A Study of Supervised Machine Learning Techniques for Structural Health Monitoring

William Nick
North Carolina A&T State U.
Department of Comp. Sci.
Greensboro, NC 27411
wmnick@ncat.edu

Joseph Shelton
North Carolina A&T State U.
Department of Comp. Sci.
Greensboro, NC 27411
jashelt1@ncat.edu

Kassahun Asamene
North Carolina A&T State U.
Department of Mechanical Eng.
Greensboro, NC 27411
glbulloc@ncat.edu

Albert Esterline
North Carolina A&T State U.
Department of Comp. Sci.
Greensboro, NC 27411
esterlin@ncat.edu

Abstract

We report on work that is part of the development of an agent-based structural health monitoring system. The data used are acoustic emission signals, and we classify these signals according to source mechanisms. The agents are proxies for communication- and computation-intensive techniques and respond to the situation at hand by determining an appropriate constellation of techniques. It is critical that the system have a repertoire of classifiers with different characteristics so that a combination appropriate for the situation at hand can generally be found. We use unsupervised learning for identifying the existence and location of damage but supervised learning for identifying the type and severity of damage. This paper reports on results for supervised learning techniques: support vector machines (SVMs), naive Bayes classifiers (NBs), feed-forward neural networks (FNNs), and two kinds of ensemble learning, random forests and AdaBoost. We found the SVMs to be the most precise and the techniques that required the least time to classify data points. We were generally disappointed in the performance of AdaBoost.

Introduction

Structural health monitoring (SHM) provides real-time data and consequently information on the condition of the monitored structure whose integrity may be threatened by such things as corrosion and cracking. This paper reports on research related to SHM that has been carried out as part of the NASA Center for Aviation Safety (CAS) at North Carolina A&T State University. Ultimately, the target structures will be aircraft, but experiments at this stage are carried out on laboratory specimens.

Our architecture involves a multiagent system that directs a workflow system. Agents typically serve as proxies for techniques with intensive communication or computation requirements. Wooldridge defined an agent as an autonomous, problem-solving, computational entity that is capable of effectively processing data and functioning singly or in a community within dynamic and open environments (Wooldridge 2009). The agents in our system negotiate to determine a pattern of techniques for solving the task at hand, and they communicate this pattern to our workflow engine (implemented on one or more high-performance platforms), which actually carries out the tasks on the data streams provided. The multiagent system is thus the brains and the workflow engine the brawn of our SHM system (Foster, Jennings, and Kesselman 2004). Much of the intelligence here is finding the appropriate techniques for the situation at hand. In one situation, we might want a given task done quickly with only rough accuracy, while in another situation accuracy may be paramount and speed of only secondary importance. Regarding the results of machine learning for SHM, we would like an assortment of classifiers to provide a range of possibilities for the diversity of situations that arises in SHM.

The data we use are acoustic signals, and the condition of greatest interest is crack growth. Since signal sources are unobservable, classifying acoustic signals by their source must be based on machine learning. Sensing here is passive: no energy is required to generate or sense the signals (although energy is required to store and communicate the data). Once an event that is sensed via its acoustic emission has been classified, we may address a multitude of issues and provide diagnoses of the problems. Note that there may be more than one valid classification scheme for events detected via their acoustic emissions.

In SHM, data is interpreted by extracting streams of vectors of feature values from the sensor-data streams. Feature vectors are classified as to the events producing sensed signals by classifiers that have been trained with machine-learning techniques. For our experiments, a correlation coefficient is computed between an observed waveform and six reference waveforms that are generated from numerical simulations of acoustic emission events. The vector of all six correlation coefficients characterizes the waveform. Our dataset consists of 60 samples from the work reported by Esterline and his colleagues (Esterline et al. 2010).

Worden and his colleagues (Worden et al. 2007) have formulated seven axioms for SHM that capture general aspects that have emerged in several decades of experience. Of particular interest is their Axiom III, which states that unsupervised learning can be used for identifying the existence and location of damage but identifying the type and severity of damage can only be done with supervised learning. Supervised learning tries to generalize responses based on a training set with the correct responses indicated. Unsupervised learning tries to categorize the inputs based on their similarities.

