Is it worth changing pattern recognition methods for structural health monitoring?

2017 J. Phys.: Conf. Ser. 842 012006

(http://iopscience.iop.org/1742-6596/842/1/012006)

IOP Conf. Series: Journal of Physics: Conf. Series **842** (2017) 012006 doi:10.1088/1742-6596/842/1/012006

Is it worth changing pattern recognition methods for structural health monitoring?

L A Bull, K Worden, E J Cross, N Dervilis

Department of Mechanical of Engineering, University of Sheffield, Sheffield, UK

Email: lbull1@sheffield.ac.uk

Abstract. The key element of this work is to demonstrate alternative strategies for using pattern recognition algorithms whilst investigating structural health monitoring. This paper looks to determine if it makes any difference in choosing from a range of established classification techniques: from decision trees and support vector machines, to Gaussian processes. Classification algorithms are tested on adjustable synthetic data to establish performance metrics, then all techniques are applied to real SHM data. To aid the selection of training data, an informative chain of artificial intelligence tools is used to explore an active learning interaction between meaningful clusters of data.

1. Introduction and problem statement

The main aim of structural health monitoring (SHM) is the development of diagnostic systems capable of detecting and classifying structural degradation. Machine learning techniques have become increasingly relevant to the field of SHM as advanced pattern recognition algorithms can learn complicated relationships from system data whilst offering a variety of techniques for novelty detection [1].

When following a data-based approach to SHM, a lack of information representing the damaged structure can result in a significant undersupply of labelled data. This makes machine learning tasks – such as classification – very difficult, as the information required for wholly supervised methods is not available. Consequently, a need for adaptive semi-supervised learning techniques, such as active learning, is very important for SHM.

The intention of this work is to demonstrate alternative strategies for using pattern recognition algorithms, and investigate their performances for SHM purposes. The main objective is to provide performance comparison metrics for a range of classification techniques, whilst illustrating an active learning methodology to select the training sets that carry the most information. Three datasets are introduced, two synthetic sets and an acoustic-emissions (AE) data set. The synthetic data is generated as a toy example in a two-dimensional feature space for SHM demonstration, and they will be used to quantify performance metrics for each classification algorithm. Each technique will then be applied to acoustic emission data, obtained from experiments concerning the box-girder of a bridge.

2. Background

2.1. Classification algorithms

The techniques presented in this paper are selected to provide performance metrics for alternative methods to the conventional choices in SHM. Some benchmark algorithms, such as multi-layer

perceptrons, have been excluded from this work as their correct application makes these experiments highly computationally expensive and they have already been discussed extensively in an SHM context. Comprehensive comparisons including a wider range of algorithms have been suggested (see Section 6.1) and MLPs will be included in this wider remit.

Basic descriptions of the techniques tested in this work are provided, for more details the reader is directed to reference [2]. All algorithms are applied in Python, using the machine learning package s*cikit-learn* [3]. During initial experiments, certain algorithm hyperparameters were defined through engineering judgement – relevant values are provided in braces. Of course, this is not good practice in practical engineering applications, as any hyperparameters must be optimised with the use of a validation subset (or otherwise); however, the elected values serve well to establish a general comparison for the purposes of this paper and will be optimised in future work.

2.1.1. Linear support vector machine (SVM). A sparse technique in which the learning task maximises the 'margin' around a linear decision boundary [2]. The margin is defined by the perpendicular distance between the decision boundary and the closest points either side (support vectors). A soft margin parameter C is introduced to allow for misclassification and prevent over fitting. {C = 1} {stopping criteria = 1e-4} {multiclass strategy: one vs. rest classifiers}

2.1.2. Kernel support vector machine (KSVM). Applies the 'kernel trick' to SVMs, transforming data into some new space to make clusters more separable with a linear decision boundary [2]. This is achieved through the use of a specific kernel function in place of any inner products in the original space. {C = 1} {kernels: polynomial 3rd degree γ=0.5 and Gaussian γ=0.5} {stopping criterion = 1e-4} {multiclass strategy: one vs. rest classifiers}

2.1.3. Relevance vector machine (RVM). A sparse kernel technique based on a Bayesian formulation that shares many characteristics with the SVM, but produces probabilistic outputs [4]. An Automatic Relevance Determination (ARD) prior distribution is placed over the SVM weight vector. This introduces precision hyper-parameters (α_i), which are optimised during training through likelihood maximisation. The use of an ARD prior over the parameter vector results in a number of the parameter weights concentrating at zero (large α_i); these weights have a small contribution to the classification, so if a given threshold is broken they are 'pruned' out. Relevance vectors are not available in *scikit-learn*, so an alternative implementation using the *scikit-learn* API is used here [5].
{kernel: linear} {threshold α_i = 1e09} {stopping criterion: 1e-03}

