Journal of Sound and Vibration 437 (2018) 373–388

Contents lists available at ScienceDirect

Journal of Sound and Vibration

journal homepage: www.elsevier.com/locate/jsvi

Active learning for semi-supervised structural health monitoring

L. Bull [*], K. Worden, G. Manson, N. Dervilis

Dynamics Research Group, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK

ARTICLE INFO

Article history:
Received 21 November 2017
Revised 15 August 2018
Accepted 18 August 2018
Available online 5 September 2018
Handling Editor: K. Shin

Keywords:
Structural health monitoring
Vibration monitoring
Active learning
Semi-supervised learning
Classification
Pattern recognition

ABSTRACT

A critical issue for structural health monitoring (SHM) strategies based on pattern recognition models is a lack of diagnostic labels to explain the measured data. In an engineering context, these descriptive labels are costly to obtain, and as a result, conventional supervised learning is not feasible. Active learning tools look to solve this issue by selecting a limited number of the most informative observations to query for labels. This work presents the application of cluster-adaptive active learning to measured data from aircraft experiments. These tests successfully illustrate the advantages of utilising active learning tools for SHM, and they present the first application/adaptation of active learning methods to engineering data — a MATLAB package is available via GitHub: https://github.com/labull/cluster_based_active_learning.

1. Introduction

Structural health monitoring involves the observation of a structure or mechanical system over time using periodically spaced measurements [1]. These data are usually dynamic response, but many alternative measures such as temperature, image or sound data may be used. Damage-sensitive features are extracted from these data, and the analysis of these features can be used to determine the current state of system health [1].

As digital storage gets cheaper, and sensing devices develop, it has become much easier to collect large datasets that may be indicative of system health. Using this resource enables the data-based approach to SHM, which focuses on machine learning and pattern recognition algorithms for black/grey-box modelling as a means of diagnosis and prognosis. While these datasets may be large, comprehensive labelling is rare; furthermore, investigating diagnostic labels in an engineering context is often impractical and expensive, as it is infeasible to damage structures (such as bridges or wind turbines) to obtain labelled data for the damaged states of health. This forces a dependence on partially-supervised machine learning techniques [2], which utilise both labelled and unlabelled data. This work concerns the application of active learning as a form of partially-supervised pattern recognition for SHM.

For demonstration, Dasgupta and Hsu's (DH) cluster-adaptive active learner [3] is applied to experimental engineering data, used to build a damage-locator model for the wing of a Gnat aircraft. Dataset information and implementation details for the algorithm are provided; additionally, a MATLAB package is available through GitHub: https://github.com/labull/cluster_based_active_learning.

[*] Corresponding author.
E-mail address: lbull1@sheffield.ac.uk (L. Bull).

https://doi.org/10.1016/j.jsv.2018.08.040

2. Background

2.1. Data-based SHM & machine learning

The data-based approach to SHM generates a diagnostic model that is not based on physical laws. Instead, pattern recognition methods are applied to the available training data to learn a model for classification to the relevant diagnostic label [1]. This process is typical of *machine learning* — generally defined as a set of methods that can learn and detect patterns in data, and then use these uncovered patterns to predict future data, or perform other kinds of decision making [4].

Supervised learning describes the situation where diagnostic labels, $Y = \{y_i\}_{i=1}^{N} \in \mathcal{Y}$, are available for all input observations, $X = \{\boldsymbol{x}_i\}_{i=1}^{N} \in \mathcal{X}$. Classification techniques can be applied here to learn a mapping h from the observations in the training set X_{train} to their respective output labels Y_{train}, where $h : \mathcal{X} \mapsto \mathcal{Y}$. The end goal is to use the patterns learnt from the training data to predict the class label y^* of a previously unseen observation $\boldsymbol{x}^*$. Note: $\mathcal{X} \in \mathbb{R}^d$, where d is the dimensionality of input observations, and $\mathcal{Y} \in \{1, 2, \ldots, C\}$, where C is the number of classes that represent different states of structural health.

Unsupervised learning methods are used when diagnostic labels are not available for the input observations X. This problem is less well-defined than supervised learning, as the algorithm must learn a relationship from the properties of the input data alone [4]. Clustering techniques are a family of algorithms that work with unlabelled data by finding K groups/clusters of similar observations within the feature space. These are usually defined by calculating the dissimilarity, d, between observations in X, through the use of a distance metric. Outlier analysis and novelty detection are another group of methods that utilise unlabelled data, regularly applied in engineering industry [1]. These techniques look to highlight observations in X that are significantly different; therefore, these data are assumed to be generated by some alternative mechanism [1,5] — such as damage, noise effects or environmental changes.

Semi-supervised learning is a *partially-supervised* learning framework; that is, a family of pattern recognition algorithms lying somewhere between the definitions of unsupervised and supervised learning [2,6]. Partially-supervised methods are required when input observations are available with limited supervision information — a common occurrence with engineering data. Generally, the input data X can be divided into two parts, the points for which the labels are provided X_l, and the points for which labels are unknown X_u [2,6]. More specifically, a *semi-supervised* framework uses the information in the labelled data, while utilising any unlabelled instances to further constrain a classification algorithm [2]. This work focusses on *active learning* as another variation of partially-supervised pattern recognition [2].

2.2. Active learning

Active learning, or query learning, is motivated by scenarios where it is relatively easy to amass large quantities of data but costly/impractical to obtain their labels [7]. The key philosophy is that a pattern recognition algorithm can achieve greater performance, using fewer training labels, if it is allowed to select the data from which it learns [7]. Like supervised learning, the goal is to ultimately learn a mapping from observations to labels; however, here the data are initially unlabelled — more precisely, the algorithm systematically builds an informative training set X_l, limited to a budget of n observations [2].

Consider the data (arriving as a stream or pool) $X = \{\boldsymbol{x}_i\}_{i=1}^{N} \in \mathcal{X}$, each of which has a hidden label that can be queried, $Y = \{y_i\}_{i=1}^{N} \in \mathcal{Y}$. An active learner looks to find a classifier h that provides an accurate mapping of the observations in $\mathcal{X}$ to the labels in $\mathcal{Y}$, while keeping queries to a minimum [8]. In summary, an active learner tries to get the most out of a limited budget, by choosing n query points ($X_l = \{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$) in an intelligent and adaptive manner [8]. The *generalised* steps behind active learning are summarised below and illustrated in Fig. 1.

1. Start with a pool of unlabelled data, these may arrive as a stream, $X \in \mathcal{X}$.
2. By some querying regime, establish which n data carry the most information, $X_l = \{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n} \subset X$.
3. Provide labels for these data, $\{\widehat{y}_i\}_{i=1}^{n} \subset Y \mid \{\widehat{\boldsymbol{x}}_i\}_{i=1}^{n}$.
4. Train a classifier h on this informative subset, $h : \mathcal{X} \mapsto \mathcal{Y} \mid \{\widehat{\boldsymbol{x}}_i\}, \{\widehat{y}_i\}_{i=1}^{n}$.

In the context of SHM, each observation $\widehat{\boldsymbol{x}}_i$ would represent a vector of damage sensitive features. Respective diagnostic labels in Ydescribe the operating condition for all observations. To obtain a label $\widehat{y}_i$, the structure in question will have to be investigated, often at a cost. For example, this might involve the manual assessment of a wind turbine blade, 80 miles from land, at an offshore wind-farm.

2.3. Query frameworks for active learning

The fundamental issue in active learning is determining how to select the most critical instances to be labelled [2,9]. In the machine learning literature [3,8,9], there are two generalised frameworks used to describe various active learning regimes; these approaches are summarised (and compared) below.

