Active Learning for Regression by Inverse Distance Weighting

Alberto Bemporad[1]

Abstract

This paper proposes an active learning (AL) algorithm to solve regression problems based on inverse-distance weighting functions for selecting the feature vectors to query. The algorithm has the following features: (*i*) supports both pool-based and population-based sampling; (*ii*) is not tailored to a particular class of predictors; (*iii*) can handle known and unknown constraints on the queryable feature vectors; and (*iv*) can run either sequentially, or in batch mode, depending on how often the predictor is retrained. The potentials of the method are shown in numerical tests on illustrative synthetic problems and real-world datasets. An implementation of the algorithm, which we call IDEAL (Inverse-Distance based Exploration for Active Learning), is available at `http://cse.lab.imtlucca.it/~bemporad/ideal`.

Keywords: Active learning (AL), inverse distance weighting, pool-based sampling, query synthesis, supervised learning, regression, neural networks.

1. Introduction

Active learning (AL) strategies are used in supervised learning to let the training algorithm "ask questions" [34], i.e., choose the feature vectors to query for the corresponding target value during the training phase, usually based on the model learned so far. The main aim of AL is to possibly reduce the number of training samples required to train the model, or in other words, to get a model of the same prediction quality with a smaller dataset. This is particularly useful when knowing the target value associated with a given combination of features is an expensive operation, for example, it may involve asking a human to "label" samples manually, running a costly and time-consuming laboratory experiment, or performing a complex computer simulation.

[1]IMT School for Advanced Studies Lucca, Piazza San Francesco, 19, 55100 Lucca, Italy. Email: `alberto.bemporad@imtlucca.it`

AL methods are usually categorized in *query synthesis* (or *population-based*) methods, in which the feature vector to query can be chosen arbitrarily, *pool-based* sampling methods, in which the vector can only be chosen within a given finite set (or "pool") of unlabeled values, and *selective-sampling* methods, in which vectors are proposed in a streaming flow and the AL algorithm can only decide online whether to ask for the corresponding target or not [34].

Several approaches to AL are available in the literature, see, e.g., the survey papers [34, 39, 16, 1, 22]. Most of the literature focuses on classification problems [1, 33], although AL has been investigated also for regression [27, 11, 38, 13, 12, 10, 9, 41, 42, 25]. We will describe in detail some of the most popular AL algorithms for regression in Section 3.4. For a detailed and updated taxonomy of AL methods for classification, regression, and clustering we refer the reader to the recent survey paper [22].

As pointed out in [41], AL methods should collect data that are *informative*, *representative*, and *diverse*, i.e., respectively, contain rich information for reducing modeling errors, cover portions of the feature vector space where the predictor is evaluated most frequently and in particular reject outliers, and explore such a space trying to avoid sampling the same regions too often. AL methods are often linked to a specific class of predictors, such as neural networks [27] or mixtures of Gaussians and locally weighted regression [11], or to a particular learning algorithm [38, 13, 12, 9]. Moreover, AL methods can be computationally involved in the case optimal sampling is sought, or in query-by-committee (QBC) methods [35, 31, 8] in which multiple predictors need to be retrained repeatedly to measure their disagreement.

In general, in AL the *acquisition function* that is used to drive the selection of the next sample has two components. The first is related to the position of the feature vector within the feature-vector space and is used for pure *exploration* of that space. The second aims at the *exploitation* of the target values acquired so far, learning a model on the available feature vectors/target pairs and using it for predicting target values. Such a model-based approach usually tries to estimate a form or another of target uncertainty, such as to locate feature vectors whose target is supposed to be farthest from the target values already acquired [42], sample where a committee of predictors mostly disagree [35, 31, 8], or select the feature vector that is expected to make the most change in the prediction function [9].

AL is related to the problem of optimally designing experiments, whose origins date back at least to the 30s [15], and has attracted an extensive literature for decades [7]. Another problem related to AL is black-box derivative-free optimization [32] in which a surrogate of the objective function is learned incrementally from a finite number of samples of it, such as in Bayesian optimization

methods [36]. Compared to solving a supervised learning problem, where the objective is to find a model that reproduces well the underlying process over the entire set of feature-vectors of interest, in black-box optimization the problem is somehow simpler, as the interest is limited to approximating the objective function well around one of its global minimizers.

1.1. Contribution

In this paper, we provide an AL framework for regression that is applicable to any prediction model, can address both pool-based and population-based settings, and is not computationally involved. By leveraging on ideas we previously investigated for global optimization based on surrogate functions [3, 5], we propose an AL method in which the uncertainty associated with the currently available predictor and the exploration function used to sample the feature-vector space are characterized by *inverse-distance weighting* (IDW) functions [23, 37].

The proposed algorithm that we call IDEAL (Inverse-Distance based Exploration for Active Learning) blends different requirements: informativeness, by sampling regions of the feature-vector space where model uncertainty is estimated to be large; representativeness, in the case of pool-based sampling, by possibly taking into account a density function similar to the one used in density-based spatial clustering approaches [14]; and diversity, using an IDW exploration term that is higher far away from samples that have already been queried. The algorithm can also handle constraints on the feature vectors that can be queried, that can either be known a priori or even *unknown*. The latter case covers the situation in which one discovers only after querying certain combinations of features that the corresponding target cannot be retrieved; for example, because a specific physical experiment cannot be performed or a computer simulation does not converge. Finally, the proposed algorithm can be run either sequentially, by retraining the predictor after each successful query, or in batch mode, by retraining only after querying a certain prescribed finite number of samples.

IDEAL belongs to the class of model-based AL methods for regression, in that the prediction model learned on the currently available samples is used in combination with IDW terms to quantify model uncertainty and look for samples that are expected to provide maximum informativeness. Moreover, the latter only requires the currently learned predictor, contrarily to QBC methods that require instead training multiple predictors, and does not involve complex computations required by optimal sampling methods, which makes them only applicable to relatively simple prediction models.

Similarly to the greedy sampling method proposed in [42], which combines

diversity in the feature vector and (predicted) target spaces, IDEAL combines the informativeness measure mentioned above with the diversity quantification in the feature-vector space provided by IDW terms only. As we will show in several numerical examples, such a combination of model-based uncertainty characterization and feature-vector diversity is beneficial with respect to uncertainty characterization only, as in QBC methods [31, 8], and input diversity only, as in the greedy method [43, Algorithm 1] and the improved representativeness-diversity maximization method [25].

The paper is organized as follows. After formulating the AL problem in Section 2, we describe the proposed algorithm in Section 3. Numerical tests on synthetic and real-world regression problems are reported in Section 4 and some conclusions are drawn in Section 5.

A Python implementation of IDEAL, and of other passive and active learning methods we have compared with, is available at `http://cse.lab.imtlucca.it/~bemporad/ideal`.

2. Active learning problem

We consider a process $y : \mathcal{X} \to \mathcal{Y}$ generating data $y_k = y(x_k)$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of feature vectors, $x_k \in \mathcal{X}$, and $\mathcal{Y} \subseteq \mathbb{R}^m$ the set of corresponding targets y_k, $y_k \in \mathcal{Y}$. As the process y is unknown, we wish to find a predictor $\hat{y} : \mathcal{X} \to \mathcal{Y}$ solving the supervised learning problem

$$\min_{\hat{y}} \int_{\bar{\mathcal{X}} \cap \mathcal{X}} \ell(y(x), \hat{y}(x), x) dx \tag{1}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$ is a loss function, for instance $\ell(y(x), \hat{y}(x), x) = \|y(x) - \hat{y}(x)\|_2^2$, and $\bar{\mathcal{X}} \subseteq \mathbb{R}^n$ is a bounded set of feature vectors x of interest, i.e., for which we want to obtain a good approximation $\hat{y}(x)$ of $y(x)$. While the set $\bar{\mathcal{X}}$ is known, for example, it may be defined by the set of inequality constraints

$$\bar{\mathcal{X}} = \{x : \mathbb{R}^n : \ g_i(x) \leq 0, \ i = 1, \ldots, n_c\}$$

$g_i : \mathbb{R}^n \to \mathbb{R}$, the set $\mathcal{X}$ for which $y(x)$ is defined could be *unknown*, as we might not be able to know a priori whether for a given $x \in \bar{\mathcal{X}}$ its corresponding target $y(x)$ can be obtained. For example, all features x_i of interest may take any value between -10 and 10, i.e., $\bar{\mathcal{X}} = \{x : \mathbb{R}^n : \ |x_i| \leq 10, \ i = 1, \ldots, n\}$ and $y(x) = \log(x)$, which is only defined for $x > 0$. In this case, $y(x)$ cannot be queried when $x \leq 0$, i.e., $\mathcal{X} = \{x : x_i > 0, \ i = 1, \ldots, n\}$ and we are in the presence of the unknown constraint $x \in \mathcal{X}$.

In practical real-world applications, unknown constraints may arise when evaluating $y(x)$ may require running a complex experiment or computer simulation, and this could not be completed for various reasons for the particular

parameter settings defined by x. In such cases, characterizing the shape of $\mathcal{X}$, if of interest, would be a binary classification problem itself that is amenable for active learning. Note that in the case of multiple targets ($m > 1$), we could generalize the setting by assuming that each process component $[y]_i : \mathcal{X}_i \to \mathcal{Y}_i$, $i = 1, \ldots, m$. However, for simplicity of notation, we assume here that $\mathcal{X} = \cap_{i=1}^{m} \mathcal{X}_i$, i.e., that either the entire output vector $y(x)$ is defined or it is entirely undefined at a given x.

