Fast Time Series Classification with Random Symbolic Subsequences

Thach Le Nguyen and Georgiana Ifrim

School of Computer Science, University College Dublin, Ireland
{thach.lenguyen, georgiana.ifrim}@ucd.ie

Abstract. Symbolic representations of time series have proven to be effective for time series classification, with many recent approaches including BOSS, WEASEL, and MrSEQL. These classifiers use various elaborate methods to select discriminative features from symbolic representations of time series. As a result, although they have competitive results regarding accuracy, their classification models are relatively expensive to train. Most if not all of these approaches have missed an important research question: are these elaborate feature selection methods actually necessary? ROCKET, a state-of-the-art time series classifier, outperforms all of them without utilizing any feature selection techniques. In this paper, we answer this question by contrasting these classifiers with a very simple method, named MrSQM. This method samples random subsequences from symbolic representations of time series. Our experiments on 112 datasets of the UEA/UCR benchmark demonstrate that MrSQM can quickly extract useful features and learn accurate classifiers with the logistic regression algorithm. MrSQM completes training and prediction on 112 datasets in 1.5h for an accuracy comparable to existing efficient state-of-the-art methods, e.g., MrSEQL (10h) and ROCKET (2.5h). Furthermore, MrSQM enables the user to trade-off accuracy and speed by controlling the type and number of symbolic representations, thus further reducing the total runtime to 20 minutes for a similar level of accuracy. With these results, we show that random subsequences extracted from symbolic transformations can be as effective as the more sophisticated and expensive feature selection methods proposed in previous works. We propose MrSQM as a strong baseline for future research in time series classification, especially for approaches based on symbolic representations of time series.

1 Introduction

Symbolic representations of time series are a family of techniques to transform numerical time series to sequences of symbols, and were shown to be more robust to noise and useful for building effective time series classifiers. Two of the most prominent symbolic representations are Symbolic Aggregate Approximation (SAX) [18] and Symbolic Fourier Approximation (SFA) [26]. SAX-based classifiers include BOP [18, 19], FastShapelets [23], SAX-VSM [28]; SFA-based classifiers include BOSS [24], BOSS VS [25] and WEASEL [27]. MrSEQL [17] is

a symbolic classifier which utilizes both SAX and SFA transformations, which further improved the accuracy and speed of classification. Several state-of-the-art ensemble methods, e.g., HIVE-COTE [3, 20, 2] and TS-CHIEF [29], incorporate symbolic representations for their constituent classifiers and are the current state-of-the-art with regard to accuracy.

Symbolic representations of time series enable the adoption of techniques developed for text mining. For example, SAX-VSM, BOSS VS and WEASEL make use of tf-idf vectors and vector space models [28, 25], while MrSEQL is based on a sequence learning algorithm developed for text classification [15]. These apparently different approaches can be summarized as methods of extracting discriminative features from symbolic representations of time series, coupled with a classifier. While achieving high accuracy, the key challenge for symbolic classifiers is to efficiently select good features from a large feature space. For example, even with fixed parameters, a SAX bag-of-words can contain as many as α^w unique words, in which α is the size of the alphabet (the number of distinct symbols) and w is the length of the words. Even for moderate alphabet and word sizes, this feature space grows quickly, e.g., for typical SAX parameters $\alpha = 4, w = 16$, there can be 4 billion unique SAX words. SAX-VSM works with a single optimized SAX representation, but the process for optimizing the SAX parameters is expensive. WEASEL has high accuracy by using SFA unigrams and bigrams but a high memory demand, due to needing to store all the SFA words before applying feature selection. MrSEQL uses the feature space of all subsequences in the training data, in order to find useful features inside SAX or SFA words. It employs greedy feature selection and a gradient bound to quickly prune unpromising features. Despite these computational challenges, these methods are still vastly faster and less resource demanding than most state-of-the-art classifiers, in particular ensembles, e.g., HIVE-COTE, TS-CHIEF, and deep learning models, e.g., InceptionTime [10].

Recently, these symbolic classifiers had been outperformed by ROCKET [6] and MiniROCKET [7], a family of methods that uses random kernels to extract features from raw time series. Rather than finding the most discriminative features, ROCKET generates 10,000 random kernels regardless of their discriminative strength. As a result, while they also combine large feature spaces with linear classifiers, the ROCKET methods are faster and more accurate than the existing symbolic classifiers. This intriguing observation raises a simple question: are the expensive feature selection techniques employed by the previous symbolic classifiers (e.g., SAX-VSM, BOSS, WEASEL, MrSEQL) necessary? This question has inspired us to re-examine fast symbolic transformations, feature selection and linear classifiers for working with symbolic representations of time series. We thus propose MrSQM, a simple time series classifier which samples random subsequences from symbolic representations of time series. Our experiments on the UEA/UCR benchmark show that, even without any feature selection method, a large enough number of random symbolic subsequences can be as effective for learning accurate classifiers. More importantly, the method is significantly faster than most of its counterparts: it can complete training and prediction on 112

datasets in 1.5h, for an accuracy comparable to existing efficient state-of-the-art methods, e.g., MrSEQL (10h) and ROCKET (2.5h). Furthermore, MrSQM enables the user to trade-off accuracy and speed by controlling the type and number of symbolic representations.