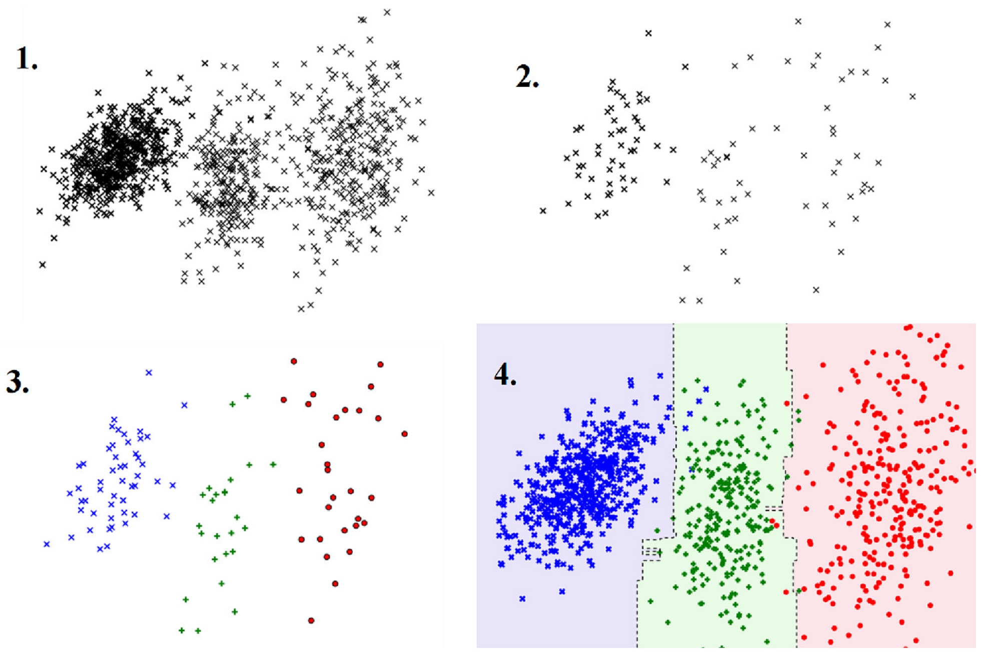


Fig. 1. Visualisation of active learning steps.

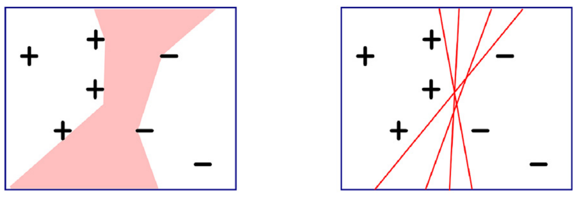


Fig. 2. Left: version space for a binary linear classifier (shaded). Right: some of the plausible hypotheses/classifiers (h) in the current version space. Image credit: [8].

2.3.1. Classification-based

Several query regimes have been based on supervised classification algorithms [9,10]. Typical examples include query by committee and uncertainty sampling [7,11]. Query by committee (QBC) approaches build an ensemble/committee of classifiers using a small initial sample of labelled data, leading to multiple predictions for unlabelled instances. Observations with the most *conflicted* label predictions are viewed as informative, thus, they are queried [9]. Alternatively, uncertainty sampling methods build a single classifier, and observations with the *least confident* predicted label are generally deemed the most informative [10].

Uncertainty sampling approaches can be conceptualised as a search through hypothesis space [8,10]. The hypothesis space $\mathcal{H}$ is used to describe all the possible boundaries that a classifier can take. The version space is a subset of these hypotheses ($\mathcal{H}_t \subset \mathcal{H}$, where $\mathcal{H}_t = \{h\}$) consistent with the labelled data seen so far [4], shown in Fig. 2. As more labels are observed by the learner, the set of plausible hypotheses will shrink, restricting the current version space $\mathcal{H}_t$ [4]. Using active learning, observations who's labels explicitly shrink the version space as fast as possible can be selected [3,10] — in other words, data that lie in/near the shaded region of Fig. 2.

2.3.2. Cluster-based

The second active learning regime exploits cluster structure in data [7,12,13]. A key advantage of cluster-based heuristics is that the framework can naturally utilise the unlabelled data X_u, as well as optimising the selection of the training data X_l [9,13].

Roughly speaking, various cluster-based methods follow a similar framework, introduced by Dasgupta and Hsu [8]. In an ideal scenario, defined, separable clusters will exist that are pure in terms of labels. Following definition by unsupervised learning, a few informative points X_l can be selected from each cluster; any remaining unlabelled points X_u can then be assumed to have their most confident (majority) label [3,9,13] — as in Fig. 3. (Throughout this work, this process is referred to as *label propagation*.) A supervised classifier can then be trained on the labelled dataset X_L, including queried and propagated labels Y_L, such that $X_L = (X_l \cup X_u, Y_L)$.

The active/guided sampling element of cluster-based techniques is defined by the sampling procedure. Various methods have been proposed. Dasgupta and Hsu suggest a heuristic that favours instances from clusters that appear mixed as querying

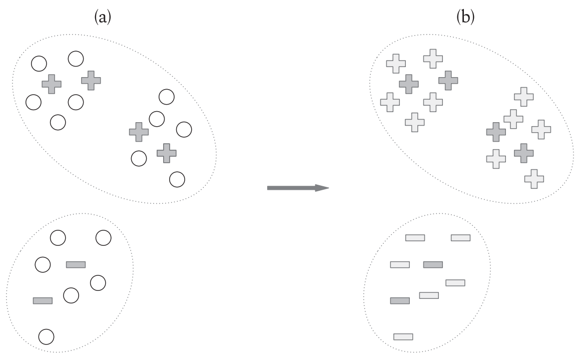


Fig. 3. Ideal clusters (separable and pure): (a) clustering of query points [+/−] and unlabelled instances [∘]; (b) query points and propagated labels (X_L).

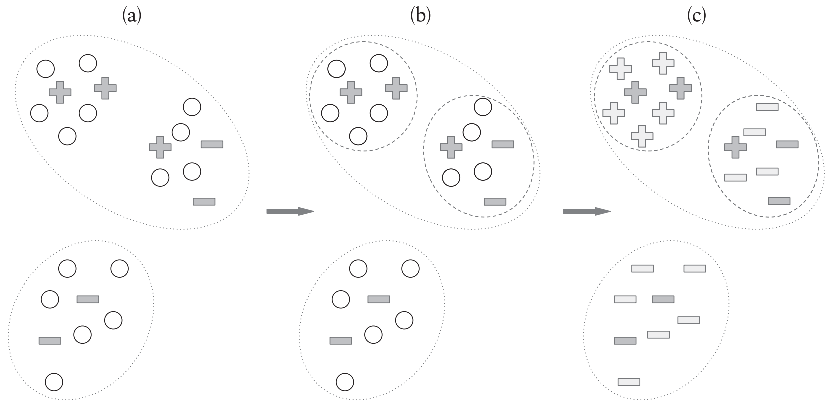


Fig. 4. (a), (b) Identification of viable clusterings at different resolutions; (c) label propagation.

progresses [3]. Alternatively, the density clustering algorithm, by Wang et al. [9], favours queries in regions populated by (relatively) dense groups of data. In this work, queries are directed to areas of the feature space that appear to be mixed in terms of labels, as these clusters are assumed the most informative to both the cluster structure and final classification.

In reality, the ideal case shown in Fig. 3 is extremely rare. The relationship between labels and clusters could be insignificant, or there might be viable (near pure) clusters but at many different resolutions [3] — as in Fig. 4. For this reason, the performance of cluster-based methods heavily depends on the quality of the clustering results [9,14]; thus, data clustering must be adaptive — *actively* changing as more information becomes available. Provided that there is some relationship between clustered groups of data and diagnostic labels, at whatever resolution, cluster-based active learning can exploit these patterns [8,9].

Label propagation steps are typical of *semi-supervised* learning [2], as unlabelled instances X_u are used to constrain the classifier by assuming their labels. In the semi-supervised literature, label prorogation is also referred to as self-labelling or self-training [6,7]. Intuitively, the ability to naturally incorporate unlabelled data brings further benefits to cluster-based active learning, normally associated with semi-supervised algorithms [6].

2.4. Sampling bias

For *classification-based* active learning, exclusively selecting *uncertain* observations can focus too much on specific regions of the feature space (i.e. areas close to the decision boundary). This can neglect alternative regions that might be more representative of the underlying data distribution [10].

To demonstrate this problem, consider the one-dimensional example in Fig. 5, originally presented by Dasgupta and Hsu [3]. In this problem, the data lie in four groups, and the classifier used to separate them is defined by some threshold value, $\omega \in \mathbb{R}$. The proportion of the dataset in each group is given by a percentage. Grey blocks have a (**1**) label, and white blocks have a (**0**) label. Most of the data lie in the two most external groups; therefore, a small, initial random sample has a high likelihood of coming from these. In this case, the initial hypothesis or classifier ($h_\omega \in \mathcal{H}$, Eq. (1)) would lie somewhere between the two

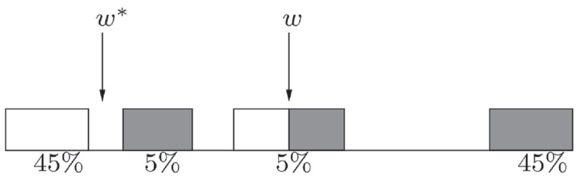


Fig. 5. One-dimensional classification problem to demonstrate sampling bias. Image credit: [3].

external groups shown in Fig. 5.

$$h_\omega(x) = \begin{cases} 0 & x < \omega \\ 1 & x > \omega \end{cases}. \tag{1}$$

As active learning proceeds, selecting uncertain observations, the classifier would most likely converge to ω, in the centre of Fig. 5. However, the classifier ω has 5% error, while ω^* has only 2.5% error [3]. This occurs as the most probable initial sample is poorly representative of the underlying distribution in the data [8]. It includes no observations in the second group from the left (5% grey block), and as a result, this group is overlooked; therefore, the learner is mistakenly confident that these data have a **0** label [3]. To clarify, this group *hides* behind the decision boundary ω due to a poorly placed initial classifier h. This example presents just one-dimension; in higher dimensions the problem can get worse, as there are more spaces for groups of data to hide [3]. This phenomenon is referred to as *sampling bias*.

