Exploration Techniques in Active Learning in Classification

Peter Kuchling
Faculty of Engineering and Mathematics
Hochschule Bielefeld – University of Applied Sciences and Arts
Bielefeld, Germany
peter.kuchling@hsbi.de

***Abstract*—Active Learning is the process of selectively querying unlabelled data to be classified by an expert for efficient supervised learning on small data to, among other things, reduce labelling costs. While many works are concerned with the notion of uncertainty quantification as a measure to query informative data points, less work has been dedicated to the preceding exploration phase. In this article, we introduce techniques which can be used to explore the unlabelled data before switching to an exploitation algorithm such as query by uncertainty. Besides benchmarking various algorithms, a novel "fake class"-technique is introduced, which is specialized on discovering new classes within the unlabelled dataset, but also generally improves the performance of the algorithms.**

***Index Terms*—Active Learning, Exploration, Supervised Learning, Pool-based query**

I. Introduction

A. Active Learning

Nowadays, a large amount of unlabelled data can be accessed for free or at relatively low cost. However, the labelling process of such data can take up large amounts of resources, such as time (e.g. annotating large texts or long videos) and money (expensive experiments, medical procedures, expert's assessment). Therefore, one is inclined to reduce the amount of resources needed by only labelling points which are deemed "useful". The hope is that this way, a learner can be trained on a low amount of high-quality data to match the performance of a classical supervised learner with a larger training set. The idea of active learning is to let the learner itself decide what it deems useful data to be labelled. This process is called querying, where one or more points of the unlabelled data are sent to an expert, or oracle, to be assigned their true label.

Usually, active learning can be categorized as one of three settings. First, we have *membership query synthesis*, where the learner generates synthetic points to be labelled by an expert. This setting may not work in situations where the learner may produce examples that are not recognizable, e.g., an arrangement of pixels that does not represent a number in handwritten digit recognition [1]. The second scenario is the *stream-based* scenario, where the examples are presented to the learner sequentially and it needs to decide whether to keep an example and have it labelled, or to discard it. The third setting is the *pool-based* scenario, where the learner has access to the whole dataset from the beginning. It can then decide which examples it wants to query for labelling. Depending on the precise conditions, the querying process may be done sequentially or at once. The focus of this article is the pool-based scenario with query processes depending on the algorithm. For more information on the various scenarios and overviews of active learning in general, we refer to the surveys [2], [3].

An illustration of the workflow of an active learner is given by [4, Figure 1]: Assume that a set of already-labelled data is available to the learner. Given a probabilistic model, each unlabelled point can be assigned an uncertainty score. The point with the highest uncertainty is then queried, since this point is deemed to be the most informative. This part of the active learning process is called exploitation phase, since the learner exploits the previously gained knowledge of the available class labels to improve the model. The technique is used in various works, e.g. [5]. However, this technique can fail if the initial data is not chosen properly, which is known as sampling bias. One example is presented in Section I-C.

To mitigate this problem, one needs to employ an efficient exploration technique on the data before exploiting the knowledge gained there. This part is known as the exploration phase, and it focuses on representative instead of informative data. In this article, we introduce possible exploration techniques and compare them to previously analysed methods. Furthermore, a new exploration mechanism is introduced, which also employs the probabilistic learner for exploration by generating a "fake class".

In summary, the contribution of this article is as follows:

- Introduce the two novel exploration algorithms "Latin Hypercube Sampling Dirac" and the center of gravity-based approach.
- Introduce the novel "fake class"-technique, which modifies any exploration algorithm to especially look for undiscovered classes.
- Discuss practical considerations and benchmark the methods on various synthetic and real datasets against the state of the art.

The author is supported by the SAIL network, funded by the Ministry of Culture and Science of the State of North Rhine-Westphalia (Germany) under the grant no NW21–059B.

The article is structured as follows. The remaining part of Section I explains the datasets used for benchmarking and the quantities used to measure the performance of the algorithms. We close the section by commenting on some practical considerations. Section II presents the algorithms in more detail. In Section III, we introduce and discuss the new "fake class"-technique for exploration. Section IV presents and discusses the results of the benchmarks, while Section V gives a conclusion and outlook on possible future research.

B. Datasets used for benchmarking

Let us briefly describe the underlying tests conducted for each algorithm. To focus the attention on the core aspects of the exploration algorithms, the datasets considered for benchmarking all have continuous real-valued features with no missing data. Nevertheless, the datasets can be distinguished by number of features and balance of classes. Besides testing on some binary artificial datasets with varying degrees of balance as in [4], we considered three real-world datasets with more than two classes. Especially for class detection, the glass dataset is interesting, as there are some classes with a low number of occurences. Additionally, two instances of the synthetic two moon dataset are generated to compare the performance of the exploration algorithms on binary data which is not linearly separable as well.

The tested datasets are as follows:

- Synthetic binary datasets, points uniformly distributed in the unit square $[0,1]^2$. The class boundaries are given by
  1) $B_1 = \{x_1 = x_2\}$, balanced classes
  2) $B_2 = \{x_2 = x_1 + \frac{1}{2}\}$, ratio 7:1
  3) Checkerboard pattern, balanced classes
- Dry Bean Dataset [6]
- Yeast Dataset [7]
- Glass identification [8]
- Two moons, synthetic dataset, 2000 data points
- Two moons, synthetic dataset, 2000 data points with Gaussian noise with standard deviation 0.2.