Special cases of (1) are (multivariate) regression problems ($\mathcal{Y} = \mathbb{R}^m$) and classification problems ($\mathcal{Y} = \{0, 1\}^m$). We assume that possible discrete features have been one-hot encoded, and that hence in general $\mathcal{X} \subseteq \{0, 1\}^{n_b} \times \mathbb{R}^{n_n}$, where n_b and n_n are the number of binary and numeric features, respectively, $n = n_b + n_n$, and that the loss ℓ contains impulsive terms (Dirac delta terms) so that (1) can be rewritten as

$$\min_{\hat{y}} \sum_{x_b \in \mathcal{X}_b \cap \bar{\mathcal{X}}_b} \int_{\mathcal{X}_c \cap \bar{\mathcal{X}}_c} \ell(y(x), \hat{y}(x), x_c, x_b) dx_c \tag{2}$$

where x_b denotes the subvector of binary components of the feature vector x, $\mathcal{X}_b$ ($\bar{\mathcal{X}}_b$) the corresponding set of their admissible combinations (of interest), and $\mathcal{X}_c$ ($\bar{\mathcal{X}}_c$) the set of admissible subvectors x_c of numeric features (of interest).

In order to address problem (1), we will solve its empirical approximation

$$\min_{\hat{y}} \frac{1}{N} \sum_{k=1}^{N} \ell(y_k, \hat{y}(x_k), x_k) \tag{3}$$

where $D_N \triangleq \{(x_k, y_k)\}_{k=1}^{N}$ is a training dataset, with $y_k = y(x_k)$ for some unknown function y[2].

In (supervised) *passive* learning the training dataset D_N is given, where clearly $x_k \in \mathcal{X}$ for all $k = 1, \ldots, N$, as the corresponding targets y_k have been acquired. Instead, in *active* learning we are free to select the training vectors x_k to *query*, i.e., for which we want to get the corresponding target value y_k, if it is defined, or a declaration that $x_k \notin \mathcal{X}$. We have a *pool-based* AL problem when x_k can only be selected from a pool

$$\mathcal{X}_P = \{\bar{x}_j\}_{j=1}^{M} \tag{4}$$

of samples, $M \geq N$, with $\mathcal{X}_P \subseteq \bar{\mathcal{X}}$, or a *population-based* AL problem when x_k can be chosen freely within the given bounded set $\mathcal{X}_P = \bar{\mathcal{X}}$.

[2]Although function y is rather arbitrary, the formulation could be extended to explicitly include a noise term $\eta_k \in \mathbb{R}^{n_\eta}$, so that $y_k = y(x_k, \eta_k)$ is available rather than $y(x_k)$. This would allow modeling non-reproducible queries, i.e., $y_k \neq y_j$ for $x_k = x_j$, $k \neq j$.

3. Active learning algorithm

Let $[x_{\min}, x_{\max}] \subset \mathbb{R}^n$ be the smallest hyper-box containing the feature vectors we are allowed to sample, i.e.,

$$[x_{\min}]_i \triangleq \min_{x \in \mathcal{X}_P} [x]_i, \;\; [x_{\max}]_i \triangleq \max_{x \in \mathcal{X}_P} [x]_i \tag{5a}$$

which in case of pool-based AL (4) is equivalent to setting

$$[x_{\min}]_i \triangleq \min_{j=1,\dots,M} [\bar{x}]_j, \;\; [x_{\max}]_i \triangleq \max_{j=1,\dots,M} [\bar{x}]_j \tag{5b}$$

In order to be immune to different scaling of the individual features, when querying samples we consider the scaling function $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ defined as

$$\sigma_i(x) \triangleq \frac{2}{[x_{\max}]_i - [x_{\min}]_i} \left(x_i - \frac{[x_{\max}]_i + [x_{\min}]_i}{2} \right), \;\; i = 1, \dots, n \tag{5c}$$

where clearly $\sigma(x) \in [-1, 1]^n$ for all $x \in [x_{\min}, x_{\max}]$.

Let $N_{\max}$ be the total budget of queries we have available to perform the AL task. During AL, we collect in the set[3] $\mathcal{Q} \subseteq \{1, \dots, N_{\max}\}$ the indices of the samples x_k that have been selected and for which the corresponding target could be acquired, i.e., $k \in \mathcal{Q}$ if and only if $x_k \in \mathcal{X}$. Moreover, in the case of pool-based sampling, we keep track of the indices of samples already extracted and queried from the pool $\mathcal{X}_P$ in the set $\mathcal{E} \subseteq \{1, \dots, M\}$, to avoid possibly querying them again.

3.1. Initialization

Before fitting any prediction model, as commonly done in most AL approaches we must first select N_i samples $x_1, \dots, x_{N_i} \in \bar{\mathcal{X}} \cap \mathcal{X}$. As also mentioned in [25, Section 4.3], unsupervised AL (i.e., AL that only selects samples based on their position within the feature-vector space, without querying targets) can be superior to model-based AL when the number of samples is small, due to the possibly high inaccuracy of $\hat{y}$ (and of the estimate of its uncertainty) when trained on a small set of samples. In fact, without first selecting $x_1, \dots, x_{N_i}$ in an unsupervised way, the first trained predictors $\hat{y}$ could drive the search quite inefficiently, especially when the exploration term is not dominant, leading to collecting weakly informative samples. As a consequence, model-based criteria would remain quite inexact, leading to

[3] In case of multiple targets $m > 1$ and different feasible sets $\mathcal{X}_i$, i.e., $[y]_i : \mathcal{X}_i \to \mathcal{Y}_i$, one could define a separate set $\mathcal{Q}_i$ for each target $i = 1, \dots, m$, with $k \in \mathcal{Q}_i$ if and only if $x_k \in \mathcal{X}_i$

further collect not-so-relevant samples, with consequent performances possibly even worse than just randomly sampling $\bar{\mathcal{X}}$.

In the case of population-based AL, we use Latin Hypercube Sampling (LHS) [28] on the hyper-box $[x_{\min}, x_{\max}]$; in the case of pool-based AL, we run instead the K-means algorithm [26] on the pool $\mathcal{X}_P^\sigma \triangleq \sigma(\mathcal{X}_P)$ of scaled samples with $K = N_i$ and pickup the N_i different vectors $\sigma(\bar{x}_1), \ldots, \bar{\sigma}(\bar{x}_{N_i}) \in \mathcal{X}_P^\sigma$ that are closest to the centroids obtained by K-means in terms of Euclidean distance (cf. [41]). As some vectors may be infeasible ($\bar{x}_k \notin \bar{\mathcal{X}}$) or cannot be queried ($\bar{x}_k \notin \mathcal{X}$), similarly to the LHS algorithm with constraints described in [3, Algorithm 2] the vectors $\bar{x}_k \notin \bar{\mathcal{X}} \cap \mathcal{X}$ are discarded, and the above procedure is repeated until a set of N_i pairs $(\bar{x}_k, \bar{y}_k)$ is collected.

We denote by N_{init}, $N_{\text{init}} \geq N_i$, the total number of samples queried during the initialization phase and by $\{(x_i, y_i)\}$, $i = 1, \ldots, N_i$ the resulting set of collected samples. Note that in the case $\mathcal{X} \subset \bar{\mathcal{X}}$, $N_{\text{init}} > N_i$ queries might be required to get N_i pairs (x_k, y_k), as samples $x \in \bar{\mathcal{X}} \setminus \mathcal{X}$ might be encountered for which $y(x)$ is not defined. In this case, $N_{\text{init}} \in \mathcal{Q}$, as the initialization phase terminates as long as N_i pairs have been successfully collected. Note also that in case N_i valid samples cannot be retrieved at initialization within the total budget $N_{\max}$ of queries we have available, the AL task cannot proceed further. In the case of absence or irrelevance of unknown constraints ($\bar{\mathcal{X}} \subseteq \mathcal{X}$), we always have $N_{\text{init}} = N_i$.

3.2. Query-point selection

Assume that we have collected N samples x_k and, $\forall k \in \mathcal{Q}$, the corresponding target values y_k, and that we have fit a predictor $\hat{y}(x)$ on them by solving the supervised learning problem as in (3)

$$\hat{y} = \arg\min_{\hat{y}} \sum_{k \in \mathcal{Q}} \ell(y_k, \hat{y}(x_k), x_k) \tag{6}$$

Then, we need to define a criterion to select the remaining $N_{\max} - N_{\text{init}}$ samples x_k to query. In this paper, we will select the next sample x_{N+1} to query by maximizing an *acquisition function* $a : \mathbb{R}^n \to [0, +\infty)$ that we will introduce in the sequel

$$x_{N+1} = \arg\max_{x \in \mathcal{X}_P} a(x) \tag{7}$$

retrain $\hat{y}$, update the acquisition function a, increase N, and so on, until $N = N_{\max}$, i.e., the total available budget for queries is exhausted. In case $y(x_{N+1})$ is not defined because $x_{N+1} \notin \mathcal{X}$, we clearly do not need to retrain $\hat{y}$. The approach can be extended easily to batch-mode active learning by retraining $\hat{y}$ only after T new queries have been performed, $T > 1$.

To define the acquisition function a, we want to use an empirical estimation of the uncertainty $s_i(x)$, $s_i : \mathbb{R}^n \to [0, +\infty)$, associated with each component i of the prediction $\hat{y}(x)$, $i = 1, \ldots, m$, that we define here as we proposed in [3] to promote exploration in global optimization using surrogate functions.

