Dimension selection strategies for multivariate time series classification with HIVE-COTEv2.0

Alejandro Pasos Ruiz and Anthony Bagnall

School of Computing Sciences, University of East Anglia, UK. ajb@uea.ac.uk

Abstract. Multivariate time series classification (MTSC) is an area of machine learning that deals with predicting a discrete target variable from multidimensional time dependent data. The possible high dimensionality of multivariate time series can affect the training time and possibly accuracy of complex classifiers, which often scale poorly in dimensions. We explore dimension filtering algorithms for high dimensional MTSC used in conjunction with the state of the art MTSC algorithm, HIVE-COTEv2.0. We apply and adapt recently proposed selection algorithms and propose new methods based on the ROCKET classifier built on single dimensions. We find that, for high dimensional MTSC problems, the best approach can on average filter between 50% and 60% of dimensions without significant loss of accuracy, reducing train time by a similar proportion.

1 Introduction

Time series classification (TSC) is an area of machine learning that deals with predicting a discrete target variable from time dependent data. Recently there has been a focus on a specific time series problem called multivariate time series classification (MTSC) i.e. TSC problems where each time series has more than one dimension or channel. MTSC are generally more common than univariate TSC problems, and occur in a wide range of domains such as EEG classification, human activity recognition and classification of medical sensor data. MTSC problems have the added complexity of high dimensionality, which can affect the training time and possibly accuracy of the time series classifier; redundant or highly correlated dimensions may confound the classifier. There have been many recently proposed algorithms for MSTC, based on, for example, distance measures, transformation, ensembles and deep learning. A review of MTSC algorithms [18] found that there was no one approach significantly better than the others, when assessed on a benchmark set of data referred to as the UEA MTSC archive [17]. The conclusion of [18] was that an algorithm called the ROCKET [5] was recommended due to its superior speed compared to the rest of the state of the art. Subsequently, a new algorithm, HIVE-COTEv2 (HC2) [16] was shown to be significantly more accurate than all of the algorithms for MTSC evaluated in [18], including ROCKET. HC2 represents the current state of the art for both univariate and multivariate TSC, but it does not scale well in terms of number of dimensions. Our aim is to investigate whether we can improve the efficiency (and possibly the accuracy) of HC2 for high dimensional problems using a pipeline approach of filtering dimensions prior to classification.

Standard approaches for classifying high dimensional data are to employ a filter to select a subset of attributes or to transform the data into a lower dimensional feature space using, for example, principal component analysis. Our focus is on dimensionality reduction through filtering. For MTSC, filtering is generally accepted to be selecting the most important dimensions to use before training the classifier. Dimension selection can, on average, either increase, not change or decrease the accuracy of classification. The first case implies that the higher dimensionality is confounding the classifier's discriminatory power. In the second case it is often still desirable to filter due to improved training time. In the third case, filtering may still be desirable, depending on the trade-off between performance (e.g. accuracy) and efficiency (e.g. train time): a small reduction in accuracy may be acceptable if build time reduces by an order of magnitude. We address the task of how best to select a subset of dimensions for high dimensional data so that we can speed up and possibly improve HC2 on high dimensional MTSC problems.

Detecting the best subset of dimensions is not a straightforward problem, since the number of combinations to consider increases exponentially with the number of dimensions. Selection is also made more complex by the fact that the objective function used to assess a set of features may not generalise well to unseen data. Furthermore, since the primary reason for filtering the dimensions is improving the efficiency of the classifier, dimension selection strategies themselves need to be fast. HC2 is not as fast as ROCKET. We investigate whether we can use the speed and competitive accuracy of ROCKET to serve as a dimension filter for HC2. We use a stripped back version of ROCKET to assess and select dimensions through cross validation, then measure the impact this has on HC2 on both the train and the test data. We compare the ROCKET filter to recently proposed algorithms from the literature. Our contribution is to incrementally improve our understanding of how best to classify high dimensional time series: we introduce four new high dimensional MTSC problems to the UEA archive; we propose a hybrid approach for classifying high dimensional MTSC problems using ROCKET as a filter; and we compare our approach to a range of alternative algorithms and analyze the results.

The rest of the document is presented as follows. We review current approaches for dimension selection in Section 2, then provide a description of the ROCKET based method in Section 3. Section 4 describes the new data we are adding to the archive and our experimental design, and our results are described in Section 5. Finally, we draw general conclusions from our experiment and highlight areas of future investigation in Section 6.

