Improved SAX-VSM in Multivariate Time Series

Jiefeng FANG
jiefeng.fang@polytechnique.edu

This report proposes five techniques to ameliorate SAXVSM [1] to solve time series classification (TSC) problem and three versions of SAXVSM (INDEPEND, CONJOINT and MIX) to be used in multivariate context. According to experiments, improved SAXVSM significantly outperforms the original version, and MIX version for multivariate TSC (MTSC) shows great potential and has at least as good performance as baseline NN-DTW, if it is not better.

1 Introduction

TSC has become increasingly important with more and easier collections of temporal data in different domains, e.g. medicine, finance, biology and in our case, steel industry. Such interest can arise, for example, when an automatised process to recognise the states of plant machines, or to understand the reasons behind failures of machines, is demanded.

While our intuition is restricted to low dimension, insights gained from states of the art for time series problems can be a critical tool for a deeper understanding of the data.

As TSC is an active domain for the research, there are many approaches proposed to attack TSC problem:

1.1 Traditional Ensemble Classifier

Traditional ensemble classifiers, like Rotation Forest (RotF) [2] , see TSC as a general classification problem and provide an important baseline for other time-series-specific approaches. RotF, specially, shows a strong robustness to noise [3]. However, when the noise comes down to a relatively low level, these classifiers become less performant. Also, usually, traditional ensemble classifiers do not provide an evident way for interpretation.

1.2 Similarity-Based Approach

Similarity-based models are the first type to be proposed to solve TSC problem. These models work on invariances that lie in time series, e.g. vertical invariance (amplitude/offset), horizontal invariance (warping/scaling/phase), complexity and occlusion [4]. Models use an appropriate measure of similarity that takes into account these invariance. In conjunction, a Nearest Neighbour (NN) classifier is used. For example, Dynamic Time Warping (DTW) saw the light since 4 decades, was introduced from speech recognition to TSC later and has dominated until these years. Other modified versions of DTW and Edit Distance (ED) are proposed. MoveSortMerge (MSM) [5] combines both DTW and ED, forms a well defined metric and shows a competitive performance even against recently proposed approaches. However, while NN does provide an idea of how a case is classified into some class by providing a nearest neighbour in training set, it has two disadvantages:

1. It has a large space complexity and time complexity during the test.
2. It does not provide an intuitive interpretation of what is the characteristics of a class.

1.3 Heterogeneous Ensemble Approach

Heterogeneous ensemble models are among the states of the art these years, e.g. Collection Of Transformation Ensemble (COTE) [6], Shapelet Transformation (ST) [7], Elastic Ensemble (EE) [8]. The general idea of these approaches is to combine as many different approaches as possible, following the principal of diversity of ensemble. While having a high accuracy, it is hard to draw some insight from the result given by these approaches. Besides, these ensembles are more time-consuming than any other existing techniques for TSC problem.

1.4 Artificial Neural Network

Artificial Neural Network (ANN) has also been tried [9] to attack this problem and proves to be competitive against states of the art. Some of the most advanced idea in neural network nowadays, e.g. convolutional layer to detect local structure, a multiple layer structure applied to allow learning of complex features. While applying a general classifier and obtaining a state-of-the-art performance against problem-specific approaches is exciting, a good interpretation of complex

neural network is still to be discovered.

1.5 Structure-Base Approach

Several models that allow an intuitive interpretation and have a reasonable performance are proposed. All of these approaches, to our knowledge, belongs to the structure-based categories. Structure-based approaches are inspired by the fact that for a time series, attributes that are closer in time usually have a stronger correlation in between.

1.5.1 Interval-Based Approach

Interval-based approaches are structure-based. These approaches are inspired by the observation that usually we are capable to distinguish different classes by observing levels (offset) and changes (slope). They use statistics computed from sampled intervals to represent the original time series [10]. One interpretable model that belongs to this category is Time Series Forest (TSF) [11], which is capable to give an insight, by giving a heat map for each class, of which parts of the time series one shall concentrate on to tell whether it belongs to this class. However, this does not provide a direct insight of which features in the time series characterise this time series.

1.5.2 Shapelet-Based Approach

Shapelet-based approaches are structure-based and are also the first type of models that are created to directly address the problem of interpretability. These approaches search for length-fixed subsequences that can be regarded as discriminative features (Fast Shapelet (FS) [12], Logical Shapelet [13], Learned Shapelet (LS) [14]). For those most interpretable models of this kind, a decision tree follows, with its nodes being comparisons to these discriminative subsequences. However, shapelet as a concept is defined on uni-variate time series and has no evident way to expand it to multivariate problems.

1.5.3 Discretisation-Based Approach

Discretisation-based approaches are structure-based and the first to propose this idea is the work of Lin et al that proposes an approach called Bag-of-Patterns (BoP) [15], whose name takes the analogue to the bag-of-words technique in text mining domain. These approaches, while having the same idea as shapelet-based approaches that there exist some local fixed patterns, transforms the potential segments to words to allows reduction of noise, a quicker comparison between patterns and usage of existing text mining techniques.

For those most interpretable models of this kind, Vector Space Model is used as classifier, like Symbolic Aggregation approXimation - Vector Space Model (SAX-VSM) [1] and Bag-of-SFA-Symbols Vector Space (BOSS VS) [16].