Our main contributions in this paper are as follows:

- We propose **Multiple Representations Sequence Miner (MrSQM)**, a new symbolic time series classifier which builds on multiple symbolic representations, random sequence mining and a linear classifier, to achieve high accuracy with reduced computational cost.
- We present an extensive empirical study comparing the accuracy and runtime of MrSQM to recent state-of-the-art time series classifiers on 112 datasets of the new UEA/UCR TSC benchmark [1]. Our study demonstrates that random subsequences can be as effective as complex feature selection methods proposed in previous works on symbolic approaches.
- All our code and data is publicly available to enable reproducibility of our results[1]. Our code is implemented in Python and C++ (in Python wrappers) and a Python Jupyter Notebook with detailed examples to make the implementation more widely accessible.

The rest of the paper is organised as follows. In Section 2 we discuss the state-of-the-art in time series classification research. In Section 3 we describe our research methodology. In Section 4 we present an empirical study with a detailed sensitivity analysis for our methods and a comparison to state-of-the-art time series classifiers. We conclude in Section 5.

2 Related Work

The state-of-the-art in time series classification (TSC) has evolved rapidly with many different approaches contributing to improvements in accuracy and speed. The main baseline for TSC is 1NN-DTW [3], a one Nearest-Neighbor classifier with Dynamic Time Warping as distance measure. While this baseline is at times preferred for its simplicity, it is not very robust to noise and has been significantly outperformed in accuracy by more recent methods. Some of the most successful TSC approaches typically fall into the following three groups.

Ensemble Classifiers aggregate the predictions of many independent classifiers. Each classifier is trained with different data representations and feature spaces, and the individual predictions are weighted based on the quality of the classifier on validation data. HIVE-COTE [20] is the most popular example of such an approach. It is an evolution of the COTE [3] ensemble and it is still currently the most accurate TSC approach. While being very accurate, this method's runtime is bound to the slowest of its component classifiers. Recent work [2, 21] has proposed techniques to make this approach more usable by improving its runtime, but it still requires more than two weeks to train on the new

[1] `https://github.com/mlgig/mrsqm`

UEA/UCR benchmark which has a moderate size of about 300Mb. TS-CHIEF [29] is another recent ensemble which only uses decision tree classifiers. It was proposed as a more scalable alternative to HIVE-COTE, but still takes weeks to train on the UEA/UCR benchmark. This makes the reproducibility of results with these methods challenging.

Deep Learning Classifiers were recently proposed for time series data and analysed in an extensive empirical survey [16]. Methods such as Fully Convolutional Networks (FCN) and Residual Networks (Resnet) were found to be highly effective and achieve accuracy comparable to HIVE-COTE. One issue with such approaches is their tendency to overfit with small training data and to have a high variance in accuracy. In order to create a more stable approach InceptionTime [10] was proposed to ensemble five deep learning classifiers. InceptionTime achieves an accuracy comparable to HIVE-COTE, but requires vast computational resources and requires days to train on the same benchmark [10, 2].

Linear Classifiers were recently shown to work well for time series classification. Given a large feature space, the need for further feature expansion and learning non-linear classifiers is reduced. This idea was incorporated very successfully for large scale classification in libraries such as LIBLINEAR [9]. In the context of TSC, this idea was first incorporated by classifiers such as WEASEL [27], which creates a large SFA-words feature space, filters it with Chi-square feature selection, then learns a logistic regression classifier. Another linear classifier, MrSEQL [17], uses a large feature space of SAX and SFA subwords, which is filtered using greedy gradient descent and logistic regression. A recent classifier ROCKET [6] generates many random convolutional kernels and uses max pooling and a feature called *ppv* to capture good features from the time series. ROCKET uses a large feature space of 20,000 features (default settings) associated with the kernels, and a linear classifier (logistic regression or ridge regression). MiniROCKET [7] is a recent extension of ROCKET with comparable accuracy and faster runtime. These approaches were shown to be as accurate as ensembles and deep learning for TSC, but are orders of magnitude faster to train [17, 6, 21]. MrSEQL can train on the UEA/UCR benchmark in 10h, while ROCKET has further reduced this time to 2.5h. Another advantage of these methods is their conceptual simplicity, since the method can be broken down into three stages: (1) transformation (e.g., symbolic for WEASEL and MrSEQL, or convolutional kernels for ROCKET), (2) feature selection and (3) linear classifier. Intuitively, these methods extract many shapelet-like features from the training data, and use the linear classifier to learn weights to filter out the useful features from the rest. While there is a vast literature on shapelet-learning techniques, e.g., [30, 23, 14, 3, 2] these recent linear classification methods were shown to be more accurate and faster than other shapelet-based approaches. In particular, the SFA transform does not require data normalisation (which may harm accuracy for some problems), it was shown to be robust to noise [24], and has very fast implementations which build on the past 20 years of work on speeding up the computation of Discrete Fourier Transform [11, 12].