2 Related Work

Time series classification can be categorized based on the number of dimensions as univariate (1 dimension) or multivariate ($d > 1$ dimensions). In univariate time series classification, an instance is a pair $\{\boldsymbol{x}, y\}$ with m scalar observations $(x_1, \ldots, x_m)$ (the time series) and discrete class variable y with c possible values. A classifier is a function or mapping from the space of possible inputs to a probability distribution over the class variable values. In MTSC, each observation of a time series has d dimensions (or channels), assumed to be aligned in time, so that for a single case $\{\boldsymbol{x}, y\}$, $\boldsymbol{x} = \{\boldsymbol{x_1}, \ldots \boldsymbol{x_m}\}$, where $\boldsymbol{x_j} = (x_{j,1}, x_{j,2}, \ldots, x_{j,d})$. If referring to multiple cases, we denote the j^{th} observation of the i^{th} case of dimension k as the scalar $x_{i,j,k}$.

2.1 MTSC Algorithms

MTSC approaches are often simple extensions of univariate TSC, which is a more extensively researched field. In a review of recently proposed algorithms [18] it was found that the best performing classifiers were Random Convolutional Kernel Transform (ROCKET) [5] and HIVE-COTEv1.0 [1]. Given equality of accuracy, ROCKET was recommended as a starting point for MTSC due to its much faster build time. We use ROCKET in our filtering approach, so a brief overview of the algorithm is appropriate.

ROCKET is based on a large number of random convolution kernels (defaulting to 10,000) used in conjunction with a linear classifier (ridge or logistic regression). It is a pipeline classifier that involves transformation followed by classification. A convolution is a transformation enacted by sliding a weight vector (the convolution) across a series, vector multiplying the weights and the window at each time point to form a new value for each window. Each randomly generated convolution is applied to the time series to create a new series. The maximum value and the proportion of positive values (ppv) are derived from the transformed series and form part of a new feature set. This huge new feature space (by default, 20,000 features for each instance), is used to train the classifier, which internally performs some feature selection/weighting to ignore non-discriminatory features. ROCKET is fast because the convolutions are randomly generated and the classifiers are simple. The original ROCKET was proposed for univariate TSC. However, an approach to enable use on multivariate datasets is available in the `sktime` toolkit[1]. For multivariate datasets, kernels are randomly assigned dimensions. Weights are then generated for each dimension.

More recently, version 2.0 of the HIVE-COTE classifier (HC2) was shown to be significantly more accurate than ROCKET on the UEA MTSC problems. HC2 is a heterogeneous meta ensemble of four ensemble classifiers built on different data representations: the Shapelet Transform Classifier (STC) [3], the Temporal Dictionary Ensemble (TDE) [14], the Diverse Representation Canonical Interval Forest (DrCIF) [15] and the Arsenal, an ensemble of ROCKET classifiers [16]. HC2 does not scale very well for high dimensional MTSC. This led us to consider whether we could combine approaches to improve HC2 efficiency. Our aim is to find the simplest approach that speeds up HC2 on high dimensional MTSC problems without significantly decreasing accuracy. Hence, our focus is pipeline

[1] https://github.com/alan-turing-institute/sktime

approaches that select dimensions prior to classification, rather than wrapper approaches that combine classification and selection.

2.2 MTSC Dimension Selection Algorithms

The Common principal component Loading based Variable subset selection method (CleVer) [21] algorithm is PCA based approach that is adapted for dimension selection for multivariate time series. CleVer uses PCA and clustering techniques to select dimensions. A PCA is performed independently on each instance, and the principal components are extracted. Next, all components that belong to the same class are combined to create common principal components through a process called Common Principal Component Analysis (cPCA). A proportion of the common components are used to create a feature space. These features are clustered, and the closest dimension to each centroid is chosen as selected dimensions. CleVer requires a separate PCA on each series, which is both time and space consuming. It is also complex, and we cannot find an open source implementation. Since we are looking for a lightweight feature selector, we do not evaluate CleVer in this study.