On the one hand, these models allow a heat map for each instance on each class that indicates if this instance belongs to this class, which parts of this instance provide the evidence and which parts say no. Such a detailed insight is naturally superior than that of interval-based models which give only a heat map for a class without telling what discriminative features actually are. On the other hand, these models not only allow a similar interpretation to shapelet-based approaches, i.g. to provide some most discriminative features for each class, but also gives the quantified importance of these discriminative features for each class, which is naturally superior than that of shapelet-based approaches.

In terms of multivariate problem, Symbolic Multivariate Time Series (SMTS) [17] is proposed and shows competitive performance. However, SMTS uses a random forest as a classifier and its interpretation is not intuitive.

Goal, Organisation and Contribution

While accuracy is important in general, to apply an approach in industry requires more in interpretability, since what is also important is an insight that can investigated later and a prediction for which we can understand the reason. What's more, in practical problem, we are usually faced with multivariate problems other than univariate. These two points lead to a desire for a multivariate version of interpretable models for TSC. Thus, we decide to look into some discretisation-based approaches that use VSM model. As SAX-VSM is more intuitive in terms of implementation compared to BOSS VS, we choose SAX-VSM as the approach we study.

Nevertheless, as is proved [18], SAX-VSM does not have a satisfying accuracy, so our work has extended to the improvement of SAX-VSM itself. Our main goal is to improve SAX-VSM and extend it to multivariate TSC.

The organisation of the report is as follow:

1. In Section 2, we will talk about the states of the arts that address both accuracy and interpretability in TSC.
2. In Section 3, we will give a review of SAX-VSM.
3. In Section 5, we will give in details our several modifications on the original version.
4. In Section 6, several ideas to extend SAX-VSM to multivariate problem are given.
5. In Section 7, results of experiments are given with analysis of statistics.
6. In the end, further discussion and conclusion are given in Section 8 and 9.

Our main contributions are as follow:

1. We propose a new type of Vector Space Model using different kind of weighting (TAN) that allows a

negative weight and distinguish partial generality.
2. We propose to add the offset and slope information to SAX transformation.
3. We propose using a random search followed by a local search other than DIRECT [19] optimisation proposed by the original author.
4. We propose three approaches to address multivariate time series problem with SAX-VSM.
5. We test the effect of numerosity reduction technique and the Z-normalisation in cosines similarity and all proposals above with a comprehensive analysis on the results.

2 Related Work

2.1 Bag-of-Patterns(BoP)

The first approach that uses discretisation technique to attack TSC problem is the work of Lin et al [15]. and is named Bag-of-Patterns. BoP uses sliding window and SAX to transform time series into a bag of SAX words. Frequency histograms of words for each training examples are then created to represent the corresponding time series. During the test, it uses NN as classifier.

2.2 SAX-VSM

SAX-VSM, inspired from BoP, is then proposed to make the full use of discretisation that allows access to text mining techniques. SAX-VSM differs from BoP by its way of treating the histograms of SAX words. Instead of using a NN, SAX-VSM make a summary of these histograms for each class and use a VSM classifier later. The advantages of this approach is that it reduces massively the test time and at the same time provides a significantly better interpretability.

2.3 Bag-of-SFA-Symbols(BOSS)

BOSS [20], also inspired from BoP, continues to use NN as classifier, but changes the discretisation process. Instead of using SAX, it uses SFA which consists of Discrete Fourier Transform (DFT) and Multiple Coefficient Binning (MCB) and therefore looks into frequency domain other than time domain. With a bagging ensemble technique, BOSS can attain performance of the states of the art nowadays.

2.4 BOSS Vector Space(BOSS VS)

BOSS VS then sees the light. Combining discretisation of BOSS, i.g. SFA and classifier of SAX-VSM i.g. VSM, BOSS VS provides a quick test, a strong interpretability and a reasonable accuracy.

As is mentioned before, the interpretability of approaches using a discretisation conjuncted to a VSM is among the best for two reasons:

1. *Instance Level Interpretation.* It provides insight not only for a class (like interval-based approach) but also for a single instance relative to a given class (heat map for each instance on each class).
2. *Quantified Importance of Discriminative Features.* It provides insight not only for what the discriminative features are (like shapelet-based approach) but also for how important the discriminative features are for different classes.

Two instantiations of this idea are SAX-VSM and BOSS VS. To our knowledge, there is not yet work on extending these approaches to *standard* MTSC. [21] does use SAX-VSM in MTSC problem, but the task at which it aims is to attribute a label to each time index in the time series, which is different from the usual TSC that assigns a label to a whole time series.)

Thus, we decide to look into one of these most interpretable models, SAX-VSM. Our focus is on improvement of the original SAX-VSM, and its extension to multivariate TSC problem, with the intention to take advantage of the interpretability of VSM in MTSC problems.

3 Outline of SAX-VSM

This section is a review of SAX-VSM.

Training a SAX-VSM model can be divided into two steps:

1. **Discretisation**, which transforms training time series into bags of words using SAX.
2. **Fitting a VSM model** to these bags of words.

3.1 Discretisation

The process of discretisation can be divided into three steps:

1. **Sliding window**, which depends on a parameter, the length of window l and extracts subsequences from time series with the length-fixed window that slides along time axis.
2. **SAX** or Symbolic Aggregate approXimation [22], which transforms each of the extracted subsequences into a SAX word.
3. **Numerosity Reduction**, which deletes all the words that are followed by an identical word.