2.1.4. Decision tree (DT): A simple yet effective algorithm, shown in the literature to give excellent parametric performance, even when compared to more complex methods [6]. During classification, the classification and regression tree (CART) algorithm partitions the input/feature space into cuboidal regions according to threshold values that minimise an impurity measure [2]. This measure is assessed with the *Gini index*, which quantifies the 'mixture' of classes in each split; minimising this value encourages the formation of regions that contain a high proportion of data assigned to one specific class [2]. The tree will continue to grow, splitting at each node, until a predefined number of samples remain, or the Gini index falls below a given value. Given the greedy learning strategy of DTs, pruning techniques may be applied to reduce their size by removing the nodes that contribute least to classification, no pruning methods are applied here. {min. samples to split = 2} {threshold Gini = 1e-7}

Ensemble methods can be applied to any classifier with the aim to reduce the risk of overfitting [2]. As DTs are computationally inexpensive and ensemble methods have been shown to work well with the algorithm in the existing literature [6], the following ensemble methods are applied:

- Bootstrap aggregating (Bagging): 'Bootstrap' sample N random subsets with replacement from the training set. Train classifiers on each sample to create a committee of models and 'aggregate' by combining whilst minimising the unweighted average error of the base classifiers [2]. $\{N = 10\}$
- Boosting (*AdaBoost*): Combines multiple base classifiers; however, unlike bagging, the models are trained on the complete training set, with weights associated to each sample. Weights are updated during the learning process to prioritise misclassified samples from preceding models [2]. $\{N = 10\}$ {learning rate = 1}

2.1.5. Random forests (RF). Similar in concept to bagged DTs, however the algorithm is modified to introduce extra variability into the learner. Rather than using all the available features, the RF classifier selects a random subset of features to determine the best split at each node. $\{N = 10\}$ {min. samples to split = 2} {threshold Gini = 1e-7} {no. features in subset = 1}

2.1.6. Gaussian process classifiers (GPC). A probabilistic method that shares many characteristics with the RVM; however, the parametric models are abandoned and a Gaussian prior is placed over the basis functions directly [7]. Samples are then taken form this Gaussian process (GP) to create a technically infinite range of functions to describe the classification [7]. In the case of GPs, the kernel function is used to describe elements of the covariance matrix. Often a spherical covariance is assumed, but it is also possible to define an elliptical covariance using the Gaussian (rbf) kernel [7]; this introduces the hyper parameter γ_i, which is optimised during training through likelihood maximisation.
{kernel: Gaussian, initial $\gamma_i = 1$} {multi class method: one vs. rest classifier}

2.2. Active learning

Active-learning, or query-learning, is a semi-supervised machine learning approach; its main premise is that a learner can perform with increased efficiency given less training data, if it can select the data from which it learns. This eliminates the time-consuming process of labelling large volumes of trivial data, but more importantly for SHM, active learning significantly reduces the need for extensive training sets.

A wide range of active learning algorithms exist within the remit of machine learning. Some examples include the 'Active Support Vector Machine' (SVM_{active}) [8] and 'Manifold Adaptive Experimental Design' (MAED) [9]. Each employ various 'querying' methods to determine which unlabeled data carry the most information; measures include the distance from the decision boundary (SVM_{active}) or the average squared predicted error (MAED). The details of these algorithms are out of scope for this specific paper, however the key concept behind active learning tools is summarised in Figure 1.

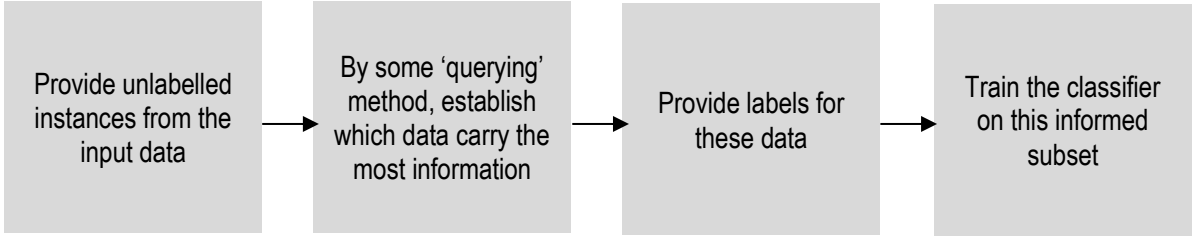

Figure 1. Active learning methodology.