Based on these observations and the success of symbolic transforms and linear classifiers, we focus our work on designing and evaluating new TSC methods built on large symbolic feature spaces and efficient linear classifiers.

3 Proposed Method

The MrSQM time series classifier has three main building blocks: (1) **symbolic transformation**, (2) **feature transformation** and (3) **learning algorithm** for training a classifier. In the first stage, we transform the numerical time series to multiple symbolic representations using either SAX or SFA transforms. We carefully analyse the impact of parameter selection for the symbolic transform, as well as integrate fast transform implementations, especially for the Discrete Fourier Transform in SFA. For the second stage, random subsequences are sampled from the symbolic representations. Each subsequence then becomes a feature for model training. The value of the feature is a binary value: 1 if the subsequence can be found in the symbolic representation of the sample and 0 if not. The output of this stage is transformed data in tabular form. For the third stage, we employ an efficient linear classifier based on logistic regression. While the choice of the learning algorithm does not depend on the previous two stages, we select logistic regression for its scalability, accuracy and the benefit of model transparency and calibrated prediction probabilities, which can benefit some follow up steps such as classifier interpretation. For example, as done in the MrSEQL approach [17], the symbolic features selected by the logistic regression model can be mapped back to the time series to compute a saliency map explanation for the classifier prediction. A schematic representation of the MrSQM approach is given in Figure 1.

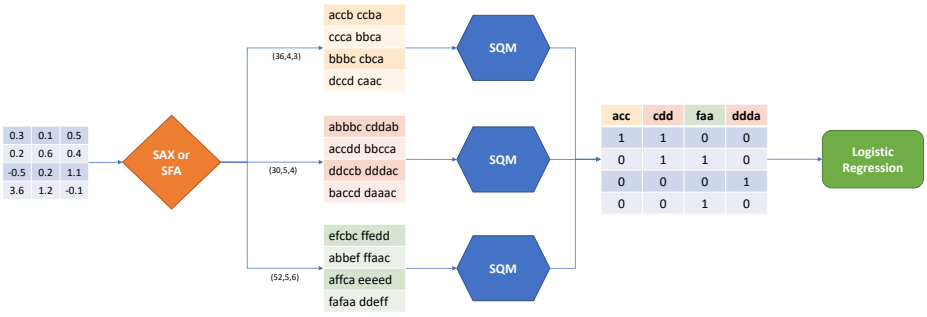

Fig. 1. Workflow for the MrSQM time series classifier with 3 stages: 1. symbolic transform, 2. feature transform, 3. classifier learning.

3.1 Symbolic Representations of Time Series

While SAX and SFA are two different techniques to transform time series data to symbolic representations, both can be summarized in three steps:

- Use a sliding window to extract segments of time series (parameter l: window size).
- Approximate each segment with a vector of smaller or equal length (parameter w: word size).
- Discretise the approximation to obtain a symbolic word (e.g., *abbacc*; parameter α: alphabet size).

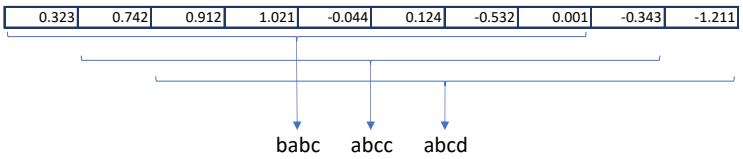

Fig. 2. Example symbolic transform using a sliding window over the time series.

As a result, the output of transforming a time series is a sequence of symbolic words (e.g., *abbacc aaccdd bbacda aacbbc*). Figure 2 shows an example symbolic transform applied to a time series, and the resulting sequence of symbolic words.

The main differences between SAX and SFA are the approximation and discretisation techniques, which are summarized in Table 1. The SAX transform works directly on the raw numeric time series, in the time domain, using an approximation called Piecewise Aggregate Approximation (PAA). The SFA transform builds on the Discrete Fourier Transform (DFT), followed by discretisation in the frequency domain. Hence these two symbolic transforms should capture different types of information about the time series structure. Each transform results in a different symbolic representation, for a fixed set of parameters (l, w, α). This means that for a given type of symbolic transform (e.g., SAX), we can obtain multiple symbolic representations by varying these parameters. This helps in capturing the time series structure at different granularity, e.g., by varying the window size l the symbolic words capture more detailed or higher level information about the time series.

MrSQM generates $k \times log(L)$ representations by randomly sampling values for (l, w, α) from a range of values, as shown in Table 2. Parameter $k \in \mathbb{N}, k > 0$ is a controlling parameter that can be set by the user. In comparison, the MrSEQL classifier creates approximately $\sqrt{L}$ symbolic representations for each time series

Table 1. Steps for SAX and SFA symbolic transforms. N is the number of time series and L is the length of time series. Piecewise Aggregate Approximation (PAA) is used in SAX and Discrete Fourier Transform (DFT) is used in SFA.