As show in Figure 1, a SAX transformation itself can also be divided into three steps:

1. **Z-normalisation**, which subtracts the mean from each attribute in the sequence and divides the result by the standard deviation.
2. **PAA**, or Piecewise Aggregation Approximation, which depends on an integer parameter w, duplicates each element in the normalised sequence w times, divides the resulting sequence into w parts

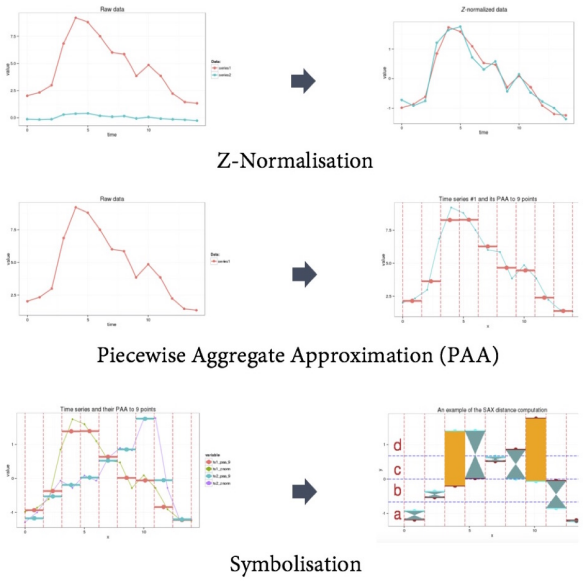

Fig. 1. SAX Transformation can be divided into three steps: Z-normalisation, PAA and symbolisation.

and replaces each part by its mean (replaces each part by one single value).

3. **Symbolisation**, which depends on an integer parameter α, takes the dimension-reduced sequence from PAA and maps each value to a symbol. This mapping is created according to a series of breakpoints that are taken to divide the real domain $\mathbb{R}$ into α continue parts so that according to Gaussian distribution, each part has the same probability.

3.2 Fitting the bags of SAX words into a VSM

VSM is a widely known technique for classification used in text mining domain. It can be divided into 2 steps:

1. **Weighting**: A weight is computed to represent how much a word implies a class.
2. **Comparison with Similarity**: During the inference, similarity is computed between test case and each class, and the class with the largest similarity is predicted.

3.2.1 Weighting

At the beginning of VSM, the importance of each word for each class is computed. The most widely used weighting is Term Frequency - Inverted Document Frequency (TF-IDF). First, TF and DF are computed:

$$TF(word, class) = \frac{F(word|class)}{F(class)}$$

$$DF(word) = \frac{F(instance|word)}{F(instance)}$$

where $F(x)$ is the number of occurrences of x in training set and $x|y$ means x under the condition of y.

The importance of a word for a class is then computed as:

$$\begin{aligned} &W(word, class) \\ &= -log(1 + TF(word, class)) \times log(DF(word)) \end{aligned} \tag{1}$$

Intuitively, TF indicates how frequent occurrences of a word are in a class, DF indicates how frequent occurrences of a word are in all classes. For example, if we were to classify an article into sports and cooking categories, words like *football*, *running*, *swimming* would give a strong evidence for the article being about sports, correspondingly, these words would probably have a higher frequency in sports article. However, words like *is*, *a*, *it*, while they are pretty likely to appear very frequently in an article, they do not seem to provide a very useful information in our problem, since they are too general. DF describes this phenomenon and is the corresponding measure for the generality of a word. The higher the DF of a word is, the more general the word is, and the less importance we attribute to this word.

3.2.2 Comparison with Similarity

After training, to perform a prediction with VSM, a similarity is chosen to allow a comparison between instance and class. The most widely used similarity is called Cosines Similarity.

Firstly, we perform the same discretisation on the given test time series x and obtain a vector of frequencies f_x. If f_c denotes the resulting weight vector from TF-IDF for the class c, then the similarity between x and c is computed as:

$$\begin{aligned} Similarity(x, c) &= cos(angle(f_x, f_c)) \\ &= \frac{DotProduct(f_x, f_c)}{Norm(f_x) Norm(f_c)} \end{aligned} \tag{2}$$

The motivation in text mining for this similarity which omits the difference caused by magnitude is to eliminate the influence of the length of the article when judging on subject matter.

4 Optimisation of Parameters

To choose hyperparameters (length of sliding window l, length of SAX word w, size of alphabet set α) for the model, DIRECT [19] algorithm is used. However, while DIRECT works on discrete space, it is based on a hypothesis of certain *smoothness* on this space, which is not evidently verified in our case, because we observe a

high sensitivity of SAX-VSM on parameters. We decide not to give more details of DIRECT in the report for it is abandoned right after some initial experiments.

5 Improving SAX-VSM

Proposed modifications can be divided into three parts:

1. Modifications on discretisation
2. Modifications on VSM
3. Modifications on optimisation of parameters

5.1 Modifications on Discretisation

We propose three modifications for discretisation:

1. Adding offset information
2. Adding slope information
3. Not doing numerosity reduction

5.1.1 Offset Information

One of the benefits we can get from Z-normalisation is that it gives vertical invariance (amplitude and offset). While such invariance is possibly satisfied by some specific problems, it could decrease accuracy sometimes [20]. The author of BOSS proposed to set it as a parameter for SFA transformation, to be chosen by cross validation. We adapt this idea to our SAX transformation.