2.3. Metric measures

To assess classification performance, accuracy (ACC) and F1-score ($F1$) metrics [10] will be used:

- Accuracy: percentage of correctly classified data.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (1)$$

Where: true positives = TP, true negatives = TN, false positives = FP, false negatives = FN

- F1-score: the balanced average of precision (PRE) and recall (RE). A weighted average is used to combine PRE and RE for each class and account for class imbalance [10].

$$PRE = \frac{TP}{TP + FP} \quad (2)$$

$$REC = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = 2 \times \frac{PRE \times REC}{PRE + REC} \quad (4)$$

2.4. Clustering – unsupervised

Various algorithms are available to find clusters in unlabelled data: examples include K-means [2], Affinity Propagation [11] and Gaussian Mixture Models (GMM) [2]. Initial experiments found that GMMs were particularly effective when applied to the data analysed in this work. K-means clustering is also referenced in Section 3.2, so brief descriptions are provided.

2.4.1. Gaussian mixture model (GMM). A linear superposition of K Gaussian components. After defining K number of components – and therefore clusters – a GMM is fitted to the data by updating parameters whilst using the Expectation-Maximisation algorithm [2]. {K = 3} {covariance = full} {stopping criterion = 1e-03}

2.4.2. K-means: Data is separated into some predefined number of clusters K. A cluster can be considered as a group of data whose inter-point Euclidean distances are small compared to points outside the cluster [2]. The algorithm aims to choose cluster centroids that minimise the cluster inertia or 'separation'. {K = 3} {stopping criterion = 1e-04}

3. Methodology

3.1. Data: synthetic

Synthetic 'toy' data have been generated using three 2D Gaussian distributions. These data aim to represent 3 classes of structural degradation in two-dimensional feature space, following dimension reduction. The classes are imbalanced and the total number of samples N_s = 1050 (class1=500, class2=250, class3=300). Two versions of these data were generated, a challenging set with significant cluster overlap (Figure 2) and a forgiving set with more separable clusters (Figure 3).

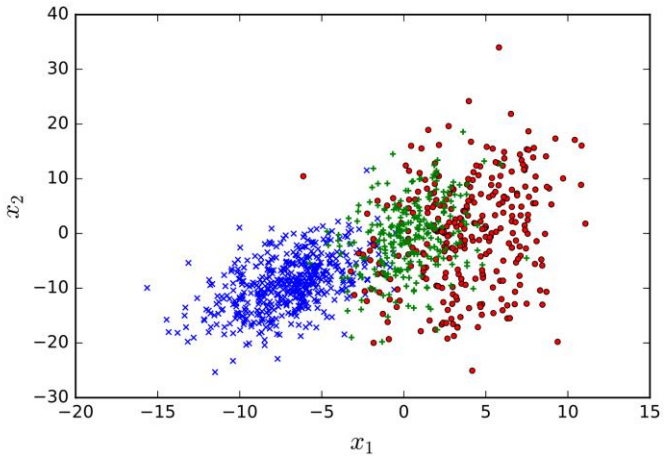

Figure 2. Challenging synthetic data: class 1 (×), class 2 (+), class 3 (○).

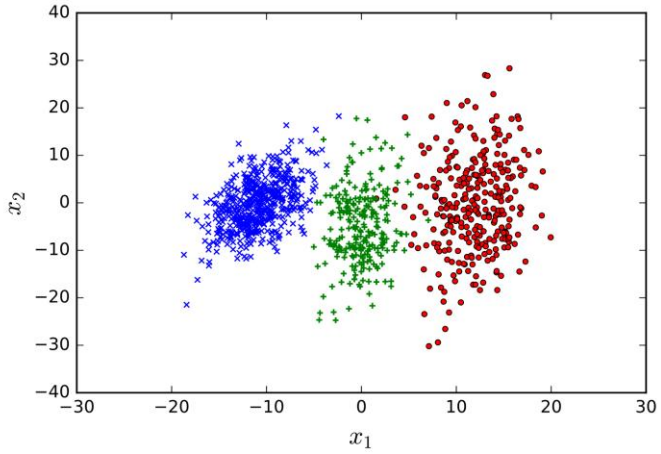

Figure 3. Easy synthetic data: class 1 (×), class 2 (+), class 3 (○).