Given a set $\{x_k\}_{k=1}^N$ of vectors of $\mathbb{R}^n$, we consider the squared (scaled) Euclidean distance function $d^2 : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$d^2(x, x_k) = \|\sigma(x_k) - \sigma(x))\|_2^2, \ i = 1, \ldots, N \tag{8}$$

In standard IDW functions [37], the weight functions $w_k : \mathbb{R}^n \setminus \{x_k\} \to \mathbb{R}$ are defined by the squared *inverse distances*

$$w_k(x) = \frac{1}{d^2(x, x_k)} \tag{9a}$$

In order to make the weight decay more quickly as x gets more distant from x_k, as suggested in [18, 3], here we adopt the alternative weighting function

$$w_k(x) = \frac{e^{-d^2(x,x_k)}}{d^2(x, x_k)} \tag{9b}$$

Then, we define the following functions $v_k : \mathbb{R}^n \to \mathbb{R}$ for $k = 1, \ldots, N$ as

$$v_k(x) = \begin{cases} 1 & \text{if } x = x_k \\ 0 & \text{if } x = x_j, \ j \neq k \\ \dfrac{w_k(x)}{\sum_{j=1}^N w_k(x)} & \text{otherwise} \end{cases} \tag{10}$$

As suggested in [18, 3], we then define $s^2 : \mathbb{R}^n \to \mathbb{R}^m$ as the *IDW variance function*

$$s_i^2(x) = \sum_{k \in \mathcal{Q}} v_k(x)([y_k]_i - [\hat{y}(x)]_i)^2, \ i = 1, \ldots, m \tag{11}$$

associated with the current training dataset $\{(x_k, y_k)\}_{k=1}^N$ and predictor $\hat{y}$. Note that for $x = x_k$ and $k \in \mathcal{Q}$ we have $s_i^2(x_k) = ([y_k]_i - [\hat{y}(x_k)]_i)^2$, which in the case of perfect interpolation $[\hat{y}(x_k)]_i = [y_k]_i$ gives $s_i^2(x_k) = 0$ (this corresponds to having no prediction uncertainty about $y_i(x)$ at $x = x_k$). Note also that the sum in (11) only considers the indices $k \in \mathcal{Q}$, as for $k \notin \mathcal{Q}$ vector $x_k \notin \mathcal{X}$ and therefore $y_k = y(x_k)$ is undefined. This is equivalent to assume that $y_k = \hat{y}(x_k)$ for all $x_k \notin \mathcal{X}$ and sum in (11) for $k = 1, \ldots, N$.

Regarding promoting diversity in exploring the feature-vector space, as suggested in [3] we also consider the *IDW exploration function* $z : \mathbb{R}^n \to \mathbb{R}$ defined as

$$z(x) = \begin{cases} 0 & \text{if } x \in \{x_1, \ldots, x_N\} \\ \frac{2}{\pi} \tan^{-1}\left(\frac{1}{\sum_{k=1}^N w_k(x)}\right) & \text{otherwise} \end{cases} \tag{12}$$

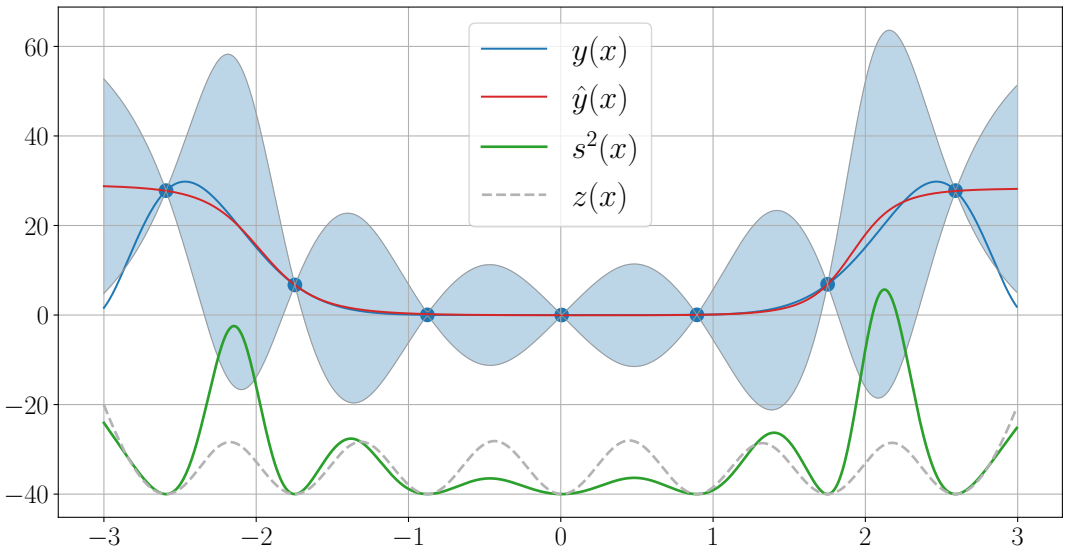

Figure 1: Function y of Example 1 (blue line), samples (x_k, y_k) (blue dots), NN predictor $\hat{y}$ (red line), band $\hat{y}(x) \pm 3\sqrt{s_i^2(x)}$ (light blue area), scaled and shifted IDW functions s^2 (green line) and z (dashed gray line)

Similarly to the passive sampling approach in [43], function z returns a pure exploration term that is only based on the geometric position of the (scaled) feature vectors $\{x_k\}$, and hence, contrarily to the IDW variance function s^2, is not influenced by the predictor $\hat{y}$ learned up to step N. Note that s^2 also promotes exploration, but only indirectly.

Example 1. Let the data y_k be generated by the following scalar function $y : \mathbb{R} \to \mathbb{R}$

$$y(x) = x^4 \sin^2\left(\frac{1}{3}x^2\right) \tag{13}$$

that we want to approximate over the interval $\bar{\mathcal{X}} = [-3, 3]$ by a simple feedforward neural network (NN) $\hat{y}$ with two layers of five neurons each, logistic activation function $\frac{1}{1+e^{-x}}$, and linear output function. As depicted in Figure 1, we assume that we have collected $N = 7$ samples (x_k, y_k) (blue dots), $y_k = y(x_k)$, and fit a NN via the `MLPRegressor` function in `scikit-learn` [30] with ℓ_2-regularization term $\alpha = 10^{-2}$, by using the L-BFGS nonlinear optimization algorithm [24]. Figure 1 also shows the original function $y(x)$ (blue line), the NN predictor $\hat{y}(x)$ (red line), and the band $\hat{y}(x) \pm 3\sqrt{s_i^2(x)}$ (light blue area). The figure also shows scaled and shifted versions of the IDW functions $s^2(x)$ (green line) and $z(x)$ defined in (12) (dashed gray line).

Let us now define the acquisition function

$$a(x) = \left(\sum_{i=1}^{m} s_i^2(x) \right) + \delta z(x) \tag{14}$$

where $\delta \geq 0$ is a hyperparameter balancing the role of IDW variance $s^2(x)$ and IDW distance $z(x)$. Note that δ trades off between model-based learning (small δ) and learning based on the pure exploration of the feature-vector space to promote diversity (large δ).

In the case of population-based AL, the maximization problem (7) can be solved by global optimization; in this paper, we will use the derivative-free Particle Swarm Optimization (PSO) algorithm [21], as $a(x)$ is a cheap function to evaluate whenever $\hat{y}(x)$ is easy to compute. In pool-based sampling, when the number M of samples in the pool is not too high, problem (7) can be solved by enumeration by setting

$$x_{N+1} = \bar{x}_{k^*},\ k^* = \arg \min_{k \in \{1,\ldots,M\} \setminus \mathcal{E}} \{a(\bar{x}_k)\} \tag{15a}$$

We assume that possible duplicates $\bar{x}_k = \bar{x}_j$, $k \neq j$, are removed upfront from the pool $\mathcal{X}_P$.

When (15a) is impractical due to a large number M of samples in the pool, one can first use PSO to optimize over the entire set $\bar{\mathcal{X}}$ to get $\bar{x}^* = \arg\max_{x \in \bar{\mathcal{X}}} a(x)$ as in population-based AL and then set (cf. [40])

$$x_{N+1} = \bar{x}_{k^*},\ k^* = \arg \min_{k \in \{1,\ldots,M\} \setminus \mathcal{E}} d^2(\bar{x}_k, x^*) \tag{15b}$$

Algorithm 1 reports the pseudocode of the proposed AL algorithm that we call Inverse-Distance based Exploration for Active Learning (IDEAL). The complexity of the algorithm will be discussed in Section 3.5. Note that output data scaling can be updated before retraining $\hat{y}$ at Step 6.1, such as by applying standard scaling based on the currently available values $\{y_k\}$, $k \in \mathcal{Q}$.

3.3. Extensions of the acquisition function

The basic acquisition function (14) can be extended in two directions. First, for pool-based AL we can consider the density function $\rho : \mathcal{X}_P \to (0, +\infty)$ that measures how much "isolated" is a sample $\bar{x}_k \in \mathcal{X}_P$ with respect to the remaining samples. Similar to density-based spatial clustering approaches [14], one can use the average (scaled) distance of $\bar{x}_k$ from its n nearest neighbors,

$$d_k = \frac{1}{n} \sum_{j \in \mathcal{N}_k} \|\sigma(\bar{x}_k) - \sigma(\bar{x}_j)\|_2$$

Algorithm 1 Inverse-Distance based Exploration for Active Learning (IDEAL).

Input: Set $\mathcal{X}_P = \{\bar{x}_k\}_{k=1}^M$ (pool-based) or $\mathcal{X}_P = \bar{\mathcal{X}}$ (population-based) of queryable feature vectors; budget $N_{\max}$ of available queries; number N_i of initial samples to acquire; pure exploration hyperparameter $\delta \geq 0$.

1. Remove possible duplicates $\bar{x}_k$ from $\mathcal{X}_P$ (pool-based only);
2. Compute scaling functions σ_i as in (5);
3. Extract N_i samples (x_k, y_k) as described in Section 3.1 by K-means (pool-based) or LHS (population-based); if it is not possible to extract them within $N_{\max}$ queries go to Step 7, otherwise set $N_{\text{init}} =$ number of queries done;
4. $\mathcal{Q} \leftarrow \{k \in \{1, \dots, N_{\text{init}}\} : \ x_k \in \mathcal{X}\}$;
5. $\mathcal{E} \leftarrow \left\{i \in \{1, \dots, M\} : \ \bar{x}_i = x_k \text{ for some } k \in \{1, \dots, N_{\text{init}}\}\right\}$ (pool-based only);
6. **For** $N = N_{\text{init}}, \dots, N_{\max}$ **do**:
   - 6.1. **If** $N \notin \mathcal{Q}$ **then** update predictor $\hat{y}$ by solving (3);
   - 6.2. Compute new sample x_{N+1} as in (7) (population-based) or (15) (pool-based);
   - 6.3. **If** $x_{N+1} \in \mathcal{X}$ acquire y_{N+1} and set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{N+1\}$;
   - 6.4. $\mathcal{E} \leftarrow \mathcal{E} \cup \{k^*\}$ (pool-based only);
7. **End**.