Different from the case in BOSS, SAX includes itself a Z-normalisation to obtain a Gaussian-like distribution for later work, and therefore loses this information by default, while SFA, using Discrete Fourier Transform (DFT), keeps the offset information by default.

Our approach is to concatenate behind each word a symbol that indicates the *Level* of this word relative to the total time series. This symbol is obtained by the way as follow:

Let map_α denote the symbolisation process in SAX that depends on the extermal parameter α, which receives a value from $\mathbb{R}$ and returns a symbol. The symbol for a subsequence s in time series x can be computed as:

$$Level(s) = map_\alpha(\frac{mean(s) - mean(x)}{stddev(x)})$$

5.1.2 Slope Information

Inspired by both [10] and [21], we decide to try adding slope information in a symbol.

Remind that in PAA, we represent each of the w segments by one single value, i.g. the average. Such idea is very similar to interval-based approaches in the sense that a segment is represented by its statistics. With a little generalisation, instead of representing each segment only by its average, we decide to add a slope value to give a more exhaustive description.

Note that adding more information to each word does not guarantee a better performance, because it can be the case where the total number of possible words increases, dimension of the VSM increases, and with the curse of dimensionality, information that lies in each dimension (word) will decrease and it could be more difficult for the model to capture the discriminative features.

We use a small number of symbols to represent different states of slope: S(stable), U(goingUp) and D(goingDown), as is experimented by [21].

To compute the slope of each segement, *least square method* is used. Symbols are assigned according to the sign of the slope.

Also, the idea is to provide this as a choice that can be done during cross validation.

5.1.3 Numerosity Reduction

We also tried to ignore the numerosity reduction to see how its actual effect to the performance.

5.2 Modifications on VSM

We propose two modifications for VSM:

1. TAN Weighting
2. Dot Product Similarity

5.2.1 TAN Weighting

While TF-IDF is widely used in telling the subject matter, it has one potential defeat, that is it can not distinguish between *partial generality* and *global generality* of a word.

DF is the term that reflects a consideration of generality of a word, but only a global one.

A word may be general among a certain subset of classes but can still be a good evidence for the test NOT belonging to this subset. There should also be a strong negative effect of a word for a certain class if this word has never been in this class but appears frequently in all the other classes. Such an effect may be missed by TF-IDF, because a word appears frequently in all classes but one can be too general and the weight of this word for all classes may be smoothed to having only small difference of weights between classes.

For example, to classify articles into 10 sports among which 9 are balls and the tenth sport is swimming. *Ball* is probably a word too general to distinguish the 9 first sports, but can have a good chance to tell us that it is not about swimming. With TF-IDF, though weight of *ball* for swimming is smaller than for the other sports, but the difference can be very small since $DF(ball)$ is very large. Thereby, what we are doing is actually smoothing the negative effect of the word *ball* for the class swimming just because it is too general among the ball sports classes!

Such a couterintuitive effect is why we try to propose another weighting called TAN whose formula

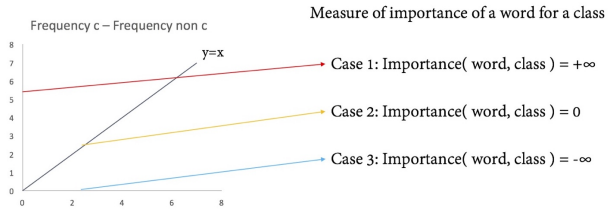

Fig. 2. Three cases when weighting a word for a class

are given as follow:

$$\begin{aligned} & W_{TAN}(word, class) \\ =& tan(2 \times angle(f(word|class) + \epsilon \\ & + j \times (f(word|nonclass) + \epsilon)) - \frac{\pi}{2}) \end{aligned} \tag{3}$$

where $f(w|c)$ is frequency of w in c per instance, $j = \sqrt{-1}$, *angle* maps a complex number to its angle in range $[0, 2\pi]$ and ϵ an infinitesimal to avoid infinite value.

First thing to notice is that in the formula, other than DF we use a new term $f(word|nonclass)$ which gives the frequency of the word out of the class. This is the critical part to address the partial generality problem that appears in *TF-IDF* weighting.

The intuition behind this formula is as follow:

There are three special cases when we are determining the importance of a word for a class. To simplfy notations, f_c, f_n denote $f(word|class)$ and $f(word|nonclass)$ respectively.

1. When f_c is nonzero and f_n is zero, this word strongly indicate the test case belongs to this class. The bigger f_c, the stronger the implication, the huger the importance of this word for this class.
2. When f_c is equal to f_n, this word can not help to tell whether the test case belongs to this class. The importance should be assigned zero.
3. In the case opposite to the first case, the importance of a word for this class should be negative.

As is shown in Figure 3, we are actually looking at the angle formed between vector (f_{nc}, f_c) and axis. To amplify the effect, we map the angle to $\mathbb{R}$ with *tan*, as a result of which we call *TAN* (also Take into Account Negative effect).

5.2.2 Dot Product Similarity

We also test a non normalised version of cosines similarity, which is dot product similarity. The reason is that in TSC, unlike text classification, instances do not vary too much in term of length of time series.

