we must supply the classifiers with massive collections of highly redundant features, and then aggregate the results to classify individual items [5].

1.2 Critique of Methodology

In this position paper, we present a case study of k-NN classification using a well-known MIR genre classification benchmark with well-known music features from the literature. Unlike previous studies, however, we generate an extensive collection of music features to play to the strength of k-NN classifiers. We first use an exhaustive (exact) k-NN classifier, which scans the entire feature collection sequentially, to tune the parameters of the feature generation. Then we use a very simple approximate k-NN classifier to show that equivalent results can be obtained in a fraction of the time required for the exhaustive search.

The results of this case study show that the approximate k-NN classifier strongly outperforms any k-NN classification results reported in the literature. Furthermore, the approximate k-NN classifier actually outperforms learning-based results from the literature obtained using this particular music collection and these particular music features. We therefore argue that nearest-neighbor-based classification is much more competitive that previously thought. Furthermore, we argue that when evaluating the quality of k-NN classification, the evaluators must work with the strengths of k-NN classification, namely scalability and efficiency, for a fair comparison. Anything else is much like inviting a fish to a tree-climbing competition with a monkey!

2 Music Classification: A (Very Brief) Literature Review

Automatic genre classification of music has received considerable attention in the MIR community in the last two decades, albeit partly (and recently) in the form of criticism [26, 27, 11]. Due to space limitations, we only survey a few key works of content-based methods that are particularly relevant to our case study. For a more detailed discussion on the general subject we refer the reader to [27], and for a review of context/web-based methods to [10].

When talking about the performance of genre classification approaches, it is notable that results reported in literature are realized on very different datasets with different numbers of classes/genres, as there is (for valid reasons) no commonly agreed upon standard dataset for this task (cf. [1, 17, 6]).For instance, the ISMIR2004 dataset partitions its 1,458 full song examples into 6 genre classes, while the the USPOP collection used in the MIREX 2005 Genre Classification set comprises 8,746 tracks by 400 artists classified into 10 genres (of which 293 artists belong to "Rock") and 251 styles. The seminal work of Tzanetakis and Cook [28] uses a dataset with 10 musical genres, and proposes to classify songs based on timbral features, rhythmic content, and pitch content. Their collection (referred to as GTZAN) has since been used by several researchers, and is indeed the collection we use for the experiments in this paper.[1]

Tzanetakis and Cook [28] used a 30-dimensional feature vector extracted from each of the 30-second-song-excerpts in their 1,000-excerpt dataset. An accuracy of 61% is reported using a Gaussian Mixture Model classifier and 60% using k-NN. Li et al. [13] used the same features as Tzanetakis and Cook, with the addition of Daubechies Wavelet Coefficient Histograms (DWHC). Wavelets decompose the audio signal based on octaves with each octave having different characteristics. Support Vector Machine (SVM) classification yielded 78.5% classification accuracy on GTZAN while only 62.1% was achieved using k-NN. One feature vector per song was used for the experiments. Instead of using just one vector for each song, Bergstra et al. [4] extract features in frames of 1,024 samples and aggregate frames into segments before using AdaBoost for classification, yielding 82.5% accuracy on GTZAN. They state that the best results are achieved by aggregating between 50 and 100 frames for a segment. In the MIREX 2005 Audio Genre Classification competition, however, this setting could not be pursued as calculation for all songs would have taken longer than the allowed 24 hours. Panagakis et al. [19] achieved 78.2% accuracy using "multiscale spectro-temporal modulation features," where each audio file is converted to a third-order feature tensor generated by a model of auditory cortical processing. Non-negative matrix factorization is used for dimensionality reduction and

[1] It is known that the GTZAN collection exhibits inconsistencies, repetitions, and mislabeling, as confirmed by Tzanetakis and investigated in detail by Sturm [25]. From our own investigation of the dataset, with the help of musical experts, we gained similar insights and tried to "correct" the ground truth [18]. As classification results between the original and the "corrected" ground truth did not change drastically, we decided to refer in this work to the original ground truth to allow for comparability of results with the literature.

SVM for classification. The highest accuracy results for GTZAN are achieved by Panagakis et al. [20] with 91% accuracy. The authors extract auditory temporal modulation representation of each song and examine several classifiers. Their best results are achieved using sparse representation-based classification (SRC), while their results using k-NN are below 70% for GTZAN. Seyerlehner et al. [24] propose to compute several spectrum- and rhythm-based features from cent-scaled audio spectrograms on the level of blocks. Each song is split into a number of such (overlapping) blocks, the block size being dependent on the actual feature to compute. Training a SVM, they obtain an accuracy of 78%, but when replacing the SVM with a Random Forest classifier, the authors are able to achieve a classification accuracy of 87% [23].