Following Axiom III, our previous research investigated two unsupervised and three supervised learning techniques for different aspects of the SHM problem. The objective is

to explore these techniques and note their characteristics so that various combinations of them may be used appropriately in various circumstances. The results of all five techniques for acoustic test data are reported in (Nick et al. 2015). The current paper reviews the results for the three previously investigated supervised learning techniques and reports results for two new techniques, which are both varieties of ensemble learning. The previously-investigated supervised learning techniques are support vector machines (SVM), naive Bayes classifiers, and feed-forward neural networks (FNN). For each technique, we tested a version with principal component analysis (PCA) as a frontend to reduce the dimensionality of the data (usually to three principal components), and we tested another version without PCA. Since PCA generally did not result in significant improvement, the new techniques were tested only without PCA.

For our supervised-learning experiments, class labels on data points indicate one of six possible source types: impulses of three different durations applied to the neutral axis (equidistant between the two surfaces) or to the surface of the specimen. These are cleanly defined events ideal for testing our learning techniques. In practice, class labels would include sources that are crack growth and fretting (friction-producing), the former being a threat, the latter generally being innocuous.

The approach followed here can be generalized for exploring the characteristics of machine-learning techniques for monitoring various kinds of structures. One must first determine what signals are appropriate for monitoring the structures, (For example, acoustic signals are appropriate for monitoring metallic structures while signals propagated through optical fiber are appropriate for bridge type structures.) One then determines the sensor and communication infrastructure. Finally, as per this paper, one determines the characteristics of various supervised and unsupervised learning techniques for monitoring the structures in question (given the signals and infrastructure chosen). Admittedly, the repertoire of techniques explored here is far from complete, but we have included the ones most often encountered in structural health monitoring.

The remainder of this paper is organized as follows. The next sections provides a brief overview of SHM, and the following section looks into previous work in machine learning for SHM. The section after that explains the supervised machine learning techniques we use, and the penultimate section presents our results. The last section concludes.

Structural Health Monitoring

In general, damage is defined as change introduced into a system that will adversely affect its current or future performance (Farrar and Worden 2007). For mechanical structures, damage can be defined more narrowly as change to the material and/or geometric properties. SHM provides real-time information on the integrity of the structure. It allows better use of resources than scheduled maintenance, which may take place when there is no need.

In characterizing the state of damage in a system, we can ask whether there is damage, where in the system it is, what kind of damage it is, and how severe it is. Damage prognosis is the estimation of the remaining useful life of a mechanical structure (Farrar and Lieven 2007).

The field of SHM has matured to the point where several fundamental axioms or general principles have emerged. Worden and his colleagues (Worden et al. 2007) suggest seven axioms for SHM. The following are three that are particularly relevant to this paper.

> **Axiom IVa**: Sensors cannot measure damage. Feature extraction through signal processing and statistical classication is necessary to convert sensor data into damage information.
>
> **Axiom IVb**: Without intelligent feature extraction, the more sensitive a measurement is to damage, the more sensitive it is to changing operational and environmental conditions.
>
> **Axiom V**: The length- and time-scales associated with damage initiation and evolution dictate the required properties of the SHM sensing system.

The following, however, is the most relevant.

> **Axiom III**: Identifying the existence and location of damage can be done in an unsupervised learning mode, but identifying the type of damage present and the damage severity can generally only be done in a supervised learning mode.

As we address supervised learning in this paper, we expect our techniques to be able to identify the type of damage and its severity.