5.3 Optimisation of Parameters

DIRECT optimisation is proposed by the original author, which is a method of optimisation by sampling, which should work for discrete space like in our case. However, for what we have observed, DIRECT, faced with the sensitivity of SAX-VSM to parameters, does not adapt itself well, having a tendency to either oversample the region in one single sample unit hypercube or under-sample a huge region where lies potentially an optimal, which introduces compromise between efficiency and optimality of solution. This dysfunction may result from the fact that the optimised function is far rougher than DIRECT can bear.

Therefore, during the experiments, instead of DIRECT, we decide to simply use a random search followed by a local search, for random search enables a global view on the search space and also is easier in terms of implementation. The pseudocode is given as follow:

Input:
1. optimised function e
2. search space s
3. maximal number of iteration t
4. early stop constant **for** local search c

Output: (locally) optimal solution b:

1. cs := [] (empty **list**) // ordered candidates
2. ss := {} (empty **set**) // searched points
   //random search
3. **for** i **from** 1 to t:
   1. let a randomly chosen **in** s but **not in** ss
   2. cs := **sorted**(insert((e(a),a),cs)
   3. ss := union(ss,{a})
4. end **for**
   //local search
5. bsc, b := cs[0] // best score, best solution
6. impt := 0 //impatience **for** early stop
7. **while** true:
   1. **if** impt >= c then **break** end **if**
   2. impt := impt + 1
   3. **if** cs **is** empty then **break** end **if**
   4. bcsc, bc := cs[0]
      // best candidate score, best candidate
   5. remove_first_element(cs)
   6. **for** a **in** neighbours(bc,s) − ss:
      1. score := e(a)
      2. ss := union(ss,{a})
      3. cs := **sorted**(insert((score,a),cs)
      4. **if** score > bsc:
         1. bsc := score
         2. b := a
         3. impt := 0
      5. end **if**
   7. end **for**
8. end **while**
9. **return** b

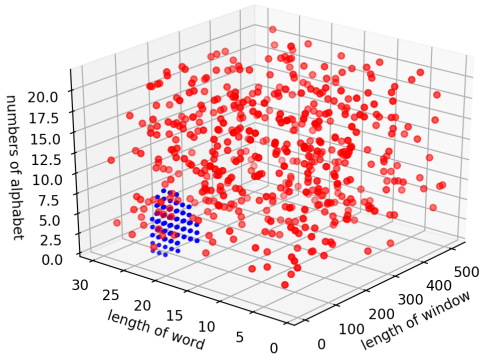

Fig. 3. Points tested by random search in red, points tested by local search in blue

Figure 3 is a visualisation of the searched region. Scattered points are all searched during the optimisation. Red dots are searched during random search, and blue dots are searched during local search.

6 Multivariate SAX-VSM

Our another goal is to extend SAX-VSM to multivariate TSC problem. To our knowledge, this is the first attempt to extend a discretisation-based approach using VSM to the typical MTSC where a whole time series is classified into exactly one class. A similar work is done by [21], which tries to do a online classification, labelling each time index of the time series with a class, but the problem itself is very different from a typical MTSC problem where unique label is given for each time series.

We propose in total three modes of SAXVSM to address this problem, with the first two making different assumptions on the data and the third combining the first two. Their details are as follow:

6.1 INDEPEND

INDEPEND mode is based on a very basic assumption of variate-independency, which is to say this mode do not consider explicitly the fact that there could be dependency between variates. This mode serves also as a baseline and provide a sanity check to whether the consideration of intervariate correlation actually helps. Practically, what we do is just to add all bags of SAX words from each variate up to one single bag, as shown in Figure 4.

6.2 CONJOINT

CONJOINT mode is based on the opposite case of INDEPEND and suppose a perfect synchronised correlation between variants, which means for each time index, the values of all variants are seen as one single

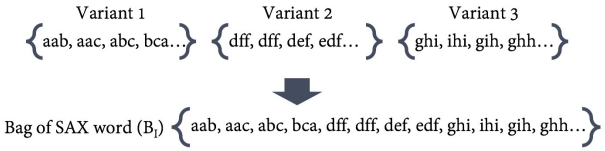

Fig. 4. INDEPEND simply unites all bags of words into one

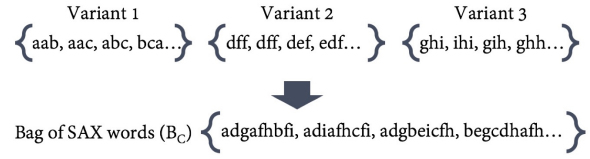

Fig. 5. CONJOINT creates new symbols by concatenating the symbols of every variant and thus new words

state. Concretely, we create a new table of symbols which is computed as the Cartesian product between symbol sets from different variates, as shown in Figure 5.

6.3 MIX

MIX is simply a combination of first two ideas, leading a bag of words that is union of the former two.

7 Experiment

We conduct experiments on two benchmarks. The first benchmark is UCR dataset [23], which is a traditional benchmark for univariate TSC problem. This benchmark allows us to verify the effectiveness of improvement measures in Section 2 and compare our approaches to the other potentially competing approaches. For reason of runtime, experiment is only conducted on 79 datasets, as is detailed later, Another benchmark is a collection of datasets from a website [24]. To our knowledge, work on discovering interpretable approaches for multivariate TSC is far less diverse as in univariate problem, and therefore number of datasets is much less, concretely 13.