The issues of sampling bias are a significant challenge for active learning, particularly those based on efficient search through hypothesis space [3,10,15]. If the learning algorithm is incorrectly confident about regions far away from the decision boundary, the results can be worse than standard supervised learning [10]. To counteract this effect, sampling should systematically include *representative* observations (i.e. those far away from the version space) as well as uncertain observations [14]. Several methods have been suggested; typical heuristics, such as the pre-clustering algorithm by Nguyen and Smeulders [13], or the QUIRE algorithm by Huang et al. [14], combine an unsupervised clustering with the classification algorithm, to inform the active learning process. This leads to a hybrid framework, where a balance of uncertain observations (close to the decision boundary) and representative observations (near cluster centroids) are selected.

Active learning frameworks that are purely cluster-based [3,8,13] can automatically mitigate sampling bias by querying across the entire cluster structure, even after a poorly representative initial sample. As discussed, the generalised cluster-based framework [3] completely removes the classifier from the active learning steps; thus, this approach prevents the learner from being constrained by an ill-informed hypothesis. Considering the issues of sampling bias, as well as the benefits associated with label prorogation, this work focusses on the DH algorithm [3] as a cluster-based variation of active learning.

3. The DH learner

Hierarchical sampling for active learning — applied via the DH algorithm — is an active learning tool proposed by Dasgupta and Hsu [3]; this technique utilises a cluster-adaptive framework for guided sampling and label propagation. The heuristic is clearly defined in the original papers [3,8]; however, each stage of the algorithm is explained here — with some slight differences in implementation, indicated in Section 3.3.

3.1. A cluster-based framework for guided sampling

3.1.1. Clustering

The DH learner starts with a hierarchical clustering of the input data. In the experiments here, agglomerative clustering is used; an unsupervised technique that works by sequentially joining groups of data. Initially, it compares N groups, each containing one observation ($K = N$). At each step the dissimilarity matrix d is assessed (Eqs. (2) and (3)), and the two most similar groups are merged, until there is a single cluster containing all the data ($K = 1$) [4].

The dissimilarity between objects is calculated using the Euclidean distance for single data points,

$$d(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{D} (x_{ij} - x_{i'j})^2}, \tag{2}$$

and Ward's average linkage for groups of data,

$$d_{r,s} = \sqrt{\frac{2n_r n_s}{n_r + n_s}} \times d(\overline{x}_r, \overline{x}_s), \tag{3}$$

where n_s and n_r are the number of data in groups r and s respectively, while $\overline{x}_r$ and $\overline{x}_s$ are the cluster centroids. Pseudocode for

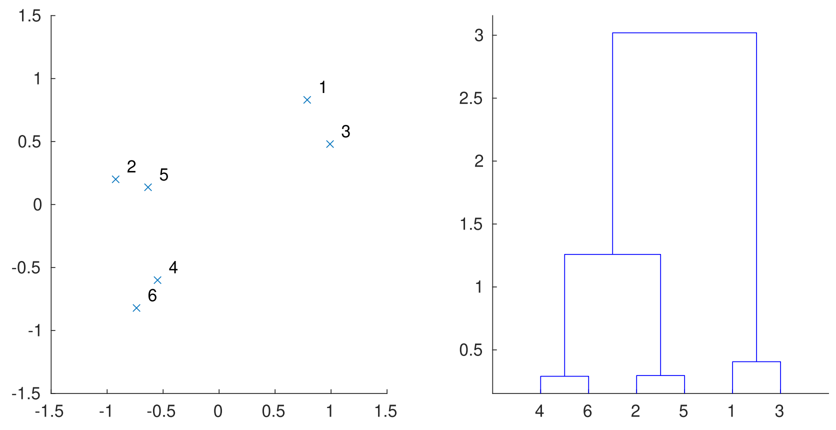


Fig. 6. Dendrogram of hierarchical clustering, down to single observations $N = 6$.

the agglomerative clustering algorithm is provided in Algorithm 1.

Algorithm 1 Agglomerative clustering [4].

```
1: compute dissimilarity matrix $d$ between all observations in $X$
2: initialise clusters as singletons: for $i \leftarrow 1$ to $N$ do $Ci \leftarrow \{i\}$
3: initialise set of clusters available for merging: $S \leftarrow \{1, \dots, N\}$
4: repeat
5:    Pick the two most similar clusters to merge:
      $(j, k) \leftarrow \text{argmin}_{j,k \in S}(d_{j,k})$
6:    Create new cluster $C_l \leftarrow C_j \cup C_k$
7:    Mark $j$ and $k$ as unavailable: $S \leftarrow S\{j, k\}$
8:    if $C_l \neq \{1, \dots, N\}$ then
9:       Mark $l$ as available, $S \leftarrow S \cup \{l\}$
10:   end if
11:   for $i \in S$ do
12:      Update dissimilarity matrix $d(i, l)$
13:   end for
14: until no more clusters are available for merging
```

The merging process can be represented with the use of a binary tree T, called a dendrogram, shown in Fig. 6. The initial groups (singleton observations) are represented by the leaves of the tree, at the bottom of the graph. Each time two groups are merged they are joined in the tree at a node u. The tree T can be defined as a set of nodes, $T = \{u_i\}_{i=1}^{N-1}$, and the height of branches represents the dissimilarity between two respective groups [4]. The root of the tree, at the top of the dendrogram, represents one group containing all the data.

If the tree is cut at any given height, a clustering is induced for a given number of groups K. For example, if the tree in Fig. 6 was cut at height 2.5, this induces a clustering where $K = 2$, with groups: $\{\{4, 6\}, \{2, 5\}\}, \{1, 3\}$.

3.1.2. Cluster-adaptive guided sampling & label propagation

To illustrate guided sampling and label propagation, one can return to Dasgupta and Hsu's one-dimensional example [3]. In Fig. 7, the dendrogram represents the top few nodes of a hierarchical clustering; therefore, each leaf defines a group of data rather than singleton observations. Proportions of the total data in each leaf are provided.

Following hierarchical clustering, the DH algorithm will work with a particular partition of the dataset at any given time, defined by a *pruning* P of the tree T [3]. A pruning of the tree is a subset of nodes that are disjoint and together cover all the data [3]; $P = \{v_1, \dots, v_m\} \subset T$. Initially this is set as the root node from agglomerative clustering, a single group containing all the data, $P = \{1\}$. A small number of random points are drawn from this cluster and queried; these initial labels provide the first indication of the underlying distribution in the data, for all levels of the hierarchy. These samples will usually reveal that the top node is very mixed, while nodes $\{2\}$ and $\{3\}$ are relatively pure [3]. Once this transpires, partition $\{1\}$ will be replaced with a pruning of $P = \{2, 3\}$ [3]. The next set of observations will then be selected according to a querying strategy that favours the less pure node [3].

After further rounds of sampling, the P would most likely be refined to $\{2, 4, 9\}$. At this stage, the benefits of cluster-based sampling become most obvious [3]. Considering the observations seen so far, it can be concluded that cluster $\{9\}$ is relatively pure, so fewer queries will be made from this group [3]. Instead, future samples will be directed towards groups $\{2\}$ and $\{4\}$. Guided sampling continues in this way, working down the dendrogram.