The three artificial datasets on $[0,1]^2$ as well as the noisy moons are depicted in Figure 1. For each dataset, various amounts of query budget were used for exploration. Furthermore, to account for randomness in the initial sampling process, each experiment was repeated 50 times with different initial data.

As probabilistic classifier, a random forest with $M = 200$ trees was used, since it showed better performance than, say, a neural network in the small-data setting of active learning.

C. Performance measures

The first measure for performance is the classification accuracy of the active learning algorithm. However, as noted in e.g. [9], exploration queries do not produce the most accurate models, unless combined with an exploitation algorithm. This is especially visible in [9, Figure 1]. Therefore, one is inclined to use another way of judging the performance of exploration techniques. To this end, let us look at the following motivating example.

When examining active learning queries, one may be tempted to start with exploitation query right away. After all, the idea to eliminate uncertain classifications by obtaining a definite label sounds like a good idea. However, if the model is not prepared, this may have negative consequences. For our example, let us look at an active learning situation given the Iris dataset [10] with three random points as well as eight queries by uncertainty. The result is depicted in Figure 2: Here, the learner deemed the left region to be certainly belonging to the blue class, while Figure 3 shows that this is actually not the case. The example described above motivates adding class discovery rate as an alternative performance measure. Here, the number of classes discovered after the initial data is recorded for each seed, then averaged over the total number of classes. Especially in multiclass scenarios it may make sense to take class discovery into account, which is done in our benchmarks for the multiclass datasets. But even in an imbalanced binary classification problem, this measure can be used as an indicator of the capabilities of discovering the minority class.

D. Practical Considerations

Besides performance in general, there are some factors one should keep in mind when considering which exploration algorithm to use. Some of the algorithms presented below query points sequentially, while others divide the feature space and distribute all points in one step. Such algorithms are not suitable if one wishes to move freely between exploration and exploitation, such as in ε-greedy algorithms.

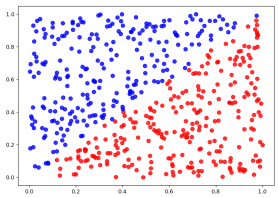
The algorithms presented below are tested in a pool-based scenario. However, most of them generate a synthetic point on the way. Therefore, one may also think about using them in a membership query scenario.

The nearest-neighbour search used in Section II needs an underlying distance function on the feature space. When replaced with the fake class-algorithm presented in Section III, this requirement is no longer present. This also means that the fake class-algorithm can be used in conjunction with other algorithms not presented here, as long as they produce a synthetic point in the feature space, even with categorical features. However, this has not been tested for performance yet. One drawback of the fake class-algorithm is the much higher computational time, as the probabilistic learner has to be trained for each query.

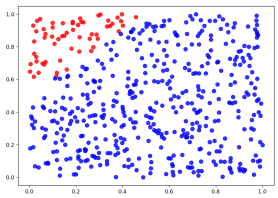
One last property to keep in mind is structure-agnostic vs. structurally sensitive algorithms: Some of the algorithms presented below only need the boundaries of the feature space to work, while others take the structure of the unlabelled dataset into account. This may have benefits, but also drawbacks, depending on the dataset.

II. Exploration queries

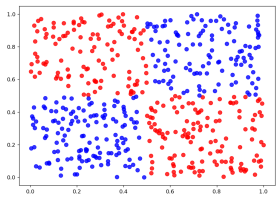
As stated before, an exploration algorithm should focus on finding "representative" data points. The notion of representativeness is used in statistics when talking about e.g. surveys, indicating that each participant of the underlying dataset is



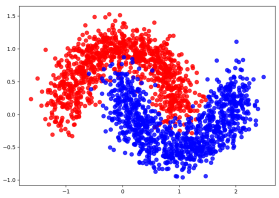
(a) Linear decision boundary B_1, balanced classes



(b) Linear decision boundary B_2, imbalanced classes



(c) Checkerboard pattern



(d) Two moons dataset, $\sigma = 0.2$

Fig. 1. Artificial datasets used for benchmarking.

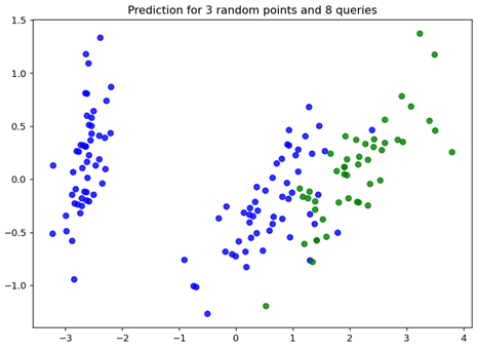


Fig. 2. 3 random points and 8 uncertainty queries may result in a missing class.

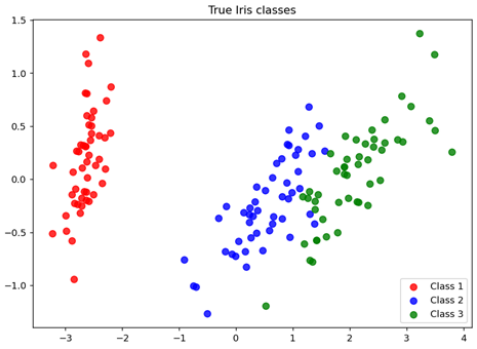


Fig. 3. True classes of the first two features of the PCA-transformed Iris dataset

equally likely to be selected. Hence, the first interpretation of "representative" is simply "randomly chosen". There are actually already two ways to consider randomness on the underlying dataset: On the one hand, one may pick a point from the dataset at random, which we shall call "random choice". On the other hand, one may say that the points stem from some underlying (unknown) distribution and we select the points by uniformly sampling a point from the whole feature space, then selecting the closest real data point, which we call "spatial random" sampling.