7.1 Experiment on UCR Dataset

7.1.1 Control Variables

In this experiment, to test the effectiveness of modifications, we fix each test to one single setting (e.g. instead of choosing invariance of offset by cross validation, we fix it to true or false for each test). Therefore, we have in total 32 settings for each dataset. (Weighting, similarity, numerosity reduction, offset and slope in total 5 bianry choices).

Table 1. Details for the experiment

length of window l	$[1, minlength]$
length of word w	$[1, 30]$
size of alphabets α	$[1, 20]$
number of iterations for random search	500
early stop for local search	20
number of folds for cross validation	3
number of repetitions of accuracy test	100

7.1.2 Test Process

The seed of random number generator is set the same for all settings to prevent bias.

During optimisation, each dataset is first divided into a training set and a test set according to original UCR dataset. Training set is divided into several folds to allow cross validation for evaluation of parameters (folds may be different for different parameters).

After optimisation, test set and training set are merged into one. 100 random stratified splits (with the same test size) on this entire dataset are done to evaluate the accuracy of models. Such method is designed to give comparable result to results uploaded on the website. While the each 100 splits may not be exactly the same as are used on the website, with 100 tests the variance of results should be negligibly small, which justifies our further comparison against other algorithms presented on the website.

7.1.3 Optimisation Setting

During experiments, 6 huge datasets requires a smaller search space on length of SAX word and size of alphabets, since our default search space is quite huge (maximum 30 for the word length and 20 for the alphabet size) which leads to a crash for memory being exhausted in these huge datasets. A potential solution is simply to reduce the search space. The actual optimum results usually from parameters where the word length is below 15 and the size of alphabets is below 8. However, due to constraint of time, we did not re-conduct experiments on these datasets. These 6 datasets are ElectricDevices, FordB, NonInvasiveFatalECGThorax1, NonInvasiveFatalECGThorax2, StarlightCurves, UWaveGestureLibraryAll.

Other details of experiments are shown in Table 1.

For convenience, a specific setting is described in the form of 5-tuple, e.g. (T, T, T, T, T) for using TAN weighting (F otherwise), with dot product similarity (F otherwise), without numerosity reduction (F otherwise), with offset information (F otherwise), with slope information (F otherwise). (Therefore, the original version is (F, F, F, F, F))

7.1.4 Result and Analysis

Due to the constraint of screen/paper size, test result is appended in the file called **test_result.xlsx** which can be opened by Excel.

To give a brief navigation on that table file, there are in total 16 sheets, 11 for results on UCR dataset and 5 for results on MTS dataset.

There are four prefix:

1. **uts** means the sheet compares 32 settings.
2. **temp** means the sheet compares univariate improved SAXVSM to states of the art.
3. **mts** means the sheet compares 3 multivariate versions of SAXVSM.
4. **mtmp** means the sheet compares multivariate SAXVSM to states of the art.

There are seven suffix:

1. **err** means the sheet contains error rate.
2. **acc** means the sheet contains accuracies.
3. **worst** means the sheet counts on how many datasets algorithms perform worst among all.
4. **best** means the sheet counts on how many datasets algorithms perform best among all.
5. **winlost** means the sheet counts on how many datasets CV(for univariate)/MIX(for multivariate) win/lose to this algorithm.
6. **stat** means the sheet computes statistics of the result, using Friedmen test [25] and its post-hoc test to test whether algorithms are significantly different.

According to the sheet *uts_stat*, there are several observations:

1. On average over UCR dataset, two settings are 80% confidently better than all the other settings (and are significantly better than original version (99.92% confidence). These two are: (T, T, F, T, F) and (T, F, F, T, F)
2. We can study the effect of each factor by fixing all the other factors:
   (a) TAN (resp. without invariance of offset) wins over TFIDF (resp. with invariance of offset) for all 16 pairs (100% confidently better).
   (b) Dot product similarity wins only 6 times against cosines similarity (85% confidently worse).
   (c) Deleting numerosity reduction wins only once against using numerosity reduction (99.96% confidently worse).
   (d) Additional slope information only wins 3 times among 16 comparisons (99.4% confidently worse).

These two observations coincide and give us three conclusions:

1. Replacing TFIDF weighting with TAN weighting can bring better results on average over UCR dataset, and it is the same case for adding offset information.
2. Numerosity reduction does help in promoting accuracy on average.
3. Additional slope information makes results worse on average.

Apart from these, we also know that according to results, maybe cosines similarity is slightly better than dot product similarity, but with only 85% confidence and we can not draw any further conclusion.

Based on these observations, we create a version (CV) with cross validation choosing whether to add offset information and add slope information.

7.1.5 Comparison to States of the Art

In the sheet *temp_stat*, we also compare our results to other algorithms. We give the ranking of our implementation of original version of SAX-VSM (Original), best single setting (Best), cross validation on choice of invariance and slop information (CV) with the list of approach on the website. Other than only interpretable models, we give a more comprehensive comparison to states of the art to provide a more concrete idea of the performance of our algorithms and to better show the improvement of our modifications.

As we can observe, there are three leaps of performance of different version of SAX-VSM:

1. The difference between the result of SAX-VSM on the website and our original version can be explained by a more exhaustive optimisation we have done. As is mentioned, we observe a non trivial sensitivity of SAX-VSM on parameters, so a more comprehensive search can give a significantly better result.
2. The difference between Original and Best can be explained by the usage of TAN weighting and adding offset information as is discussed before.
3. The difference between Best and CV is another interesting observation. While it proves that on average, adding offset information can help, adding slope information makes thing worse and TAN outperforms TF-IDF, but it turns out to be dataset-specific. Therefore, using cross validation to choose the right settings can help, which ultimately improves the result.