The querying can be stopped at any stage; when this is done, the unlabelled data X_u associated with each cluster in final

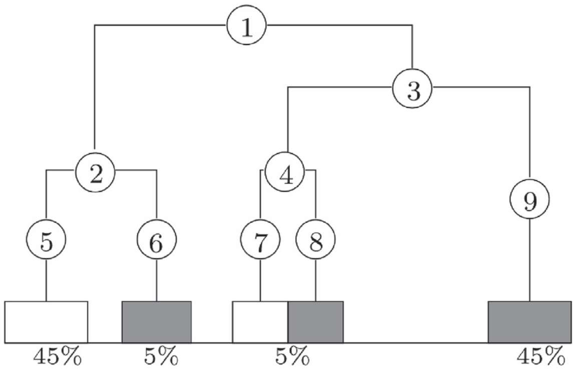


Fig. 7. The top few levels of a hierarchical clustering. Clustered groups are shaded according to their majority label: (1) grey, (0) white. Image credit [3].

pruning P are assigned their majority label, according to the queried data seen so far X_l. (Provided that a set of criterion are met, Section 3.3). In this way, the learner looks to label the entire dataset, such that $X_L = (X_l \cup X_u, Y_L)$, while keeping the number of erroneous propagated labels in Y_L to a minimum [3].

3.2. Pruning & node properties

For any node u in the tree T, T_u denotes the subtree rooted at node u, as well as all the data contained in that node [3]. Therefore a pruning of the tree $P = \{v_1, \ldots, v_m\}$, is such that T_{v_i} are disjoint and together cover all the data [3]. Partial prunings are also considered when working with sub-trees, here the associated leaves do not cover all the data.

The *weight* w_u of a node $u \in T$ is the proportion of total data contained in the subtree of that node, where N_u is the number of data in T_u.

$$w_u = \frac{N_u}{N}. \tag{4}$$

The weight of a pruning $w(P)$ is the fraction of the total data contained in the pruning P [3]:

$$w(P) = \sum_{v \in P} w_v. \tag{5}$$

For a complete pruning, $w(P) = 1$, and for a partial pruning, $0 < w(P) < 1$.

If there are k possible labels, the *proportion* of each label observed in node u is [3]:

$$p_{u,l} = \frac{n_{u,l}}{n_u}. \tag{6}$$

where $n_{u,l}$ is the number of times l has been observed in u, while n_u is the total number of queries taken from node u.

Let the label for node u be $L(u) \in \{1, 2, \ldots, k\}$. The approximate *error* induced when assigning all the data in cluster T_u with the label $L(u)$ is given in Eq. (7) [3]. Intuitively, it makes sense to assign a cluster u with is majority label [3], so $L(u) = l_{\text{argmax}_l(p_{u,l})}$.

$$\epsilon_{u,l} = 1 - \max_l (p_{u,l}). \tag{7}$$

For a partial or complete pruning, the *error* introduced when assigning each cluster with its majority label is defined as [3]:

$$\epsilon(P, L) = \frac{1}{w(P)} \sum_{v \in P} w_v \epsilon_{v,L(v)}. \tag{8}$$

Due to limited sampling, labels are only available in the queried nodes, and these queries are not necessarily indicative of the majority label. At a *given time*, $l(t)$ labels have been observed, and there has been $n_u(t)$ queries; so based on the labels seen so far, the current estimate for the label proportions is $p_{u,l}(t)$ [3]. The corresponding error at this time is given by $\epsilon_{u,l}(t) = 1 - p_{u,l}(t)$ [3].

The *quality* of these estimates can be assessed using generalisation bounds. At any given time the label proportion estimates can be assigned *confidence intervals*, $\{p^{LB}_{u,l}, p^{UB}_{u,l}\}$ [3]. The true value of $p_{u,l}$ is expected to lie within these bounds. The confidence intervals are defined using a variation of *Wald's interval* [3,16]:

$$\{p^{LB}_{u,l},\ p^{UB}_{u,l}\} = \{\max[p_{u,l}(t) - \delta_{u,l}(t),\ 0],\ \min[p_{u,l}(t) + \delta_{u,l}(t),\ 1]\} \tag{9}$$

for,

$$\delta_{u,l}(t) \approx \frac{1}{n_u(t)} + \sqrt{\frac{p_{u,l}(t)(1 - p_{u,l}(t))}{n_u(t)}}. \tag{10}$$

3.3. Algorithm properties

3.3.1. Admissible clusters

When pruning the tree it is useful to work down the dendrogram as far as possible [3]; this way, clusters can be analysed at a higher resolution, so queries can be directed to specific areas of the feature space, and label propagation can be applied to more complex clusterings. However, to justify descending into lower levels of the hierarchy, the learner should first be confident about majority label estimates $L(u)$ for all nodes in the potential pruning.

Considering this, the *admissibility* $A_{u,l}(t)$ is defined to establish when and where the learner can be confident about a majority label estimate [3]:

$$A_{u,l}(t) = \text{True} \;\Leftrightarrow\; (1 - p^{LB}_{u,l}(t)) < \beta \cdot \min_{l \neq l'} (1 - p^{UB}_{u,l'}(t)). \tag{11}$$

In words, for each cluster, a label is admissible if its (largest) expected error is at least β times less than the (smallest) expected error of any other label. For these experiments the hyper-parameter β is set to a value of 1.5, so Eq. (11) becomes:

$$A_{u,l}(t) = \text{True} \;\Leftrightarrow\; p^{LB}_{u,l}(t) > (1.5p^{UB}_{u,l'}(t) - 1) \;\; \forall \;\; l \neq l'. \tag{12}$$

The set of admissible cluster-label (u, l) pairs is defined using $\mathcal{A}(t)$; at any given time there may be several labels associated with each node. The set $\mathcal{A}(t)$ is used throughout sampling to identify any new set of nodes that could make up a refined pruning — with increased homogeneity in each cluster.

3.3.2. Adjusted empirical error

The error estimates $\epsilon_{u,L(u)}(t)$ can be inaccurate when a node has been inadequately sampled, as the learner has weak confidence about the label proportion estimates $p_{u,L(u)}(t)$.

$$\epsilon_{u,L(u)}(t) = 1 - p_{u,L(u)}(t). \tag{13}$$

With this in mind, the admissibility can be used to adjust the empirical error and define a more conservative estimate in areas of sparse sampling [3]:

$$\widetilde{\epsilon}_{u,L(u)}(t) = \begin{cases} 1 - p_{u,L(u)}(t) & \text{if} \;\; (u, L(u)) \in \mathcal{A}(t) \\ 1 & \text{if} \;\; (u, L(u)) \notin \mathcal{A}(t) \end{cases}. \tag{14}$$

In words, label proportion estimates are only valid when their cluster-label pairings are admissible.

The *adjusted* empirical error is now defined as:

$$\widetilde{\epsilon}(P, L, t) = \frac{1}{w(P)} \sum_{v \in P} w_v \widetilde{\epsilon}_{v,L(v)}(t). \tag{15}$$

3.3.3. The select procedure for guided sampling

This describes how the learner *actively* directs sampling in the current working partition (P) of the tree. As suggested by Dasgupta and Hsu [3], the select procedure will favour nodes v that appear most mixed. Once a mixed node is chosen, a random sample is taken from the cluster that it represents, and the label is queried. Formally, the select procedure is defined as,

$$\textbf{Select} \; v \in P \quad \text{with probability} \quad \mathbb{P}(v) \propto w_v(1 - p^{LB}_{v,L(v)}(t)). \tag{16}$$

This definition is used in the experiments; however, the select procedure is flexible and can be modified according to the application [3].

3.3.4. Pruning refinements

When refining the current pruning, $P = \{v_i\}_{i=1}^{m}$, it is convenient to think of the process one node at a time. Therefore, for each node $v \in P$, the best pruning and labelling of the associated subtree T_v is (P_v, L_v). The following rules are used to define (P_v, L_v), where $P_v = \{\widehat{v}_i\}_{i=1}^{m_v}$:

- $L(u)$ is defined for $\widehat{v} \in P_v$ and ancestors of P_v in T_v [3],
- $(u, L(u)) \in \mathcal{A}(t)$ is defined for for $\widehat{v} \in P_v$ and ancestors of P_v in T_v.

For this implementation, while searching through T_v for the best pruning P_v (from the root node down), any new set of nodes must meet the above criteria. Additionally, any two child nodes $ch_u = \{u_{ch1}, u_{ch2}\}$ can only replace their parent node u if a reduction in the adjusted empirical error is observed:

$$\widetilde{\epsilon}(ch_u, L, t) < \widetilde{\epsilon}_{u,L(u)}(t) \quad \text{where} \quad \widetilde{\epsilon}(ch_u, L, t) = \frac{1}{w(u)} \sum_{i=1}^{2} w_{chi} \widetilde{\epsilon}_{chi,L(chi)}(t). \tag{17}$$

3.3.5. *Label propagation*

An additional rule is added to this implementation to prevent inconsistent performance at low query budgets n. It states that label propagation (assumption) to the unlabelled instances X_u only occurs if the number of clusters in the final, admissible pruning is ≥ number of unique labels observed so far:

$$\text{Assign each unlabelled point in } T_v \text{the label } L(v) \;\Leftrightarrow\; |P| \geq |l(t)| \tag{18}$$

This is intuitive; for example, it is illogical to assume labels for three admissible clusters across the whole data, when a total of seven classes have been observed.