3.2. Data: experimental acoustic emissions (AE).

Experimental data were obtained from tests at Cardiff University, concerned with collecting AE data from the box girder of a bridge [12]. Each sample represents a typical AE burst (Figure 4) and the total number of samples N_{AE} = 91. For further pre-processing and experimental details, see Reference [13]. Four traditional AE features – rise time, peak amplitude, duration and ring down count – are extracted from each burst, normalised, then projected though a linear transformation on to two dimensions using Principal Component Analysis (PCA) – Figure 5. The new co-ordinates, or principal component scores (x_1, x_2), are a linear combination of the information retained within the original features and account for the maximum variance within the data [2]. As definite labels are not available, K-means clustering is used to predict 3 class labels, which are then assumed as the definitive classes (class1=22, class2=59, class3=10).

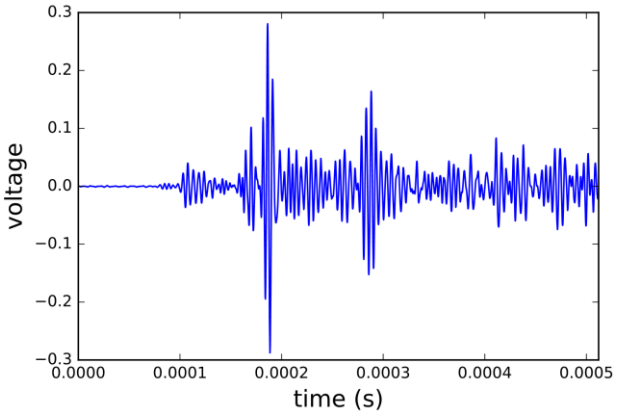

Figure 4. Time signal of a typical AE burst.

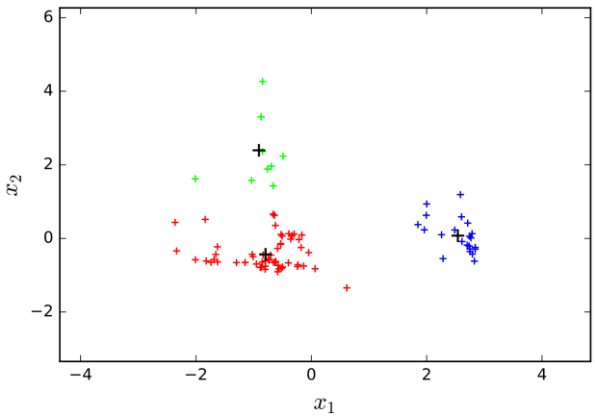

Figure 5. PCA reduction applied to AE features, labelled following K-means clustering (K = 3).

3.3. Method

Each data set is initially split into two subsets: a training set and test set. Basic active learning principles are demonstrated by selecting a given fraction of the most informative points for training – each classifier is then trained on these data. All the remaining data will be used as a test set to establish the accuracy and F1-score.

3.3.1. Selecting the training set. A simplified active learning regime is used to select 15% - 45% of the most informative synthetic data in 10% increments. Steps are listed below and illustrated in Figure 6:

I. Start with unlabelled data.
II. Fit a GMM model to find clusters within the data, without knowledge of labels.
III. Randomly sample a given fraction of data from each cluster to select representative examples.
IV. Provide class labels to this informed training-set.