Output: Predictor $\hat{y}$, or declaration of failure in collecting N_i feasible initial samples.

where $\mathcal{N}_k$ is the set of indices corresponding the n nearest neighbors of $\bar{x}_k$ in $\mathcal{X}_P \setminus \{\bar{x}_k\}$, to estimate the density as proportional to the normalized inverse volume of the sphere of radius d_k, i.e.,

$$\rho(\bar{x}_k) = \frac{\frac{1}{d_k^n}}{\max_{j=1,\dots,M} \left\{\frac{1}{d_j^n}\right\}} = \frac{\min_{j=1,\dots,M} \{d_k^n\}}{d_k^n} \tag{16}$$

Note that (16) is always defined, as n duplicates cannot exist such that they have a zero average distance due to the fact that we have assumed that all possible duplicates $\bar{x}_k = \bar{x}_j$, $k \neq j$, have been removed. Note that ρ does not depend on the predictor $\hat{y}$ learned and can be therefore computed upfront. Regarding population-based AL, one can simply set $\rho(x) = 1$, $\forall x \in \bar{\mathcal{X}}$.

Next, we can introduce weight functions $c_i : \mathbb{R}^n \to [0, +\infty)$ to actively learn the predictor in a non-uniform way with respect to the target index i and x (or uniformly, if $c_i(x) \equiv 1$ for all $i = 1, \ldots, m$). Accordingly, we extend (14) to

$$a(x) = (1 + \omega\rho(x)) \sum_{i=1}^{m} c_i(x) \left(s_i^2(x) + \frac{\delta}{m} z(x) \right) \tag{17}$$

where $\omega \geq 0$ is a scalar weight on density. Note that ω is redundant in the case of population-based AL, having assumed that $\rho(x) \equiv 1$.

Let us show that the active learning mechanism (7) under (14), possibly extended as in (17), follows criteria of *informativeness*, *representativeness*, and *diversity*, which are listed in [41] as essential for AL. Regarding the first, maximizing $a(x)$ implies looking for large values of the uncertainty $s(x)$ associated with the current predictor $\hat{y}(x)$, i.e., to select the next sample x_{N+1} where $\hat{y}$ is considered most uncertain according to (11), so that querying x_{N+1} is expected to bring significant new information. The second, which is only applicable in the case of pool-based sampling under the extension (17), is taken care of by $\rho(x)$ when $\omega > 0$, as in the maximization (7) those samples $\bar{x}_k$ that have a low density $\rho(\bar{x}_k)$, for instance, because they are outliers, will be discouraged. Third, diversity is promoted because $s(x)$ and $z(x)$ are small close to samples that have been already visited, which ultimately makes the AL algorithm visit unexplored areas of the feature-vector space. The tradeoff between representativeness and diversity is taken care of by the coefficient ω.

In all the numerical tests reported in Section 4 we will always employ the baseline acquisition function (14), as no significant improvements were found by using $\omega > 0$ in our benchmarks, and in addition we aim at a uniform weighting $c_i(x) \equiv 1$. Nonetheless, the extra versatility allowed by (17) might be useful in certain AL applications.

3.4. Other active learning algorithms

Algorithm 1 (ideal) will be compared to some of the most common active learning methods proposed in the literature that can support rather arbitrary prediction models $\hat{y}$. The considered methods have some substantial differences, that we will see have consequences on AL performance. We review such methods here below.

3.4.1. Random sampling (random)

The method draws samples x_{N+1} from the uniform distribution defined over $\bar{\mathcal{X}}$ in the case of population-based sampling, or by selecting a random index in $\{1, \ldots, N_{\max}\} \setminus \mathcal{E}$ in the case of pool-based sampling. This is the simplest method we consider to have a baseline to compare with: any AL method should be more efficient that random, at least statistically.

3.4.2. Greedy method (GS_x)

The sampling technique GS_x proposed in [43, Algorithm 1] selects x_{N+1} by maximizing the minimum distance from existing samples, i.e.,

$$x_{N+1} = \arg\max_{x \in \mathcal{X}^P} d_x(x) \tag{18a}$$

$$d_x(x) = \min_{k=1,\ldots,N} \|\sigma(x) - \sigma(x_k)\|_2^2 \tag{18b}$$

The method is not model-based, in that the predictor $\hat{y}$ is not used to select the samples to query. Although conceived for pool-based AL, the method can be extended also to population-based AL by maximizing $d(x)$ with respect to $x \in \bar{\mathcal{X}}$ in (18). For fair comparison, in the case of population-based AL, rather than maximizing the minimum distance in our numerical tests we will also adopt LHS to acquire the first N_i samples instead of using the approach suggested in [42] for pool-based AL.

3.4.3. Greedy method (iGS)

The iGS method proposed in [42, Algorithm 3] is an extension of greedy sampling that, in addition, considers the minimum predicted distance in the y-space

$$d_y(x) = \min_{k=1,\ldots,N} \|\hat{y}(x) - y_k\|_2^2 \tag{19}$$

where $\hat{y}$ is the latest predictor trained on currently available samples, and selects

$$x_{N+1} = \arg\max_{x \in \mathcal{X}^P} d_x(x) d_y(x) \tag{20}$$

(in case of multiple targets, we assume that the values in (19) refer to scaled target values). The method can be extended also to population-based AL. In such a case, similarly to GS_x, we will use LHS for initialization.

Thanks to the additional term d_y defined in (19), iGS also aims at getting samples where the output y is expected to be different from the current values y_k observed so far. A possible drawback of (20), however, is that d_x and d_y are multiplied by each other, i.e., diversity is sought in both the x- *and* y-space, so that pure exploration of the x-space might be inhibited by *predicted* proximity

in the y-space, i.e., by small (or zero) estimated values d_y. Instead, ideal looks for diversity in the x- *or* y-space, as $z(x)$ and $s^2(x)$ are summed in (14) instead of being multiplied by each other as in (20).

3.4.4. Query-by-Committee (QBC)

After a first initialization phase in which N_i feasible samples are generated randomly, the QBC method for regression [31, 8] considered here creates K_{QBC} bootstrap samples obtained by randomly sampling the existing N samples with replacement, trains a predictor $\hat{y}^j$ on each set, $j = 1, \ldots, K_{QBC}$, and then selects x_{N+1} to maximize the output-prediction variance

$$x_{N+1} = \arg\max_{x \in \mathcal{X}^P} \sum_{j=1}^{K_{QBC}} \left\| \hat{y}^j(x) - \frac{1}{K_{QBC}} \sum_{j=1}^{K_{QBC}} \hat{y}^j(x) \right\|_2^2 \tag{21}$$

(in case of multiple targets, the terms in (21) must be considered again as scaled target values). It can be used for both pool-based and population-based AL. In QBC, we set $K_{QBC} = 5$ in all our tests and train the individual predictors $\hat{y}^j$ on bootstrapped samples as in [42], rather than leaving out a different subset of $\lfloor \frac{N}{K_{QBC}} \rfloor$ samples as suggested in [8]. In fact, the former approach better performed in our examples, in which the number of allowed samples is small compared to the number of model parameters to learn and hence removing $\lfloor \frac{N}{K_{QBC}} \rfloor$ samples can dramatically change the resulting individual predictions $\hat{y}_j(x)$. An additional disadvantage of QBC when actively learning NN models is that large disagreements may be caused by lack of global convergence of the optimization method used to train the different predictors, due to the non-convex nature of the NN training problem.

The QBC method we tested only relies on information related to the y-space, i.e., is totally based on the predictor $\hat{y}$ and its variants $\hat{y}^j$ to drive the acquisition, but not explicitly on measures defined purely on feature-vectors for promoting diversity in the x-space. This ultimately has a potential negative impact on the robustness of QBC. Extensions of QBC to improve performance by taking into account diversity and density was introduced in [20] in the context of classification.

3.4.5. Improved Representativeness-Diversity Maximization (iRDM)

The iRDM pool-based unsupervised active learning method [25] for regression generates $N_{\max}$ samples in one shot (rather than incrementally) after performing K-means [26] on $\mathcal{X}_P^g$ to create $N_{\max}$ clusters. Then, the samples closest to the resulting centroids are refined sequentially (up to $c_{\max}$ times) to optimize the tradeoff between the representativeness of the selected point x_k

within its cluster C_k (i.e., the average distance of x_k from the points in C_k) and the diversity of x_k from the other selected samples $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{N_{\max}}$ in the remaining clusters (i.e., the minimum distance $\|\sigma(x_k) - \sigma(x_j)\|_2$, $j = 1, \ldots, N_{\max}$, $j \neq k$). We set $c_{max} = 5$ in all our tests as suggested in [25]. As for GS_x, the method does not exploit the predictor $\hat{y}$, which in fact is only trained after acquiring all the $N_{\max}$ samples.

3.5. Numerical complexity

The initial phase of ideal (Algorithm 1) requires either extracting N_{init} samples by Latin Hypercube Sampling (LHS) [28] (population-based AL) or K-means [26] (pool-based AL). In addition, ideal requires retraining the predictor $\hat{y}$ at Step 6.1 and solving the optimization problem at Step 6.2 to get a new sample. Depending on the class of predictors $\hat{y}$ used, retraining can be the most expensive computation effort. As for other AL methods, warm-starting the training algorithm or incrementally learning $\hat{y}$ could be exploited, if supported by the particular class of prediction models and training algorithms chosen. Regarding Step 6.2, the computation complexity mainly depends on the number of operations required to evaluate the predictor $\hat{y}(x)$ in (11), and therefore on the complexity of the selected model class.