Our goal is to try and intelligently select points in a way that outperforms random selection, while keeping the idea of "representative". Therefore, we consider mechanisms which imitate some property of random sampling while exploiting the structure of the underlying data points and/or the already-labelled points. The three concepts based on this idea are as follows:

- "Representative" meaning "balanced": This concept can be seen within balanced exploration and in the Latin Hypercube-based queries.
- Central limit theorem: This is a mathematical property of random sampling. It can also be found in Latin Hypercube sampling.
- The law of large numbers: This mathematical result states that the average of a sample approximates the mean of the underlying distribution. This concept is used for the center of gravity-based approach.

The mathematical results mentioned above can be found in any standard work on probability theory, such as [11]. The connection of the concepts and the algorithms are explained in more detail when describing the individual methods.

In total, we present six selection techniques for data exploration which will be explained in detail below. The first two techniques ("Spatial Random" and "Random Choice") query points at random as a baseline. For comparability, all algorithms start out with one random initially labelled point, as this is required for some of the algorithms.

Most techniques usually generate a "synthetic" point in the feature space which is not included in the underlying dataset. In this case, we query the closest existing point (via Euclidean distance). This idea was also applied in another active learning context in [12] under the name "query synthesis and nearest neighbor search".

In Section III, a novel modification of the algorithms is introduced taking into account the capabilities of the probabilistic learner.

Let us explain the query techniques in detail now.

A. Spatial Random

For this method, a random point within the hypercube enclosing the dataset is generated, uniformly distributed in the feature space. The method is therefore completely dataset-agnostic, only needing the boundaries of the feature space.

B. Random Choice

In this case, a random point from the pool of unlabelled data is chosen in each step. As the first method, this one is

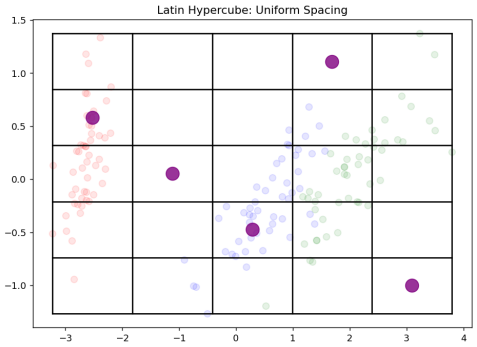


Fig. 4. Latin Hypercube Sampling with uniform division, on PCA-transformed Iris dataset

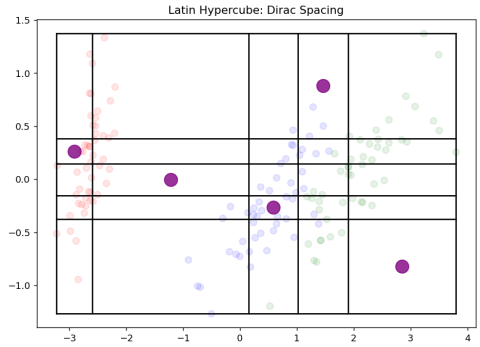


Fig. 5. Latin Hypercube Sampling with division by number of points, on PCA-transformed Iris dataset

completely random. However, since we can only query points directly and not generate some point within the feature space, this method takes into account the underlying structure of the data. From another point of view, this method is more likely than Spatial Random to sample a point from a densely populated region. Since we select a point of the underlying data directly, there is no need to query the nearest point. To keep the idea consistent with the remaining methods, one can think of it as randomly selecting a data point and querying the closest point, which is the point itself.

C. Balanced Exploration

This method aims to query points in a way that is balanced within the feature space. Namely, the query algorithm checks for each dimension whether there are more points closer to the upper bound or to the lower bound, and queries a random point within the opposite region, accordingly. The method therefore queries points from regions which are underrepresented. This method was first introduced in [4].

D. Latin Hypercube Sampling, uniform intervals

The Latin hypercube sampling (LHS) method aims to sample points uniformly in space. To this end, the space is divided into N^d hypercubes of uniform size, where d is the dimension of the feature space. A configuration of N samples is chosen in such a way that no two samples share a row or column. Figure 4 shows an example of Latin hypercube sampling with uniform division of intervals on the first two features of the PCA-transformed Iris dataset with $d = 2$ and $N = 5$.

Note that this method of sampling is data-agnostic, since the division of the feature space is done independently of the underlying data. This method, together with various variants, was used in [4] as well. The method actually has some mathematical foundations, as it exhibits similar features to random sampling, such as the central limit theorem. These statistical properties have been discussed in works such as [13]–[16].

E. Latin Hypercube Sampling, adjusted to the number of points ("LHS Dirac")

This novel approach takes into account the structure of the underlying data and combines it with the idea of Latin hypercube sampling. The structural information provided by the unlabelled dataset remains unused in the above implementation of LHS. In the novel approach presented here, the LHS approach is modified to incorporate the structure of the unlabelled dataset into the interval bounds. Namely, for each dimension, the space is divided up uniformly with respect to the number of points instead of uniformly in space. The idea here is to sample points via the Latin hypercube technique in a way that represents the data density within the feature space.

Note that to employ the hypercube exclusion principle used in this method, we are not able to divide the data uniformly simultaneously in all dimensions. Similarly to Figure 4, we illustrate this method on the PCA-transformed Iris dataset in Figure 5.