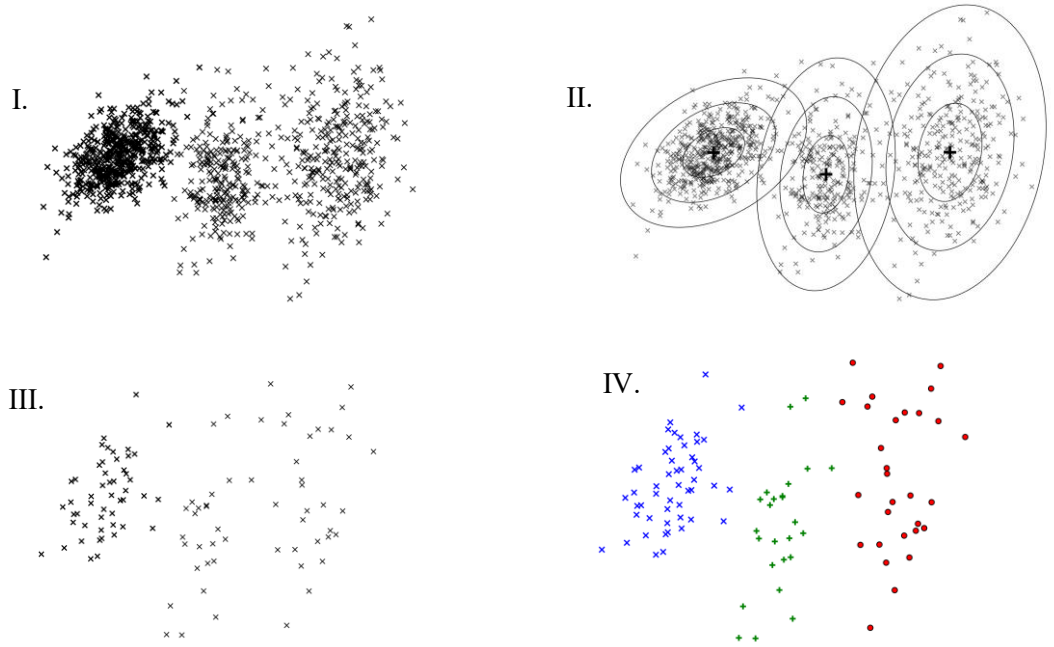

Figure 6. Visual representation of the active learning steps: 25% sample, easy synthetic data.

The querying regime used here hopes to provide a smaller, more informed training set than the alternative of one random sample across the whole data. Applying more sophisticated active learning algorithms to SHM is out of the scope of this paper but will be explored in future projects.

3.3.2. Data applications. For each training set size of the synthetic data, all classification algorithms will be trained and then tested 1000 times. The mean F1 and ACC, along with one standard deviation will be provided. Following analysis of the performance metrics, algorithms will be applied to the AE data to illustrate the relevance of this work to the remit of SHM. A 15% sample training set is used with the AE data, as this provided an array of performance metrics.

4. Performance metrics & discussion

The 'No Free Lunch Theorem' tells us that any two optimisation algorithms become equivalent when their performance is averaged across all possible problems [14]. In other words, classification results are highly data-dependant and there exists no single, optimal algorithm. The metrics presented in Figure 7 and Figure 8 are intended to illustrate algorithm behaviour across a range of scenarios to aid algorithm selection when investigating SHM.