Algorithms ideal, iGS, and QBC require retraining the predictor $\hat{y}$, respectively, $N_{\max} - N_{\text{init}} + 1$, $N_{\max} - N_{\text{init}} + 1$, and $(K_{QBC} + 1)(N_{\max} - N_{\text{init}} + 1)$ times, while random, GS_x, and iRDM only once at the end of the acquisition, as they are only based on the relative positions of the acquired samples x_k in the feature space.

In the case of pool-based sampling, ideal, iGS, and QBC require, after the initialization phase, evaluating the predictor $\hat{y}$, respectively, $(N_{\max} - N_{\text{init}} + 1)M$, $(N_{\max} - N_{\text{init}} + 1)M$, and $(K_{QBC}+1)(N_{\max} - N_{\text{init}} + 1)M$ times. In addition, ideal requires evaluating the acquisition function $a(\bar{x}_k)$ $(N_{\max} - N_{\text{init}} + 1)M$ times, GS_x and iGS evaluating the squared distances in (18b) $(N_{\max} - N_{\text{init}} + 1)M$ times, and in addition iGS evaluating the squared distances (19) $(N_{\max} - N_{\text{init}} + 1)M$ times, while QBC requires evaluating the output prediction variance terms in (21) $(N_{\max} - N_{\text{init}} + 1)M$ times. Then, ideal, GS_x, iGS, and QBC require computing $N_{\max} - N_{\text{init}} + 1$ times the minima defined by (15a), (18a), (20), and (21), respectively.

Regarding iRDM, it requires solving K-means to partition M points into $N_{\max}$ clusters, computing the representativeness measure M times, and then repeat $c_{\max}$ times the construction of the diversity measure on all candidate samples within the updated cluster with respect to the samples already fixed in each one of the other clusters. Note that iRDM, contrarily to the other methods, is not an incremental AL method, and therefore a change of $N_{\max}$

(such as due to allowing more queries) would require redefining all the $N_{\max}$ samples to acquire.

Finally, we remark that the computation time required by the AL algorithm is often negligible with respect to the time required to acquire a new target value, which is often the most dominating effort in the practical situations AL algorithms are employed.

4. Numerical tests

In this section, we test the proposed AL approach on synthetic illustrative examples and real-world datasets, comparing it to the different AL methods reviewed in Section 3.4.

Regarding the initial N_i samples, for the GS_x and iGS methods we recursively use (18b) starting from the centroid x_1 of $\mathcal{X}^P$ as proposed in [42], and random sampling for the random and QBC methods. Different approaches have been proposed in the literature for cold-starting AL, see for instance the representative sampling method proposed recently in [17] in the context of image classification, or the approach used by iRDM.

To analyze the performance of iRDM as a function of the number of acquired samples, as iRDM is not an incremental method we execute it from scratch each time we want to collect a different number of samples.

All computations were carried out in Python 3.9.15 using the scikit-learn package [30] to train feedforward NNs for regression (`MLPRegressor` function) and support vector regression (SVR) (`SVR` function). Regarding the considered AL methods, for pool-based active learning we used the Python implementation developed by the author and available at `http://cse.lab.imtlucca.it/~bemporad/ideal`.

4.1. Scalar example

We first test the proposed AL approach on the simple regression problem defined in Example 1, i.e., with $y(x)$ as in (13). Since $n = 1$, we generate a grid $\mathcal{X}_P$ of $M = 1000$ equally-spaced points on the line segment $\bar{\mathcal{X}} = [-3, 3]$ and use pool-based sampling on the entire pool $\mathcal{X}_P$, so that problem (7) can be solved by enumeration (15a). While training the NN, the parameter vector is not warm-started when executing Step 6.1, to avoid possible low-quality local minima inherited by the early steps of Algorithm 1 when only a few data are available.

The median over 50 runs of the root-mean-square error (RMSE)

$$RMSE = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(y_k - \hat{y}(x_k))^2}$$

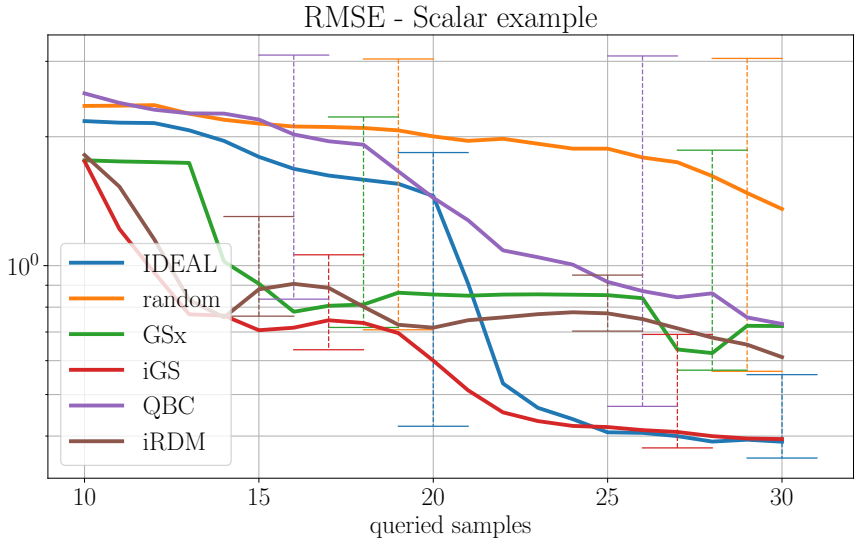

Figure 2: AL of function (13): median RMSE as a function of the number of queries. Vertical lines denote min and max RMSE values

and its range (min and max values) obtained with $\delta = 5$, $N_i = 10$, $N_{\max} = 30$, as a function of the number N of acquired samples, is depicted in Figure 2 and compared with the RMSE obtained with random, GS_x, iGS, QBC, and iRDM sampling. It is apparent that ideal is superior to random and QBC, behaves better than GS_x and iRDM (which are not model-based methods) and similarly to iGS after about half of the allowed samples have been acquired. The high variance of QBC is possibly due to the small number of samples queried and, consequently, the even smaller number of training samples used to train the predictors forming the committee that may lead to large disagreements among them.

Table 1 shows the mean and standard deviation of the RMSE obtained by ideal when $N = N_{\max}$ for different values of the hyperparameter δ and, for comparison, by random sampling.

δ	0.0	0.1	1.0	5.0	10.0	R
mean	0.528	0.470	0.439	0.402	0.402	1.495
std	0.307	0.239	0.084	0.042	0.036	0.548

Table 1: AL of function (13): mean RMSE and its standard deviation after $N_{\max} = 30$ steps obtained on 50 different runs of Algorithm 1 for different values of δ (R = random sampling)

We will take $\delta = 5$ in all our remaining tests. For such a value of δ, to test the robustness of AL against measurement noise we repeat the same test by

perturbing the measurements $y_k = y(x_k)+\eta_k$, where $\eta_k \sim \mathcal{N}(0, \sigma_\eta^2)$ for different values of the standard deviation σ_η. The mean and standard deviation of the resulting RMSE over 50 runs after $N_{\max} = 30$ iterations is shown in Table 2. The table shows that for increasing values of σ_η the RMSE deteriorates without an excessive increase of variance and in a gradual way for ideal (proving its robustness with respect to noisy target measurements), iGS, and iRDM, while such a trend is less marked for GS_x, random, and QBC.

σ_η	ideal	random	GS_x	iGS	QBC	iRDM
0.0	0.40 (0.04)	1.44 (0.60)	0.76 (0.14)	0.41 (0.04)	1.16 (0.88)	0.63 (0.12)
1.0	0.62 (0.02)	1.45 (0.55)	0.80 (0.11)	0.62 (0.04)	1.15 (0.79)	0.74 (0.16)
2.0	0.86 (0.09)	1.60 (0.48)	0.92 (0.14)	0.86 (0.05)	1.38 (0.78)	0.91 (0.14)

Table 2: AL of function (13) with noise: mean (std) RMSE after $N_{\max}$ =30 steps for different values of σ_η

4.2. Multiparametric quadratic programming

Model predictive control (MPC) is a popular engineering technique for controlling dynamical systems in an optimal way under operating constraints [6]. Evaluating the MPC law requires solving a quadratic programming (QP) problem of the form

$$\begin{array}{rcll} z^*(x) & = & \arg\min_z & \frac{1}{2}z'Qz + x'F'z \\ & & \text{s.t.} & Az \leq b + Sx \\ & & & \ell \leq z \leq u \\ y(x) & = & [I_m\ 0\ \ldots\ 0] & z^*(x) \end{array} \tag{22}$$

where $z \in \mathbb{R}^{n_z}$ is a vector of future control moves, $n_z \geq m$, and $x \in \mathbb{R}^n$ is a vector of parameters that change at run time, such as estimated states and reference signals, and the Hessian matrix $Q = Q' \succ 0$. To alleviate the effort of solving (22) online for each given vector x, multiparametric QP (mpQP) was proposed in [4], showing that the solution $z^* : \mathbb{R}^n \to \mathbb{R}^{n_z}$, and therefore $y(x)$, is continuous and piecewise affine over a polyhedral partition of a convex polyhedron $\mathcal{X} \subseteq \mathbb{R}^n$. The main drawback of such an *explicit* form of MPC is that the number of polyhedral cells tends to grow exponentially with the number of constraints in (22).