A method based around one nearest neighbour classification with dynamic time warping (1-NN DTW) is described in [10]. A merit score function (MSTS) is used to assess the quality of a subset of dimensions. The DTW distance function between cases and dimensions is precalculated. A prediction for each dimension pair is found through a three fold cross validation of 1-NN DTW. Similarity between each dimension is estimated using the adjusted mutual information (AMI) between the predictions of dimensions (dimension-to-dimension) and for the predictions of each dimension and the class (dimension-to-class). The MSTS for any subset of dimensions is a function of the average of the dimension-to-dimension and dimension-to-class AMI. A subset of features is chosen either through enumerating MSTS for all 2^d feature combinations, or using a wrapper on the top 5% of subsets. The algorithm first calculate the dimension-to-class (DC) correlation for each dimension which is the accuracy of the predictions $\hat{y}$ on train data by cross validation with 3 folds. Second, the dimension-to-dimension (DD) is calculated by the adjusted mutual information (AMI) between the predictions of each pair of dimensions. Finally, for each possible subset, the merit score function is calculated as follows:

$$MS(subset) = \frac{k\overline{DC}}{\sqrt{k + k(k-1)\overline{DD}}} \tag{1}$$

Where $\overline{DC}$ is the average of dimension-to-class of each dimension in the subset and $\overline{DD}$ is the average of dimension-to-dimension of each pair of dimensions in the subset. The evaluation of all dimension combinations makes MSTS infeasible for very high dimensional problems. MSTS has recently been applied to sensor data, and used in conjunction with ROCKET [9].

The recent research most closely aligned to our work is described in [7], where dimensions are selected based on distances between series within classes.

Algorithm 1 MSTS($\boldsymbol{X}, y, |X|$)

```
Parameters: Training data $\boldsymbol{X}$, labels $y$, the number of dimensions $|X|$
 1: for $i \leftarrow 1$ to $|X|$ do
 2:     $\hat{y_i} \leftarrow$ CrossValidate($\boldsymbol{X_i}$, $classifier : DTW$, $folds : 3$)
 3:     $\boldsymbol{DC_i} \leftarrow accuracy(\hat{y_i}, y_i)$
 4: for $(i, j)$ in $pairs(|X|)$ do
 5:     $\boldsymbol{DD_{i,j}} \leftarrow AMI(\hat{y_i}, \hat{y_j})$
 6: $bestSubset \leftarrow \emptyset$
 7: $bestScore \leftarrow -\infty$
 8: for each $subset \subseteq |X|$ do
 9:     $subsetScore \leftarrow MS(subset)$
10:     if $subsetScore > bestScore$ then
11:         $bestSubset \leftarrow subset$
12:         $bestScore \leftarrow subsetScore$
13: return $bestSubset$
```

A synthetic series that characterises each dimension/class combination is found through averaging the relevant dimension of series belonging to that class. A matrix of the pairwise Euclidean distance between all dimension/class centroids is then found. Three algorithms were proposed to use this $d \times c \cdot (c-1)$ distance matrix for dimension selection.

1. The **KMeans** approach applies k-means clustering (with $k = 2$) on the distance matrix to separate the channels. The cluster centroid represents the mean distance of dimensions across all class pairs and the average of the centroid describes the within cluster variation of dimensions. The kmeans algorithm selects all dimensions in the cluster with the largest average.
2. The **Elbow Class Sum (ECS)** algorithm sums each row of the distance function, then uses the elbow cut method [20] to select dimensions based on the rank order of the sums.
3. The **Elbow Class Pairwise (ECP)** iterates through every class pair, selects the best set of dimensions for that pair using the same elbow cut method as ECS and finally takes the union of dimensions over all pairs.

KMeans, ECS and ECP are compared to full enumeration of dimension subsets and random selection with the accuracy of a range of TSC classifiers, including ROCKET, used to measure performance on evaluate the effectiveness on the multivariate problems in the UEA/UCR time series archive [4].

3 Dimension Selection for HIVE-COTEv2.0

We propose a range of methods for dimension selection, including adaptations of the algorithms described in Section 2, with the goal of making HC2 more efficient.

Our classifier pipeline involves dimension selection followed by the HC2 classifier. We want to evaluate the effect of changing the dimension selection mechanism whilst keeping everything else the same. Dimension selection is either through scoring and ranking then selection or dimension subset evaluation.

