Annals of Telecommunications
https://doi.org/10.1007/s12243-020-00775-2

Active feature acquisition on data streams under feature drift

Christian Beyer[1] · **Maik Büttner**[1] · **Vishnu Unnikrishnan**[1] · **Miro Schleicher**[1] · **Eirini Ntoutsi**[2] · **Myra Spiliopoulou**[1]

Received: 17 December 2019 / Accepted: 1 June 2020

Abstract
Traditional active learning tries to identify instances for which the acquisition of the label increases model performance under budget constraints. Less research has been devoted to the task of actively acquiring feature values, whereupon both the instance and the feature must be selected intelligently and even less to a scenario where the instances arrive in a stream with feature drift. We propose an active feature acquisition strategy for data streams with feature drift, as well as an active feature acquisition evaluation framework. We also implement a baseline that chooses features randomly and compare the random approach against eight different methods in a scenario where we can acquire at most one feature at the time per instance and where all features are considered to cost the same. Our initial experiments on 9 different data sets, with 7 different degrees of missing features and 8 different budgets show that our developed methods outperform the random acquisition on 7 data sets and have a comparable performance on the remaining two.

Keywords Active feature acquisition · Data streams · Feature drift

1 Introduction

Active learning (AL) usually concerns itself with a scenario where we deal with label scarcity and have the option to acquire labels for a cost with the help of an oracle. The goal is to intelligently pick instances whose labels, when acquired, improve the performance of our predictive model once we trained it on the chosen instances. A more recent development is to consider the scenario where we cannot acquire labels for a cost but missing features. This is called active feature acquisition (AFA). We propose new AFA methods for data streams. Settles describes the goal of AFA in [1] as the following: "The goal in active feature acquisition is to select the most informative features to obtain during training, rather than randomly or exhaustively acquiring all new features for all training instances." For example if we want to predict whether a patient has a certain complex disease or not, we could choose from multiple medical tests and have to find a trade-off between which tests are the most predictive and which tests are the cheapest. The results of a test represent the value to a feature that was initially missing and we would like to have a strategy that tells us if we still need to acquire features for an instance in order to give a confident prediction and if so which feature should we acquire under budget constraints. Differently from most of the works in the active feature acquisition community we deal with an instance-wise stream instead of a pool- or batch-based scenario where the developed strategies can look at multiple instances, evaluate them and then choose for which instances which features should be acquired. For example the predictors for a patients discomfort might change with the time of a year, e.g., during

The first-author position is shared between the first two authors Christian Beyer and Maik Büttner.

✉ Christian Beyer
christian.beyer@ovgu.de

Maik Büttner
maik.buettnern@st.ovgu.de

Vishnu Unnikrishnan
vishnu.unnikrishnan@ovgu.de

Miro Schleicher
miro.schleicher@ovgu.de

Eirini Ntoutsi
ntoutsi@kbs.uni-hannover.de

Myra Spiliopoulou
myra@ovgu.de

[1] Otto-von-Guericke University, Magdeburg, Germany

[2] Leibniz University, Hannover, Germany

Published online: 08 July 2020

winter month it might be best to test for flue, while in spring it might be allergies.

In this paper we are interested in a stream-based scenario where the decision if a feature should be acquired for an instance must be made immediately before seeing the next instance of the data stream. In order to achieve that goal we used an existing budgeting strategy from a stream-based AL algorithm that requires an instance quality estimate [2] and derived such estimates. These quality estimates are based on multiple metrics developed in [3], Average Euclidean Distance, Information Gain and Symmetric Uncertainty, all of which were formerly used in an active feature selection (AFS) scenario. We pose the following questions: (1) Is the chosen metric suitable for AFA? and (2) What effect does the budget management strategy have on our performance? This work delivers the following contributions:

Contributions

- We provide new methods for AFA in data streams.
- We also provide an evaluation framework for AFA in data streams along with a baseline strategy which randomly selects features to be acquired.

The paper is structured as follows: First we discuss related work and describe our proposed method in detail. Afterwards we discuss the developed evaluation framework and present the experimental setup. The last chapters cover the results, the conclusion and the future work.

2 Related work

In the area of active feature acquisition two concepts often overlap and are not as strictly separated from paper to paper. Those two concepts are active feature acquisition (AFA) and active feature selection (AFS). AFA deals with feature incomplete instances and their most efficient completion to gain performance benefits. AFS deals with predominantly feature complete instances and the selection of specific features to reduce dimensionality of the feature space while keeping similar or equal model performance. Though both subjects differ, the concepts and methods involved in determining the most relevant features often overlap. Table 1 provides an overview of all AFA methods discussed hereafter. Since this paper uses a metric originally described in an AFS paper, we will also discuss it briefly in this section.

A recent approach to AFA comes from Huang et al. [4] combining their method with a matrix completion algorithm. By minimizing both the empirical classification error and the matrix reconstruction error simultaneously through the means of an accelerated proximal gradient descend, the reconstructed values of unknown features vary from iteration to iteration serving as an informativeness indicator. With the help of this informativeness value potentially interesting features are acquired. Further dividing this informativeness value by the acquisition cost of the feature implements a simple cost consideration. Tests performed on 6 data sets against other matrix completion methods showed constantly high results for the method. This method was developed for the static use case and was not employed on data streams.

A different AFA approach is described by Saar-Tsechansky et al. [5] named Sampled Expected Utility that makes use of an utility value approach based on estimations of expected acquisition values and their impact on the model. To get the utility value of an acquisition candidate, it calculates two factors for all possible feature values. First it estimates the likeliness of a certain feature value given an estimated distribution of feature values based on the label. Second it predicts the impact on a performance metric if the classifier were to learn on this additional data. Once all values are calculated they may be divided by the acquisition cost and the sum of all these final products returns the utility value. Since the complexity of such calculations is immense, the paper considers a further reduction of the set of potential acquisitions by simply limiting the amount of uniformly randomly chosen queries or by favoring those candidates, that are part of a highly uncertain instance. While the method provides highly competitive performance gains, its complexity prohibits usage in a stream setting.

Melville et al. [6] proposed an AFA method for the partial completion of training data within a pool-based environment. It requires label knowledge at training time. With the help of a classifier it builds a model from which it randomly selects up to m misclassified, incomplete instances. If less than m examples are selected, the remaining examples are filled by incomplete but correctly classified examples based on the uncertainty score provided by the classifier. Those incomplete instances will be feature completed by acquiring all features in question and the completed data is used to build another model. This process of selection and acquisition continues until a stopping criterion is met or no incomplete instances remain. Evaluation using a C4.5 decision tree on five data sets from the UCI machine learning repository showed that the discussed method gained about 17% better error reduction compared against randomly selecting incomplete instances for acquisition.