	SAX	SFA
Approximation	PAA	DFT
Discretisation	equi-probability bins	equi-depth bins
Complexity	$\mathcal{O}(NL)$	$\mathcal{O}(NL \log L)$

Table 2. An example of parameter sampling for a dataset of time series length $L = 64$.

	MrSQM	MrSEQL
window size	$2^{3+i/k}$ for i in $(0, 1, \ldots log(L))$	16,24,32,40,48,54,60,64
word length	6,8,10,12,14,16	16
alphabet size	3,4,5,6	4

(where L is the length of the time series). This new sampling strategy helps MrSQM to scale better for long time series. Moreover, MrSQM samples the window size using an exponential scale, i.e., it tends to choose smaller windows more often, while MrSEQL gives equal importance to windows of all sizes (see Table 2 for an example).

3.2 MrSQM Variants

We create two variants for MrSQM. The first variant (MrSQM-R) is the base variant with three stages of transforming and learning as described in Figure 1. The second variant (MrSQM-RS) includes an extra step of feature selection. After the features are extracted from each representation, a feature selection module (*SelectKBest* from the *sklearn* library) ranks the features according to their importance. Only the most important features are kept for the later stage. While the number of features per representation is also configurable, in our experiments we set it so both variants produce 500 features from each representation for learning.

Time Complexity. All our classifier variants have a time complexity dominated by the symbolic transform time complexity. In our case, the SFA transform, which is $\mathcal{O}(NL \log L)$ is the dominant factor. This is repeated $\mathcal{O}(\log L)$ times (the number of symbolic representations being generated) hence the overall time complexity of MrSQM is $\mathcal{O}(NL(\log L)^2)$. Although SFA has a time complexity of $\mathcal{O}(NL \log L)$, we build our SFA implementation using the latest advances for efficiently computing the Discrete Fourier Transform[2], which results in significant time savings as compared to older SFA implementations.

4 Evaluation

4.1 Experiment Setup

We ran experiments on 112 fixed-length univariate time series classification datasets from the new UEA/UCR TSC Archive. MrSQM also works with variable-length time series, without any additional steps being required (i.e., once it is supported by the input file format). Since the majority of state-of-the-art TSC implementations only support fixed-length time series, for comparison, we have restricted our experiments to fixed-length datasets.

[2] FFTW is an open source C library for efficiently computing the Discrete Fourier Transform (DFT): `https://www.fftw.org`

MrSQM is implemented in Python and C++ (wrapped with Cython). For our experiments we use a Linux workstation with an Intel Core i7-7700 Processor and 32GB memory. To support the reproducibility of results, we have a Github repository[3] with all the code and results. All the datasets used for experiments are available from the UEA/UCR TSC Archive website[4]. We also obtained the accuracy results for some of the existing classifiers from the same website. For the classifiers that we ran ourselves, we have used the implementation provided in the sktime library[5].

For accuracy comparison of multiple classifiers, we follow the recommendation in [8, 13, 4]. The accuracy gain is evaluated using a Wilcoxon signed-rank test with Holm correction and visualised with the critical difference (CD) diagram. The CD shows the ranking of methods with respect to their average accuracy rank computed across multiple datasets. For computing the CD we use the R library *scmamp*[6] [5]. While CDs are a useful visualization tool, they do not tell the full story since minor differences in accuracy can lead to different ranks. We thus supplement the CDs with tables and pairwise scatter plots for a closer look at the accuracy and runtime performance.

4.2 Sensitivity Analysis

Comparing MrSQM Variants. Here, we investigate the two variants of MrSQM (R and RS) in relation to the symbolic transformation (i.e., SAX and SFA). Figure 3 compares four different combinations of MrSQM.

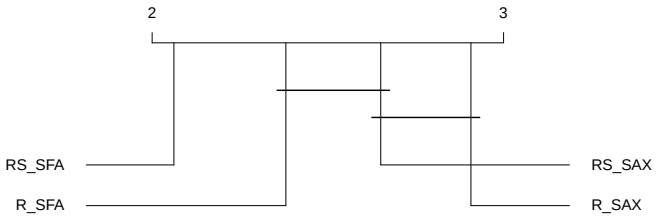

Fig. 3. Comparison of combinations between two variants of MrSQM and two symbolic representations.

Table 3. Time (minutes) comparison for SAX versus SFA combined with different feature selection strategies with $k = 1$.

Symbolic Transform	MrSQM-R	MrSQM-RS
SAX	27	28
SFA	17	22

It is clear from this experiment (Figure 3) that the SFA symbolic transform is generally superior to SAX. On the other hand, feature selection (R versus

[3] `https://github.com/mlgig/mrsqm`
[4] `https://http://timeseriesclassification.com`
[5] `https://www.sktime.org/en/stable/get_started.html`
[6] `https://github.com/b0rxa/scmamp`

RS) appears to be useful with minimal cost (Table 3). All of these variants are very fast, totaling less than 30 minutes for training and predicting on the entire 112 datasets. Since the RS-SFA combination is more accurate than the others, from this point onward it is our default choice for the experiments unless stated otherwise.