Recent approaches for genre recognition, music similarity, and related retrieval tasks, aim at modeling finer grained musical concepts, also as an intermediate step for genre classification [21], or try to optimize audio features using user preference patterns by means of deep learning [15]. While these approaches are specifically devised with scalability of indexing in mind, they cannot be compared to existing work due to the proprietary industrial data used. In an effort to make large-scale learning approaches comparable, van den Oord et al. [16] apply deep learning models, trained using the Million Song Dataset, to classification of GTZAN (transfer learning), achieving 88.2% accuracy.

3 Case Study: Genre Classification

In this section, we report on our classification experiments using two k-NN classifiers, one exact and one approximate. We start by describing the experimental setup. Then we detail experiments using a full (exact) sequential scan to analyze various aspects of the music feature configuration. We then use the best feature configuration to study the impact of the approximate k-NN classifier.

3.1 Experimental Setup

For our experiments, we used the MARSYAS framework for feature extraction [28] that accompanies the GTZAN collection. As the purpose of this case study is not to create the best classifier, or generate the best classification results, but rather to point out methodological issues in previous studies, we chose to use a collection that is a) extensively studied in the literature, and b) comes with easily available and flexible feature extraction software.

The GTZAN collection consists of 1,000 song excerpts, where each excerpt is 30 seconds long. Each song is sampled at 22,050 KHz in mono. The songs are evenly distributed into 10 genres: Blues, Classical, Country, Disco, HipHop, Jazz, Metal, Pop, Reggae and Rock. We used randomized and stratified 10-fold cross-validation, by a) shuffling songs within each genre, and then concatenating the genre files, and b) by ignoring, when computing distances, all feature vectors from songs that are located within the same 10% of the collection as the query. We ignore the first feature vectors of each song, due to overlap between songs. All experiments were performed on an Apple MacBook Pro with 2.66GHz Duo Core Intel processors, 4GB RAM and a 300GB 5.4Krpm Serial ATA hard disk.

Table 1. Effect of feature selection on accuracy

Features	**Dim.**	**Accuracy** (%)
TF	34	75.4
TF + SFM	82	80.4
TF + SFM + SCF	130	80.0
TF + SFM + LSP	118	80.0
TF + SFM + LPCC	106	79.8
TF + SFM + CHROMA	106	79.4

Table 2. Effect of varying k

Neighbors (k)	**Accuracy** (%)	**Time** (min)
1	80.4	207.4
2	80.7	208.3
3	80.8	209.3
4	80.6	210.1
5	80.5	212.3

3.2 Exact Classification: Impact of Feature Parameters

The first k-NN classifier performs a sequential scan of all database descriptors and calculates the Manhattan distance between each query vector and each database vector.[2] Once the scan has computed the nearest neighbors for all query feature vectors, the total score for the query song is aggregated, by counting how many neighbors "vote" for each genre and returning the ranked list of genres.

For feature extraction, we used window and hop sizes of 512 and a memory size of 40, as they performed best in experiments. For feature selection, we started from timbral features (TF), namely MFCCs, and added additional features; the results are summarized in Table 1. The table shows that adding spectral flatness measure (SFM) increases the accuracy for this test set from 75.4% to 80.4%. Adding further information to the feature vector, however, does not improve results, and in fact actually hurts accuracy slightly. In the remainder of this study, we hence use the TF+SFM features, with a dimensionality of 82.

Finally, we experiment with the k parameter: how many neighbors from the collection are considered for each query feature. We ran the sequential scan with k ranging from 1 to 10. Table 2 shows the results (no further change is observed beyond $k = 5$). As the table shows, varying k does not affect the classification accuracy much, nor does it have significant impact on the classification time.

3.3 Approximate Classification: Impact of eCP Parameters

Having reached a respectable 80.8% accuracy in 3.5 hours using a sequential scan, we now turn our attention to reducing the classification time by employing the eCP high-dimensional indexing method [7]. Very briefly, the algorithm works as follows. For index construction, C points are randomly sampled from the feature collection as cluster representatives. Each feature vector is then assigned to the closest representative, and the resulting clusters are stored on disk. In order to facilitate efficient assignment, as well as efficient search, a cluster index is created by recursively sampling and clustering the representatives. At search time, the b clusters that are closest to each query feature are found via the cluster index, and then scanned to find the closest k neighbors. The approximation comes from

[2] Manhattan distance can be implemented very efficiently using fast intrinsic SSE code and was therefore selected over Euclidean distance.