3.4. *The algorithm*

The pseudocode in Algorithm 2 formalises this implementation of the DH learner; it follows the same flexible structure presented in the original paper [3].

Algorithm 2 Cluster-adaptive active learning.

```
    Input: Agglomerative clustering $T = \{u_i\}_{i=1}^{N-1}$ of the input observations $X$
 1: $P \leftarrow \{\text{root}\}$                        ▷ Initialise current pruning as the root node
 2: $L \leftarrow \{0\}$                                   ▷ Initialise arbitrary root label
 3: #——— GUIDED SAMPLING ———#
 4: for $t = 1 : T$ do                                     ▷ Algorithm run budget $T$
 5:     for $i = 1 : B$ do                                 ▷ Guided sampling, batch size $B$
 6:         $v \leftarrow \text{select}(P)$                ▷ select $v$ from $P$ according to Equation 16
 7:         randomly sample $\hat{x}$ form $T_v$           ▷ Adding to $X_l$
 8:         query $l$ the label of $\hat{x}$, update $X_L$  ▷ Provided by engineer/oracle
 9:         update $(n_u(t), p_u(t))$                      ▷ For all nodes containing new sample $\hat{x}$
10:     end for
11:     for all nodes $u \in T$ do                         ▷ Compute the admissibility and error for all nodes
12:         update $(\mathcal{A},\ \tilde{\epsilon}_{u,L(u)})$
13:     end for
14:     #—— PRUNING REFINEMENTS ——#
15:     for each $v \in P$ do                              ▷ Refine the current pruning, node by node
16:         let $(P_v, L_v)$ be the best pruning
            and labelling of $T_v$:                        ▷ According to the rules in Section 3.3
17:         $P \leftarrow P_v \cup (P \setminus v)$        ▷ Update node $v$ to refine $P$
18:         $L(v) \leftarrow L_v(\hat{v})$ for all $\hat{v} \in T_v$ ▷ Update $L(v)$ to reflect the refined pruning
19:     end for
20: end for
21: #——— LABEL PROPAGATION ———#
22: for each cluster $v \in P$ do                          ▷ For each cluster in the *final* pruning
23:     if $|P| \geq |l(t)|$ then                          ▷ Additional rule, Equation 18
24:         assign each unlabelled point in                ▷ Propagate labels to $X_u$
            $T_v$ the label $L(v)$, update $X_L$
25:     end if
26: end for
27: return: final pruning and labelling $(P, L)$, labelled data $X_L$
```

3.4.1. Classification

Following the cluster-based active learning process, any supervised classifier can be trained using X_L. The classification algorithm is independent of the active learning heuristic; therefore, it does not affect the *active* elements of the learner. Furthermore, as the 'no free lunch' theorem suggests [17], the performance of any algorithm is entirely data dependant. Thus, the choice of classifier is trivial when focussing on the active learning characteristics of the process (provided the same model is used throughout tests).

In fact, as suggested by Wang et al. [9], a classification algorithm is not necessary for cluster-based active learning. Future data can be classified according the final pruning P of the feature space and a majority vote [9]. Nonetheless, a classification algorithm is applied in the experiments for direct comparison to conventional techniques; some justification is provided.

Initially a Multi-Layer Perception (MLP) was considered, however, when label propagation does not occur, the training data is far too sparse to correctly train an MLP. A Bayesian classifier, such as the Relevance Vector Machine, was also considered to provide probabilistic outputs for the classification; however, these probabilities are less meaningful when labels have been assumed in a non-probabilistic manner. Eventually it was decided to use bagged decision trees for computational efficiency. Combining the Classification and Regression Tree (CART) algorithm with bootstrap aggregating provides a simple yet effective classifier, shown in the literature to give excellent parametric performance, even when compared to more complex methods [18]. Model parameters where kept constant, as proper validation was not possible for small values of n — this provides a simple classification metric, suitable for the purposes of this paper.

4. Experiments

Cluster-adaptive active learning is applied to engineering data from aircraft experiments.

4.1. Methods

DH active learning will be compared to two benchmark methods: random sample training and standard supervised learning. For each experiment, the input data and hidden labels (X, Y) are split into a test set (X_{test}, Y_{test})(33%) and a *potential* training set (X_{pt}, Y_{pt})(66%) using random indices.

1. **Standard supervised learning:** The traditional approach for *passive* learning in engineering applications. All the available training data are used to train the classifier, $(X_{pt}, Y_{pt}) = (X_{train}, Y_{train})$. As a result, this is the most expensive method (in terms of labels); therefore, the achieved accuracy should be considered the target performance.
2. **Random sample training:** Another form of passive learning [10,11], which takes a simple random sample of n data from the potential training set, then queries the labels: $(X_{train}, Y_{train}) \subset (X_{pt}, Y_{pt})$, where $(X_{train},\ Y_{train}) = (\{\hat{\mathbf{x}}_i\}_{i=1}^n, \{\hat{y}_i\}_{i=1}^n)$. The classifier is trained using this subset alone.
3. **DH cluster-adaptive active learning:** X_{pt} is presented as a pool of unlabelled instances. Following Algorithm 2, guided sampling *actively* selects n of the most informative data, according to the *select* procedure; such that $X_l = \{\hat{\mathbf{x}}_i\}_{i=1}^n$ and the queried labels are given by $\{\hat{y}_i\}_{i=1}^n$. When the budget runs out, these labels are propagated to the unlabelled data X_u, throughout the admissible cluster structure, giving $X_L = (X_l \cup X_u, Y_L)$. A classifier is trained using this dataset, where $X_L = (X_{train}, Y_{train})$.

4.2. Procedure

For standard supervised learning (X_{train}, Y_{train}) is resampled from X 100 times, then the classifier is trained/validated 10 times for each sample. The model generalisation is evaluated using the test set (1000 runs in total).

For methods 2 and 3 the same procedure applies while increasing the sample budget n, where $\hat{n} = \{n_i\}_{i=1}^{198} = \{3, 6, 9, \ldots, 594\}$:

```
1: for $n$ in $\hat{n}$ do
2:   for s = 1:100 do
3:     split $(X, Y)$ into $(X_{test}, Y_{test})$ and $(X_{pt}, Y_{pt})$ by random sample
4:     define $(X_{train}, Y_{train})$ according to method, query budget $n$
5:     for r = 1:10 do
6:       train a supervised classifier using $(X_{train}, Y_{train})$
7:       test the classifier using $X_{test}$, record the misclassification error
8:     end for
9:   end for
10: end for
```

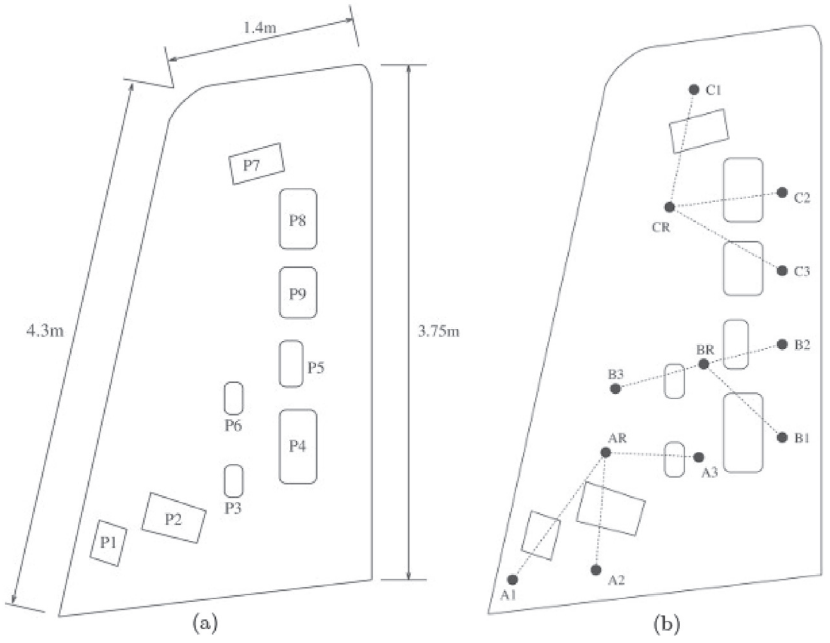


Fig. 8. Gnat aircraft wing schematics; (a) panel locations [23]. (b) transducer groups and transmissibility paths [23].