F. Center of gravity-based queries

One way to approach systematic exploration is to query points in regions which are less explored. To do so, starting from the center of the unlabelled data, we want to explore "away" from the already-labelled data, i.e.,

$$X_{\text{synth}} = C_{\mathcal{U}} + A(C_{\mathcal{U}} - C_{\mathcal{L}}) \tag{1}$$

where $A > 0$ and $C_{\mathcal{U}}$ and $C_{\mathcal{L}}$ denote the centers of the unlabelled and labelled data, respectively. Of course, one may ask how to choose the parameter A correctly. This is answered by considering a different idea. Recall from statistics that random sampling follows the law of large numbers, i.e., under certain assumptions, the average of a sample converges to the theoretical mean of its distribution:

$$\frac{1}{N}\sum_{i=1}^{N} X_i \xrightarrow{N\to\infty} \mathbb{E}X$$

where $X, X_1, X_2, \ldots$ are all independent samples following the same distribution. While we do not know the true distribution of the data, we can take the center of gravity (COG) of the whole dataset as an approximation of $\mathbb{E}X$. To this end, we

query the point which moves the COG of the labelled data as close to the COG of the whole dataset as possible. The idea here is to use the COG as an indicator of similarity between the labelled data and the whole dataset, so trying to match these centers would indicate that the labelled data is representative of the whole dataset.

Denote by $\mathcal{L} = \{x_1, \dots, x_{|\mathcal{L}|}\}$ and $\mathcal{U} = \{y_1, \dots, y_{|\mathcal{U}|}\}$ the set of labelled and unlabelled data points, respectively, where $|\mathcal{L}|$ and $|\mathcal{U}|$ denote the number of points in each of the sets. Furthermore, $C_\mathcal{L}$ and $C_\mathcal{U}$ are explicitly defined as

$$C_\mathcal{L} := \frac{1}{|\mathcal{L}|}\sum_{i=1}^{|\mathcal{L}|} x_i \text{ and } C_\mathcal{U} := \frac{1}{|\mathcal{U}|}\sum_{j=1}^{|\mathcal{U}|} y_j.$$

Denote by $\mathcal{D}$ the total dataset, i.e., $\mathcal{D} = \mathcal{L} \cup \mathcal{U}$ and

$$C_\mathcal{D} = \frac{1}{|\mathcal{D}|}\Big[\sum_{i=1}^{|\mathcal{L}|} x_i + \sum_{j=1}^{|\mathcal{U}|} y_j\Big].$$

Note that $|\mathcal{D}| = |\mathcal{L}| + |\mathcal{U}|$. We also consider the centers after querying a new datapoint $q \in \mathcal{U}$:

$$C_{\mathcal{L}\cup\{q\}} = \frac{C_\mathcal{L}|\mathcal{L}| + q}{|\mathcal{L}| + 1} \text{ and } C_{\mathcal{U}\setminus\{q\}} = \frac{C_\mathcal{U}|\mathcal{U}| - q}{|\mathcal{U}| - 1}$$

We want to find the unlabelled point which - once added to the labelled set - would move the center of the labelled data as close as possible to the center of the whole dataset. In other words, our goal is to query the point $q \in \mathcal{U}$ such that

$$\begin{aligned} q &= \operatorname*{argmin}_{q\in\mathcal{U}} |C_{\mathcal{L}\cup\{q\}} - C_\mathcal{D}| \\ &= \operatorname*{argmin}_{q\in\mathcal{U}} \Big|\frac{C_\mathcal{L}|\mathcal{L}| + q}{|\mathcal{L}| + 1} - \frac{C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}|}{|\mathcal{L}| + |\mathcal{U}|}\Big| \\ &= \operatorname*{argmin}_{q\in\mathcal{U}} \Big|\frac{C_\mathcal{L}|\mathcal{L}|}{|\mathcal{L}| + 1} - \frac{C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}|}{|\mathcal{L}| + |\mathcal{U}|} + q\cdot\frac{1}{|\mathcal{L}| + 1}\Big| \end{aligned}$$

where the outer $|\cdot|$ denotes the Euclidean distance. Note that the argmin stays invariant if we multiply the whole expression by the positive number $|\mathcal{L}| + 1$:

$$\begin{aligned} q &= \operatorname*{argmin}_{q\in\mathcal{U}} |C_{\mathcal{L}\cup\{q\}} - C_\mathcal{D}| \\ &= \operatorname*{argmin}_{q\in\mathcal{U}} \Big|C_\mathcal{L}|\mathcal{L}| - \frac{|\mathcal{L}| + 1}{|\mathcal{L}| + |\mathcal{U}|}\big(C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}|\big) + q\Big| \\ &= \operatorname*{argmin}_{q\in\mathcal{U}} \Big|\underbrace{\frac{|\mathcal{L}| + 1}{|\mathcal{L}| + |\mathcal{U}|}\big(C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}|\big) - C_\mathcal{L}|\mathcal{L}|}_{(**)} - q\Big| \end{aligned}$$

where in the last step, we multiplied the part within the absolute value by (-1) (keeping its value equal).