Another static method is described in desJardins et al. [7] paper about a confidence-based approach to active feature acquisition. Its base concept is the creation of successive models that learn on an increasingly wider catalogue of features while limiting the test set to those instances the previous models deemed uncertain. The first of these models is trained on all instances using only the zero cost features. The instances the model is trained on have all

Table 1 A comparison of the mentioned pool-based active feature acquisition methods

Ref	Complexity	Budgeting	Use case	Method description
Huang et al. [4]	Medium	Yes	Feature value acquisition	Successive application of matrix reconstruction methods leads to variance within imputed values which are used to judge the utility of feature value acquisitions
Saar-Tsechansky et al. [5]	High	Yes	Feature value acquisition	Judges the utility of a feature value acquisition through estimations of expected acquisition values and their impact on the model
Melville et al. [6]	Low	No	Instance completion of train data	Feature completion of incomplete, labeled instances based on misclassification and confidence of a model
desJardins et al. [7]	Medium	Limited	Feature set acquisition	Greedy feature acquisition of increasingly smaller instance sets using cascaded models of increasingly higher feature dimensionality by means of model confidence analysis

the features acquired that are currently part of the feature subset. Instances above a set threshold are kept for the next model, that will also include the next cheapest feature for training, to be reevaluated. Once no more features may be added or all instances stay below the uncertainty threshold the acquisition process is done.

Most relevant to this paper is the work of Yuan et al. [3] who developed a batch-based method for AFS in data streams. The authors derived multiple metrics which describe how well a feature separates the existing classes in a data set. They then used this information in order to decide which features should be retained and used to train their model. Our work uses the proposed metrics but for a different purpose. Whereas the authors are trying to reduce the dimensionality of given data in order to improve classifier performance, we want to acquire missing features which would otherwise have to be imputed in order to increase classifier performance.

We combine the metric which we derived from [3] with a budget management strategy for traditional AL which was proposed by Kottke et al. [2]. The budget strategy is based on an incremental percentile filter which keeps a sorted list of usefulness values and makes an acquisition if the incoming usefulness value of a new instance is in the top percentile of the sorted list. Once the list reaches its defined length new incoming values lead to the deletion of the oldest value in the list irrespective of its usefulness. The authors showed that while budget consumption fluctuates it is around the defined budget on average. This approach makes it possible to make acquisition decisions in streams without having to see future instances while considering the recent past and adapting to concept drift.

3 Method

We perform instance-based feature acquisition on a data stream and thus have to make two decisions. The first decision is whether we should spend budget on a given instance with missing features and the second decision is which feature or features should be acquired. For both decisions we rely on common functions which were proposed in [3] and which estimate the discriminatory power of a feature. The estimates are based on either average euclidean distance (AED), information gain (IG) or symmetric uncertainty (SU). The two latter are entropy-based methods necessitating a discretization step before they can be applied on numerical data. We derive a new metric from said estimates called *merit* for each feature which approximates which feature will improve the prediction of an instance the most given the feature's cost. Once we know the *merits* of all features we use them to estimate the *quality* of an instance with missing features. The *quality* should give us an estimate how beneficial a feature acquisition is for the model. Since the aforementioned entropy-based methods require an additional discretization step and show a similar performance, we focus our reporting on the AED-based method. Our general approach is shown in Algorithm 1 and described hereafter.

Algorithm 1 AFA-Stream.

Input: stream X, budgetmanager bm, costs C
Ensure: window W initialized
  for all instance x in X **do**
    $m \leftarrow merits(W, C)$ // described in 3.1
    $q \leftarrow quality(x, m)$ // described in 3.2
    if $bm(q)$ **then**
      acquire feature with highest *merit* // described in 3.3
    end if
  end for

3.1 Merit of missing features using average euclidean distance

Let W be our window containing all recently labeled instances, let F describe a potentially incomplete feature

matrix with all features of W and let F_{ic} describe all feature values of feature i in our window W with class label c. We want a metric $g(F_i)$ that estimates the goodness of a feature in W for the predictive model. Yuan et al. suggest multiple such metrics in [3] based on average euclidean distance, information gain and symmetric uncertainty.

The average euclidean distance (AED) is a simple method of evaluating the separation between subgroups via their mean distances in euclidean space. In terms of our task, the values of these subgroups are the feature values with regards to the existing classes. This means for each feature we calculate how well this features separates the classes in euclidean space. Depending on the feature type, categorical or numerical, the algorithm to calculate the average distance changes.

Following the method as presented in [3] $MV(F_{ic})$ describes the mean value of all feature values of feature i given class label c. L describes all possible class labels.

$$MV(F_{ic}) = \frac{1}{|F_{ic}|} \sum_{n=0}^{|F_{ic}|-1} F_{ic}^n \tag{1}$$

$$AED_{num}(F_i) = \sqrt{\sum_{0 \le c < k < L} (MV(F_{ic}) - MV(F_{ik}))^2} \tag{2}$$

Categorical features are more complex to calculate since they require further comparisons of the amount of each occurrence of a possible feature value to each other possible feature value within a feature. Therefore let V_i denote all feature values feature i can take and let F_{icv} list all values of feature i given class c that are of value v.

$$AED_{nom}(F_i) = \sum_{0 \le c < k < L} \left[\frac{1}{|V_i|} \sum_{v \in V_i} \left(\frac{|F_{icv}|}{|F_{ic}|} - \frac{|F_{ikv}|}{|F_{ik}|} \right) \right] \tag{3}$$

Note that NaN cases in either feature type are skipped. Once all features have been given their average euclidean distances a cost factor may be applied to this value leaving us with a *merit* value for each feature. As such let C be the costs for all features and let C_i be the cost for acquiring the feature i.

$$merit(F_i, C_i) = \frac{g(F_i)}{C_i} \tag{4}$$

In our runs $g(F_i)$ would either be $AED(F_i)$, $IG(F_i)$ or $SU(F_i)$. For each feature F_i we store the respective $merit(F_i)$ in a vector based on the current window W.

$$merits(W, C) = (merit(F_0, C_0), \ldots, merit(F_{|F|-1}, C_{|C|-1}))^T \tag{5}$$

With each arriving instance the content of our window W changes making an update of our *merit* values necessary. We implemented two different ways of calculating the AED which will be described next.

3.1.1 Merit as single window average euclidean distance

As the name suggests in a single window average euclidean distance strategy all past instances are placed in a single window. Thus forgetting strategies orient themselves on the instances rather than individual features providing a more simple and compact implementation for stream-based classification tasks. In our experiments the sliding window containing our active batches was shared between the framework and our AFA strategy. The merit calculation behavior of the single window average euclidean distance (SWAED) method can be seen in Fig. 1.

3.1.2 Merit as multi window average euclidean distance

In a multi window average euclidean distance strategy each feature-label combination gets a separate window into which the last n values of those combinations are saved. Forgetting mechanism may be applied directly on these combinations allowing for more complexity while increasing the required data storage. For example one could use change detection [8] on the individual features independent of each other and only invoke a forgetting mechanism on a specific feature once a change has been detected. We

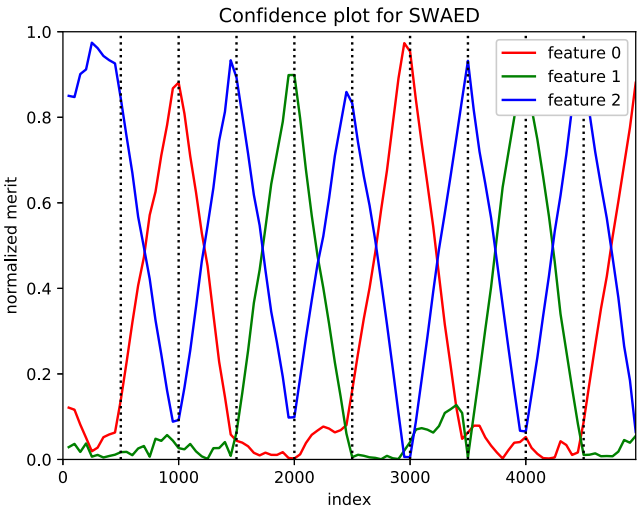

Fig. 1 The calculated merits of a SWAED method along the gen data stream. The data stream has always one feature identical to the label and two features which are randomly distributed. Every 500 instances the label associated feature changes which is depicted by the vertical lines. The SWAED shown has a window with the size of 500 instances and all data is available for its merit calculation. The peak merit values are reached once the window has been filled completely with instances of a single concept thus the SWAED method would only start favoring the most important feature after roughly half of the instances within the window belong to the new concept

could then forget all old feature values prior to the detected change.