Our first filtering approach is to employ ROCKET as a mechanism for scoring features from the training data, then using the elbow method to select features. This involves scoring a ROCKET classifier on each dimension independently, then ranking dimensions. We consider three scores all based on ROCKET predictions found through three fold cross validation:

- Accuracy (A): proportion of cases correctly classified.
- Silhouette (S): As alternative to using accuracy the silhouette method used in clustering to determine the optimal numbers of clusters. It is a score that goes from -1 to 1 indicating how good is the clustering based on the distances within a cluster and their differences to the points from other clusters. To use in dimension selection, the train data is cross validated with 3 folds and the predictions are used as clusters. The formula for the silhouette method is:

$$S = \frac{(b-a)}{max(a,b)} \tag{2}$$

  where a is the mean distance between data points in the same cluster and b is the mean distance between all other data points of the next nearest cluster.
- Adjusted Mutual information (M). It is a variation of mutual information that adds an element of chance, usually used in clustering. The formula is:

$$AMI(U,V) = \frac{MI(U,V) - E(MI(U,V))}{avg(H(U),H(V)) - E(MI(U,V))} \tag{3}$$

We also consider using both ECP and ECS as a filtering algorithm for HC2. As another cluster variant, we propose that instead of calculating the centroid distances as with ECP and ECS, we calculate the distance between each instance and the centroid which is calculated as the mean vector of all instances that belong to that class. This method is called CLUSTER. Finally, we also evaluate using MSTS subset selection algorithm, although we make two changes to the version described in 2.2.

- We use ROCKET instead of DTW as classifier on line 2 of Algorithm 1;
- The exhaustive subset selection done on lines 8-12 of Algorithm 1 is infeasible for some problems, because there are 2^d possible subsets of attributes. Instead, a forward selection procedure is used where the best k subsets starting with size two are selected and one dimension is added per step until the merit score function MSTS stops improving.

Table 1 summarises the attribute selection methods used in our evaluations.

Table 1. Summary of different dimension selection ranking methods with elbow method.

Attribute ranking then selection with elbow method	
Algorithm	Ranking
ECS	Sum of difference between centroid pair distance [7]
ECP	Union of sum of individual centroid pair distances [7]
CLUSTER	Error between centroid and examples
ROCKET$_A$	Accuracy of ROCKET predictions on each dimension
ROCKET$_S$	Sillouette of ROCKET predictions on each dimension
ROCKET$_M$	AMI of ROCKET predictions on each dimension
Attribute subset selection	
MSTS	Subset selection using merit score [10]
KMeans	Cluster distance function [7]

Table 2. Summary of 15 data sets used in experimentation. (*) indicates a padded series, bold indicates a data set new to the UEA archive.

Name	Train size	Test size	Dimensions	Length	Classes
ArticularyWordRecognition	275	300	9	144	25
DuckDuckGeese	50	50	1345	270	5
EMOPain	1093	50	30	180	3
FingerMovements	316	100	28	50	2
MotionSenseHAR	217	144	12	200	6
HandMovementDirection	160	74	10	400	4
Heartbeat	204	205	61	405	2
JapaneseVowels(*)	270	370	12	25	9
MindReading	727	653	204	200	5
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PEMS-SF	267	173	963	144	7
PhonemeSpectra	3315	3353	11	217	39
Siemens	700	395	39	180	10
SpokenArabicDigits (*)	6599	2199	13	65	10

4 Evaluation

We use the time series machine learning toolkit sktime [12] for our experiments. All of the algorithms used have been implemented in sktime format and are available at the GitHub repository associated with this page[2]. We are in the process of porting a subset of the algorithms into the main sktime repository, and they should be available directly from sktime version 0.14.

[2] https://github.com/sktime/dimension-selection

4.1 Data

The UEA multivariate time series repository contains 30 datasets from a wide range of fields such as EEG classification and human activity recognition [19]. In our experience, filtering does not improve the performance of HC2, so our priority is improving efficiency. Low dimensional data can mask the performance differences of filtering algorithms, so we restrict our attention to higher dimensional problems, which we define as nine or more dimensions. Ideally, we would set the threshold even higher, but there are just nine equal length problems in the archive with nine or more dimensions. There are also two high dimensional data that are unequal length: JapaneseVowels and SpokenArabicDigits. We made these equal length by padding to the longest length. We also include four new high dimensional datasets into the archive to help improve the power of our tests of performance. These four datasets are available on the UEA archive website[3].

EMOPain: The goal of the project that generated this data was the automatic detection of pain behaviours [8] and pain levels, based on data collected from people with chronic pain performing movements that are identical to those that make up daily physical functioning. The data consist of 26 sensor calculating angle positions on distinct parts of the body and 4 electromyography sensor that have the objective of measure the electric signals generated by a muscle when is moved. The sensors are positioned on the upper fibres of trapezium and on the lumbar para spinal muscles approximately at the 4/5 lumbar vertebra.