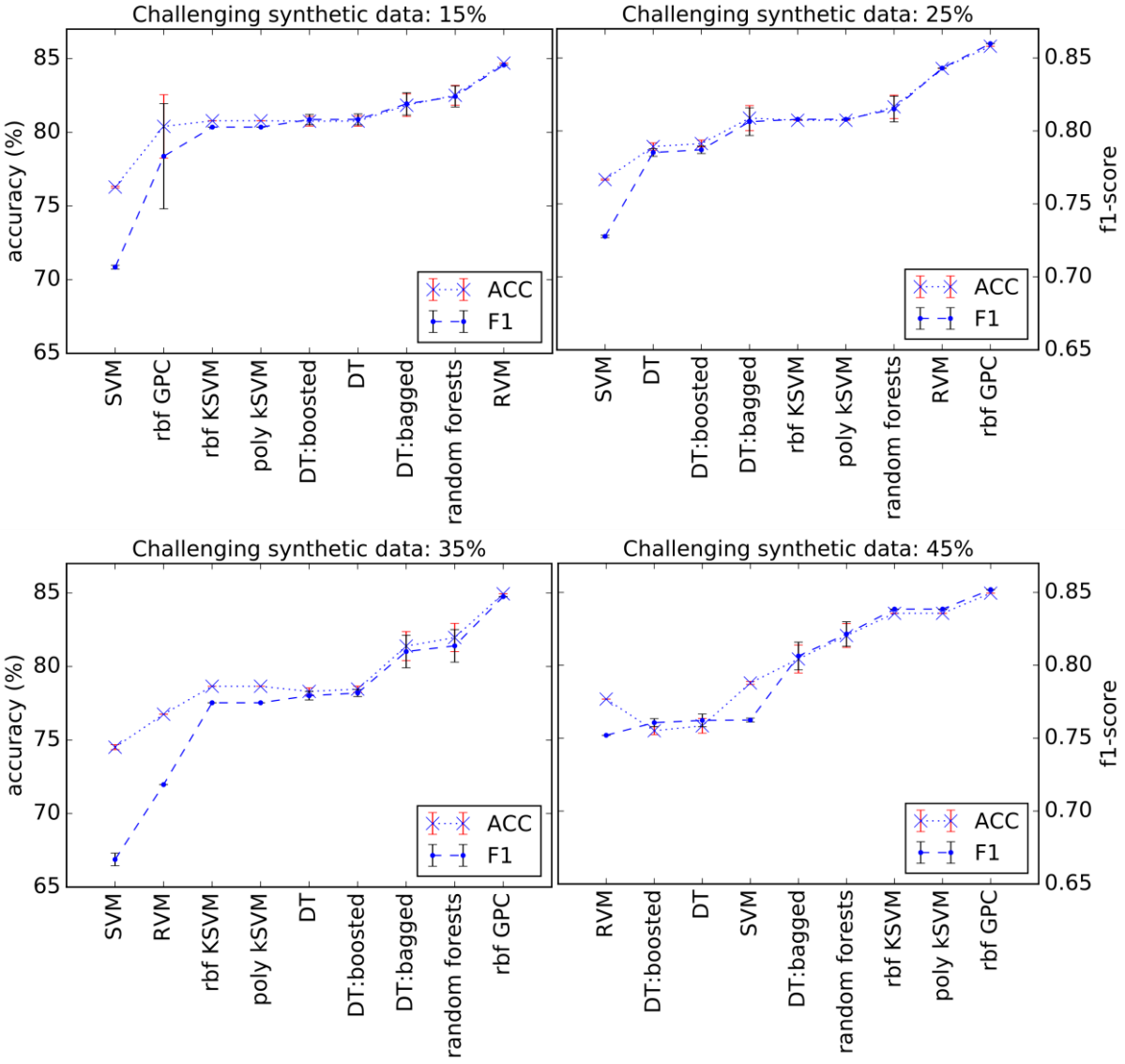

Figure 7. Challenging synthetic data performance metrics, ranked by F1-score.