3.2 Estimating instance quality using merits

With the *merits* calculated incoming instances may now be evaluated according to their expected usefulness. In order to estimate the usefulness of an incomplete instance before any feature is acquired we sum up the *merits* of all known features of that instance. The potential usefulness that can be achieved would then be that sum plus the *merit* of the best missing feature since we only allow for the acquisition of one missing feature per instance.

$$best\_f(x) = \underset{f \notin x.known\_f}{\arg\max} \; (merit(f)) \tag{6}$$

We are normalizing that usefulness by the number of known features plus one so that we do not favor instances which have only a few features missing because the cumulative *merits* of the known features of these instances would be high to begin with. As such we name this new usefulness value of an instance $quality$. Let x be the instance that our strategy is applied upon. The $quality$ of our acquisition choice is then calculated as follows:

$$quality(x, merits) = \frac{\sum_{f \in x.known\_f} merits[f] + best\_f(x)}{|x.known\_f| + 1} \tag{7}$$

Once the instance x with its corresponding label is added to our window W, distances and *merit* values for the updated features have to be recalculated. The *quality* of an instance is used by the budgeting strategy in order to decide whether features should be acquired for a given instance.

3.3 Budget management with and without instance quality

In this section we describe two different budget management (BM) strategies. The first one was developed by Kottke et al. [2] and takes the aforementioned instance *quality* into account. The second one on the other hand is a simple method that ignores the instance *quality* and was developed in order to investigate the effect of the budgeting on the AFA process. We acquire the feature with the highest *merit* if the BMs decision is positive.

3.3.1 Incremental percentile filter budgeting

The incremental percentile filter is a dynamic online budget manager as a means to quickly choose more qualitative instances while adapting to the changing usefulness a certain decision provides [2]. Its core idea is to store a limited amount of usefulness values in two windows: one sorted by the value itself, the other in the order they were received. If a newly added value exceeds the window size, the oldest value is removed from both windows. Based on the position the new value is then placed into the value-ordered window the decision of acquisition is made. In the case of these experiments the usefulness values are the $quality(x)$ values introduced in Section 3.2. When using the incremental percentile filter (IPF) as the BM we decide to make an acquisition if the *quality* of an instance is placed within the top percentile of the value-ordered window (vow).

$$bm(q) = \begin{cases} true & if\ q\ in\ top\ percentile\ of\ vow \\ false & otherwise \end{cases} \tag{8}$$

3.3.2 Simple budget manager

The simple budget manager implements a basic solution for staying below a desired budget. Incoming values are only allowed for acquisition if we are below the desired budget. For example if we have a budget of 10%, this strategy will always acquire missing features as long as we have acquired below 10% of the incomplete instances seen so far. Once our used budget is above or equal to our desired budget it will cease to acquire features until we are below the threshold again. When using the simple budget manager (SBM) we decide to make an acquisition if we still have budget.

$$bm(q) = \begin{cases} true & if\ used\_budget \leq available\_budget \\ false & otherwise \end{cases} \tag{9}$$

4 Evaluation framework and experimental setup

In this section we will discuss the evaluation framework[1] for AFA on data streams that we developed and how we used it in our experiments.

4.1 Evaluation framework for AFA on data streams

The goal of our evaluation framework is to be able to compare different AFA strategies on data streams. We use the model performance as a proxy for the quality of an AFA strategy, so a good AFA strategy should lead to a better model performance than a bad AFA strategy. In order to control for different degrees of missing data and to have comparable runs we require a feature complete data stream X. We first introduce a user specified amount of missing features which are randomly and uniformly distributed. We then impute the missing values with the mean value of the

[1] Code is provided at https://github.com/Buettner-Maik/afa-stream

respective feature. We can now calculate the lower and the upper performance bounds which help us evaluate an AFA strategy.

The lower bound is a model's performance without any AFA, that is it relies solely on imputation techniques. The upper bound on the other hand is a model's performance on the feature complete stream, so with no missing values. The choice of imputation method directly influences the lower bound performance making visualizing and evaluating the effectiveness of the proposed AFA methods harder if the imputation is better. Thus we opted to apply a simplistic mean imputation for numerical features and most frequent value imputation for nominal features only. The imputation method itself is replaceable in the framework.

As part of our framework we also provide a random baseline which selects missing features for acquisition at random. For an AFA strategy to be successful it must be better than the lower bound and better than the random baseline. We use prequential evaluation on the datastream with the AFA happening before the prediction.

Our framework uses a relative budget with respect to the number of incomplete instances in the data stream and only allows for at most one feature to be acquired per instance. For example a budget of 20% means for 20% of the incomplete instances we can acquire one feature. Since AFA on data streams is a very new topic we will introduce two simplifying assumptions to give us greater control over the experiments and help us untangle the interplay between AFA and budgeting strategy.

4.1.1 Assumptions

The first simplifying assumption that we made was to unify the costs of all features. This means all feature acquisition costs were set to be one. Thus the *merit* values equal our chosen feature goodness metric. This allows us to use the model performance as a proxy for the AFA quality which is not influenced by some arbitrarily chosen feature costs.

Secondly instances were only reviewed once during retrieval allowing only one feature to be acquired at maximum making the choice of the acquisition feature more critical. While it is possible for the partially restored instance to be fed back into the evaluation step, it was not done so in our experiments in order to disentangle the AFA strategy from the budgeting mechanism.

4.2 Experimental setup

All experiments were run with a stochastic gradient descend support vector machine (SGD) provided in scikit-learn's library with a limit of 100 iterations to achieve a tolerance of 0.001 using log-loss. The choice of the classifier is not mission-critical: we always refer to our bounds when evaluating our model performance. Different classifiers will behave similarly but with other bounds.

As we make no assumptions about the class distribution in the data, we chose Cohen's kappa as the performance metric as it is robust against unbalanced class distributions. Our experiments also showed similar results using F1-score, accuracy and log-loss so we omitted them from this paper.

The experiments were conducted on six static data sets handled as streams and three stream data sets of which two are synthetic. The single concept nature of the six static data sets allowed for the task to be run on a randomized permutation further enabling us to cross-validate different degrees of missingness on the same data set without having to consider concept drift. For each of these new missingness a new permutation was used for the run. Each stream data set was only run once in their chronological order. No additional steps were taken to mitigate concept drift in either of them.