Looking at the final expression, our goal is to find the point $q \in \mathcal{U}$ which is closest to the vector $(**)$. Let us further examine this expression:

$$\begin{aligned} (**) &= \frac{|\mathcal{L}| + 1}{|\mathcal{D}|}\big(C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}|\big) - \frac{|\mathcal{L}|(|\mathcal{L}| + |\mathcal{U}|)}{|\mathcal{D}|}C_\mathcal{L} \\ &= \frac{1}{|\mathcal{D}|}\big(C_\mathcal{L}|\mathcal{L}| + C_\mathcal{U}|\mathcal{U}| + |\mathcal{U}||\mathcal{L}|(C_\mathcal{U} - C_\mathcal{L})\big) \\ &= C_\mathcal{D} + \frac{|\mathcal{U}||\mathcal{L}|}{|\mathcal{D}|}(C_\mathcal{U} - C_\mathcal{L}) \end{aligned}$$

In an active learning situation, we usually have $|\mathcal{U}| \gg |\mathcal{L}|$ and hence, $|\mathcal{D}| = |\mathcal{U}| + |\mathcal{L}| \approx |\mathcal{U}|$ and $C_\mathcal{U} \approx C_\mathcal{D}$. Using these approximations above, we get

$$\begin{aligned} (**) &= C_\mathcal{D} + \frac{|\mathcal{U}||\mathcal{L}|}{|\mathcal{U}| + |\mathcal{L}|}(C_\mathcal{U} - C_\mathcal{L}) \\ &\approx C_\mathcal{U} + \frac{|\mathcal{U}||\mathcal{L}|}{|\mathcal{U}|}(C_\mathcal{U} - C_\mathcal{L}) \\ &= C_\mathcal{U} + |\mathcal{L}|(C_\mathcal{U} - C_\mathcal{L}) \end{aligned}$$

Comparing this term with the construction of the synthetic point (1), we see that the expressions coincide when choosing $A = |\mathcal{L}|$. Hence, we see that querying away from the labelled data, towards the unlabelled data is achieving a similar task as querying with the objective of aligning the centers of gravity.

Remark 1:

1) In $(**)$, we replace $C_\mathcal{D}$ by $C_\mathcal{U}$. Both are valid as an interpretation of exploration. The former ($C_\mathcal{D}$) means we start in the center of the whole data and move towards the higher mass concentration of the unlabelled set, the latter ($C_\mathcal{U}$) means we already start in the middle of the unlabelled set and move even further away from the center of the labelled set. Both interpretations make sense, and in an AL situation, the centers are usually close anyway, as argued above.
2) A similar calculation to the one above also shows that the query minimizing the distance between $C_{\mathcal{L}\cup\{q\}}$ and $C_\mathcal{D}$ also minimizes the distance between $C_{\mathcal{L}\cup\{q\}}$ and $C_{\mathcal{U}\setminus\{q\}}$, i.e., the query minimizes the distance between the COG of the labelled and unlabelled data as well.

III. Fake class-synthetic query

In the pool-based scenario, an expert can only label data points which are actually present in the real dataset. Therefore, instead of asking for the label of the point that the learner generated, the algorithms above query the closest point available in the unlabelled pool. This may not always be the optimal query, though, as the closest point may e.g. already be in a well-explored region. However, the learner itself is able to process this synthetic point directly. Since we are also concerned with class detection, we want to "tell" the learner that we suspect an undiscovered class at the synthetic point. The learner shall then use this information to query the (real) point that it deems most likely to belong to an undiscovered class. The learner can process the synthetic point just fine, while we end up with a real data point to be labelled.

The way to accomplish this is as follows: The synthetic point is temporarily added to the training set. Assuming the "real" classes are given by $y = 0, 1, 2, \ldots$, the synthetic point is assigned class -1. A probabilistic model is then trained, and the point with the highest probability belonging to this class is queried:

$$q = \operatorname*{argmax}_{x \in \mathcal{U}} \mathbb{P}(y = -1|x)$$

The point q is queried, labelled by an expert and added to the training set with its true label. The synthetic point is discarded and the process is repeated if desired. The query method is listed in more detail in Algorithm 1.

In summary, the idea of the algorithm is to answer the question, "If there was an undiscovered class, which point is most probable to belong to it?"

Since most of the algorithms introduced in the section above generate synthetic points anyway, these algorithms are predestined to be used for this technique. For the random choice exploration, we may treat the chosen point as a synthetic point, also enabling the fake class procedure.

Algorithm 1 Query a point based on a synthetic fake class-point.

Input: $\mathcal{L} = \{(x_1, y_1), \ldots, (x_{|\mathcal{L}|}, y_{|\mathcal{L}|})\}$ with $y_i \in \mathbb{N}_0$ and $\mathcal{U} = \{x_1, \ldots, x_{|\mathcal{U}|}\}$, point generator $f(\mathcal{L}, \mathcal{U})$, classifier C

Output: Point $X \in \mathcal{U}$ to be queried

$X_{\text{synth}} \leftarrow f(\mathcal{L}, \mathcal{U})$
$y_{\text{synth}} \leftarrow -1$
Train probabilistic classifier C on $(\mathcal{L} \cup \{(X_{\text{synth}}, y_{\text{synth}})\})$
$X \leftarrow \operatorname{argmax}_{X \in \mathcal{U}} \mathbb{P}(C(X) = -1)$
return X

Remark 2: One may wonder if this method selects the closest point to a generated (synthetic) point, just as the nearest neighbour search as described above. However, the probabilistic learner may also choose a different point. Imagine a situation where the synthetic point is close to a cluster of labelled points sharing one class, let's say y_0. Then any unlabelled points in this area will also belong to the class y_0 with high probability, even though our synthetic point with fake class is close. This means that the algorithm will not query points which are already (relatively) safely classified as one of the known classes. The hope is that the learner benefits from this technique as it prevents oversampling of regions.