Previous Work in Machine Learning for SHM

Bridge-like structures have been the main structures addressed in the literature on machine learning for SHM. We have a quick look at this rather mature area before turning to our subject, which targets aircraft. (Farrar and Worden 2012) is a text that addresses machine learning for SHM in general. It is directed to mechanical engineers and dedicates most of its space to background. Considering original results, Figueiredo and his colleagues performed an experiment on a three-story frame aluminum structure that used a load cell and four accelerometers (Figueiredo et al. 2011). For each test of state conditions, the features were estimated by using a least squares technique applied to time-series from all four accelerometers and stored into feature vectors. They used four machine learning techniques in an unsupervised learning mode: 1) auto-associative neural network (AANN), 2) factor analysis (FA), 3) singular value decomposition (SVD), and 4) Mahalanobis squared distance (MSD). First the features from all undamaged states were taken into account. Then those feature vectors were split into training and testing sets. In this case, a feed-forward neural network was used to build-up the AANN-based algorithm to perform mapping and de-mapping. The network had ten nodes in each of the mapping and de-mapping layers and two nodes in the bottleneck layer. The network was trained using back-propagation. The AANN- and MSD- based algorithms performed better at detecting damage. The SVD- and FA- based algorithms performed better at avoiding false indications of damage.

Tibaduiza and his colleagues (Tibaduiza et al. 2013), in investigating SHM for an aircraft fuselage and a carbon fiber reinforced plastic (CFRP) composite plate, made use of multiway principal component analysis (MPCA), discrete wavelet transform (DWT), squared prediction error (SPE) measures and a self-organizing map (SOM) for the classification and detection of damage. Each PCA was created using 66 percent of the whole data set from the undamaged structure. Signals from the remaining 34 percent of this data set plus 80 percent of the data set of the damaged structure were used in classifying with the SOM. This approach had an area under the ROC curve of 0.9988. A ROC chart is a display of the performance of a binary classifier, with true positive rate vs. false positive rate.

Esterline and his colleagues (Esterline et al. 2010) (also targeting aircraft) ran an experiment with two approaches. Their first approach used as training instances experimental data with eighteen traditional acoustic emission features to train a SVM, while their second approach used six correlation coefficients between basic modes and waveforms from simulation data also to train a SVM. The SVM with the second approach performed as well or better than the SVM using the first approach, suggesting the superiority of a set of correlation coefficients over a substantial set of traditional acoustic emission features for learning to identify the source of acoustic emissions. It is for this reason that the work reported here uses the six correlation coefficients.

Approach

Recall that the supervised learning techniques we previously investigated are FNN, SVM, and nave Bayes classifiers and that the supervised learning techniques we are reporting on for the first time here are ensemble techniques, specifically random-forest learning and AdaBoost. An artificial neural network (ANN) is a computational model based on the structure and functions of a biological neural network (Bishop 2006). In a FNN, or multilayer perceptron, input vectors are put into input nodes and fed forward in the network. The inputs and first-layer weights will determine whether the hidden nodes will fire. The output of the neurons in the hidden layer and the second-layer weights are used to determine which of the output layer neurons fire. The error between the network output and targets is computed using the sum-of-squares difference. This error is fed backward through the network to update the edge weights in a process known as back propagation.

SVMs rely on preprocessing to represent patterns in the data in a high dimension, usually higher than the original feature space, so that classes that are entangled in the original space are separated by hyper-planes at higher dimension. Training a SVM (Duda, Hart, and Stork 2001) involves choosing a (usually nonlinear) function that maps the data to a higher-dimensional space. Choices are generally decided by the users knowledge of the problem domain. SVMs can reduce the need for labeled training instances.

Naïve Bayes' classifiers (NBs) form a supervised learning technique that belongs to a family of classifiers based on Bayes' theorem with a strong assumption about the independence of features (Duda, Hart, and Stork 2001). Assumptions and the underlying probabilistic model allow us to capture any uncertainty about the model. This is generally done in a principled way by determining the probabilities of the outcomes. NBs were introduced to solve diagnostic and predictive problems. Bayesian classification provides practical learning through the use of algorithms, prior knowledge, and observation of the data in combination. A Gaussian NB assumes that the conditional probabilities follow a Gaussian or normal distribution.