Parameter Sampling for the Symbolic Transform. In this set of experiments, we study the impact of the symbolic transformation in terms of both quality and quantity of representations. Figure 4 shows results comparing dif-

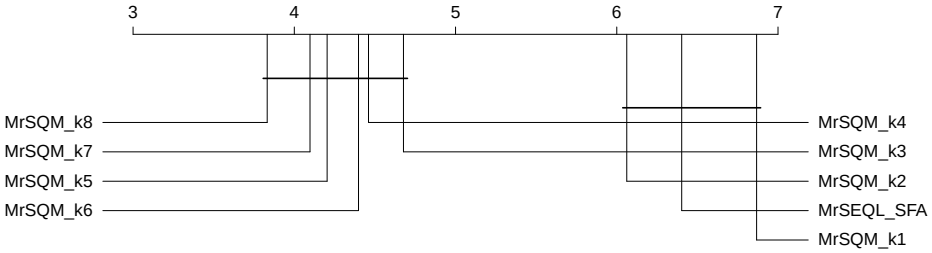

Fig. 4. Comparison of average accuracy rank for MrSQM-SFA variants at variable k and MrSEQL-SFA as baseline.

ferent numbers of SFA representations (with k varying from 1 to 8), when using the RS feature selection strategy for MrSQM. It also includes a comparison to the MrSEQL classifier restricted to only using SFA features, in order to directly compare the accuracy and speed, using the same type of representation. The results show that adding more symbolic representations by varying the control parameter k can benefit MrSQM, albeit with the cost of extra computation reflected in the runtime. In addition, MrSQM at $k = 3$ is already significantly more accurate than MrSEQL, while still being four times faster.

Figure 5 shows a comparison of the accuracy and runtime of MrSQM (for different values for k) and the MrSEQL classifier. Overall, the MrSQM variant with $k = 5$ seems to achieve a good trade-off between accuracy and speed, taking slightly over 100 minutes total time.

Combining SAX and SFA Features. Next we explore the option of combining SAX and SFA feature spaces. The work of [17] found that the combination of SAX and SFA features (with a 1:1 ratio) is very effective for the MrSEQL classifier. For MrSQM, we do not find the same behaviour when combining the two types of representations. Figure 6 shows that the MrSQM variant that only uses the SFA transform is as effective as when using combinations of SAX and SFA representations in different ratios. These results suggest that, to maximise accuracy and speed, the recommended choice of symbolic transformation for MrSQM is SFA. However, it is worth noting that in practice this choice can depend on the requirements of the application. Across the 112 datasets coming from a wide variety of domains, SFA seems to be outperforming the SAX transform in both

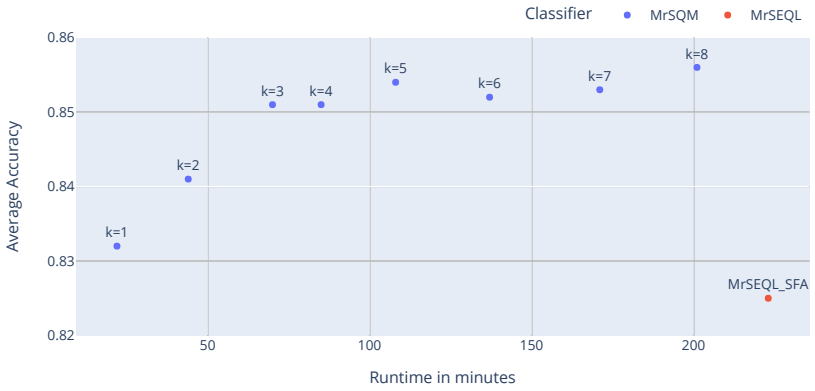

Fig. 5. Comparison of average accuracy and total training and prediction time (minutes) for MrSQM-SFA variants at varying k and MrSEQL variants as baseline.

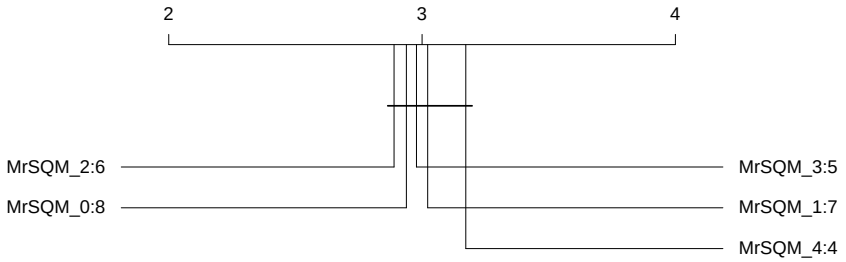

Fig. 6. Comparison between variants of MrSQM with different ratios of SAX and SFA representations. $k_1 : k_2$ means MrSQM generates $k_1 \times log(L)$ SAX representations and $k_2 \times log(L)$ SFA representations.

accuracy and speed. Nevertheless, for many datasets, SAX and SFA models have similar accuracy. Furthermore, like MrSEQL, the MrSQM classifier can produce a saliency map for each time series, from models trained with SAX features. This can be valuable in some scenarios where classifier interpretability is desirable. MrSQM enables the user to select the transform (type and number) that best fits their application scenario.