4.2.1 Data sets

Six static data sets were run under the same conditions. These data sets by name are *abalone*, *adult*, *magic*, *nursery*, *occupancy* and *pendigits* from the UCI machine learning repository[2]. The synthetic data set *sea* [9] has four different concepts each lasting for 15,000 consecutive instances. Another synthetic data set *gen*[3] was created for this paper and contains 10 concepts each lasting for 500 consecutive instances with one of the three categorical features randomly chosen to be identical to the label. Finally the *electricity* [10] data set presents the only non-synthetic data stream.

The structure and purpose of these data sets are listed in Table 2.

4.2.2 Batches

For a run within the given framework each data set was split into batches. This was necessary because our implementation was extending a given framework for AL in streams which was developed for large streams with changing feature spaces.[4] The methods we describe here however do not rely on batches and can also be used in conventional streams. The first batch acted as a set of complete and labeled instances the classifier was initially trained on. To ensure at least one

[2]The used UCI data sets can be found here https://archive.ics.uci.edu/ml/datasets.html

[3]Generation script and data set is provided at: https://github.com/Buettner-Maik/afa-stream

[4]https://github.com/elrasp/osm

Table 2 All data sets used for the experiments

Data set	Instances	Labels	Features	Type	Purpose
sea	60,000	2	0 cat. 3 num.	Synth. stream	Determine if two specific features greater than threshold
electricity	45,312	2	1 cat. 7 num.	Stream	Determine if the market price of electricity rises or drops
adult	32,561	2	4 cat. 8 num.	Static	Determine if yearly income of individual is above $50,000
occupancy	20,560	2	1 cat. 7 num.	Static	Determine whether an office room is occupied
magic	19,020	2	0 cat. 10 num.	Synth. static	Determine if signal is gamma ray based on Cherenkov radiation
nursery	12,960	5	8 cat. 0 num.	Static	Determine the rank of child application for nursery school
pendigits	10,992	10	0 cat. 16 num.	Static	Determine digit written on a pad
gen	5000	2	3 cat. 0 num.	Synth. stream	Find the current feature describing the label
abalone	4177	3	0 cat. 8 num.	Static	Determine sex of abalones

instance of each label was part of this initially labeled data, we randomly picked an instance for each label to be added to this initial batch. After this selection the remaining data was shuffled. Since six of the data sets in question inherit no chronological order as they are static data sets handled as stream data, shuffling should not affect classification in any drastic manner. The so shuffled data was now split into batches of 50 instances with the last batch containing however many instances are left. The first of these batches was added to the initial batch such that the data the classifier was first trained on contains $50 + |labels|$ instances.

The data sets *sea* and *gen* have a chronological order which prohibits randomizing their permutation. Thus their order was left unaltered with the initial batch only containing 50 instances. For either data set type the rest of the batches were further altered by removing a set percentage of feature values for all features using a uniformly randomized selection.

4.2.3 Process of missing data generation

A single iteration on a data set consists of a specified feature missingness, several budgets and run configurations. With these parameters a new permutation of the data set is generated per iteration and missingness and the batches split according to the process described before. Once the data set is prepared and split into batches the lower and upper bound for that permutation may be calculated. After both bounds are determined, the data set can be evaluated given all combinations of budget values and run configurations.

4.2.4 Active feature acquisition setups

To evaluate the effectiveness of each of our components, we combined our four AFA methods (*single window average euclidean distance (SWAED), multi window average euclidean distance (MWAED), single window information gain (SWIG)* and *single window symmetric uncertainty (SWSU)*) with both our budget managers (*incremental percentile filter (IPF)* and *simple budget manager (SBM)*). Both the SWIG and SWSU required the discretization of numerical features which we implemented using the method proposed by Gama et al. [11] which noticeably slowed down the experiments. Since the random acquisition strategy *RA* does not provide a *quality* value for the *IPF* to use, it was only paired with the *SBM*. Thus we are left with nine different AFA strategy and budget manager combinations.

4.2.5 Parameters

In conclusion each static data set was run with seven different missingness values (0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875) and 8 budget values (0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1.0) for the nine combinations of strategies and budget managers mentioned before and the lower and upper bound methods described. Each of the described tests were run ten times for static data sets totaling the number of individual runs to 5180 and 518 on the non-static data sets.

5 Results

We will start our results with some general observations and continue on by asking and answering the following questions. First: Is the average euclidean distance a useful metric for acquisition? Second: How does the choice of budget manager impact the performance? Finally we will reflect on our findings.

5.1 General observations

We depict our experiment results in multiple tables each for a different data set. The tables show the mean kappa values derived from the different strategies, using a stochastic gradient descend support vector machine as classifier with different combinations of missingness values, budget values, AFA strategies and budget managers. For the static data sets we used ten runs each with a different permutation of the data and for the stream data set we used one run because of potential concept drift. The elongated missingness cell further shows the previously discussed upper and lower bounds. For example Table 3 depicting the *electricity* data set results has a lower bound of 0.308 and an upper bound of 0.478 for a missingness of 0.5. Values written in italics highlight the best column-wise value, that is the best mean kappa for a specific missingness and budget value among the five compared AFA strategies.

The results for our biggest data sets can be seen in Table 3 and Tables 6, 8 and 9 in Appendix and they differ substantially. The *sea* data set shows overall improvements when using our methods compared with the random approach. The *electricity* data set shows a similar improvement when used with our methods compared with the random approach. On the *adult* data set all methods performed very similarly with the entropy-based approaches slightly outperforming the AED methods. On the *occupancy* data set our methods vastly outperformed the baseline. We suspect that the methods all performed similarly on *adult* because of our restriction that we can only acquire one feature per instance which seems not to be enough. It is also the case that the lower and upper bound on this data set are very close to each other which further indicates that only marginal improvements can be made by any method. On the *occupancy* data set we can see that our methods lead to big improvements over the random baseline with a maximum difference of 0.436 in kappa (missingness = 0.75, budget = 1). We attribute this advantage to the data set having a single highly predictive feature in "weekday" making its acquisition particularly effective. This also explains why our method gets so close to the upper bound even when we only allow for one feature to be acquired per instance.

We also briefly investigated if data sets with exclusively numerical or categorical features have an impact on our method. For this we contrast the *magic* data set which is purely numerical (Table 4) and the *nursery* data set seen in Table 5 which has solely categorical features. On both data sets our methods using AED consistently outperform the random baseline which indicates to us that our method is suitable for both types of features but this needs to be more diligently investigated in the future. The results of the *nursery* data set further suggest that the entropy-based methods work more favorably on categorical features than the AED-based methods.