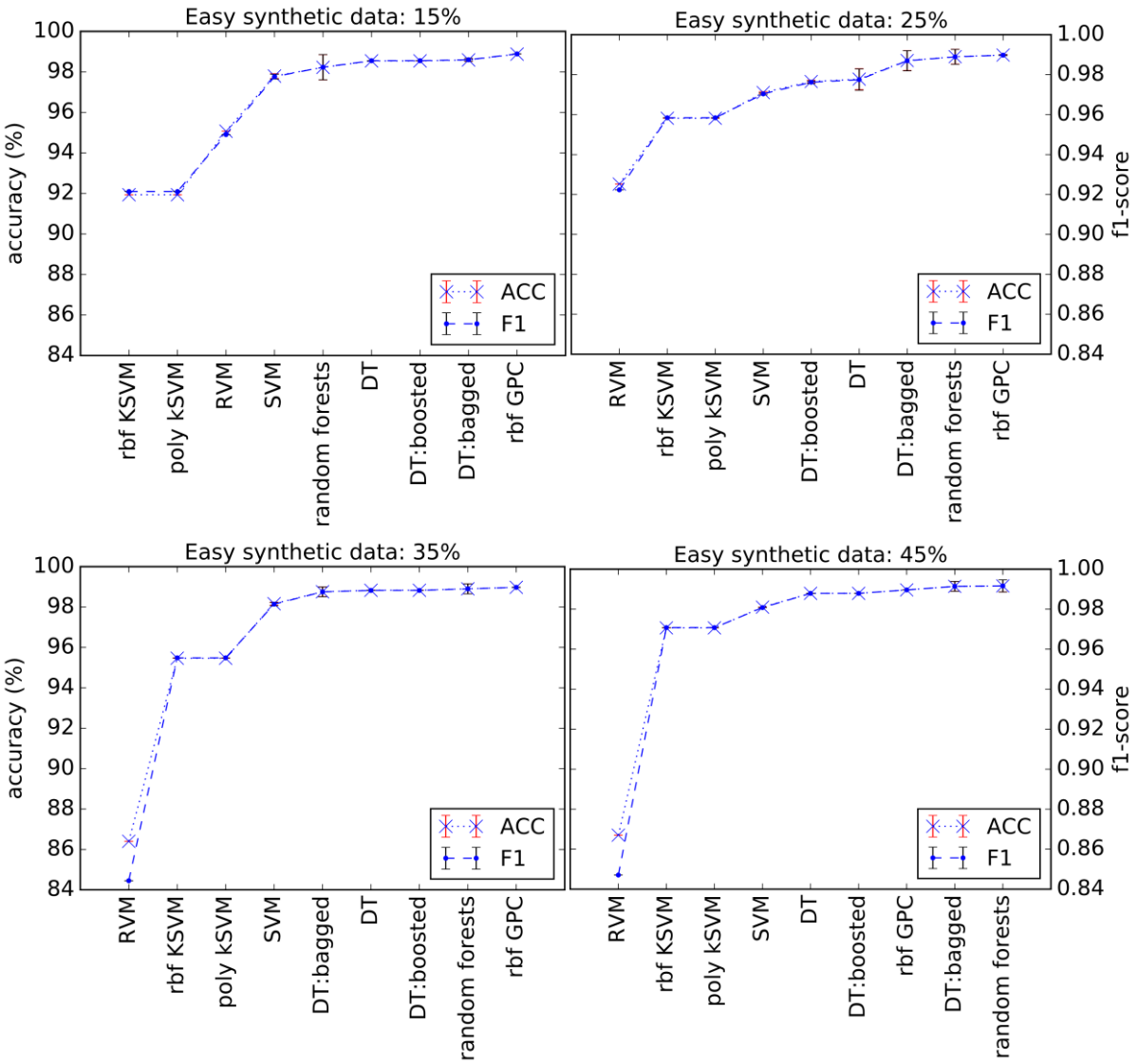

Figure 8. Easy synthetic data performance metrics, ranked by F1-score.

4.1. Linear support vector machine

Linear SVMs poorly predict class labels when applied to challenging data (Figure 7). Considering the linear decision boundary this is not surprising – it can be seen from Figure 2 that the samples are far from linearly separable. Intuitively, it then makes sense that the labels predicted for the easy data (Figure 8) are more accurate, where SVMs show good overall performance. A low variance is produced when compared to alternative methods, which is most likely due to the sparse nature of the algorithm, as every classifier is defined by similar support vectors.

4.2. Kernel support vector machine

For challenging datasets, the kernel trick succeeds in improving SVM performance – particularly with a polynomial kernel. However, when applied to the easy data, kernel techniques become a disadvantage. This is reasonable, as the data is already relatively separable in the original space (Figure 3), therefore transforming the data into some alternative space is unlikely to improve separation.

4.3. Relevance vector machine

RVMs perform very well when used with smaller training sets of the challenging data. The Bayesian framework of the classifier appears to reduce the risk of overfitting, enabling it to perform especially well with smaller training samples; however, their relative performance drops as more training data become available. When applied to the easy data, the RVMs rank lower, again their performance deteriorates as the size of the training set increases. The pruning regime could hinder competitiveness as more data becomes available and the complexity reduces, so optimisation of the precision threshold $\{\alpha_i = 1e09\}$ must still be considered. The RVMs tested in this work are shown to be optimal when applied to complicated data sets with limited training resources.

4.4. Decision tree methods

In agreement with the findings in the existing literature [6], DT methods appear to provide the best overall performance across various training sets and data complexities. Despite their simplicity, respectable ACC and F1-scores are produced throughout, even with smaller training sets. As expected, metric variance increases with the use of ensemble methods – particularly RFs. However, this increase in variance comes with increased performance, showing that averaging techniques are successfully reducing the susceptibility of DTs to overtraining.

4.5. Gaussian process classification

GPCs provide very accurate predications in most experiments, indicating the resilience of the probabilistic algorithm to overfitting. However, metric rankings fall (with increased variance) when applied to the challenging data with a 15% training set. In this case, the mapping of the Gaussian (rbf) kernel might not be suited to the specific manifold of the smallest training sample.

5. Acoustic emission application

The AE data can be described as relatively ‘easy’ as it has linearly separable clusters. A small training set (15%) is used to challenge each classifier. Each classification method is then trained 100 times, and the mean along with one standard deviation is provided – see Figure 9.

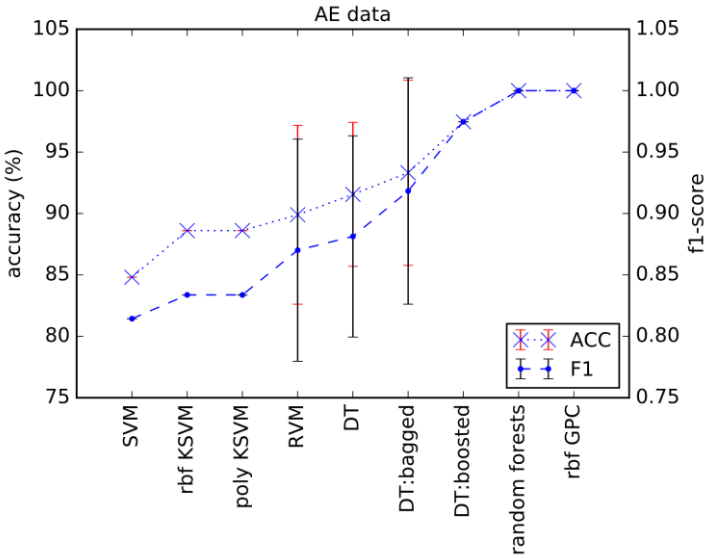

Figure 9. Classification methods applied to AE data.