MindReading: The data consists of MEG recordings [11] of a single subject, made during two separate measurement sessions (consecutive days). In each session the subject was watching five different movie categories without audio. The goal is to predict the category of the movie the subject is watching.

Siemens: This data consist of a group of sensors from four tanks that pumps water from a reservoir tank to three small tanks [2]. The goal is to detect the type of failure the tank is experiencing based on the value of the different sensors.

MotionSenseHAR: This dataset includes time series data generated by accelerometer and gyroscope sensors (attitude, gravity, user acceleration, and rotation rate) [13]. A total of 24 participants in a range of gender, age, weight, and height performed 6 activities in 15 trials in the same environment and conditions: downstairs, upstairs, walking, jogging, sitting, and standing. With this dataset, we aim to look for personal attributes fingerprints in time-series of sensor data, i.e. attribute-specific patterns that can be used to infer gender or personality of the data subjects in addition to their activities.

[3] www.timeseriesclassification.com

4.2 Experiments

The experiments were carried out on the High Performance Computing Cluster supported by the Research and Specialist Computing Support service at the University of East Anglia. Each classifier was trained on the same 30 train test resamples of the 15 high dimensional datasets. Build time was limited to 7 days. Our performance metric is test set accuracy. To compare multiple classifiers on multiple data sets with use ranks rather than accuracy, and we use critical difference diagrams [6] to display average ranks and cliques: a clique is a group of classifiers that are labelled as not significantly different to each other. We find cliques through pairwise comparison at the 5% alpha level with an adjustment for multiple testing commonly called the Holm correction. This adjustment is less severe than a Bonferroni adjustment: we order classifiers by rank then start with the best performing classifier as our control. We pairwise test using Wilcoxon sign-rank test in order, making the adjustment as to the maximum size of the clique. Thus, if testing 11 classifiers, the maximum number of tests to find the top clique is 10, so we require a p-value of alpha/10 to be considered significantly different. Once we find a classifier that fails the pairwise test, we form a clique of those prior to it. We then repeat the process with the second best classifier, with the caveat that if a clique is found that is contained with one found already, we ignore it.

This is the most robust way we have found to form cliques, but it can still lead to anomalies. Given three classifiers, A, B and C, where A is the highest rank and C the lowest, it is possible that A is significantly better than B but not significantly better than C. However, our approach would put A and C in different cliques. We intend to move towards a more graphical display of pairwise tests and recommend that CD diagrams should only form part of the methodology of presenting results that compare classifiers.

5 Results

Our first experiment defines the scope of further experiments by bounding our expectations as to the accuracy of filtering prior to training HC2. Figure 1 shows the ranks of full HC2, full ROCKET, HC2 with 20% of dimensions selected randomly and HC2 with 60% of dimensions selected randomly. Figure 1 illustrates that HC2 is significantly more accurate than ROCKET. We reran the experiments with the sktime implementation of HC2, so this serves to recreate the results reported in [16]. HC2 is, on average, 2.5% more accurate than ROCKET, winning on 11 problems and losing on 4. By default, HC2 is not configured for speed: it takes several hours to complete one resample of experiments, whereas ROCKET takes minutes. Figure 1 also shows that our basic straw man for comparison, randomly selecting 20% or 60% of attributes, results in a significant loss of accuracy in HC2. Our second experiment addresses the question as to whether applying any of the dimension selection algorithms listed in Table 1 can speed up HC2 without loss of accuracy. Figure 2 shows the relative ranked performance of eight different filtering algorithms in addition to the four classifiers

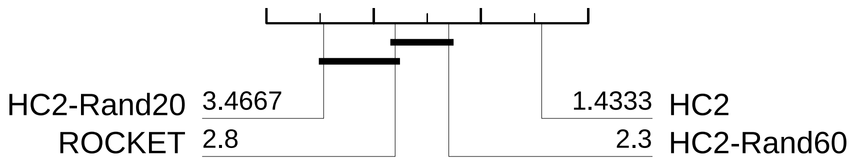

Fig. 1. Critical difference diagram for comparing ROCKET, full HC2 and HC2 with random dimension selection.

shown in Figure 1. Cliques were formed with the p-values shown in Table 3. The average accuracy data for four classifiers is provided in Table 4, and full results are available in the associated repository.