Our method leads to slight improvements over the random baseline on the *pendigits* data set shown in Table 10 in Appendix and all methods were almost identical again on

Table 3 Mean kappa values over 1 run on electricity data set using a SGD classifier

	Mean kappa over 1 run on data set electricity											
Missingness	0.25 (kappa ∈ [0.395, 0.477])				0.5 (kappa ∈ [0.308, 0.478])				0.75 (kappa ∈ [0.181, 0.486])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.404	0.418	0.431	0.437	0.325	0.335	0.347	0.37	0.212	0.23	0.253	0.263
MWAED+SBM	0.406	0.415	0.416	0.432	0.331	0.336	0.355	0.361	0.204	0.22	0.249	0.265
SWAED+IPF	0.41	0.422	0.435	0.459	0.33	0.362	0.375	0.424	0.248	0.281	0.331	0.386
SWAED+SBM	0.414	0.424	0.441	0.456	0.331	0.359	0.379	0.427	0.231	0.281	0.327	0.388
SWIG+IPF	0.419	0.431	0.452	0.463	0.34	0.351	0.385	0.445	0.244	0.272	0.334	0.403
SWIG+SBM	0.414	0.426	0.448	0.472	0.331	0.352	0.391	0.438	0.233	0.265	0.33	0.394
SWSU+IPF	0.427	0.436	0.439	0.471	0.336	0.361	0.398	0.44	0.249	0.3	0.349	0.417
SWSU+SBM	0.41	0.424	0.454	0.467	0.332	0.366	0.405	0.442	0.237	0.29	0.347	0.425
RA+SBM	0.404	0.416	0.43	0.443	0.317	0.322	0.333	0.352	0.205	0.22	0.231	0.255

Table 4 Mean kappa values over 10 runs on magic data set using a SGD classifier

	Mean kappa over 10 runs on data set magic											
Missingness	0.25 (kappa ∈ [0.319, 0.409])				0.5 (kappa ∈ [0.229, 0.409])				0.75 (kappa ∈ [0.131, 0.41])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.344	0.357	0.375	0.395	0.27	0.303	0.333	0.367	0.185	0.226	0.276	0.321
MWAED+SBM	0.337	0.357	0.375	0.394	0.26	0.292	0.326	0.364	0.177	0.224	0.274	0.321
SWAED+IPF	0.347	0.358	0.377	0.391	0.265	0.302	0.339	0.366	0.185	0.23	0.27	0.323
SWAED+SBM	0.34	0.357	0.376	0.395	0.261	0.296	0.333	0.368	0.174	0.223	0.274	0.317
SWIG+IPF	0.341	0.353	0.372	0.391	0.259	0.29	0.32	0.358	0.173	0.205	0.256	0.301
SWIG+SBM	0.34	0.357	0.375	0.392	0.259	0.289	0.326	0.361	0.165	0.209	0.256	0.299
SWSU+IPF	0.336	0.353	0.37	0.391	0.256	0.287	0.32	0.359	0.171	0.205	0.251	0.299
SWSU+SBM	0.337	0.354	0.376	0.393	0.257	0.29	0.325	0.361	0.169	0.212	0.254	0.304
RA+SBM	0.326	0.334	0.34	0.355	0.232	0.248	0.26	0.266	0.142	0.151	0.159	0.171

the *abalone* data set as seen in Table 11 in Appendix. We suspect the reason for the similar performance on *abalone* and *adult* across all methods are due to the the lower and upper bounds of both data sets being close together. As such the span for improvement is already small. Our method consistently outperformed the random baseline on the *gen* data set.

5.2 Is the average euclidean distance a useful metric for acquisition?

To answer this question we compare the mean kappa values of our classification results between the SWAED combinations and the random acquisition strategy. If our metric has value it must outperform the random acquisition approach. Since the pool of sub optimal decisions increases with higher budgets and missingness values, we expect greater performance gains for both of our SWAED strategies in those scenarios. The performance gap between these methods increases the higher the budget gets as seen in Tables 4 and 5 and Table 9 in Appendix. When comparing different missingness values in the tables we can also see a bigger impact of AFA for the higher missingness values. These results suggest that AED is in fact a useful metric for this AFA method but depending on the data set and the amount of categorical data within, other quality functions might perform better. On the data sets *occupany, adult, nursery and gen* the use of Information Gain and Symmetric Uncertainty lead to better results but at the cost that an additional discretization

Table 5 Mean kappa values over 10 runs on nursery data set using a SGD classifier

	Mean kappa over 10 runs on data set nursery											
Missingness	0.25 (kappa ∈ [0.516, 0.84])				0.5 (kappa ∈ [0.3, 0.843])				0.75 (kappa ∈ [0.129, 0.842])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.577	0.632	0.69	0.75	0.385	0.458	0.549	0.65	0.225	0.317	0.43	0.566
MWAED+SBM	0.572	0.634	0.691	0.751	0.377	0.462	0.557	0.649	0.219	0.324	0.437	0.567
SWAED+IPF	0.586	0.645	0.704	0.761	0.385	0.467	0.558	0.654	0.222	0.313	0.424	0.568
SWAED+SBM	0.577	0.638	0.697	0.762	0.378	0.465	0.561	0.653	0.215	0.32	0.441	0.568
SWIG+IPF	0.599	0.661	0.735	0.779	0.401	0.494	0.594	0.683	0.232	0.335	0.458	0.591
SWIG+SBM	0.583	0.646	0.71	0.778	0.381	0.477	0.58	0.68	0.225	0.333	0.454	0.592
SWSU+IPF	0.598	0.666	0.734	0.777	0.401	0.494	0.598	0.684	0.233	0.336	0.46	0.589
SWSU+SBM	0.578	0.646	0.711	0.779	0.386	0.475	0.58	0.682	0.222	0.331	0.454	0.589
RA+SBM	0.549	0.584	0.614	0.646	0.321	0.346	0.371	0.398	0.147	0.169	0.189	0.209

step was necessary for the continuous features. An investigation on which kind of data which quality function is expected to be most beneficial and how to handle the trade-off between additional computational cost and expected performance gains lies in the scope of future experiments.