Ensemble learning is a supervised machine learning technique that uses multiple hypothesis spaces for predicting a solution to a problem (Dietterich 2000) (Bennett, Demiriz, and Maclin 2002) (Maclin and Opitz 1999). Generally, a solution found in a hypothesis space may be a weak solution, even if the space is constrained to optimal solutions. Ensemble methods combine different solutions to form accurate decisions for a problem. A unique characteristic of ensemble methods is that the ensemble of solutions can all be accurate yet diverse. Diversity, however, will occur only if the problem is unstable. "Unstable" means that minor changes to the training set affect the classifying performances greatly. We investigate two forms of ensemble learning: random forest and AdaBoost.

Choosing a structure for a tree and training decision trees is time consuming for deep trees. Creating the leaves for the trees is relatively less time consuming. One solution to this is to use fixed tree structures and random features. By using a collection of trees, classifiers can be built. The collection of trees and the randomness of features lead to this algorithm being called random forest (Breiman 2001) (Liaw and Wiener 2002). The random forest algorithm works as follows. A number of user specified trees are randomly created, and each tree has the same depth. The training data is then used to fill in the leaves, which forms predictions for the classifier. The many trees are formed as a committee machine of sorts to form a classifier. If features are too irrelevant, then the classifying performance will not be adequate since there will be a small number of features chosen. The number of trees is important for the classifying process. If there are enough trees, the randomness of features chosen will be overridden by the number of relevant features selected. Meanwhile, the effects of the completely random features will be diminished.

The concept of boosting involves using a series of weakly performing classifiers to form some strong performing classifier. Each classifier can be given some weight that has some correlation to its performance. As different classifiers are added, the weights are readjusted. Weights can be minimized or maximized depending on the boosting algorithm. One popular boosting algorithm is the AdaBoost, or adaptive boosting algorithm (Schapire 1999) (Rätsch, Onoda, and Müller 2001). AdaBoost works as follows. Each data point in a classifier is given some weight based on its significance. A series of classifiers is then trained on training data. A classifier's weight is then determined based on the predictions it makes on the training data. The weight can be used to determine some adaptive value, which is the importance of some classifier. The adaptive value changes based on the classifiers that have been checked. The poorer performing

classifiers have lower weights then better performing classifiers.

Results

The learning techniques were run on a machine running a Windows 7 64-bit operating system with a 2.4 GHz quad core processor and 4 GB of RAM. Software from scikit-learn (Pedregosa et al. 2011) was used for SVM, Gaussian NBs, random forests, and AdaBoost. Software from PyBrain (Schaul et al. 2010) was used for the FNN. Both scikit-learn and PyBrain are written in Python. We recorded the time taken by the classifiers produced by each technique to classify the data points in our test set. This involved executing Python code.

To avoid overfitting, we used stratified five-fold cross-validation with our set of 60 data points. In five-fold cross-validation, the data points are divided into five sets (called folds), all as nearly as possible of the same size. The classifier is learned using four of the folds, and the remaining fold is held out for testing. In multiple runs, different folds are held out for testing. In stratified five-fold cross-validation, the folds are stratified, that is, each contains approximately the same proportion of the labels as the complete data set.

For each learning technique, we had 26 groups of cross-validation runs. In each group, we performed stratified five-fold cross-validation five times, each time holding out a different fold. For each cross-validation run, we computed the precision for the test fold. The precision is defined as $tp/(tp + fp)$, where tp is the number of true positives, and fp is the number of false positives. We also recorded the time it took to classify the 12 data points in the test fold. We then computed the average precision and average classification time for all five runs in the group. We found the minimum, maximum, and standard deviation of the average precision and average time to classify 12 points across the 26 groups of runs.

We ran a SVM with four types of kernel function: linear, radial basis (RBF, with $\gamma = 0.03125$), polynomial and sigmoid (again with $\gamma = 0.03125$). Table 1 displays the mean (over 26 groups of five runs each) precision with which our SVMs classified the 12 data points in our test folds. Note that, for each kernel function, the mean precision and the standard deviation turned out the same for each of the 26 groups of runs.