4.3. Gnat aircraft data

When following Rytter's hierarchy [19], an SHM process accumulates information to inform damage detection, followed by location and classification of that damage. The Gnat data are defined to consider the first two levels of this hierarchy.

A series of four research papers explain how the data were acquired and processed [20–23]. The first two publications establish the experimental procedure and damage detection strategy. These methods utilise vibration data and outlier analysis to flag faults in a test rig [22] and aircraft wing [20]. The third and fourth papers explore damage location methods, using wholly supervised classification techniques [21]. The active learning tests is this work return to the classification problem, with limited supervision information.

4.3.1. Data capture and signal processing

Throughout experiments with the Gnat aircraft, it was undesirable to permanently damage the wing structure, so damage was simulated by sequentially removing one of nine inspection panels. The advantage of this method is that the 'damage' was reversible, allowing for repeatability of measurements [22]. It should be considered that the removal of each panel imitates a fairly large, significant fault.

The nine inspection panels are shown in the schematic, as in Fig. 8a (not to scale). Panels were chosen based on ease of removal, and to cover a range of damage locations/sizes [22]. Each panel is held in place with number of screws, ranging from 8 to 26. These were removed using an electric screwdriver with controllable torque, in an attempt to keep constant boundary conditions [20]. It was estimated that panels 3 and 6 would cause the most problems for any pattern recognition techniques, as they are the smallest and placed relatively close together [21].

Transmissibilities across each of the selected panels were used as the main measurements, for justification see Refs. [20–22]. The panels were split into three groups (A, B & C) and assigned a central *reference* transducer; nine additional *response* transducers were then associated with each specific panel. The transducer layout, with the relevant transmissibilities, is shown in Fig. 8b.

The transmissibility associated with each response transducer is obtained by taking the ratio of the acceleration response spectrum with the reference acceleration spectrum. Spectra were estimated using the fast Fourier transform and an appropriate windowing average [23]. The wing was excited using an electrodynamic shaker with white Gaussian excitation. In all cases, 1024 spectral lines were measured between 1024 and 2048 Hz [23]. The real and imaginary parts of the response functions were recorded, then converted into magnitudes for *feature selection*. In total, 700 one-shot measurements were made for the normal condition and 198 for each damage condition [23].

4.3.2. Feature extraction and novelty detection

In Ref. [20], damage sensitive features were established by selecting regions from each transmissibility that were observed to be unambiguously different from the normal condition when damage was simulated. 44 features were found [20], each

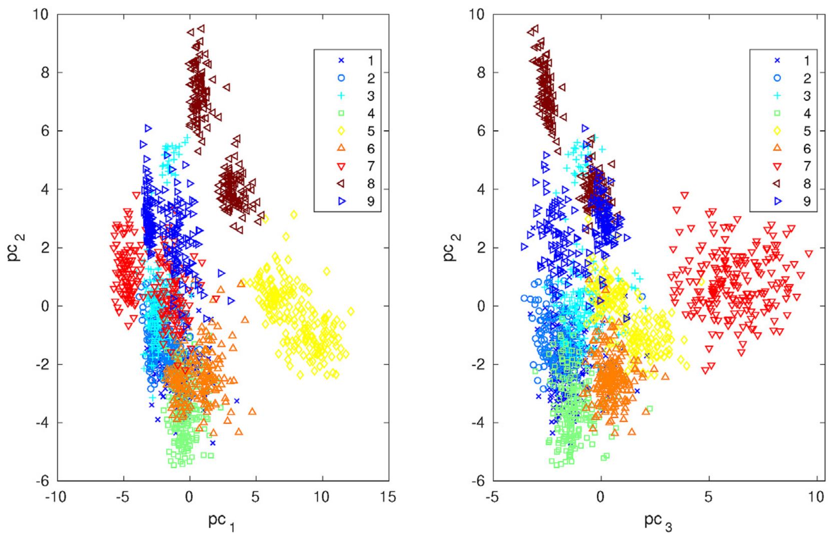


Fig. 9. Gnat data for the three principal components (pc_1, pc_2, pc_3).

characterised by a vector of spectral lines. A large amount of 'engineering judgement' was used in these initial feature extraction steps, and the (damage location) labels were used informally to aid the process. Ideally, the selection of transmissibility regions should follow a pragmatic framework — e.g. included in the dimension reduction regime, summarised in the next Section.

For each damage sensitive feature, a novelty detector was constructed using multivariate outlier analysis and Mahalanobis-squared distance [22]. Each detector provides a scalar novelty index $x^{(d)}$ which is assessed against a threshold, calculated using the normal condition and the theory outlined in Ref. [1]. If this threshold is exceeded, damage is inferred. The outcome of these data-compression steps leads to the processed input data $X = \{\boldsymbol{x}_i\}_{i=1}^{1782}$ (198 observations for each panel). Each observation is characterised by 44 damage sensitive features, defined by novelty indices: $\boldsymbol{x}_i = \{x_i^{(d)}\}_{d=1}^{44}$. The log of each novelty index is used to expose information at low levels of the scalar output [23]; this high-dimensional dataset is summarised in Eq. (19).

$$(X, Y) = (\{\boldsymbol{x}_i\}_{i=1}^{1782}, \ \{y_i\}_{i=1}^{1782}) \quad \text{where} \quad \boldsymbol{x}_i \in \mathbb{R}^{44}, \ \ y \in \{1, 2, 3, \ldots, 9\}. \tag{19}$$

4.3.3. Genetic dimension reduction

The work in the final paper [23] outlines a pragmatic approach for dimension reduction, applied to the extracted features of X, where $\boldsymbol{x}_i \in \mathbb{R}^{44}$. A Genetic Algorithm (GA) was used to determine the optimal subset of damage sensitive features, used as the inputs for a damage location model. Briefly, the GA implementation iterates though a *population* of different features sets — represented by a vector of integers (ranging from 1 to 44) [23]. The *fitness* of each set is assessed using a simple multi-layer-perception and a validation set — the inverse validation error is used [23]. The *fittest* sets are passed on to the next generation by combining their solutions. Mutation is also included by the occasional random switch of a feature [23]. In these experiments, the optimal feature set containing 9 novelty indices is used — providing the compressed input data, $X = \{\boldsymbol{x}_i\}_{i=1}^{1782}$, now with nine dimensions, $\boldsymbol{x}_i \in \mathbb{R}^9$.

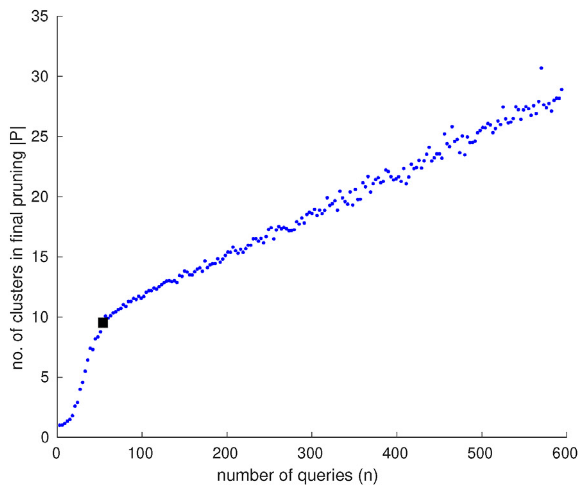
It is worth mentioning that training/validation sets must be used to assess the fitness when using a genetic algorithm for dimension reduction, and the availability of these sets can negate the need for active learning. However, if these data groups are small, they could be used as the initial sample for the DH learner. The investigation of further data could then be dictated by active learning. Alternatively, effective, wholly unsupervised methods for feature extraction (with high-dimensional engineering data) would be ideal for active learning applications; the investigation of such techniques is suggested as further work.