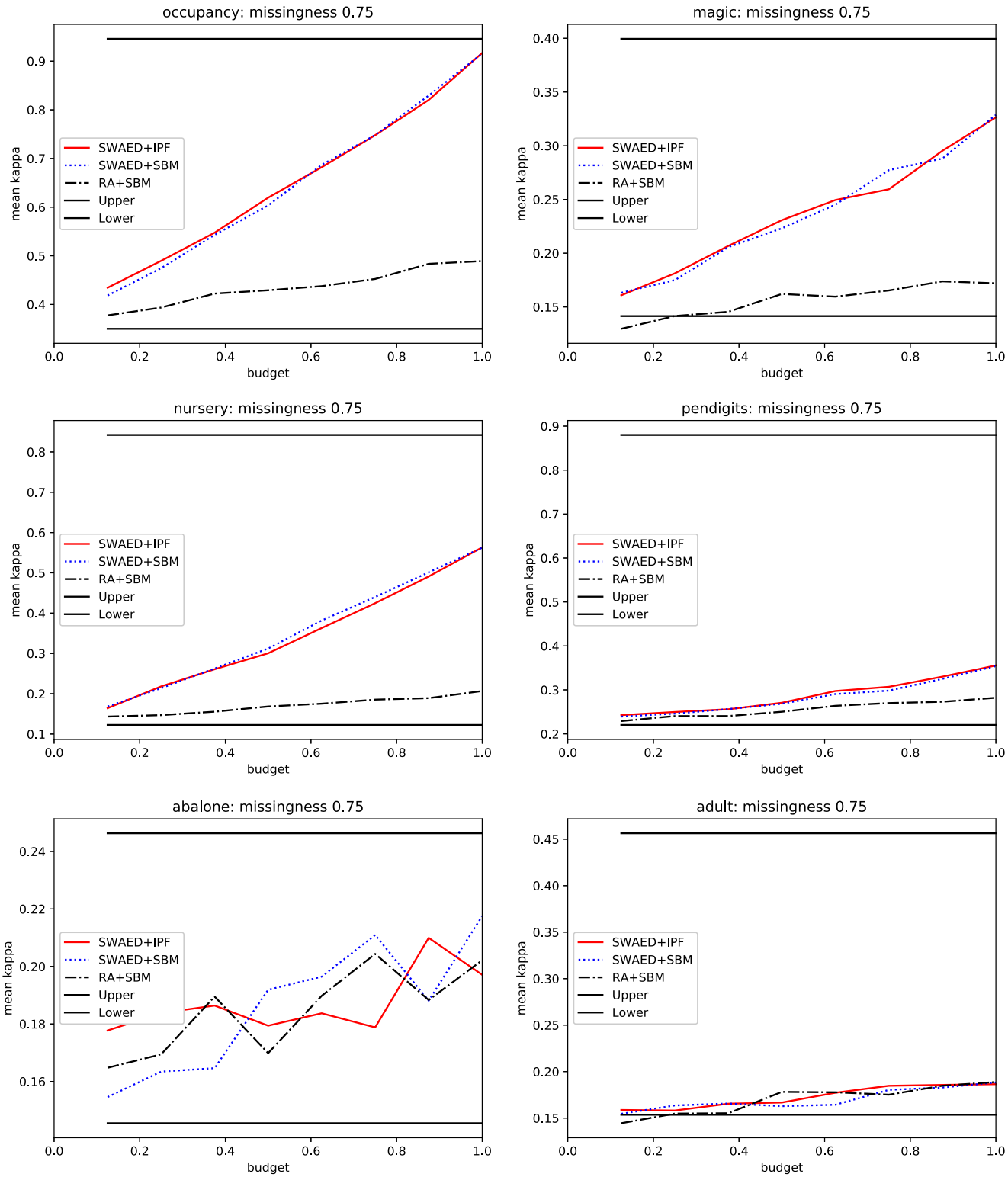

Fig. 2 The mean kappa performance comparison over ten runs of the *single window average euclidean distance (SWAED)* configurations on the six static data sets with a fixed feature missingness of 0.75

5.3 How does the choice of budget manager impact the performance?

Both the simple budget manager and incremental percentile filter perform equally when dealing with a budget of 0 or 1, as we make no acquisitions in the former and always acquire the feature with the highest *merit* in the latter case. As such only the steepness of the performance gain in between maximum and minimum budget varies. Naturally a good budget manager achieves a higher performance with less budget than a worse one. Figures 2 and 3 show us a quick comparison between the mean kappa values for the relevant AFA+BM combinations for a given missingness of 75%. Do note that while we presented two implementations of a window for the average euclidean distance and listed both in the tables mentioned before, we will only assess the results of our single window approach (SWAED) due to their similar performances and omitted MWAED strategies in our graphs for better readability. We can see little to no difference between the SWAED+IPF and SWAED+SBM implementations on most data sets except *gen*. This also holds true for the other missingness values which are not explicitly shown in this paper. There seems to be a slight advantage in using the incremental percentile filter but the difference in performance is so small that further experiments are needed which are part of our future work.

5.4 Summary

Our results show that the proposed AFA methods outperform a random baseline on seven data sets and have comparable performance on the other two. The performance difference is usually most pronounced in scenarios where we have a high degree of missingness and a high budget. This result is intuitive as these are the situations where our

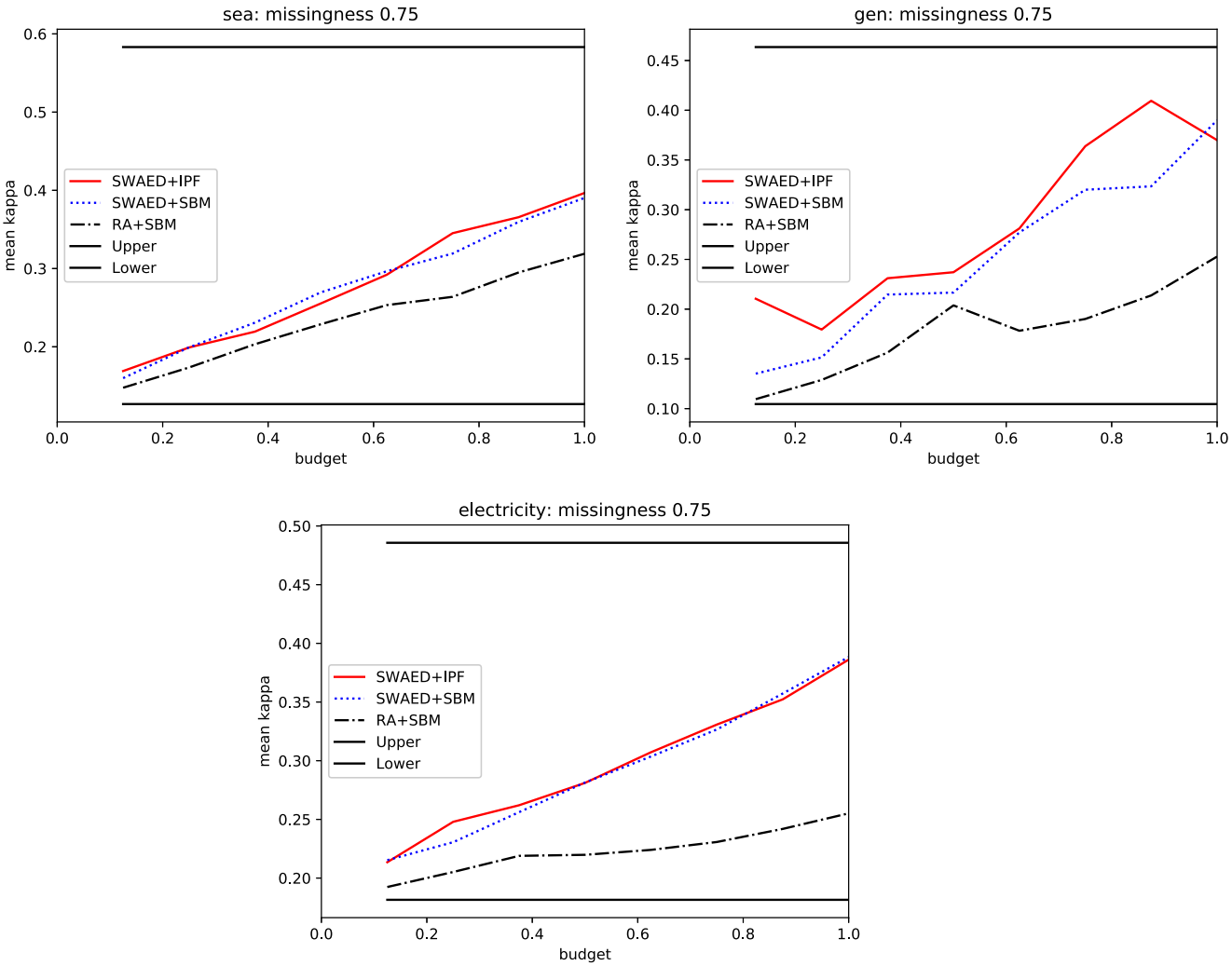

Fig. 3 The mean kappa performance comparison over ten runs of the *single window average euclidean distance (SWAED)* configurations on the three stream data sets with a fixed feature missingness of 0.75

methods can differentiate the most from the random baseline. The biggest gains in performance can be seen on the *nursery* and *occupancy* data sets where we achieve a maximum mean difference in kappa of 0.383 and 0.436 between our best method and the random baseline respectively. On the two data sets where our methods were not the clear winner the maximum mean difference in kappa is negligible. These results underpin our assertion that our proposed methods for AFA in data streams are superior in most cases and at worst comparable in performance with a random baseline. Considering the question whether average euclidean distance is a suitable basis for our AFA strategy, we would answer with yes based on our results but entropy methods (IG and SU) seem to perform better on categorical data. The question whether budgeting affects our results we would tentatively answer with yes as the IPF usually slightly outperforms the SBM but this needs further investigation as the differences were not very large. The time complexity of our method was also not evaluated in detail but in the worst case our AED methods were three times slower than the random baseline with the entropy solutions performing even worse due to the discretizer.