Kernel	RBF	Polynomial	Linear	Sigmoid
Mean	0.83	0.70	0.88	0.87
St. Dev.	0.07	0.11	0.11	0.04

Table 1: Precision of the SVMs (12 points, 26 groups of 5 runs)

A Gaussian NB classifier and an FNN were trained and tested again with 26 groups of five runs each of five-fold cross-validation. Table 2 shows the resulting ranges of mean precision values and standard deviations of the precision values for the 12 data points in the test fold.

Random-forest and AdaBoost classifiers were also trained and tested with 26 groups of five runs each of five-fold cross-validation. Both of these techniques were implemented with 6, 50, and 75 constituent classifiers. For AdaBoost, the mean precision for each of the 26 runs for all numbers of constituent classifiers was 0.57 and the standard deviation for each run was 0.10. Table 3 shows, for the random forest classifiers, the range in the mean precision values of each group of five runs and the range in the standard deviations of the precision values for these runs.

Technique	Gaussian NB	FNN
Mean	0.78 - 0.78	0.58 - 0.74
St. Dev.	0.10 - 0.10	0.00 - 0.009

Table 2: Precision of the Gaussian NB and FNN (12 points, 26 groups of 5 runs)

No. of Est.	6	50	75
Mean	0.73 - 0.87	0.82 - 0.88	0.82 - 0.92
St. dev.	0.05 - 0.19	0.06 - 0.17	0.04 - 0.17

Table 3: Precision of the Random Forests with various numbers of estimators (12 points, 26 groups of 5 runs)

Regarding precision, the best techniques were SVM with linear (88%) and sigmoid (87%) kernel functions. The random forest with 75 estimators had average precision values in the range 82-92% and ranks up with these two SVM classifiers. The random forest with 50 estimators is close behind (82-88%). Next comes the SVM with an RBF kernel function (83%), followed by the random forest with six estimators (73-87%), and then the Gaussian NB (78%). The FNN performed poorly (58-74%), and the AdaBoosts with any number of estimators were the worst performing techniques (57%).

Turning to the time it took the classifiers trained with various techniques to classify the 12 data points in the test fold, Tables 4 and 5 shows the range of the 26 five-run means of this time (in milliseconds) for each of the kernel functions of our SVM. It also shows the standard deviations for these times. All techniques classified the 12 points in 0.08 to 0.12 msec. Table 6 shows the range of the means and standard deviations for this time in milliseconds for Gaussian NB and FNN to classify the 12 data points.

Kernel	RBF	Polynomial
Mean	0.09 - 0.11	0.09 - 0.10
St.dev.	0.0004 - 0.02	0.0002 - 0.02

Table 4: Time (msec.) for SVMs to classify the 12 data points in the test fold (26 groups of 5 runs)

Finally, Table 7 shows these times (in msec.) for AdaBoost with 6, 50, and 75 estimators, respectively, to classify 12 data points, and Table 8 shows the same for random forest.

The SVM classifiers with all the kernel functions investigated, at around 0.10 msec. to classify 12 data points, were significantly faster than the next fastest technique, which

Kernel	Linear	Sigmoid
Mean	0.08 - 0.11	0.10 - 0.12
St.dev.	0.001 - 0.04	0.001 - 0.02

Table 5: Time (msec.) for SVMs to classify the 12 data points in the test fold (26 groups of 5 runs)

Technique	Gaussian NB	FNN
Mean	0.22 - 0.27	1.60 - 1.91
st.dev.	0.0003 - 0.07	0.001 - 0.64

Table 6: Time (msec.) for Gaussian NB and FNN to classify the 12 data points in the test fold (26 groups of 5 runs)

was Gaussian NB, in the range 0.22-0.27 msec. Random forest with 6 estimators (0.32-0.36 msec.) was close behind, followed at a significant interval by Adaboost with 6 estimators (0.49-0.59). The remaining classifiers took well over one msec. FNN (1.60-1.91) was close to random forest with 50 estimators (1.58-1.75 msec.). AdaBoost with 50 estimators (3.70-3.99 msec.) was slower than random forest with 75 estimators (2.29-2.45 msec.), and AdaBoost with 75 estimators (5.59-5.88 msec.) was significantly slower still.