4.3.4. The classification problem

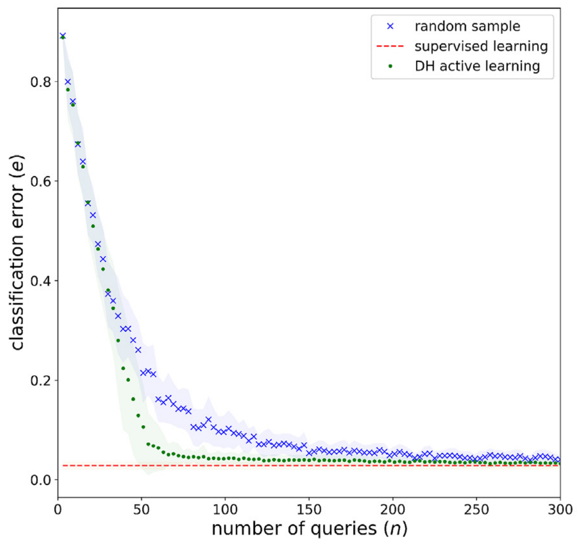
To summarise, the damage location model is trained using observations following feature extraction and dimension reduction; the lower-dimensional dataset (X, Y) is summarised in Eq. (20). Observations in X are described by 9 novelty indices (features), and the hidden labels Ydefine which damage state has occurred (the removal of panel 1–9):

$$(X, Y) = (\{\boldsymbol{x}_i\}_{i=1}^{1782}, \ \{y_i\}_{i=1}^{1782}) \quad \text{where} \quad \boldsymbol{x}_i \in \mathbb{R}^{9}, \ \ y \in \{1, 2, 3, \ldots, 9\}. \tag{20}$$

Therefore the Gnat data present a 9-class classification problem, with 9-dimensional input data. The dataset was designed to be wholly supervised; however, in these tests the labels are hidden, to demonstrate active learning. The data are projected through a linear transformation onto three dimensions for visualisation, as shown in Fig. 9. The new co-ordinates, or principal component scores (pc_1, pc_2, pc_3), are a linear combination of the information retained within the original features and account



(a)



(b)

Fig. 10. (a) Average number of clusters in the final pruning $|P|$ for an increasing query budget n; marker ■ indicates the point at which label propagation becomes admissible, $(n, |P|) = (54, 9.52)$. (b) Misclassification error e for an increasing query budget n. Plots are provided for the DH learner and both benchmark methods.

for the maximum variance within the data [4]. It is worth noting the tests are *not* applied to this projection of the data; however, Fig. 9 is still used to reference the separability of the data in X, as principal component analysis highlights this variance.

4.4. Results & discussion

The first admissible pruning and labelling of T (leading to label propagation) was generally found after 54 queries. According to the rules set out in Section 3.3, this occurs when the number of clusters in the refined pruning P is greater than or equal to the number of labels seen so far. Intuitively, this will usually occur when $|P| \geq 9$ — this threshold is shown by the highlighted datum in Fig. 10a. Interestingly, after this point, the number of clusters in the final pruning grows almost linearly with n; suggesting the additional rule (Eq. (18)) works well to define when label propagation is suitable/stable.

The classification error e is plotted against an increasing query budget n — shown in Fig. 10b. Each curve has a shaded region representing one standard deviation about the mean. Results show that using the DH learner provides a significant increase in classification performance, particularly for lower query budgets. As to be expected, there is a notable increase in the classification performance as label propagation becomes admissible, $n \gtrapprox 54$. At this stage, just 3.0% of the hidden labels are used, and the average associated error on the test set is 7.2%. This is compared to the supervised learning error, 2.8%, which requires *all* the hidden labels. In other words, at $n = 54$, the DH active learner achieves 95.5% of the performance of the supervised learning

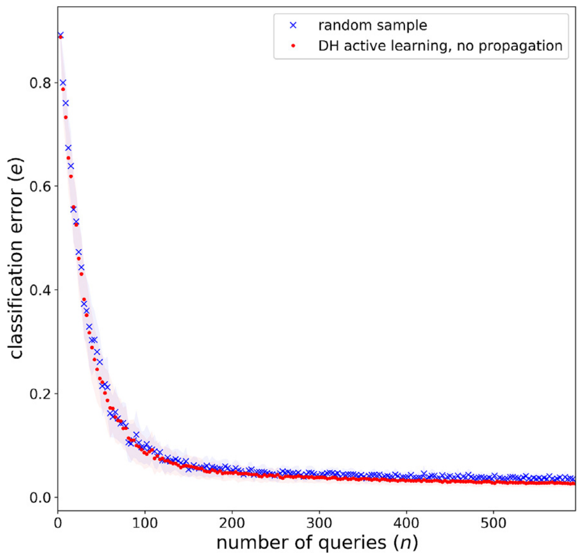
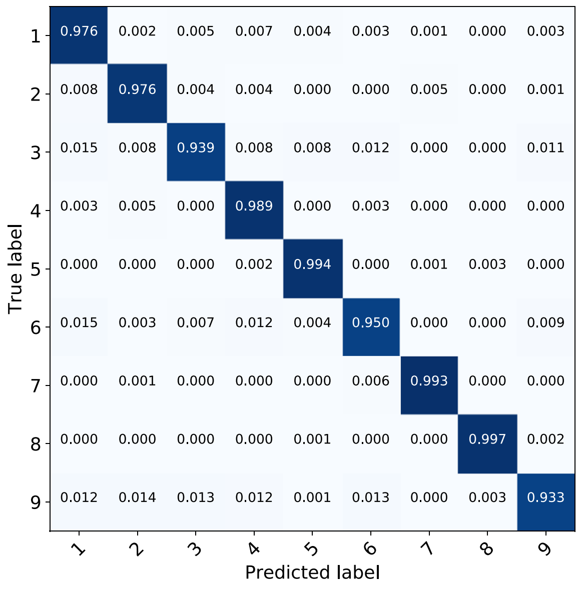


Fig. 11. Misclassification error e for an increasing query budget n. Plots are provided for classifiers trained using guided sampling (the DH learner *without* label propagation) vs. random sample training.

benchmark, while using just 3.0% of the labels; this is a significant achievement for engineering applications. At the same query budget, random sample training reaches 80.5% of the performance of supervised learning. This reduced performance (15.1%) further demonstrates the advantages brought about by cluster-adaptive active learning. Following 102 queries, the DH learner achieves 98.5% of the wholly supervised benchmark performance, while using only 5.7% of the hidden labels. Here random sample training achieves 92.4% of supervised learning performance, for the same label budget n.

In order to highlight any advantages from *guided sampling* alone, the classification error (without label propagation) is compared to random sample training in Fig. 11. Ideally, a classifier trained using a subset selected via guided sampling would outperform one trained by a plain random sample. However, Fig. 11 fails to illustrate a significant advantage, particularly for lower values of n. As a result, it is safe to deduce that improvements provided by the DH learner, in these specific experiments, are a result of cluster-adaptive label propagation. In order to increase the influence of guided sampling, the select procedure (Equation (16)) could be adapted for applications to engineering data. However, it is acknowledged in the original paper [3] that guided sampling will only provide a significant benefit when the hierarchical clustering has some large, fairly pure clusters near the top of the tree. (These will quickly be identified, and very few queries will subsequently be made in those regions [3].) It is clear



	1	2	3	4	5	6	7	8	9
1	0.976	0.002	0.005	0.007	0.004	0.003	0.001	0.000	0.003
2	0.008	0.976	0.004	0.004	0.000	0.000	0.005	0.000	0.001
3	0.015	0.008	0.939	0.008	0.008	0.012	0.000	0.000	0.011
4	0.003	0.005	0.000	0.989	0.000	0.003	0.000	0.000	0.000
5	0.000	0.000	0.000	0.002	0.994	0.000	0.001	0.003	0.000
6	0.015	0.003	0.007	0.012	0.004	0.950	0.000	0.000	0.009
7	0.000	0.001	0.000	0.000	0.000	0.006	0.993	0.000	0.000
8	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.997	0.002
9	0.012	0.014	0.013	0.012	0.001	0.013	0.000	0.003	0.933

Fig. 12. Confusion matrix: averaged classification accuracy for supervised learning.

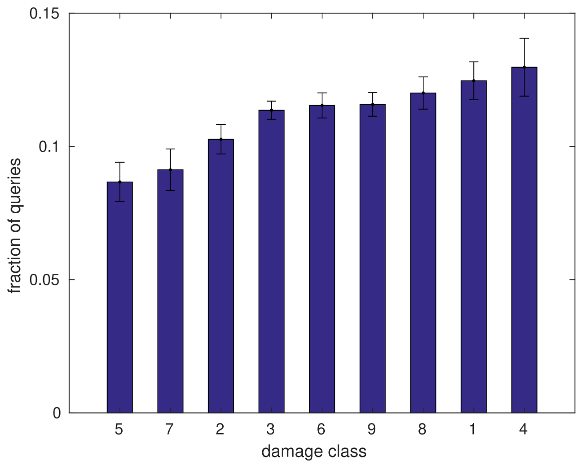


Fig. 13. Average faction of (n) queries per class.

from Fig. 9 these data do not present the ideal case; although, some relatively pure, separate groups are still shown in the data projections (classes 5 and 7).