Remark 3: The fake class technique also works in feature spaces without a distance function. Furthermore, it enables membership query synthesis techniques to be used in a pool-based scenario.

IV. Results and Discussion

Let us now compare the performance of the algorithms introduced above. Each algorithm was tested using nearest neighbour query and fake class query. The results are displayed in the tables. The first column presents the query budget of the learner, i.e., the number of points that the algorithm was allowed to query. The other columns represent the various exploration algorithms and their accuracy or discovery rate, respectively. While we don't show both accuracy and discovery rate for all datasets, note that there is some correlation between the two, as the number of unknown classes obviously limits the accuracy theoretically possible.

TABLE I
Binary classes, boundary B_1, nearest point query, accuracy

	SR	RC	BE	LHS U	LHS D	COG
5	0.654	0.704	0.676	0.720	0.739	**0.765**
10	0.767	0.787	0.776	0.797	**0.806**	0.758
15	0.809	0.825	0.827	0.834	**0.835**	0.769
20	0.838	0.849	0.852	0.862	**0.863**	0.774
25	0.862	0.872	0.870	0.877	**0.878**	0.789
30	0.875	**0.889**	0.880	0.887	0.887	0.804

A. Synthetic datasets

Let us present the results for the synthetic datasets. For the binary datasets with balanced class distribution, the class discovery record is not very meaningful, as both classes were detected by all algorithms on a small query budget. Hence, the analysis presented here focuses on the accurcy of the algorithms. For the artificial datasets, we only display the data up to a budget of 30 data points, as all of them except for the COG-based methods show similar performances.

For the unbalanced dataset, we show the discovery rates, as these are more meaningful than accuracy here, and extend the query budget to 60. One sees that the discovery performance is quite similar for all datasets, except for the COG-based query, which in most cases was not able to discover the minority class. Adding the fake class-method, all performances are improved, and even the COG-query is able to discover the minority class, given a sufficiently high query budget.

Note that in general, all algorithms except COG show similar performance on each of the three datasets shown in Fig. 1a, 1b and 1c. A possible explanation is that the points are uniformly distributed in the feature space, and there is no structural difference between the classes that could be used by the exploration algorithms.

On the two moon datasets (with and without noise), all algorithms except the COG-based approach performed similarly, while COG showed much worse accuracy. However, when combining the algorithms with the fake class-method, this difference in performance was equalized.

B. Real world datasets

Here, we discuss the performance on the beans, yeast and glass identification datasets. Since the novel approaches did not show significant differences or no improvement over the well-known methods in terms of accuracy, the accuracy results are not explicitly presented here.

Comparing the algorithms without the fake class method, one sees that the success of class discovery is highly dependent

TABLE II
BINARY CLASSES, BOUNDARY B_1, FAKE CLASS QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
5	0.683	0.671	0.691	0.715	**0.722**	0.712
10	0.782	0.778	0.778	0.786	**0.794**	0.779
15	0.827	0.824	**0.840**	0.823	0.829	0.794
20	0.854	0.845	**0.866**	0.859	0.855	0.808
25	0.867	0.870	0.876	0.875	**0.878**	0.821
30	0.881	0.885	0.881	0.885	**0.889**	0.823

TABLE III
BINARY CLASSES, BOUNDARY B_2, NEAREST POINT QUERY, DISCOVERY RATE

	SR	RC	BE	LHS U	LHS D	COG
5	0.420	0.520	**0.620**	**0.620**	**0.620**	0.320
10	0.780	0.720	**0.800**	0.780	0.760	0.320
15	0.880	0.780	0.940	**0.960**	**0.960**	0.320
20	0.920	0.860	0.960	**0.980**	**0.980**	0.320
25	0.960	0.920	0.960	**1.000**	**1.000**	0.320
30	0.980	0.960	0.960	**1.000**	**1.000**	0.320
40	**1.000**	**1.000**	0.980	**1.000**	**1.000**	0.320
50	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.320
60	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.320

TABLE IV
BINARY CLASSES, BOUNDARY B_2, FAKE CLASS QUERY, DISCOVERY RATE

	SR	RC	BE	LHS U	LHS D	COG
5	0.480	0.520	0.500	**0.600**	0.580	0.340
10	**0.840**	0.740	0.820	0.760	0.780	0.340
15	0.920	0.800	0.880	0.960	**0.980**	0.460
20	0.960	0.900	0.980	**1.000**	**1.000**	0.580
25	0.960	0.920	0.980	**1.000**	**1.000**	0.640
30	0.980	0.980	**1.000**	**1.000**	**1.000**	0.800
40	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.920
50	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.980
60	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

TABLE V
BINARY CLASSES, CHECKERBOARD, NEAREST POINT QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
5	0.519	0.522	0.534	0.531	**0.540**	**0.540**
10	0.595	0.621	**0.626**	0.618	0.623	0.593
15	0.650	0.697	0.689	**0.701**	0.691	0.657
20	0.704	0.744	0.730	0.750	0.751	**0.767**
25	0.753	0.783	0.772	0.771	0.774	**0.823**
30	0.790	0.816	0.801	0.800	0.803	**0.853**

on the dataset. For instance, the COG-method falls behind on either the beans and the yeast dataset, while it discovers all classes present in the glass dataset, which is highly