4.3 MrSQM versus State-of-the-art Symbolic Time Series Classifiers

We compare the best classifier variant for MrSQM (MrSQM_k5 has a good accuracy-time trade-off as shown in Figure 5) with state-of-the-art symbolic time series classifiers. This group includes WEASEL, MrSEQL, BOSS and cBOSS [22].

All five classifiers use SFA representations to extract features, while MrSEQL uses both SAX and SFA representations (Figure 7).

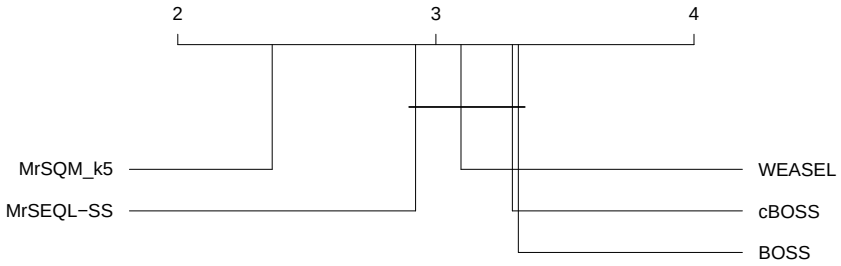

Fig. 7. Comparison of state-of-the-art symbolic time series classifiers across 112 UEA/UCR TSC datasets. The leftmost method has the best average rank.

MrSQM has the highest average rank and is significantly more accurate than the other symbolic classifiers. Furthermore, all the other methods require at least 5-12 hours to train, as shown in Figure 10 and results reported in [2, 17]. We note that the ensemble methods (e.g., BOSS, cBOSS) are outperformed by the linear classifiers. With regard to runtime, as shown in Table 5, MrSQM is significantly faster than the other symbolic classifiers (MrSQM takes 1.5h to complete training and prediction, versus 10h for MrSEQL-SS).

4.4 MrSQM versus other State-of-the-Art Time Series Classifiers

The group of the most accurate time series classifiers that have been published to date include HIVE-COTE, TS-CHIEF, ROCKET (and its extension MiniROCKET), and InceptionTime. With the exception of the ROCKET family, these classifiers are very demanding in terms of computing resources. Running them on 112 UEA/UCR TSC datasets takes days and even weeks to complete training and prediction [2, 21].

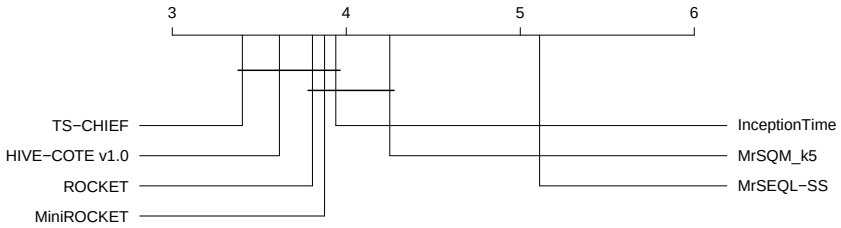

Fig. 8. Comparison with state-of-the-art time series classifiers across 112 UEA/UCR TSC datasets. The leftmost method has the best average rank.

Figure 8 shows the accuracy rank comparison between these methods and MrSQM. Among the methods compared, only TS-CHIEF and HIVE-COTE were found to have a statistically significant difference in accuracy when compared to MrSQM. Nevertheless, these methods require more than 100 hours to complete training [2], for a relatively small gain in average accuracy, typically of about

2% (see Table 4). In this diagram, MrSQM is in the same accuracy group as InceptionTime, MiniROCKET and ROCKET. In terms of runtime, MrSQM is in a group with ROCKET: MrSQM takes 100 minutes to complete training and prediction on 112 datasets, while ROCKET, in our run on the same machine, takes 150 minutes (see more details on runtime in Table 5 and Figure 10). The MrSQM-k1 variant takes only 20 minutes and for many datasets this variant is enough to achieve high accuracy. This variant is comparable in accuracy and runtime to the MiniROCKET classifier. In Figure 10 we show a comparison of some of these methods with regards to the accuracy versus runtime (we only include the methods that we ran ourselves on the same machine).

Figure 9 shows the pairwise-comparison of accuracy between these methods and MrSQM. Each dot in the plot represents one dataset from the benchmark. MrSQM is more accurate above the diagonal line and highly similar methods cluster along the line. We note that the accuracy across datasets is similar for MrSQM versus ROCKET or MiniROCKET, the only other two methods in the same runtime category.