Table 3. Effect of varying number of clusters C

Clusters (C)	**Features** (average per cluster)	**Clustering** (sec)	**Classification** (sec)	**Accuracy** (%)
50	25,867	7	784	80.0
100	12,933	7	470	80.6
500	2,587	7	114	80.2
1,000	1,293	7	55	78.4
5,000	259	9	15	79.7

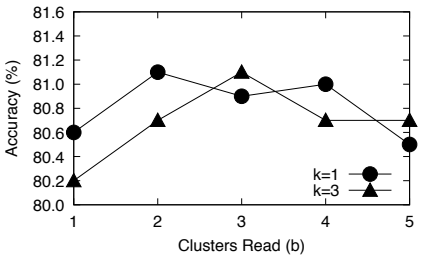

Fig. 1. Effect of b on accuracy

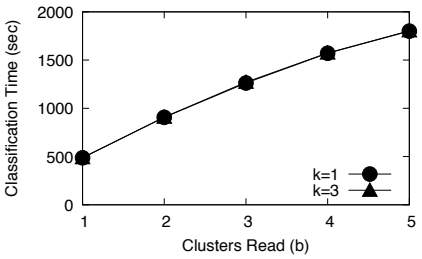

Fig. 2. Effect of b on classification time

the fact that only b clusters are scanned (retrieving a fraction b/C of the feature collection, on average), and some neighbors may be missed, because they are close to a boundary or due to the approximate nature of the index tree traversal.

Table 3 shows the impact of the number of clusters C on clustering time, classification time and classification accuracy. In this experiment $b = k = 1$. We observe that even with very small clusters (larger C), where only about 260 features are read on average, the classification accuracy is nearly the same as with the sequential scan. The classification time, however, is in the order of a few minutes or less, compared to over 200 minutes for the sequential scan.

Figure 1 shows the impact of b on accuracy (note the narrow y-axis), for $C = 100$ and $k = \{1, 3\}$. In other configurations, not reported here due to space constraints, similar effects were observed. When b is increased, the eCP algorithm retrieves and analyses more feature vectors. When the first clusters are added, results are improved. As more clusters are added, however, near vectors belonging to other genres are also found, but no change is observed beyond $b = 5$. These fluctuations are due to the fact that eCP is an approximate technique and since songs quite often receive a similar number of votes from two genres, limiting the number of audio features retrieved can have this effect. But with several configurations we actually achieve higher accuracy than with the sequential scan.

Figure 2, on the other hand, shows the time required to retrieve the clusters. As the figure shows, there is some initial time needed to rank the cluster representatives, but after that the cost of reading clusters grows linearly. The cost, however, is not affected significantly by the k parameter.

Table 4. Comparison to some genre classification results for the GTZAN collection

Reference	**Accuracy**	***k*-NN**	**Comments**
Panagakis et al., 2009 [20]	91.0%	≃68%	Sparse repr. + dim. reduction
van den Oord et al., 2014 [16]	88.2%	–	Transfer learning from MSD tags
Seyerlehner et al., 2011 [23]	87.0%	–	Random forest, optimized feat.
This work	*80–81%*		*k-NN classifier + MFCC + SFM*
Li et al., 2003 [13]	78.5%	62.1%	SVM
Panagakis et al., 2008 [19]	78.2%	–	Multilinear approach
Tzanetakis & Cook, 2002 [28]	61.0%	60.0%	Gaussian classifier + MFCC

3.4 Discussion

The key result is that using approximate k-NN classification we can very efficiently obtain accuracies in the range 80-81%. Table 4 summarizes classification results from the literature for the used GTZAN collection, as well as k-NN classification results where reported. As the table shows, the results from our case study rank fourth and far outperform any reported k-NN classification results from the literature. While previous results have denounced k-NN classification, this is presumably because those studies focused on using a single feature, or very few features, for each song. The fact that we can obtain these results within minutes, on very limited hardware, leads us to believe that k-NN classification is a viable option at a large scale.

4 Conclusions

Very efficient and scalable approximate algorithms for high-dimensional k-NN retrieval have recently been developed in the computer vision and databases communities. Using one such approximate algorithm, we have shown that k-NN classification may be used with good results for music genre classification, if provided the appropriate feature collection to work with. Indeed, the reported results are the best realized with a k-NN classifier for this particular music collection (GTZAN). Observing the difference between our results and previous results reported for k-NN classification points to a methodological problem with previous studies: the k-NN classifiers were compared unfairly to the competing approaches. We thus argue that when evaluating the quality of k-NN classification, the evaluators must work with the strengths of k-NN classification, namely scalability and efficiency, for a fair comparison.