SVM with an linear or sigmoid kernel function was the most precise technique (87 or 88%) and the technique that classified data points fastest (taking about 0.1 msec. to classify 12 data points). Random forest had an increase in precision of only 1 to 12% going from 6 to 50 estimators, but the time required to classify 12 data points went from 0.32-0.36 msec. to 1.58-1.75 msec. Increasing the number of estimators from 50 to 75 (50%) increased the precision modestly (from 0.82-0.88 msec. to 0.82-0.92 msec.), but enough to rival the SVMs, while increasing the time to classify 12 data points by 40-45%. So the random forest technique proved reasonably precise if somewhat on the slow side. AdaBoost was a complete disappointment as its precision (56%) was worse than any other technique, the second worst being FNN (58-74%). With six estimators, Adaboost was about three times faster than FNN, but with just 50 estimators (3.70-3.99 msec.) it is significantly slower than FNN (1.60-1.91 msec.).

Conclusion

We report here on work that is part of our development of an agent-based structural health monitoring (SHM) system. The data used are acoustic signals, and one attempts to classify these signals according to source. The agents are for the most part proxies for communication- and computation-intensive techniques. They negotiate to determine a pattern of techniques for understanding the situation at hand. Such a pattern determines a workflow. The agents respond in an intelligent way by determining a constellation of techniques appropriate for the situation at hand. It is critical that the system have a repertoire of classifiers with different characteristics so that a combination appropriate for the situation at hand can generally be found.

Following Worden and his colleagues (Worden et al. 2007), we use unsupervised learning for identifying the existence and location of damage but supervised learning for identifying the type and severity of damage. Our objective at this stage is to explore various machine-learning techniques and note their characteristics so that various combinations of them may be used appropriately in various circumstances. This paper in particular reports on experiments with supervised learning techniques using data typical of our domain. The supervised learning techniques investigated are support vector machines (SVMs), naive Bayes classifiers (NBs), and feed-forward neural networks (FNNs) as well as those newly reported with this paper, the ensemble techniques random forests and AdaBoost. SVMs were used with four kernel functions: linear, radial basis (RBF, with $\gamma = 0.03125$), polynomial, and sigmoid (also with $\gamma = 0.03125$). Random forest and AdaBoost both were implemented with 6, 50, and 75 estimators.

No. of Est.	6	50	75
Mean	0.49 - 0.59	3.70 - 3.99	5.59 - 5.88
St. dev.	0.001 - 0.18	0.04 - 0.18	0.04 - 0.57

Table 7: Time (msec.) for AdaBoost with various numbers of estimators to classify the 12 data points in the test fold (26 groups of 5 runs)

No. of Est.	6	50	75
Mean	0.32 - 0.36	1.58 - 1.75	2.29 - 2.45
st. dev.	0.0004 - 0.06	0.02 - 0.28	0.03 - 0.15

Table 8: Time (msec.) for random forest with various numbers of estimators to classify the 12 data points in the test fold (26 groups of 5 runs)

As before, SVM with a linear or sigmoid kernel function was the most precise technique and the technique that classified data points fastest. The random forest technique proved reasonably precise but somewhat slow. Increasing the number of estimators made no difference in the precision of AdaBoost and only a modest improvement for random forest, but the time required to classify data points appeared to be nearly linear in the number of estimators. AdaBoost was a complete disappointment as it produced the worst precision of any of the techniques, and even with just six estimators it took twice as long to classify data points as Gaussian NB.

These results apparently leave no room for intelligent decision by our multiagent system as it appears that a classifier trained as an SVM with either a linear or sigmoid kernel function should be chosen every time. But recall that we consider combinations of classifiers trained in unsupervised and supervised learning mode, the first to find existence and location of damage and then the second to determine the extent and type of damage. For unsupervised learning, we found (Nick et al. 2015) that self-organizing maps (SOMs) appear to give more reliable classifications than k-means classifiers although they take much longer to classify data points. So with unsupervised learning there are tradeoffs and a meaningful choice. In fact, there is still a large number of techniques to investigate, even when restricting ourselves to ensemble techniques. And many techniques can be adapted in subtle ways not considered here. Finally, even among super-

vised learning techniques, some might be better than others in specific circumstances while being inferior in general.