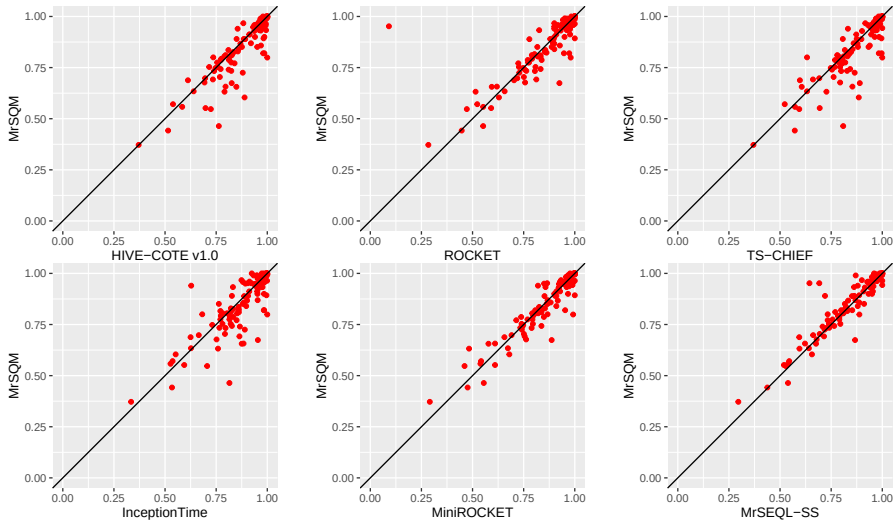

Fig. 9. Pairwise comparison between state-of-the-art time series classifiers and MrSQM with regard to accuracy across 112 UEA/UCR TSC datasets.

In Table 4, we summarize the accuracy differences between MrSQM and the other classifiers. For context, in Table 5 we also provide the runtime for all the methods. We observe that when taken together, the average difference in accuracy and the total time to complete training and prediction over the 112 datasets, we see a clear grouping of methods. If we focus on fast methods that can complete training and prediction in a couple of hours, only the ROCKET/MiniROCKET methods and MrSQM can achieve this. If we look at the average difference in accuracy versus the other methods, there is only about 2% difference in accuracy, for orders of magnitude faster runtime. In the group of symbolic classifiers,

Fig. 10. Comparison of accuracy and total time (minutes) trade-off for MrSQM variants and state-of-the-art methods that complete training within 12 hours.

MrSQM is both significantly more accurate and much faster than existing symbolic classifiers. Furthermore, while it is expected that MrSQM's results are aligned with the other symbolic methods (WEASEL, MrSEQL), it is surprising that they are also very similar to MiniROCKET (second-highest correlation) but not ROCKET (lowest correlation). Perhaps MiniROCKET is better than ROCKET at extracting frequency domain knowledge from time series data.

Table 4. Statistical summary of differences in accuracy between MrSQM and state-of-the-art time series classifiers.

Classifiers	Mean Diff	Std Diff	Correlation
HIVE COTE 1.0	0.028	0.067	0.882
TS-CHIEF	0.026	0.071	0.866
InceptionTime	0.021	0.084	0.816
ROCKET	0.002	0.099	0.797
MiniROCKET	0.007	0.052	0.936
WEASEL	-0.025	0.069	0.91
MrSEQL-SS	-0.011	0.055	0.927
MrSQM_K1	-0.021	0.038	0.97

5 Conclusion

In this paper we have presented MrSQM, a new symbolic time series classifier which works with multiple symbolic representations of time series, fast feature selection for symbolic sequences and a linear classifier. We showed that while conceptually very simple, MrSQM achieves state-of-the-art accuracy on the new UEA/UCR time series classification benchmark, and can complete training and

Table 5. Runtime of state-of-the-art classifiers for training and prediction over 112 datasets. For HIVE-COTE1.0 and TS-CHIEF the time is taken directly from [2].

Classifier	Total (hours)
MiniROCKET	0.1
MrSQM_K1	0.3
MrSQM_K5	1.5
ROCKET	2.5
WEASEL	5
MrSEQL-SS	10
HIVE-COTE1.0	400
TS-CHIEF	600

prediction in under two hours on a regular computer. This compares very favorably to existing methods such as HIVE-COTE, TS-CHIEF and InceptionTime, which achieve only slightly better accuracy, but require days to train on the same datasets and require advanced compute infrastructure. MrSQM is comparable to the recent classifier ROCKET, in regard of both accuracy and speed. This work has shown again that methods from the group of linear classifiers working in large feature spaces are very effective for the time series classification task. For future work we intend to study methods to further reduce the computational complexity of symbolic transformations and extend MrSQM to work on multivariate time series classification.

Acknowledgments

This publication has emanated from research supported in part by a grant from Science Foundation Ireland through the VistaMilk SFI Research Centre (SFI/16/RC/3835) and the Insight Centre for Data Analytics (12/RC/2289_P2).