Suboptimal methods were proposed to *approximate* $y(x)$, such as via neural networks [29, 19]. In order to find an approximation $\hat{y}(x)$ of $y(x)$, one must collect a training dataset of pairs (x_k, y_k), where evaluating $y_k = y(x_k)$ requires solving a QP problem as in (22). Randomly sampling a given set $\bar{\mathcal{X}} \subset \mathbb{R}^n$ of

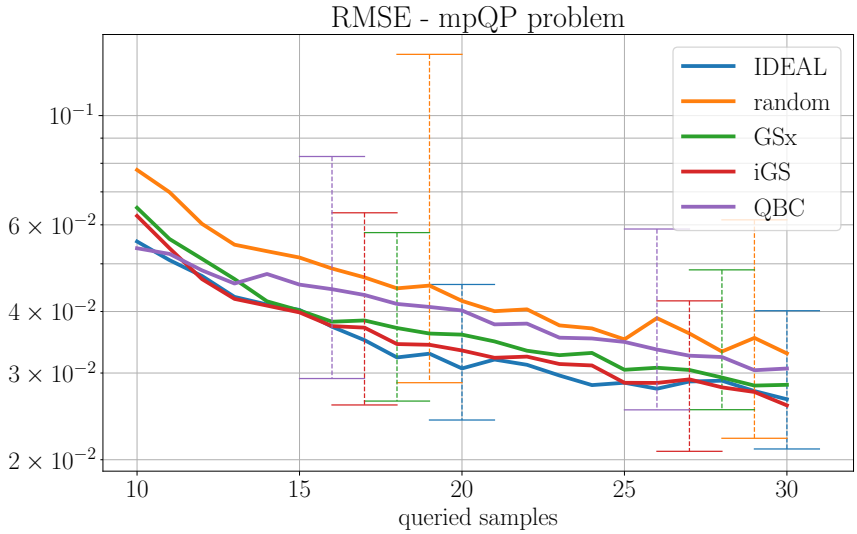

Figure 3: mpQP problem: median RMSE as a function of the number of queries. Vertical lines denote min and max values. Only the population-based versions of ideal, GS_x, iGS, and QBC were used (iRDM is a pure pool-based method).

parameters x may result time-consuming, especially when the dimension n of the parameter vector is large. To minimize the number $N_{\max}$ of QP problems solved to get a proper approximation quality, we use Algorithm 1 to actively generate samples x_k.

We consider here a mpQP problem with $n = 2$, $n_z = 12$, $m = 1$, $b \in \mathbb{R}^{12}$, $S = 0$, and all matrices in (22) generated randomly, with the entries of A, $F \sim \mathcal{N}(0, 1)$ and the entries of $b, u, -\ell \sim \mathcal{U}[0, 1]$, where $\mathcal{U}[0, 1]$ is the uniform distribution over the interval $[0, 1]$, $Q = Q' \succ 0$, is randomly generated so that its condition number equals 10^3, and $\bar{\mathcal{X}} = \{x : \|x_i\|_\infty \leq 3\}$. Algorithm 1 is applied using population-based sampling with $N_i = 10$ and $N_{\max} = 30$ for training a feedforward neural network with 3 layers of 10 neurons each and ReLU activation function, without using warm starting while retraining the model. The median RMSE and its range over 50 runs is shown in Figure 3, where it is apparent that ideal performs better than random, GS_x, and QBC, and similar to iGS. Note that in this population-based AL example we could not use iRDM, which is a pure pool-based method. Figure 4 shows the polyhedral partition associated with the exact mpQP solution (unknown to the active learning algorithms) computed as described in [2] along with the queried samples and initial samples generated by one of the runs of Algorithm 1. It is evident that the points acquired by ideal are not distributed uniformly.

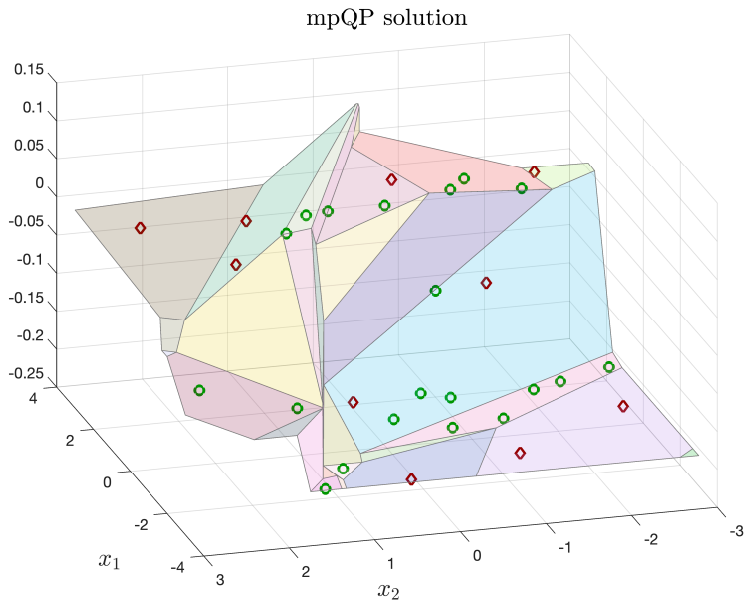

Figure 4: Exact mpQP solution, initial samples $x_1, \ldots, x_{N_i}$ (red diamonds), and samples $x_{N_i+1}, \ldots, x_{N_{\max}}$ queried by ideal (green circles)

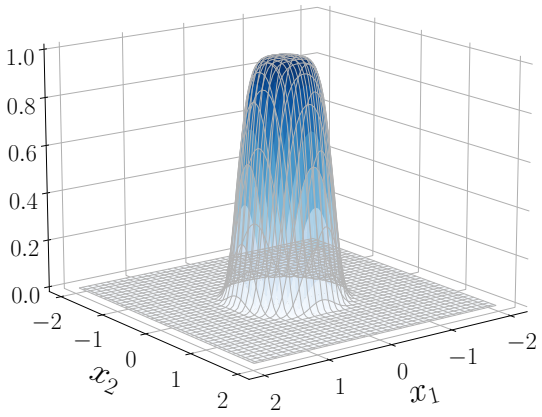

Figure 5: Bell-shaped function (23)

4.3. Active learning with unknown constraints

In order to test Algorithm 1 in the presence of unknown constraints, we consider data generated by the following bell-shaped function $y : \mathbb{R}^2 \to [0, 1]$

$$y(x) = e^{-\left(\left(\frac{3}{2}x_1\right)^2 + \left(\frac{3}{2}x_2\right)^2\right)^3} \tag{23}$$

plotted in Figure 5. Algorithm 1 is applied with $N_i = 10$ and $N_{\max} = 120$ to fit a nonlinear model via support vector regression with radial basis function (RBF) kernel, with penalty $\frac{1}{C} = 0.1$ for ℓ_2-regularization and threshold $\epsilon = 0.1$. Pool-based sampling is used on a set $\mathcal{X}_P$ of $M = 1000$ random feature vectors generated uniformly in $[-2, 2] \times [-2, 2]$. The median RMSE and its range computed on all vectors $\bar{x}_i \in \mathcal{X}_P$ over 50 runs is shown in Figure 6 (upper plot). A possible reason for the poor performance of QBC in this example is that the prediction uncertainty estimated by QBC is inaccurate, which leads to sampling feature vectors that are in reality not worth sampling. In addition, as mentioned earlier, improper sampling leads to poor predictors and hence a poor target-uncertainty estimation, so that weak sampling persists. This leads to a waste of queries.

Next, we add an unknown constraint by only defining $y(x)$ for $x \in \mathcal{X}$, where

$$\mathcal{X} \triangleq \{x : \ 3x_2 \leq \sqrt{3}|x_1|\} \tag{24}$$

and repeat the same test, obtaining the RMSE results shown in Figure 6 (lower plot), where the RMSE is computed only on the feasible vectors $\bar{x}_i \in \mathcal{X}_P \cap \mathcal{X}$.

Note that for ideal, random, GS_x, iGS, and QBC the RMSE values are not available for $k < N_{\text{init}}$. This is due to the fact that, as described in Section 3.1, the first predictor is trained only after N_i feasible samples have been collected, which may require $N_{\text{init}} > N_i$ queries. On the contrary, due to its non-incremental nature, when running iRDM to acquire $k = N_i, N_{i+1}, \ldots, N_{\max}$ samples, the predictor is always constructed on the available feasible samples, no matter how many feasible samples have been collected (unless all samples are infeasible, a situation that never occurred in our experiments).

In the above tests, when using GS_x and iGS random sampling was employed to get the first N_i samples, as in the case of unknown constraints the initialization method suggested in [42] was sometimes failing to get N_i feasible samples within the maximum budget $N_{\max}$ of queries.