6 Conclusion and outlook

In this paper we have shown a new set of AFA methods for data streams. We also provide an evaluation framework for AFA in data streams including a baseline which acquires features randomly. Our proposed methods were extensively evaluated on nine data sets using eight different budgets and seven different degrees of missingness of which a representative subset is shown in this paper. We have shown the performance of these methods in combination with two different budget managers. Our results show that our methods outperform the random baseline in most cases and are at worst similar to the random baseline. The methods performed well when facing purely numerical, purely categorical and mixed data sets. When comparing our two different budgeting strategies we showed that the incremental percentile filter sometimes outperforms our simple budgeting manager but the gains in performance were small.

At the beginning of this work we posed two questions: (1) Is the chosen metric suitable for AFA? and (2) What effect does budget management strategy have on our performance? The first question can be answered with "yes" as our proposed metric consistently outperforms the random baseline or is at least as good. The second question is harder to answer. There seems to be a small tendency that the incremental percentile filter works better than the simple budget manager but the performance is very similar in most cases so this question needs a more detailed investigation.

Our work has the following limitations which we want to address in the future: Firstly we only allowed for one feature to be acquired per instance which might also explain the lower performance on some data sets as discussed in Section 5.1. Secondly we set the cost of all features be equal in order to enable our method to pick the most valuable feature for our model. Our methods themselves on the other hand are already prepared to deal with varying feature costs and we plan to run such experiments in the future.

Additional future work includes the usage of further real stream data sets with concept drift which would allow for a deeper focus on drift detection methods and forgetting mechanisms. Also, the experimental results varied vastly from data set to data set. Further comparisons of different classifiers may reveal performance gains when it comes to specific classification and method combinations. Furthermore, we plan to investigate the effects of different feature costs in the future as this is to be expected in a real world application, as well as the acquisition of multiple features per instance. Such considerations may also include the conception of different budget managers and different metrics for the calculations of our feature *merit* and instance *quality*. Such new metrics should for instance take into account if features are heavily correlated which could lead to a waste of budget on superfluous features.

Funding information Open Access funding provided by Projekt DEAL. This work is partially funded by the German Research Foundation, project OSCAR "Opinion Stream Classification with Ensembles and Active Learners." The principal investigators of OSCAR are Myra Spiliopoulou and Eirini Ntoutsi. Additionally, Christian Beyer is also partially funded by a PhD grant from the federal state of Saxony-Anhalt.

Appendix

Here we list the remaining results of our experiments in tables. The results of the streaming data sets *sea* and *gen* are shown in Tables 6 and 7. Tables 8, 9, 10 and 11 show the remaining static data sets *adult*, *occupancy*, *pendigits* and *abalone* respectively.

Table 6 Mean kappa values over 1 run on sea data set using a SGD classifier

	Mean kappa over 1 run on data set sea											
Missingness	0.25 (kappa ∈ [0.428, 0.593])				0.5 (kappa ∈ [0.275, 0.588])				0.75 (kappa ∈ [0.127, 0.583])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.469	0.493	0.529	0.562	0.339	0.402	0.442	0.49	0.189	0.254	0.34	0.389
MWAED+SBM	0.458	0.49	0.528	0.565	0.331	0.384	0.434	0.495	0.193	0.261	0.327	0.394
SWAED+IPF	0.468	0.5	0.526	0.564	0.343	0.399	0.447	0.497	0.199	0.255	0.345	0.397
SWAED+SBM	0.456	0.494	0.526	0.561	0.327	0.387	0.444	0.499	0.199	0.27	0.319	0.39
SWIG+IPF	0.467	0.496	0.523	0.561	0.336	0.412	0.441	0.499	0.19	0.263	0.34	0.393
SWIG+SBM	0.448	0.493	0.528	0.565	0.33	0.385	0.446	0.502	0.197	0.264	0.33	0.395
SWSU+IPF	0.476	0.499	0.518	0.567	0.341	0.408	0.444	0.496	0.195	0.255	0.335	0.394
SWSU+SBM	0.46	0.493	0.527	0.559	0.334	0.384	0.446	0.502	0.197	0.264	0.332	0.396
RA+SBM	0.453	0.483	0.518	0.544	0.314	0.359	0.399	0.442	0.174	0.229	0.264	0.319

Table 7 Mean kappa values over 1 run on gen data set using a SGD classifier

	Mean kappa over 1 run on data set gen											
Missingness	0.25 (kappa ∈ [0.376, 0.503])				0.5 (kappa ∈ [0.225, 0.508])				0.75 (kappa ∈ [0.105, 0.463])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.403	0.415	0.438	0.471	0.27	0.339	0.379	0.437	0.169	0.223	0.303	0.355
MWAED+SBM	0.391	0.418	0.449	0.482	0.248	0.29	0.352	0.442	0.12	0.197	0.288	0.37
SWAED+IPF	0.406	0.424	0.459	0.436	0.289	0.35	0.395	0.465	0.179	0.237	0.364	0.37
SWAED+SBM	0.41	0.405	0.434	0.444	0.266	0.32	0.404	0.431	0.152	0.217	0.32	0.39
SWIG+IPF	0.404	0.437	0.437	0.459	0.312	0.353	0.415	0.438	0.226	0.275	0.319	0.415
SWIG+SBM	0.41	0.421	0.442	0.478	0.259	0.298	0.415	0.45	0.156	0.209	0.32	0.401
SWSU+IPF	0.437	0.445	0.433	0.509	0.321	0.354	0.413	0.478	0.185	0.291	0.305	0.392
SWSU+SBM	0.395	0.4	0.452	0.467	0.307	0.324	0.39	0.43	0.174	0.23	0.296	0.414
RA+SBM	0.365	0.436	0.444	0.476	0.265	0.3	0.298	0.371	0.129	0.204	0.19	0.253