References

1. Aucouturier, J.J., Pachet, F.: Representing musical genre: A state of the art. Journal of New Music Research 32(1), 83–93 (2003)
2. Babenko, A., Lempitsky, V.S.: The inverted multi-index. TPAMI 37(6) (2015)
3. Babenko, A., Lempitsky, V.S.: Efficient indexing of billion-scale datasets of deep descriptors. In: Proc. CVPR. Las Vegas, NV, USA (2016)

4. Bergstra, J., Casagrande, N., Erhan, D., Eck, D., Kégl, B.: Aggregate features and ADABOOST for music classification. Machine Learning 65(2-3) (2006)
5. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: Proc. CVPR (2008)
6. Fabbri, F.: A theory of musical genres: Two applications. Popular music perspectives 1, 52–81 (1981)
7. Guðmundsson, G.Þ., Amsaleg, L., Jónsson, B.Þ.: Distributed high-dimensional index creation using Hadoop, HDFS and C++. In: Proc. CBMI (2012)
8. Guðmundsson, G.Þ., Amsaleg, L., Jónsson, B.Þ., Franklin, M.J.: Towards engineering a web-scale multimedia service: A case study using Spark. In: Proc. MMSys. Taipei, Taiwan (2017)
9. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. TPAMI 33(1) (2011)
10. Knees, P., Schedl, M.: A survey of music similarity and recommendation from music context data. ACM TOMCCAP 10(1) (2013)
11. Knees, P., Schedl, M.: Music Similarity and Retrieval — An Introduction to Audio- and Web-based Strategies. Springer, Berlin and Heidelberg, Germany (2016)
12. Lejsek, H., Jónsson, B.Þ., Amsaleg, L.: NV-Tree: Nearest neighbours at the billion scale. In: Proc. ICMR (2011)
13. Li, T., Ogihara, M., Li, Q.: A comparative study on content-based music genre classification. In: Proc. SIGIR. Toronto, Canada (2003)
14. Moise, D., Shestakov, D., Guðmundsson, G.Þ., Amsaleg, L.: Indexing and searching 100M images with Map-Reduce. In: Proc. ICMR. Dallas, TX, USA (2013)
15. van den Oord, A., Dieleman, S., Schrauwen, B.: Deep Content-based Music Recommendation. In: Proc. NIPS (2013)
16. van den Oord, A., Dieleman, S., Schrauwen, B.: Transfer learning by supervised pre-training for audio-based music classification. In: Proc. ISMIR (2014)
17. Pachet, F., Cazaly, D.: A taxonomy of musical genre. In: Proc. RIAO (2000)
18. Pálmason, H., Jónsson, B.Þ., Schedl, M., Knees, P.: Music genre classification revisited: An in-depth examination guided by music experts. under review (2017)
19. Panagakis, I., Benetos, E., Kotropoulos, C.: Music genre classification: A multilinear approach. In: Proc. ISMIR. Philadelphia, PA, USA (2008)
20. Panagakis, Y., Kotropoulos, C., Arce, G.R.: Music genre classification via sparse representations of auditory temporal modulations. In: Proc. EUSIPCO (2009)
21. Prockup, M., Ehmann, A.F., Gouyon, F., Schmidt, E.M., Celma, Ò., Kim, Y.E.: Modeling genre with the music genome project: Comparing human-labeled attributes and audio features. In: Proc. ISMIR (2015)
22. Schnitzer, D.: Indexing Content-Based Music Similarity Models for Fast Retrieval in Massive Databases. Dissertation, Johannes Kepler University, Austria (2012)
23. Seyerlehner, K., Schedl, M., Knees, P., Sonnleitner, R.: A refined block-level feature set for classification, similarity and tag prediction. In: Proc. MIREX. Miami, FL, USA (2011)
24. Seyerlehner, K., Widmer, G., Pohle, T.: Fusing block-level features for music similarity estimation. In: Proc. Digital Audio Effects (DAFx). Graz, Austria (2010)
25. Sturm, B.L.: An analysis of the GTZAN music genre dataset. In: Proc. MIRUM. Nara, Japan (2012)
26. Sturm, B.L.: Classification accuracy is not enough. JIIS 41 (2013)
27. Sturm, B.L.: A survey of evaluation in music genre recognition. In: Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation (2014)
28. Tzanetakis, G., Cook, P.: Musical genre classification of audio signals. IEEE Trans. on Speech and Audio Processing 10(5) (2002)