We may notice BoP performs badly in fact, while BOSS is among the best approach, which may be caused by the fact that SFA is a more adaptive discretisation approach than SAX and BOSS uses also a bagging ensemble. Unluckily, we did not find a similar test process done on BOSS VS, which would have allowed a more comprehensive comparison.

Two more observations:

1. All approaches that are significantly better than CV are much less interpretable, in the form of ensemble or in conjunction with classifiers that do not have a good interpretability.
2. TSF as an interpretable interval-based approach, and LS (Learned Shapelet) as an interpretable shapelet-based approach, have the same preformances as CV if it is not worse, while as is analysed before, discretisation-based approaches with VSM have a superior interpretability than interval-based and shapelet-based ones.

These two observations indicate that CV is making a good compromise between accuracy and interpretability, being one of the most interpretable models among accurate ones and being the most accurate among the interpretable ones.

Of course, there is no free lunch. CV requires a huge computation for training, to choose a good configuration (slope information and offset information) and optimise the parameters.

7.2 Experiment on Multivariate Time Series Dataset

7.2.1 Experiment Detail

We also conduct experiments for different multivariate version on 13 datasets: CharacterTrajectories, CMUsubject16, ECG, JapaneseVowels, Libras, 5 datasets in RobotFailure (LP1, LP2, LP3, LP4, LP5), PenDigits, uWaveGestureLibrary and Wafer in the collection mentioned above.

Based on the result we obtain from above, we only experiment with our CV version other than any other single settings. The search space for parameters is the same as before. Accuracy estimation is done in exactly the same (100 splits) as before to make results consistent.

7.2.2 Result and Analysis

As is shown in the sheet mts_rank, MIX as expected gives the best performance but is with only 70% confidence better than INDEPEND (8 wins over 13). MIX and INDEPEND are all over 98% confidently better than CONJOINT (MIX 13 wins over 13, and INDEPEND 11 wins over 13). This result implies that most of the information can actually be found within different variants independently, but still a consideration on pattern that lies in different variants can help. The evidence for the latter belief is the test results on Libras and LP5 datasets. For these two datasets (the only two), CONJOINT has a significantly better accuracy than INDEPEND, and MIX evidently benefits from this.

7.2.3 Comparison to States of the Art

We compare our MIX to results from three other algorithms that are founded in [17]. The comparison is shown in the sheet $mtmp_winlost$. SMTS has a dominant performance but is less interpretable, and if we compare MIX to the baseline NNDTW, MIX is still slightly better (8 wins over 13) and is (90% confidence) better than MTSBF (5 wins over 6).

Notice that there are two datasets for which MIX outperforms all the other algorithms, and for these two datasets, MIX has actually the same performance than INDEPEND. This indicates that, these two versions of SAX-VSM are still more comfortable with uni-variate problem and only when the dataset verifies more or less the variate-indenpency hypothesis can the approach be the most performant. This may indicate more work to do to discover how to consider multivariate correlations

8 Discuss

8.1 Problem of Complexity on Optimisation

Our CV version of SAX-VSM is performant especially in a UTSC problem or in a MTSC problem where the discriminative features can be found within single variants. However, it has a huge time complexity. Notice that the most part of the time is used to find the best parameters. This problem can be addressed by:

1. Parallelism. The random search process and the cross validation and also the choice of settings are all processes that are perfectly parallelizable.
2. Reducing search space. Our search space for parameters is actually too wide and not practical. An optimum with word size maximally 15 and a set of alphabet sized maximally 10 covers most of our cases.
3. Reducing the number of iteration for random search, and giving an early stop condition that urges earlier stop.

8.2 Future Work

Still, as a model that is perfectly separable, SAX-VSM can potentially benefit a lot from other existing discretisation techniques, e.g. SFA. SFA, for example, has shown great performance in a similar approach Bag-Of-SFA-Symbols Vector Space (BOSS VS). BOSS VS and SAX VSM are perfectly combinable in the sense that their bags of words can simply be combined together to get conjuncted to a VSM, with the former looking into frequency domain and the latter looking into time domain. The critical difference between them is that SAX uses PAA to reduce dimension while SFA use first few terms from DFT to replace the original time series. Similarily, other similar techniques like DWT (Discrete Wavelet Transform, which is a standard technique in signal processing) can possibly play a role. This sort of combination is cheap because time complexity and space complexity are both linear on adding a new type of discretisation and that the process is perfectly parallelisable. More importantly, the interpretability stays exactly the same, allowing a heat map and direct access to the most weighted words for each class, since Vector Space Model stays unchanged.

Also, other multivariate versions are to be discovered, with a reasonable complexity and a looser hypothesis on correlation between variants.

9 Conclusion

As the result shows, among 5 modifications proposed to improved SAX-VSM, on average and on the UCR dataset, using TAN weighting and adding offset information can effectively promote accuracy, while adding slope information makes the result worse. Numerosity reduction is proved to be an effective technique to apply and no significant difference between dot product similarity and cosines similarity is detected.

Using cross validation to choose whether to apply certain modification yields a better result, which is comparable against that of BOSS while using VSM reduce significantly the test time compared to Nearest Neighbour classifier in BOSS.