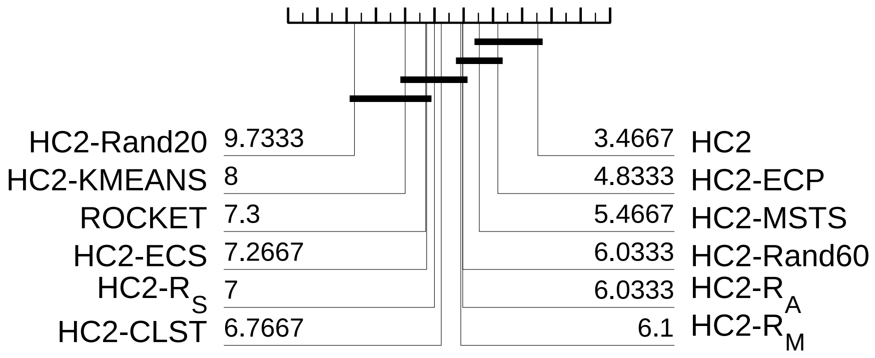

Fig. 2. Critical difference diagram for comparing all dimension selection methods proposed.

Our first conclusion is that our hypothesis that ROCKET could be a good way of filtering dimensions for HC2 is not supported by these results. The three ROCKET variants are significantly worse than HC2, and no better than randomly selecting 60% of dimensions. Table 4 shows that using HC2-R_A results on average in an approximate 1% decrease in accuracy. However, the two filters ECP and MSTS both achieve an accuracy rank that is not significantly worse than HC2. Table 4 shows that HC2-ECP performs very similarly to full HC2, but that HC2-MSTS may slightly reduce accuracy on average.

Of course, filtering will perfectly recreate HC2 results if it selects all dimensions. Table 5 shows the proportion of dimensions selected for three classifiers. On average, HC2-MSTS selects fewer dimensions, and in some cases, massively fewer. For example, with PEMS-SF[4] it selects just 13 out of 963 attributes and

[4] http://www.timeseriesclassification.com/description.php?Dataset=PEMS-SF

Table 3. P-values for pairwise Wilcoxon rank-sum test on 15 high dimensional MTSC problems.

	ECP	MSTS	R60	R_A	R_M	CLST	R_S	ECS	RCKT	KMNS	R20
HC2	0.753	0.198	0.004	0.011	0.016	0.019	0.004	0.096	0.041	0.048	0.001
ECP		0.256	0.272	0.173	0.246	0.124	0.140	0.011	0.100	0.020	0.015
MSTS			0.394	0.570	0.056	0.125	0.173	0.334	0.334	0.281	0.006
R60				0.551	0.551	0.041	0.272	0.246	0.173	0.233	0.001
R_A					0.975	0.096	0.158	0.433	0.307	0.233	0.004
R_M						0.246	0.397	0.551	0.496	0.496	0.015
CLST							0.331	0.510	1.000	0.691	0.140
R_S								0.925	0.910	0.158	0.002
ECS									0.532	0.683	0.061
RCKT										0.826	0.307
KMNS											0.364

achieves an accuracy very close to that of full HC2. Each dimension in PEMS-SF is a single traffic sensor, and the data is measured over time. There will be high correlation between adjacent sensors, and HC2-MSTS is effective at removing a high degree of the redundancy in this data.

Similarly, with DuckDuckGeese[5], HC2-MSTS chooses just 33 of the 1345 attributes, and gets comparable accuracy to HC2, although HC2-ECP actually improves on HC2 when selecting about a third of attributes. DuckDuckGeese contains audio spectrograms of different bird species, and each dimension represents a frequency range. Both HC2-MSTS and HC2-R_A over filter on the problem MindReading, whereas HC2-ECP correctly selects a larger number of dimensions and achieves a similar accuracy to full HC2.

Table 6 summarises the average training time for the HC2 and the three main filtering methods, and includes the time taken to filter. There is hardly any difference in time between the three algorithms, and each method takes about 60% of the time of full HC2.

MSTS and R_A both rely on ROCKET to score dimensions, but differ primarily in how attributes are selected. It seems that the subset selection used by MSTS may be marginally better than the elbow method used by R_A, although our tests do not have the power to reject the null hypothesis that there is no difference. ECP is based on distances between predictions and rather than accuracy, and uses 1-NN DTW to make predictions. It tends to select more attributes that MSTS, but the extra time the resultant HC2 classifier takes is offset by the more time consuming components of MSTS.