TABLE VI
BINARY CLASSES, CHECKERBOARD, FAKE CLASS QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
5	0.527	0.529	0.522	0.540	0.539	**0.543**
10	0.614	0.605	0.604	0.621	0.608	**0.655**
15	0.665	0.678	0.658	0.692	0.698	**0.727**
20	0.722	0.737	0.729	0.749	0.743	**0.775**
25	0.770	0.772	0.768	0.784	0.768	**0.829**
30	0.799	0.811	0.804	0.808	0.813	**0.873**

TABLE VII
TWO MOON DATASET WITHOUT NOISE, NEAREST POINT QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
10	0.824	0.834	0.858	0.864	**0.870**	0.672
20	0.901	0.890	0.902	**0.955**	0.939	0.687
30	0.935	0.919	0.923	**0.976**	0.968	0.688
40	0.968	0.927	0.938	**0.976**	0.965	0.674
50	0.982	0.942	0.943	**0.989**	0.971	0.673

TABLE VIII
TWO MOON DATASET WITHOUT NOISE, FAKE CLASS QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
10	0.875	0.837	0.825	**0.886**	0.881	0.787
20	0.929	0.888	0.887	**0.948**	0.941	0.884
30	0.954	0.915	0.912	**0.961**	0.958	0.944
40	0.971	0.927	0.929	0.961	0.958	**0.972**
50	0.972	0.943	0.944	0.972	0.961	**0.976**

TABLE IX
NOISY MOONS DATASET ($\sigma = 0.2$), NEAREST POINT QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
10	0.781	0.801	0.802	0.809	**0.828**	0.514
20	0.852	0.851	0.855	0.876	**0.883**	0.526
30	0.881	0.873	0.878	0.886	**0.915**	0.552
40	0.896	0.890	0.895	0.891	**0.918**	0.650
50	0.905	0.899	0.903	0.906	**0.930**	0.684

TABLE X
NOISY MOONS DATASET ($\sigma = 0.2$), FAKE CLASS QUERY, ACCURACY

	SR	RC	BE	LHS U	LHS D	COG
10	0.784	0.798	0.793	0.819	**0.837**	0.646
20	0.858	0.845	0.852	0.879	**0.893**	0.717
30	0.884	0.874	0.879	0.886	**0.919**	0.793
40	0.904	0.895	0.891	0.897	**0.920**	0.855
50	0.913	0.902	0.902	0.906	**0.924**	0.888

TABLE XI
BEANS DATASET, NEAREST POINT QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.350	0.677	**0.703**	0.460	0.637	0.567
15	0.410	**0.833**	0.823	0.527	0.693	0.617
20	0.447	**0.893**	0.877	0.597	0.720	0.630
25	0.497	**0.927**	0.913	0.613	0.733	0.630
30	0.507	**0.950**	0.940	0.637	0.890	0.630
35	0.557	**0.967**	0.950	0.680	0.883	0.630
40	0.593	**0.973**	0.957	0.710	0.863	0.630
45	0.623	**0.983**	0.963	0.703	0.870	0.630
50	0.637	**0.990**	0.970	0.727	0.887	0.630

TABLE XII
BEANS DATASET, FAKE CLASS QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.683	0.660	0.740	0.723	**0.820**	0.773
15	0.807	0.823	0.850	0.843	**0.920**	0.880
20	0.887	0.903	0.897	0.877	**0.980**	0.943
25	0.917	0.940	0.933	0.910	0.973	**0.977**
30	0.963	0.963	0.947	0.963	**0.997**	0.993
35	**1.000**	0.983	0.957	0.970	0.997	**1.000**
40	**1.000**	0.990	0.963	0.983	**1.000**	**1.000**
45	**1.000**	0.997	0.970	0.997	**1.000**	**1.000**
50	**1.000**	0.997	0.977	0.997	**1.000**	**1.000**

TABLE XIII
YEAST DATASET, NEAREST POINT QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.460	0.400	0.398	**0.496**	0.420	0.262
15	0.542	0.493	0.487	**0.580**	0.484	0.298
20	0.604	0.549	0.551	**0.647**	0.518	0.331
25	0.673	0.589	0.609	**0.678**	0.553	0.367
30	**0.702**	0.618	0.629	0.698	0.558	0.418
40	0.744	0.682	0.682	**0.762**	0.616	0.476
50	0.769	0.744	0.729	**0.791**	0.638	0.513

TABLE XIV
YEAST DATASET, FAKE CLASS QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.576	0.396	0.384	**0.591**	0.478	0.438
15	0.689	0.498	0.469	**0.713**	0.560	0.489
20	0.747	0.571	0.536	**0.771**	0.616	0.524
25	0.778	0.633	0.582	**0.807**	0.684	0.571
30	0.813	0.673	0.611	**0.824**	0.713	0.604
40	0.858	0.742	0.664	**0.869**	0.764	0.653
50	0.893	0.789	0.718	**0.896**	0.813	0.696

TABLE XV
GLASS DATASET, NEAREST POINT QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.672	0.632	0.648	0.572	0.696	**0.764**
15	0.780	0.728	0.772	0.680	0.772	**0.908**
20	0.844	0.836	0.844	0.728	0.876	**0.968**
25	0.892	0.888	0.888	0.768	0.904	**0.992**
30	0.924	0.920	0.928	0.824	0.940	**0.996**
35	0.952	0.952	0.952	0.856	0.948	**1.000**
40	0.968	0.964	0.960	0.884	0.972	**1.000**
45	0.976	0.972	0.964	0.896	0.980	**1.000**
50	0.984	0.976	0.976	0.920	0.972	**1.000**