MIX seems to be the best version among all three versions that we give (not significantly better than INDEPEND and with 70% confidence). It seems to be more performant than 1-NN-DTW (70% confidence), and is significantly more performant than the competing approach MTSBF. Although MIX is not as accurate as SMTS, MIX has a greater interpretability with VSM compared to SMTS which uses a random forest and does not give a direct interpretability.

Acknowledgements

Thanks go to Guillermo Barquero Fernández and Alonso Baldonedo Pablo for inspiring me with new ideas and sharing their points of view.

Thanks also for Iyán Méndez Veiga and Juan Redondo who kindly helped me on the problem of Internet and using the cluster.

Thanks also for Juan Redondo, Tatiana Manso and Xavier Rélo without whom this report and this internship would not have existed.

References

[1] Senin, P., and Malinchik, S., 2013. "Sax-vsm: Interpretable time series classification using sax and vector space model". In Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, pp. 1175–1180.

[2] Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J., 2006. "Rotation forest: A new classifier ensemble method". *IEEE transactions on pattern analysis and machine intelligence,* **28**(10), pp. 1619–1630.

[3] Bagnall, A., Bostrom, A., Large, J., and Lines, J., 2017. "Simulated data experiments for time series classification part 1: Accuracy comparison with default settings". *arXiv preprint arXiv:1703.09480*.

[4] Batista, G. E., Wang, X., and Keogh, E. J., 2011. "A complexity-invariant distance measure for time series". In Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, pp. 699–710.

[5] Stefan, A., Athitsos, V., and Das, G., 2013. "The move-split-merge metric for time series". *IEEE transactions on Knowledge and Data Engineering,* **25**(6), pp. 1425–1438.

[6] Bagnall, A., Davis, L., Hills, J., and Lines, J., 2012. "Transformation based ensembles for time series classification". In Proceedings of the 2012 SIAM international conference on data mining, SIAM, pp. 307–318.

[7] Lines, J., Davis, L. M., Hills, J., and Bagnall, A., 2012. "A shapelet transform for time series classification". In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 289–297.

[8] Lines, J., and Bagnall, A., 2015. "Time series classification with ensembles of elastic distance measures". *Data Mining and Knowledge Discovery,* **29**(3), pp. 565–592.

[9] Wang, Z., Yan, W., and Oates, T., 2017. "Time series classification from scratch with deep neural networks: A strong baseline". In Neural Networks (IJCNN), 2017 International Joint Conference on, IEEE, pp. 1578–1585.

[10] Rodríguez, J. J., Alonso, C. J., and Boström, H., 2001. "Boosting interval based literals". *Intelligent Data Analysis,* **5**(3), pp. 245–262.

[11] Deng, H., Runger, G., Tuv, E., and Vladimir, M., 2013. "A time series forest for classification and feature extraction". *Information Sciences,* **239**, pp. 142–153.

[12] Rakthanmanon, T., and Keogh, E., 2013. "Fast shapelets: A scalable algorithm for discovering time series shapelets". In Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, pp. 668–676.

[13] Mueen, A., Keogh, E., and Young, N., 2011. "Logical-shapelets: an expressive primitive for time series classification". In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 1154–1162.

[14] Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L., 2014. "Learning time-series shapelets". In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 392–401.

[15] Lin, J., Khade, R., and Li, Y., 2012. "Rotation-invariant similarity in time series using bag-of-patterns representation". *Journal of Intelligent Information Systems,* **39**(2), pp. 287–315.

[16] Schäfer, P., 2015. "Bag-of-sfa-symbols in vector space (boss vs)".

[17] Baydogan, M. G., and Runger, G., 2015. "Learning a symbolic representation for multivariate time series classification". *Data Mining and Knowledge Discovery,* **29**(2), pp. 400–422.

[18] Bagnall, A., Bostrom, A., Large, J., and Lines, J., 2016. "The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version". *arXiv preprint arXiv:1602.01711*.

[19] Jones, D. R., Perttunen, C. D., and Stuckman, B. E., 1993. "Lipschitzian optimization without the lipschitz constant". *Journal of Optimization Theory and Applications,* **79**(1), pp. 157–181.

[20] Schäfer, P., 2015. "The boss is concerned with time series classification in the presence of noise". *Data Mining and Knowledge Discovery,* **29**(6), pp. 1505–1530.

[21] Esmael, B., Arnaout, A., Fruhwirth, R. K., and Thonhauser, G., 2012. "Multivariate time series classification by combining trend-based and value-based approximations". In International Conference on Computational Science and Its Applications, Springer, pp. 392–403.

[22] Lin, J., Keogh, E., Lonardi, S., and Chiu, B., 2003. "A symbolic representation of time series, with implications for streaming algorithms". In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, ACM, pp. 2–11.

[23] Chen, Yanping and Keogh, Eamonn and Hu, Bing and Begum, Nurjahan and Bagnall, Anthony and Mueen, Abdullah and Batista, Gustavo, 2015. The UCR Time Series Classification Archive, July. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[24] Multivariate time series classification with learned discretization. `http://www.mustafabaydogan.com/multivariate-time-series-discretization-for-classifica.html`. Accessed: 2017-08-01.

[25] Demšar, J., 2006. "Statistical comparisons of classifiers over multiple data sets". *Journal of Machine learning research,* **7**(Jan), pp. 1–30.