Figure 7 shows the level sets of the learned classifier $\hat{y}$ during one of the tests using all the considered methods, the pool $\mathcal{X}_P$ of samples (gray circles), the queried samples (green dots), the initial samples (red diamonds), and the level sets of the prediction function (dark blue lines) and of function (23) (dashed gray circles). It is apparent how GS_x and iRDM scatter the points uniformly no matter whether they are feasible or not, which wastes a large percentage of the queries to get meaningful values y_i (all the $N_{\max}$ points acquired by iRDM are marked in red, as they are selected altogether). Similarly, iGS also samples infeasible areas of the x-space quite consistently, as it aims at sampling the co-domain of $\hat{y}$ uniformly due to the term d_y in (19). Regarding QBC, it mostly samples the infeasible set, where the K_{QBC} predictors in the

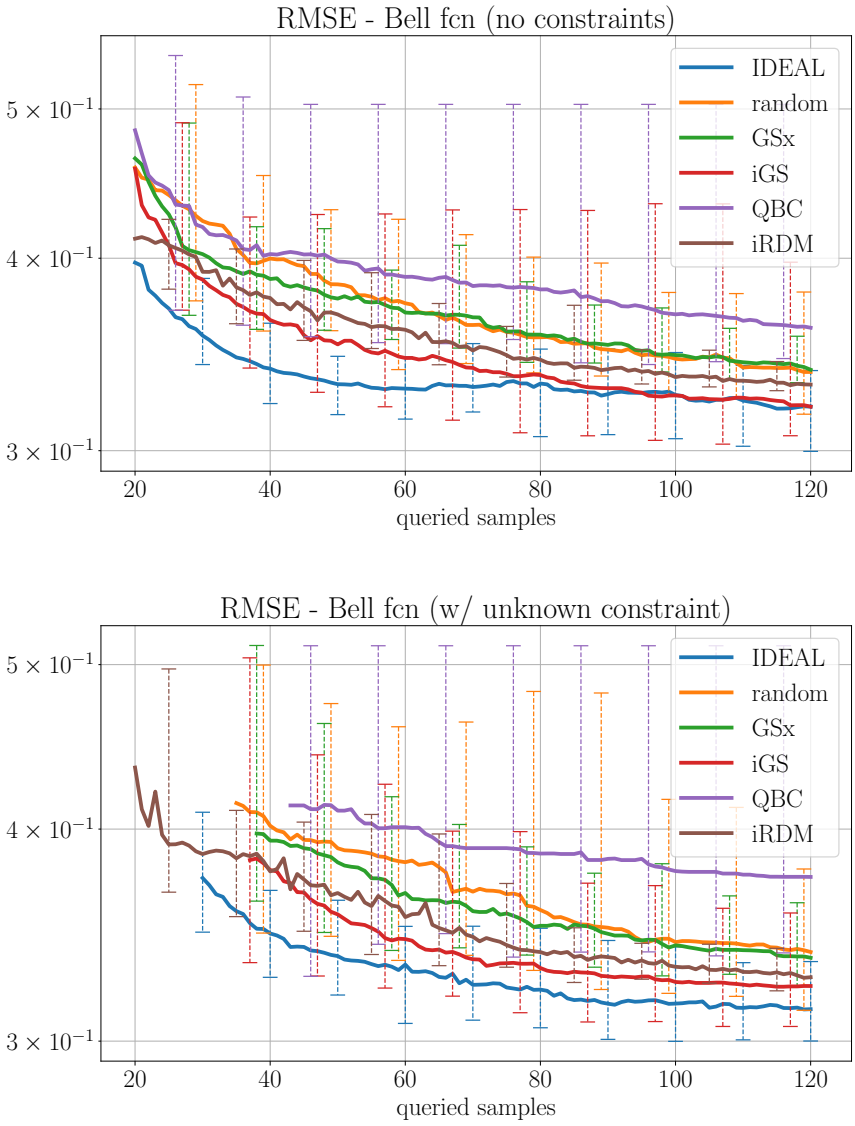

Figure 6: AL problem (23), median RMSE without (upper plot) and with unknown constraint (24) (lower plot). Vertical lines denote min and max values

committee completely extrapolate due to lack of information and hence tend to disagree the most. On the other hand, ideal spontaneously tends to avoid querying infeasible vectors $x \in \mathcal{X}_P \setminus \mathcal{X}$ and concentrates most queries where the underlying bell-shaped function has the largest variations.

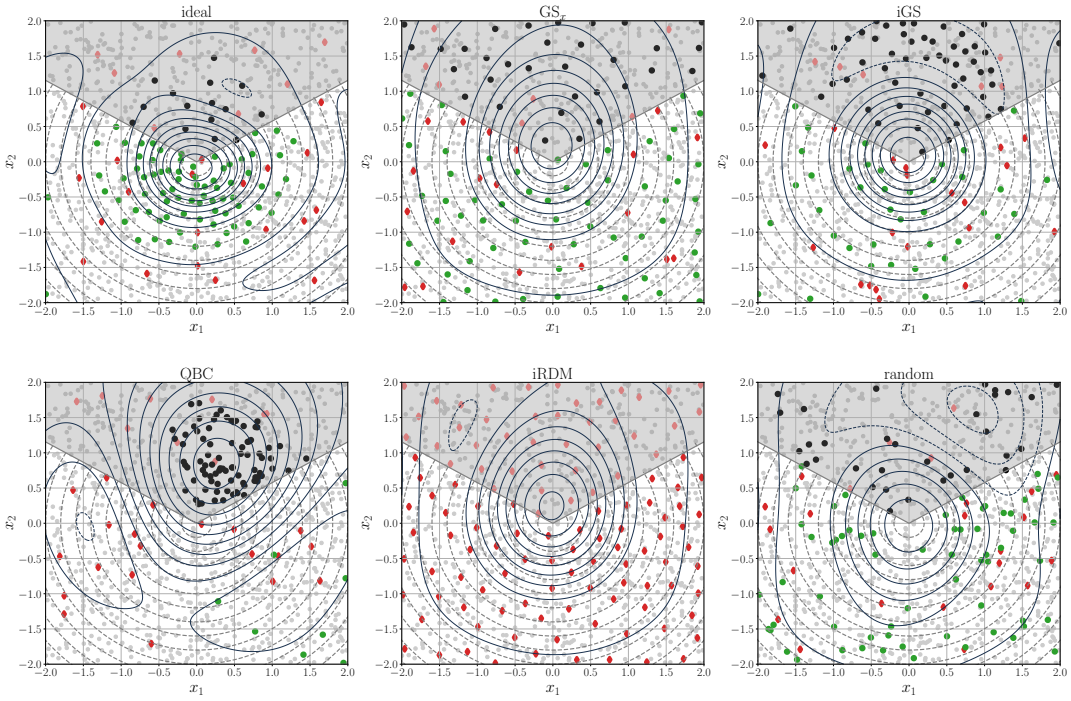

Figure 7: AL problem (23) with unknown constraint (24): pool $\mathcal{X}_P$ (gray circles), queried samples (green dots), initial samples (red diamonds), level sets of prediction function (dark blue lines) and of the true function (23) (dashed gray circles)

4.4. Real-world datasets

We test the proposed AL approach on real-world datasets for regression from the University of California, Irvine (UCI) Machine Learning Repository, Kaggle, and StatLib, summarized in Table 3.

For the tests described in the upper half of Table 3, we train neural networks $\hat{y}$ with two layers of five neurons each with logistic activation function and ℓ_2-regularization term equal to 10^{-2} on the vector of weight/bias terms of the

[4] `https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test`

[5] `https://archive.ics.uci.edu/ml/datasets/auto+mpg`

[6] `https://archive.ics.uci.edu/ml/datasets/Wine+Quality` (the first seven attributes are considered as features, the ninth as target)

[7] `https://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics`

[8] `https://archive.ics.uci.edu/ml/datasets/QSAR+aquatic+toxicity`

[9] `https://www.kaggle.com/fedesoriano/body-fat-prediction-dataset/version/1`

[10] `https://www.kaggle.com/datasets/dongeorge/beer-consumption-sao-paulo`

[11] `http://lib.stat.cmu.edu/datasets/PM10.dat`

dataset	M	n	m	$\hat{y}(x)$	$N_{\max}$
`concrete-slump` [4]	103	7	1	neural network	103
`auto-mpg` [5]	392	6	1	neural network	100
`winequality-white` [6]	4898	7	1	neural network	100
`yacht` [7]	308	6	1	neural network	100
`qsar-aquatic-toxicity` [8]	546	8	1	RBF-SVR	120
`bodyfat` [9]	252	14	1	RBF-SVR	120
`beer` [10]	365	4	1	RBF-SVR	120
`pm10` [11]	500	7	1	RBF-SVR	120

Table 3: Real-world datasets: M = number of available samples in the pool, n = number of features, $m = 1$ (single target), $N_{\max}$ = query budget.

model, while for the remaining tests we train predictors $\hat{y}$ using RBF-SVR with penalty $\frac{1}{C} = 0.02$ for ℓ_2-regularization and threshold $\epsilon = 0.05$. Pool-based AL is used with parameters $N_{\text{init}} = 20$ and the values of $N_{\max}$ reported in Table 3. Median RMSE results and their ranges over 50 tests are shown in Figures 8–11.

As expected, all the considered AL methods perform better than random when the number of queries is large enough, with QBC being the method that requires the largest number of queries to start becoming an effective AL strategy. In spite of its unsupervised AL nature, iRDM is overall quite effective, sometimes superior to model-based strategies. In all tests, ideal performs either better or comparably with respect to the other methods, and is the only method that, at least statistically, seems to perform consistently well with respect to all the considered datasets. The latter feature, consistency, makes it an "ideal" candidate to face a new active learning problem in practice.

5. Conclusion

In this paper we have introduced a new active learning method to solve a very broad set of active learning problems of regression. The approach is not linked to any particular class of predictors and supports both pool-based and population-based sampling. The objective function driving the optimal selection of the next feature vector to query only requires evaluating the prediction function that has been currently learned and compare it to the target values acquired so far. This is an advantage compared to other approaches such as query-by-committee methods in which multiple predictors must be trained and evaluated.