TABLE XVI
GLASS DATASET, FAKE CLASS QUERY, DISCOVERY RATES

	SR	RC	BE	LHS U	LHS D	COG
10	0.724	0.588	0.632	0.744	**0.756**	0.728
15	0.828	0.704	0.760	0.840	**0.856**	0.832
20	0.896	0.812	0.836	0.900	0.904	**0.920**
25	0.936	0.876	0.880	**0.964**	0.936	0.952
30	0.956	0.924	0.920	**0.976**	0.964	**0.976**
35	0.976	0.968	0.944	0.972	**0.984**	**0.984**
40	0.992	0.976	0.952	**0.996**	0.984	0.988
45	0.992	0.988	0.956	0.996	**1.000**	0.988
50	0.996	0.988	0.972	**1.000**	0.996	0.988

imbalanced. Also note that structurally agnostic algorithms, namely, spatial random query and LHS uniform, show similar performances.

Using the fake class method, we see an improvement of performance in most situations. Wherever this method is not beneficial, the discrepancy between the nearest point-query and the fake class query is minimal.

The LHS-based algorithms occasionally show performance drops when increasing the query budget. This is because the points could not be queried sequentially, meaning that each query budget may use a completely different set of data points.

V. CONCLUSION AND OUTLOOK

The presented algorithms show promising results by outperforming random search on some datsets. Combined with the fake class-technique, all algorithms showed at least similar performance to random search, often outperforming it.

The performance was highly dependent on the underlying dataset. Also, one can notice a dependence on underlying properties of the exploration algorithms, such as structural sensitivity. Therefore, the following ideas may be pursued in future research:

- Examine if performance depends on specific dataset properties, such as imbalances. To choose an optimal exploration algorithm, it is necessary to find criteria inherent to the dataset. The idea here is that in an active

learning situation, we should use the structure of the unlabelled data to optimally select the data points for labelling.

- Adding to the first point: Explore the mathematical background of the exploration queries, especially the "fake class"-technique, to gain a better understanding for the performance of each algorithm. This may help understanding and predicting in which situation each algorithm performs well or not and help in designing improved algorithms.
- Check performance in combination with exploitation algorithms: As the goal of exploration is not purely classification accuracy but preparation for the exploitation phase, we need to test the algorithms coupled with exploitation algorithms as already done in some cases in [4], [17]. The goal is to find the exploration algorithms which produce a good set of labelled points for the exploitation queries to build on for improved accuracy.
- Obtain new performance measures: Ideally, there should be a performance measure which takes an exploration algorithm and predicts the performance of a coupled exploration-exploitation-algorithm for reasons explained in the previous point. This would serve as a performance measure for exploration algorithms in general, but also would make benchmarking such algorithms more efficient, as one wouldn't need to run the (computationally expensive) exploitation algorithm to record the total performance.

REFERENCES

[1] K. J. Lang and E. B. Baum, "Query Learning Can Work Poorly when a Human Oracle is Used," *Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD, USA*, vol. 8, no. 5, 1992.

[2] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009. [Online]. Available: https://burrsettles.com/pub/settles.activelearning.pdf

[3] A. Tharwat and W. Schenck, "A survey on active learning: State-of-the-art, practical challenges and research directions," *Mathematics*, vol. 11, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/4/820

[4] ——, "Balancing exploration and exploitation: A novel active learner for imbalanced data," *Knowledge-Based Systems*, vol. 210, p. 106500, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120306298

[5] M. Sharma and M. Bilgic, "Evidence-based uncertainty sampling for active learning," *Data Mining and Knowledge Discovery*, vol. 31, pp. 164–202, 2017.

[6] "Dry Bean Dataset," UCI Machine Learning Repository, 2020, DOI: https://doi.org/10.24432/C50S4B.

[7] K. Nakai, "Yeast," UCI Machine Learning Repository, 1996, DOI: https://doi.org/10.24432/C5KG68.

[8] B. German, "Glass Identification," UCI Machine Learning Repository, 1987, DOI: https://doi.org/10.24432/C5WW2P.

[9] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 1936–1949, 2014.

[10] R. A. Fisher, "Iris," UCI Machine Learning Repository, 1988, DOI: https://doi.org/10.24432/C56C76.

[11] H. Bauer, *Probability Theory*. Berlin, New York: De Gruyter, 1996. [Online]. Available: https://doi.org/10.1515/9783110814668

[12] L. Wang, X. Hu, B. Yuan, and J. Lu, "Active learning via query synthesis and nearest neighbour search," *Neurocomputing*, vol. 147, pp. 426–434, 2015, advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231214008145

[13] J.-S. Park, "Optimal latin-hypercube designs for computer experiments," *Journal of Statistical Planning and Inference*, vol. 39, no. 1, pp. 95–111, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0378375894901155

[14] A. B. Owen, "A central limit theorem for latin hypercube sampling," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 54, no. 2, pp. 541–551, 1992. [Online]. Available: http://www.jstor.org/stable/2346140

[15] W.-L. Loh, "On Latin hypercube sampling," *The Annals of Statistics*, vol. 24, no. 5, pp. 2058 – 2080, 1996. [Online]. Available: https://doi.org/10.1214/aos/1069362310

[16] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987. [Online]. Available: http://www.jstor.org/stable/1269769

[17] A. Tharwat and W. Schenck, "A novel low-query-budget active learner with pseudo-labels for imbalanced data," *Mathematics*, vol. 10, no. 7, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/7/1068