In a practical situation, we look at a large number of events and watch for cases where hundreds are classified as originating from crack growth. So we can tolerate a certain amount of inaccuracy. Cracks, however, grow over months, yet relevant events may be only milliseconds apart, and monitoring a large structure may put a premium on speed. So the extent to which classification time is critical is an involved issue.

Future work will continue investigating supervised and unsupervised learning techniques, looking for combinations of techniques appropriate for various situations. One specific topic will be random forests with boosting. We stated how our approach can be generalized for exploring the characteristics of machine-learning techniques for monitoring various kinds of structures. We intend also to make this generalization explicit.

Acknowledgments

The Authors would like to thank Army Research Office funding for proposal number 60562-RT-REP and NASA Grant # NNX09AV08A for the financial support. Thanks are also due to members of the ISM lab and Dr. M. Sundaresun of the Mechanical Engineering Department at North Carolina A&T State University for their assistance.

References

Bennett, K. P.; Demiriz, A.; and Maclin, R. 2002. Exploiting unlabeled data in ensemble methods. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 289–296. New York,NY: ACM.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*, 1–15. New York, NY: Springer.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern Classification*. Hoboken, NJ: John Wiley & Sons.

Esterline, A.; Krishnamurthy, K.; Sundaresan, M.; Alam, T.; Rajendra, D.; and Wright, W. 2010. Classifying acoustic emission data in structural health monitoring using support vector machines. In *Proceedings of AIAA Infotech@ Aerospace 2010 Conference*.

Farrar, C. R., and Lieven, N. A. 2007. Damage prognosis: The future of structural health monitoring. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 365(1851):623–632.

Farrar, C. R., and Worden, K. 2007. An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 365(1851):303–315.

Farrar, C. R., and Worden, K. 2012. *Structural Health Monitoring: A Machine Learning Perspective*. Hoboken, NJ: John Wiley & Sons.

Figueiredo, E.; Park, G.; Farrar, C. R.; Worden, K.; and Figueiras, J. 2011. Machine learning algorithms for damage detection under operational and environmental variability. *Structural Health Monitoring* 10(6):559–572.

Foster, I.; Jennings, N. R.; and Kesselman, C. 2004. Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 8–15. Piscataway, NJ: IEEE Computer Society.

Liaw, A., and Wiener, M. 2002. Classification and regression by randomforest. *R news* 2(3):18–22.

Maclin, R., and Opitz, D. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*.

Nick, W.; Asamene, K.; Bullock, G.; Esterline, A.; and Sundaresan, M. 2015. A study of machine learning techniques for detecting and classifying structural damage. *Forthcoming*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research* 12:2825–2830.

Rätsch, G.; Onoda, T.; and Müller, K.-R. 2001. Soft margins for adaboost. *Machine learning* 42(3):287–320.

Schapire, R. E. 1999. A brief introduction to boosting. In *Ijcai*, volume 99, 1401–1406.

Schaul, T.; Bayer, J.; Wierstra, D.; Sun, Y.; Felder, M.; Sehnke, F.; Rückstieß, T.; and Schmidhuber, J. 2010. Pybrain. *The Journal of Machine Learning Research* 11:743–746.

Tibaduiza, D.-A.; Torres-Arredondo, M.-A.; Mujica, L.; Rodellar, J.; and Fritzen, C.-P. 2013. A study of two unsupervised data driven statistical methodologies for detecting and classifying damages in structural health monitoring. *Mechanical Systems and Signal Processing* 41(1):467–484.

Wooldridge, M. 2009. *An Introduction to Multiagent Systems*. Hoboken, NJ: John Wiley & Sons.

Worden, K.; Farrar, C. R.; Manson, G.; and Park, G. 2007. The fundamental axioms of structural health monitoring. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* 463(2082):1639–1664.