Table 8 Mean kappa values over 10 runs on adult data set using a SGD classifier

	Mean kappa over 10 runs on data set adult											
Missingness	0.25 (kappa ∈ [0.371, 0.445])				0.5 (kappa ∈ [0.283, 0.443])				0.75 (kappa ∈ [0.156, 0.445])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.379	0.38	0.385	0.401	0.285	0.294	0.304	0.32	0.16	0.172	0.186	0.204
MWAED+SBM	0.378	0.382	0.389	0.397	0.289	0.299	0.307	0.319	0.163	0.173	0.185	0.202
SWAED+IPF	0.376	0.381	0.388	0.399	0.286	0.288	0.299	0.312	0.164	0.172	0.185	0.199
SWAED+SBM	0.376	0.384	0.39	0.398	0.289	0.296	0.302	0.312	0.163	0.174	0.183	0.198
SWIG+IPF	0.385	0.388	0.396	0.407	0.29	0.302	0.318	0.337	0.175	0.193	0.216	0.242
SWIG+SBM	0.38	0.389	0.397	0.408	0.293	0.307	0.321	0.339	0.17	0.194	0.213	0.242
SWSU+IPF	0.379	0.385	0.392	0.407	0.29	0.299	0.309	0.335	0.171	0.188	0.208	0.232
SWSU+SBM	0.379	0.387	0.395	0.403	0.29	0.304	0.316	0.334	0.17	0.187	0.207	0.234
RA+SBM	0.375	0.386	0.392	0.396	0.291	0.295	0.304	0.314	0.165	0.177	0.187	0.204

Table 9 Mean kappa values over 10 runs on occupancy data set using a SGD classifier

	Mean kappa over 10 runs on data set occupancy											
Missingness	0.25 (kappa ∈ [0.771, 0.949])				0.5 (kappa ∈ [0.595, 0.95])				0.75 (kappa ∈ [0.364, 0.948])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.803	0.837	0.874	0.93	0.645	0.701	0.781	0.887	0.484	0.58	0.695	0.849
MWAED+SBM	0.802	0.842	0.882	0.932	0.656	0.728	0.801	0.888	0.481	0.593	0.711	0.848
SWAED+IPF	0.809	0.848	0.894	0.945	0.666	0.733	0.822	0.934	0.504	0.624	0.75	0.918
SWAED+SBM	0.804	0.846	0.896	0.944	0.666	0.738	0.827	0.933	0.486	0.613	0.751	0.916
SWIG+IPF	0.806	0.846	0.883	0.946	0.667	0.735	0.814	0.937	0.51	0.613	0.752	0.931
SWIG+SBM	0.805	0.846	0.893	0.944	0.666	0.74	0.827	0.936	0.492	0.623	0.757	0.932
SWSU+IPF	0.818	0.854	0.898	0.945	0.685	0.759	0.841	0.94	0.525	0.651	0.777	0.93
SWSU+SBM	0.804	0.846	0.895	0.945	0.669	0.743	0.831	0.94	0.49	0.621	0.756	0.933
RA+SBM	0.787	0.805	0.821	0.843	0.62	0.642	0.662	0.685	0.397	0.432	0.458	0.497

Table 10 Mean kappa values over 10 runs on pendigits data set using a SGD classifier

	Mean kappa over 10 runs on data set pendigits											
Missingness	0.25 (kappa ∈ [0.654, 0.881])				0.5 (kappa ∈ [0.46, 0.879])				0.75 (kappa ∈ [0.22, 0.879])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.664	0.677	0.693	0.727	0.481	0.498	0.522	0.566	0.247	0.273	0.306	0.354
MWAED+SBM	0.666	0.683	0.698	0.725	0.481	0.503	0.53	0.564	0.241	0.271	0.307	0.355
SWAED+IPF	0.664	0.676	0.692	0.727	0.48	0.497	0.522	0.563	0.246	0.267	0.3	0.345
SWAED+SBM	0.666	0.68	0.698	0.727	0.478	0.502	0.528	0.565	0.24	0.267	0.298	0.345
SWIG+IPF	0.665	0.675	0.688	0.713	0.474	0.484	0.508	0.546	0.238	0.253	0.275	0.309
SWIG+SBM	0.663	0.674	0.692	0.715	0.472	0.49	0.514	0.546	0.234	0.251	0.278	0.305
SWSU+IPF	0.664	0.674	0.692	0.723	0.474	0.49	0.511	0.559	0.237	0.259	0.285	0.328
SWSU+SBM	0.665	0.679	0.695	0.721	0.476	0.495	0.521	0.561	0.235	0.257	0.288	0.328
RA+SBM	0.663	0.672	0.683	0.694	0.471	0.488	0.499	0.515	0.231	0.247	0.264	0.284

Table 11 Mean kappa values over 10 runs on abalone data set using a sgd classifier

	Mean kappa over 10 runs on data set abalone											
Missingness	0.25 (kappa ∈ [0.226, 0.238])				0.5 (kappa ∈ [0.206, 0.249])				0.75 (kappa ∈ [0.163, 0.252])			
Budget	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
MWAED+IPF	0.226	0.22	0.221	0.233	0.211	0.216	0.211	0.218	0.186	0.19	0.193	0.193
MWAED+SBM	0.224	0.232	0.229	0.223	0.204	0.208	0.215	0.22	0.166	0.179	0.186	0.198
SWAED+IPF	0.23	0.229	0.225	0.234	0.21	0.211	0.213	0.214	0.185	0.188	0.193	0.197
SWAED+SBM	0.223	0.229	0.227	0.229	0.205	0.207	0.215	0.214	0.166	0.181	0.191	0.197
SWIG+IPF	0.233	0.231	0.229	0.241	0.211	0.212	0.211	0.217	0.186	0.186	0.189	0.2
SWIG+SBM	0.222	0.228	0.232	0.234	0.212	0.211	0.215	0.222	0.178	0.179	0.199	0.202
SWSU+IPF	0.228	0.219	0.236	0.237	0.21	0.212	0.217	0.219	0.176	0.194	0.192	0.21
SWSU+SBM	0.229	0.222	0.233	0.236	0.206	0.21	0.211	0.214	0.165	0.184	0.201	0.205
RA+SBM	0.227	0.228	0.232	0.222	0.213	0.208	0.216	0.211	0.17	0.169	0.185	0.2

References

1. Settles B (2009) Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences
2. Kottke D, Krempl G, Spiliopoulou M (2015) Probabilistic active learning in datastreams. In: International symposium on intelligent data analysis. Springer, pp 145–157
3. Yuan L, Pfahringer B, Barddal JP (2018) Iterative subset selection for feature drifting data streams. In: Proceedings of the 33rd annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018, pp 510–517
4. Huang S-J, Xu M, Xie M-K, Sugiyama M, Niu G, Chen S (2018) Active feature acquisition with supervised matrix completion. In: Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, pp 1571–1579
5. Saar-Tsechansky M, Melville P, Provost FJ (2009) Active feature-value acquisition. Manag Sci 55(4):664–684
6. Melville P, Saar-Tsechansky M, Provost FJ, Mooney RJ (2004) Active feature-value acquisition for classifier induction. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK, pp 483–486
7. desJardins M, MacGlashan J, Wagstaff KL (2010) Confidence-based feature acquisition to minimize training and test costs. In: Proceedings of the 2010 SIAM international conference on data mining. SIAM, pp 514–524
8. Sebastiao R, Gama J (2009) A study on change detection methods. In: Progress in artificial intelligence, 14th Portuguese conference on artificial intelligence, EPIA, pp 12–15
9. Nick Street W, Kim YS (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, New York, NY, USA. ACM, pp 377–382
10. Harries M, Wales NS (1999) Splice-2 comparative evaluation: Electricity pricing. Technical report
11. Gama J, Pinto C (2006) Discretization from data streams: applications to histograms and data mining. In: Proceedings of the 2006 ACM symposium on Applied computing. ACM, pp 662–667