Overall, these experiments show that, although there is no significant difference between HC2-ECP, HC2-MSTS and HC2-R_A, only ECP and MSTS reduce dimensionality without reducing the accuracy of HC2 significantly. Both can prove useful tools for filtering prior to using HC2 with high dimensional data, with ECP marginally preferred because it more closely recreates HC2 results.

[5] http://www.timeseriesclassification.com/description.php?Dataset=DuckDuckGeese

Table 4. Accuracy of four classifers averaged over 30 resamples of 15 high dimensional datasets.

Name	HC2	HC2-ECP	HC2-MSTS	HC2-R$_A$
ArticularyWordRecognition	**99.51**	**99.51**	99.37	99.27
DuckDuckGeese	55.07	**57.93**	55.2	54
EMO	88.78	**89.87**	89.05	88.77
FingerMovements	**55.57**	52.53	54.5	55
MotionSenseHAR	**99.71**	99.67	99.66	99.67
HandMovementDirection	41.26	**42.48**	41.53	41.89
Heartbeat	73.59	**74.29**	73.06	73.24
JapaneseVowelsEq	93.15	**94.51**	93.06	91
MotorImagery	53.63	52.77	**53.73**	53.53
MindReading	**68.36**	68.17	60.81	60.44
NATOPS	**89.04**	87.54	85.98	87.17
PEMS-SF	**99.96**	97.23	99.92	**99.96**
PhonemeSpectra	**32.01**	31.72	**32.01**	31.67
Siemens	**100**	**100**	99.92	**100**
SpokenArabicDigitsEq	**99.67**	99.62	99.64	99.63
Average	76.62	76.52	75.83	75.68
Wins	9	7	2	2

Table 5. Percentage of dimensions used.

Name	d	HC2-ECP	HC2-MSTS	HC2-R$_A$
ArticularyWordRecognition	9	100	**55.17**	57.09
DuckDuckGeese	1345	29.93	**2.48**	21.08
EMOPain	30	55.29	**31.72**	35.52
FingerMovements	28	38.18	**17.49**	50.49
MotionSenseHAR	12	86.78	**30.75**	65.8
HandMovementDirection	10	83.1	**44.14**	56.21
Heartbeat	61	15.38	**13.85**	45.56
JapaneseVowelsEq	12	75.57	97.41	**62.07**
MotorImagery	64	25.11	**6.25**	54.8
MindReading	204	61.56	**12.81**	16.23
NATOPS	24	79.31	**45.98**	63.07
PEMS-SF	963	35.03	**1.38**	13.8
PhonemeSpectra	11	**18.18**	100	59.25
Siemens	39	30.77	**10.88**	55.61
SpokenArabicDigitsEq	13	53.85	**53.85**	54.91
Average		52.54	**34.94**	47.43

6 Conclusion

There are of course limitations to this study. We have focused purely on the HC2 classifier, with the justification that this is the current state of the art. However, HC2 is a meta ensemble of four ensemble classifiers, each of which handles multi-

Table 6. Train time in hours, including the time to filter.

Name	HC2	HC2-ECP	HC2-MSTS	HC2-R_A
ArticularyWordRecognition	3.80	4.13	**3.42**	3.47
DuckDuckGeese	8.16	5.34	4.09	**3.87**
EMO	23.99	17.72	12.57	**11.52**
FingerMovements	4.15	3.29	**2.98**	3.70
HAR	21.36	21.70	**12.82**	19.39
HandMovementDirection	3.63	3.55	**3.20**	3.46
Heartbeat	6.71	**3.94**	3.96	4.42
JapaneseVowelsEq	3.04	3.07	3.06	**3.00**
MotorImagery	33.06	15.35	**9.32**	26.44
MindReading	42.38	29.9	**14.06**	15.56
NATOPS	3.06	3.00	**2.70**	2.90
PEMS-SF	32.64	13.54	**5.74**	9.44
PhonemeSpectra	112.49	**68.18**	113.20	85.18
Siemens	14.96	7.70	**4.96**	10.16
SpokenArabicDigitsEq	118.89	79.21	78.59	**76.84**
Sum	432.31	279.63	**274.65**	279.33

variate data differently. It may be interesting to explore how filtering may affect each component, and indeed other classifiers. It may be more useful to embed the dimension selection within the component classifiers to create different feature subsets for each. we have also only evaluated dimension selection algorithms. Dimension creation algorithms may also be of use in MSTC.