To investigate this further, the averaged confusion matrix for *supervised learning* experiments is provided in Fig. 12. This is shown in an attempt to highlight classes that are mixed, as these are assumed the most confused. With successful guided sampling, querying should be higher in the confused, mixed groups, while reduced in homogeneous, separable groups. Specifically, classes 9, 6, 3 should receive a high number of queries, while classes 8, 7, 5, 4 are queried less.

Averaged sample counts across each class are provided in Fig. 13. There is not a great deal of specificity for guided sampling, however, the *select* procedure does successfully direct queries to some extent. In particular, classes 5 and 7 are sampled significantly less than other groups; this makes sense, as they are among the least confused in Fig. 12; furthermore, they define clear, separable clusters in Fig. 9. Class 2 also has a low query fraction, which is justified considering its ranking in the confusion matrix.

For the remaining classes, guided sampling is more ambiguous. This is understandable, considering how mixed these classes are — see Fig. 9. Class 8, however, is observed to be relatively separate in the data projections, and it is the least confused; despite this, it is frequently queried by the learner. This can be explained as the clustering results are poorly representative of the underlying distribution of the data in class 8, for high levels of the hierarchy. As a result, guided sampling is less influential for this class. The same principle leads to higher queries in classes 1 and 4 than might seem necessary, although this is less surprising, as these clusters are visibly mixed in the data projections. To combat this, the initial clustering could be defined in an alternative manner. Experiments with alternative linkage functions and distance metrics (other than Ward's average linkage and Euclidean distance) might pose a solution; however, the issue is very application specific. In the best case scenario, the input data will define more separable, pure clusters.

5. Conclusions

To demonstrate the relevance of active learning for data-based engineering, a cluster-based heuristic has been applied to data from aircraft experiments. For the majority of engineering data, comprehensive labelling is rare; additionally, it is impractical/expensive to investigate the associated labels. Active learning addresses this issue, as the learning algorithm looks to systematically build an accurate mapping of (measured) observations to (descriptive) labels, while keeping the number of queries to a minimum.

Dasgupta's and Hsu's (DH) cluster-adaptive heuristic is applied [3]. This starts with a hierarchical clustering of the unlabelled data, which divides the feature-space into many partitions. An informative training set is built by directing queries to areas of the feature-space that appear mixed in terms of labels, while clusters that appear homogeneous are queried less. When appropriate, queried labels can be propagated to any remaining unlabelled instances, using the cluster structure and a majority vote — a process typically associated with semi-supervised learning. A standard supervised classifier can then be learnt from the resulting labelled dataset.

Experiments successfully demonstrate that cluster-adaptive active learning has the potential to significantly reduce labelling costs by utilising both labelled and unlabelled data in an active framework. The DH heuristic provides a significant increase in performance over passive training with a random sample of the same budget n; furthermore, the classification performance is highly competitive when compared to the supervised learning benchmark — which requires all the data to be labelled. Notably, following label propagation ($n \gtrapprox 54$), the DH active learner achieves 95.5% of supervised learning performance, while using just 3.0% of the labels.

In the experiments here, active learning is successful as a result cluster-adaptive label propagation — a process enabled by the hierarchical framework of the heuristic. Although guided sampling is directing queries to some extent, this procedure alone is not influential enough to directly affect the classification performance, particularly for low query budgets. Alternative select procedures might increase the influence of guided sampling, although in real terms, the success of this mechanism is very data specific. If relatively pure, separable clusters existed in high levels of the hierarchy, guided sampling should be more influential. This fact does not discredit the algorithm as an *active learner* for these data; the working partition is adapted according to the labels seen so far, and querying is successfully directed towards mixed groups, facilitating the discovery of additional admissible clusters. This allows the pruning to be further refined, which in turn enables successful label propagation.

The algorithm is well suited to engineering applications. It utilises unlabelled data, and importantly, the damage classes do not need to be defined a priori. As a result, new labels can be included as they are discovered. The heuristic is limited in some respects, as a large pool of unlabelled data has to be initially available to create an informative hierarchical clustering. To address this, future work must consider modifications to accept a stream of online data, with the underlying cluster structure updated on the fly. Alternatively, a hypothesis space active learner could be used as the final classifier — this can then accept a stream of future data online, querying when appropriate. Finally, it would be interesting to use probabilistic clustering to give certainty bounds on the propagated labels; this would also allow for the select procedure and label propagation framework to be controlled in a probabilistic manner.

A MATLAB package for this implementation of Dasgupta and Hsu's active learner is available, including a demo, through GitHub: https://github.com/labull/cluster_based_active_learning.

Declarations of interest

None.

Acknowledgements

The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) through Grant reference number EP/J016942/1.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.jsv.2018.08.040.

References

[1] Charles R. Farrar, Worden Keith, Structural Health Monitoring: a Machine Learning Perspective, John Wiley & Sons, 2012.

[2] Friedhelm Schwenker, Edmondo Trentin, Pattern classification and clustering: a review of partially supervised learning approaches, Pattern Recogn. Lett. 37 (2014) 4–14.

[3] Sanjoy Dasgupta, Daniel Hsu, Hierarchical sampling for active learning, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 208–215.

[4] Kevin P. Murphy, Machine Learning: a Probabilistic Perspective, MIT Press, 2012.

[5] Peter J. Rousseeuw, Annick M. Leroy, Robust Regression and Outlier Detection, vol. 589, John wiley & sons, 2005.

[6] Chapelle Olivier, Bernhard Scholkopf, Zien Alexander, Semi-supervised Learning, MIT Press, 2006.

[7] Settles Burr, Active Learning Literature Survey, vol. 52, University of Wisconsin, Madison, 2010, p. 11. 55-66.

[8] Sanjoy Dasgupta, Two faces of active learning, Theor. Comput. Sci. 412 (19) (2011) 1767–1781.

[9] Min Wang, Min Fan, Zhi-Heng Zhang, Yan-Xue Wu, Active learning through density clustering, Expert Syst. Appl. 85 (2017) 305–317.

[10] Jan Kremer, Kim Steenstrup Pedersen, Christian Igel, Active learning with support vector machines, Wiley Interdiscipl. Rev. Data Min. Knowl. Discov. 4 (4) (2014) 313–326.

[11] Maria E. Ramirez-Loaiza, Manali Sharma, Geet Kumar, Mustafa Bilgic, Active learning: an empirical study of common baselines, Data Min. Knowl. Discov. 31 (2) (2017) 287–313.

[12] Thomas Hofmann, Joachim M. Buhmann, Active data clustering, in: Advances in Neural Information Processing Systems, 1998, pp. 528–534.

[13] Hieu T. Nguyen, Smeulders Arnold, Active learning using pre-clustering, in: Proceedings of the Twenty-first International Conference on Machine Learning, ACM, 2004, p. 79.

[14] Sheng-Jun Huang, Rong Jin, Zhi-Hua Zhou, Active learning by querying informative and representative examples, in: Advances in Neural Information Processing Systems, 2010, pp. 892–900.

[15] Hinrich Schütze, Emre Velipasaoglu, Jan O. Pedersen, Performance thresholding in practical text classification, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM, 2006, pp. 662–671.

[16] Anirban DasGupta, Probability for Statistics and Machine Learning: Fundamentals and Advanced Topics, Springer Science & Business Media, 2011.

[17] David H. Wolpert, William G. Macready, et al., No Free Lunch Theorems for Search, Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[18] Rich Caruana, Alexandru Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference of Machine Learning, 2006.

[19] Anders Rytter, Vibrational Based Inspection of Civil Engineering Structures, PhD thesis, Dept. of Building Technology and Structural Engineering, Aalborg University, 1993.

[20] Graeme Manson, Worden Keith, David Allman, Experimental validation of a structural health monitoring methodology: Part ii. novelty detection on a gnat aircraft, J. Sound Vib. 259 (2) (2003) 345–363.

[21] Graeme Manson, Worden Keith, David Allman, Experimental validation of a structural health monitoring methodology: Part iii. damage location on an aircraft wing, J. Sound Vib. 259 (2) (2003) 365–385.

[22] Worden Keith, Graeme Manson, David Allman, Experimental validation of a structural health monitoring methodology: Part i. novelty detection on a laboratory structure, J. Sound Vib. 259 (2) (2003) 323–343.

[23] Worden Keith, Graeme Manson, G. Hilson, S.G. Pierce, Genetic optimisation of a neural damage locator, J. Sound Vib. 309 (3) (2008) 529–544.