In agreement with the findings from synthetic experiments, linear SVMs work well to classify these data as they are linearly separable in the original space; consequently, transformations with kernel methods show no advantage (KSVM). With this in mind, GPCs should be tested on these data with

alternative kernel functions, particularly a linear kernel. RVMs provide equivalent performance to linear SVMs, which is intuitive given model similarities and the simplicity of the data. DT algorithms provide reasonable classification performance, but with a lot of variation. The use of ensemble methods offers little advantage for this application as the data set is particularly small; for this reason, introducing extra variability fails to improve the already weak learners.

6. Conclusions

This work has illustrated the necessity for a wide variety of algorithms whilst investigating SHM. The performance of any given classification is highly dependent on the properties of the data themselves, as well as the choice of any relevant parameters. Decision Tree methods provide excellent overall performance across a range of scenarios despite their simplicity, even when applied to complex data and a small training set. Probabilistic algorithms – including GPCs and RVMs – show potential, particularly when applied to challenging data. With these methods, a variety of kernel functions and pruning parameters should be explored through cross-validation for optimal performance. Traditional SHM techniques, such as SVMs, continue to provide competitive metrics; especially when the data is 'easy'. Active learning interactions have been successfully demonstrated in this work; the data-dependence of classification performance highlights the potential advantages of this tool in selecting an informative data set in the context of SHM.

6.1. Future work

For comprehensive metric comparisons, any predefined parameters must be optimised through cross validation (or otherwise); particularly those relating to non-probabilistic methods. It would be interesting to compare a wider variety of kernel functions for the computationally expensive algorithms, whilst including multi-layer perceptron metrics. Ensemble methods could be applied to more techniques in an SHM context; however, the reader is directed to the work done in reference [6] as it provides a comprehensive analysis for non-probabilistic methods. Alternative clustering algorithms should be tested to provide an array of unsupervised techniques whilst selecting data for training, along with the use of more sophisticated active learning algorithms.

Acknowledgments

The support of the UK Engineering and Physical Sciences Research Council (EPSRC) through grant reference numbers EP/J016942/1 and EP/K003836/2 is gratefully acknowledged. Further thanks are extended to Karen Holford and Rhys Pullin at Cardiff University for providing the AE data.

References

[1] Farrar C R and Worden K 2012 *Structural Health Monitoring: A Machine Learning Perspective* vol 1 (Chichester: John Wiley & Sons)
[2] Bishop C M 2006 *Pattern Recognition and Machine Learning* vol 1, ed M Jordan, J Kleinberg, et al. (New York: Springer)
[3] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V and Vanderplas J 2011 Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12** 2825-30
[4] Tipping ME 2001 Sparse Bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.* **1** 211-44
[5] Ritchie J and Feinberg J 2016 scikit-rvm *GitHub repository*
[6] Caruana R and Niculescu-Mizil A 2006 An empirical comparison of supervised learning algorithms. *Proc. 23rd Int. Conf. Mach. Learn.* 161-168
[7] Rasmussen C E and Williams C K I 2006 *Gaussian Processes for Machine Learning* vol 1, ed T Dietterich, D Heckerman, et al (Massachusetts: The Massachusetts Institute of Technology Press)

[8] Tong S and Koller D 2001 Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.* **2** 45-66
[9] Cai D and He X 2012 Manifold adaptive experimental design for text categorization *IEEE Trans. Know. Data Eng.* **4** 707-19
[10] Raschka S 2015 *Python Machine Learning* vol 1, ed A Hussain, R Youe, et al (Birmingham: Packt Publishing Ltd)
[11] Givoni I E and Frey B J 2009 A binary variable model for affinity propagation *Neur. Comp.* **21** 1589-1600
[12] Rippengill S, Worden K, Holford K M and Pullin R 2003 Automatic classification of acoustic emissions data *Strain* **39** 31-41
[13] Manson G, Worden K, Holford K and Pullin R 2001 Visualisation and dimension reduction of acoustic emission data for damage detection *J. Int. Mat. Sys. Struct.* **12** 529-36
[14] Wolpert D H and Macready W G 1997 No free lunch theorems for optimization *IEEE Trans. Evol. Comp.* **1** 67-82