We have donated four new datasets to the archive, but even then we are limited to evaluation with 15 datasets. Furthermore, many of these are not genuinely high dimensional. More realistic cut off points would be 50 or 100 dimensions. MTSC data is very diverse in origin, and finding algorithms significantly better than others over all problem domains may prove unrealistic. In future work we will continue to seek out new high dimensional problems, and our intention is to focus more specifically on EEG/MEG datasets, to make our research question more specific to that problem domain.

References

1. Bagnall, A., Flynn, M., Large, J., Lines, J., Middlehurst, M.: On the usage and performance of HIVE-COTE v1.0. In: proceedings of the 5th Workshop on Advances Analytics and Learning on Temporal Data. Lecture Notes in Artificial Intelligence, vol. 12588 (2020)
2. Bierweiler, T., Labisch, D.: Four-tank batch process in smart automation. Tech. rep. (2021), https://github.com/thomasbierweiler/FaultsOf4-TankBatchProcess
3. Bostrom, A., Bagnall, A.: Binary shapelet transform for multiclass time series classification. proceedings of 17th International Conference on Big Data Analytics and Knowledge Discovery (2015)

4. Dau, H., Bagnall, A., Kamgar, K., Yeh, M., Zhu, Y., Gharghabi, S., Ratanamahatana, C., Chotirat, A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica 6(6), 1293–1305 (2019)
5. Dempster, A., Petitjean, F., Webb, G.: ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery 34, 1454–1495 (2020)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
7. Dhariyal, B., Le Nguyen, T., Ifrim, G.: Fast channel selection for scalable multivariate time series classification. In: proceedings of the 6th Workshop on Advances Analytics and Learning on Temporal Data. Lecture Notes in Computer Science, vol. 13114 (2021)
8. Egede, J.O., Olugbade, T.A., Wang, C., Song, S., Bianchi-Berthouze, N., Valstar, M.F., Williams, A.C., Meng, H., Aung, M.S.H., Lane, N.D.: Emopain challenge 2020: Multimodal pain evaluation from facial and bodily expressions. 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020) pp. 849–856 (2020)
9. Kathirgamanathan, B., Buckley, C., Caulfield, B., Cunningham, P.: Feature subset selection for detecting fatigue in runners using time series sensor data. In: proceedings of 3rd International Conference on Pattern Recognition and Artificial Intelligence. Lecture Notes in Computer Science, vol. 13363 (2022)
10. Kathirgamanathan, B., Cunningham, P.: A feature selection method for multidimension time-series data. In: proceedings of the 5th Workshop on Advances Analytics and Learning on Temporal Data. Lecture Notes in Computer Science, vol. 12588 (2020)
11. Klami, A.: Proceedings of ICANN/PASCAL2 Challenge: MEG Mind Reading. Tech. rep. (2011), http://urn.fi/URN:ISBN:978-952-60-4456-9
12. Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: A unified interface for machine learning with time series. ArXiv e-prints arXiv:1909.07872 (2019), http://arxiv.org/abs/1909.07872
13. Malekzadeh, M., Clegg, R.G., Cavallaro, A., Haddadi, H.: Mobile sensor data anonymization. In: Proceedings of the International Conference on Internet of Things Design and Implementation. pp. 49–58. IoTDI '19, ACM, New York, NY, USA (2019), http://doi.acm.org/10.1145/3302505.3310068
14. Middlehurst, M., Large, J., Cawley, G., Bagnall, A.: The temporal dictionary ensemble (TDE) classifier for time series classification. In: proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 12457, pp. 660–676 (2020)
15. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (CIF) classifier for time series classification. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 188–195. IEEE (2020)
16. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: HIVE-COTE 2.0: a new meta ensemble for time series classification. Machine Learning 110, 3211–3243 (2021)
17. Pasos-Ruiz, A., Flynn, M., Bagnall, A.: Benchmarking multivariate time series classification algorithms. ArXiv e-prints ArXiv:2007.13156 (2020), http://arxiv.org/abs/2007.13156
18. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of

recent algorithmic advances. Data Mining and Knowledge Discovery 35(2), 401–
–449 (2021)

19. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.J.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 35, 401 – 449 (2021)
20. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a "kneedle" in a haystack: detecting knee points in system behavior. In: proceedings of 31st International Conference on Distributed Computing Systems Workshops (2011)
21. Yang, K., Yoon, H., Shahabi, C.: CLeVer: A feature subset selection technique for multivariate time series. In: proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining. Advances in Knowledge Discovery and Data Mining (2005)