References

1. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The uea multivariate time series classification archive, 2018. arXiv preprint arXiv:1811.00075 (2018)
2. Bagnall, A., Flynn, M., Large, J., Lines, J., Middlehurst, M.: On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote 1.0). In: Proceedings of the 5th ECML-PKDD Workshop on Advanced Analytics and Learning on Temporal Data (04 2020)
3. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery pp. 1–55 (2016). https://doi.org/10.1007/s10618-016-0483-9, `http://dx.doi.org/10.1007/s10618-016-0483-9`
4. Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks? Journal of Machine Learning Research **17**(5), 1–10 (2016), `http://jmlr.org/papers/v17/benavoli16a.html`

5. Calvo, B., Santafé, G.: scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. The R Journal **8**(1), 248–256 (2016). https://doi.org/10.32614/RJ-2016-017, https://doi.org/10.32614/RJ-2016-017
6. Dempster, A., Petitjean, F., Webb, G.I.: ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Discov. **34**(5), 1454–1495 (2020). https://doi.org/10.1007/s10618-020-00701-z, https://doi.org/10.1007/s10618-020-00701-z
7. Dempster, A., Schmidt, D.F., Webb, G.I.: Minirocket: A very fast (almost) deterministic transform for time series classification. In: Zhu, F., Ooi, B.C., Miao, C. (eds.) KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021. pp. 248–257. ACM (2021). https://doi.org/10.1145/3447548.3467231, https://doi.org/10.1145/3447548.3467231
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (Dec 2006), http://dl.acm.org/citation.cfm?id=1248547.1248548
9. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (Jun 2008)
10. Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P., Petitjean, F.: Inceptiontime: Finding alexnet for time series classification. Data Min. Knowl. Discov. **34**(6), 1936–1962 (2020). https://doi.org/10.1007/s10618-020-00710-y, https://doi.org/10.1007/s10618-020-00710-y
11. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005), special issue on "Program Generation, Optimization, and Platform Adaptation"
12. Frigo, M., Johnson, S.G.: Fastest fourier transform in the west (2021), https://www.fftw.org
13. Garcia, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research **9**, 2677–2694 (12 2008)
14. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 392–401. KDD '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2623330.2623613, http://doi.acm.org/10.1145/2623330.2623613
15. Ifrim, G., Wiuf, C.: Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 708–716. KDD '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2020408.2020519, http://doi.acm.org/10.1145/2020408.2020519
16. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery (Mar 2019). https://doi.org/10.1007/s10618-019-00619-1, https://doi.org/10.1007/s10618-019-00619-1
17. Le Nguyen, T., Gsponer, S., Ilie, I., O'Reilly, M., Ifrim, G.: Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. Data Mining and Knowledge Discovery **33**(4), 1183–1222 (Jul

2019). https://doi.org/10.1007/s10618-019-00633-3, https://doi.org/10.1007/s10618-019-00633-3
18. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery **15**(2), 107–144 (Oct 2007). https://doi.org/10.1007/s10618-007-0064-z, https://doi.org/10.1007/s10618-007-0064-z
19. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**(2), 287–315 (2012). https://doi.org/10.1007/s10844-012-0196-5, http://dx.doi.org/10.1007/s10844-012-0196-5
20. Lines, J., Taylor, S., Bagnall, A.: Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 1041–1046 (Dec 2016). https://doi.org/10.1109/ICDM.2016.0133
21. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.J.: HIVE-COTE 2.0: a new meta ensemble for time series classification. Mach. Learn. **110**(11), 3211–3243 (2021). https://doi.org/10.1007/s10994-021-06057-9, https://doi.org/10.1007/s10994-021-06057-9
22. Middlehurst, M., Vickers, W., Bagnall, A.: Scalable dictionary classifiers for time series classification. In: Yin, H., Camacho, D., Tino, P., Tallón-Ballesteros, A.J., Menezes, R., Allmendinger, R. (eds.) Intelligent Data Engineering and Automated Learning – IDEAL 2019. pp. 11–19. Springer International Publishing, Cham (2019)
23. Rakthanmanon, T., Keogh, E.: Fast shapelets: A scalable algorithm for discovering time series shapelets. In: Proceedings of the thirteenth SIAM conference on data mining (SDM). pp. 668–676. SIAM (2013)
24. Schäfer, P.: The boss is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6), 1505–1530 (2015)
25. Schäfer, P.: Scalable time series classification. Data Mining and Knowledge Discovery **30**(5), 1273–1298 (Sep 2016). https://doi.org/10.1007/s10618-015-0441-y, http://dx.doi.org/10.1007/s10618-015-0441-y
26. Schäfer, P., Högqvist, M.: Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Proceedings of the 15th International Conference on Extending Database Technology. pp. 516–527. EDBT '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2247596.2247656, http://doi.acm.org/10.1145/2247596.2247656
27. Schäfer, P., Leser, U.: Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 637–646. CIKM '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132847.3132980, http://doi.acm.org/10.1145/3132847.3132980
28. Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on. pp. 1175–1180 (Dec 2013). https://doi.org/10.1109/ICDM.2013.52
29. Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.: Ts-chief: a scalable and accurate forest algorithm for time series classification. Data Mining and Knowledge Discovery **34**, 742–775 (2020)
30. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 947–956. ACM (2009)