For low-dimensional problems (say up to three features) amenable for population-based AL, our practical experience is that it is usually more efficient

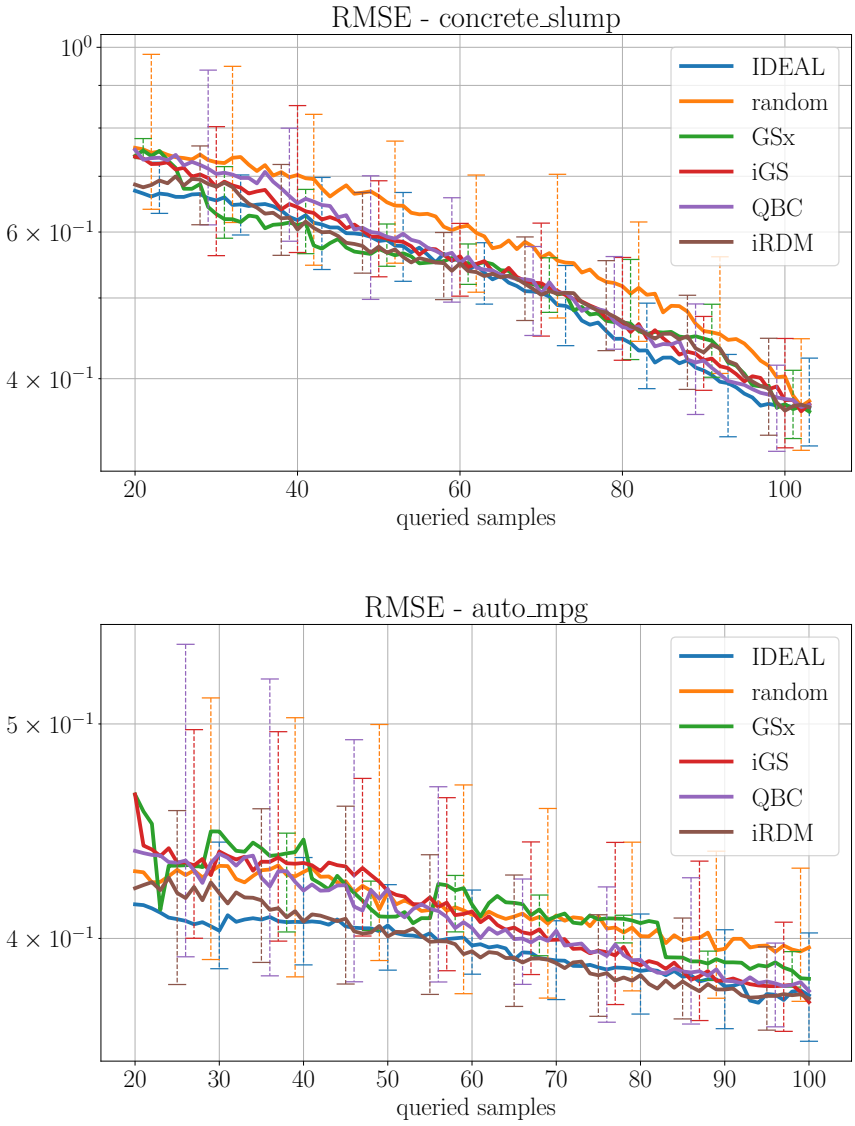

Figure 8: Regression problems, RMSE results (median and range) on `concrete-slump` (left plot), `auto-mpg` (right plot) datasets

to create a pool $\mathcal{X}_P$ containing a large but finite set of randomly-selected feature vectors and use pool-based AL instead, i.e., to optimize the sample acquisition problem by enumeration rather than by global optimization over a continuum of values. Our proposed method also seems to be particularly advantageous to learn functions that have plateaus (this would be the case if applied to classification problems), because the IDW uncertainty terms tend to be small

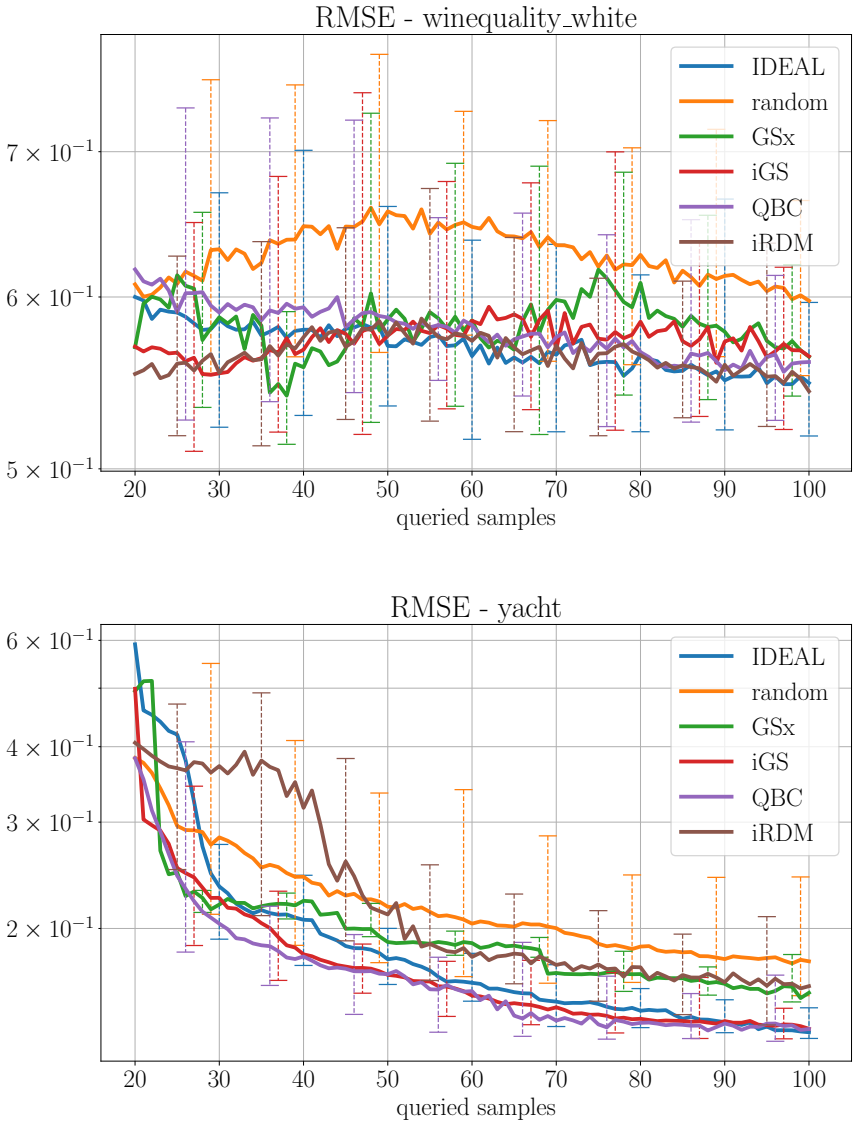

Figure 9: Regression problems, median RMSE results (median and range) on `winequality-white` (left plot), `yacht` (right plot) datasets

in regions of the feature-vector space where the acquired targets have similar values. While this is an advantage, it may also endanger the method, as it may lead to miss areas of significant change in the underlying function. For this reason, as for global optimization using surrogate functions, we found that a safeguard is to have a large-enough weight δ on pure exploration, which is entirely independent of the target values acquired and the predictor learned.

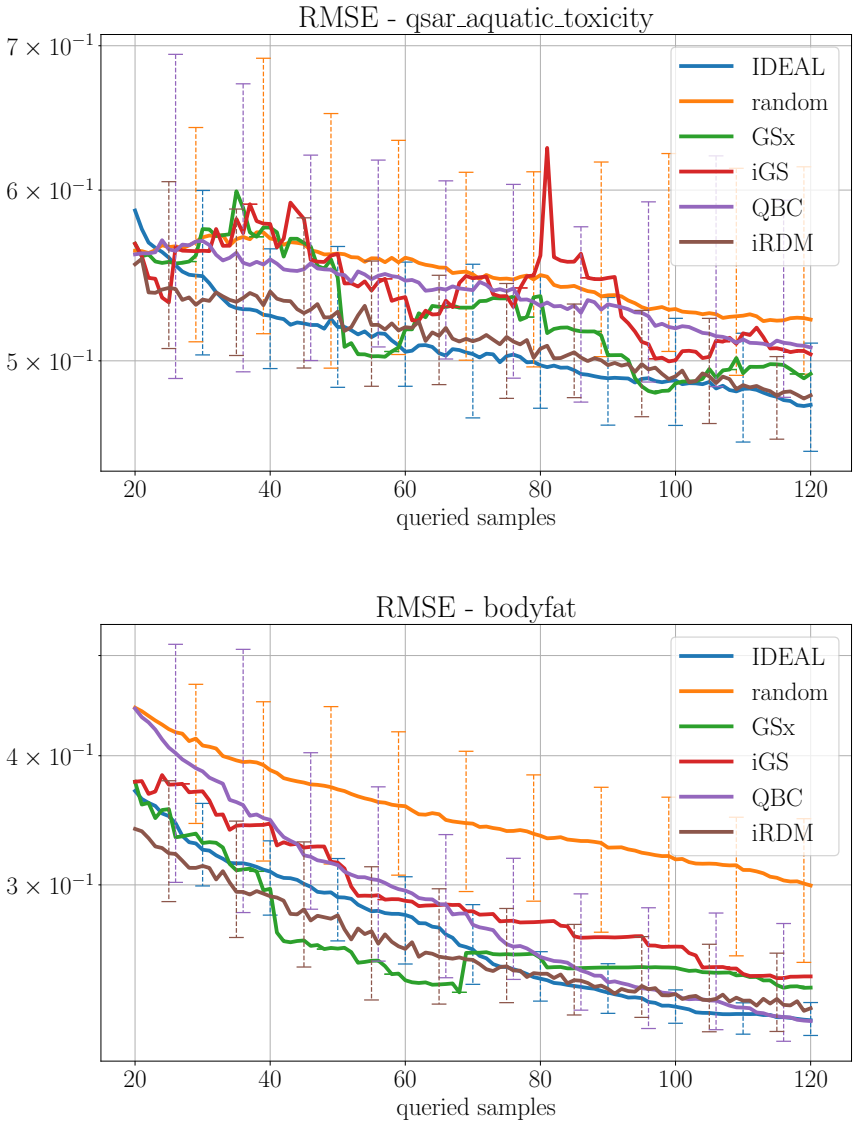

Figure 10: Regression problems, median RMSE results (median and range) on `qsar-aquatic-toxicity` (left plot), `bodyfat` (right plot) datasets

As mentioned at the beginning of Section 3.1, unsupervised AL (such as GS_x, iRDM, K-means, or simply random) is sometimes superior to model-based AL (such as ideal, iGS, QBC), see for example Figures 10–11. It would be interesting to investigate the combination of efficient unsupervised AL and model-based AL methods, in particular use iRDM to perform the initialization phase of ideal.

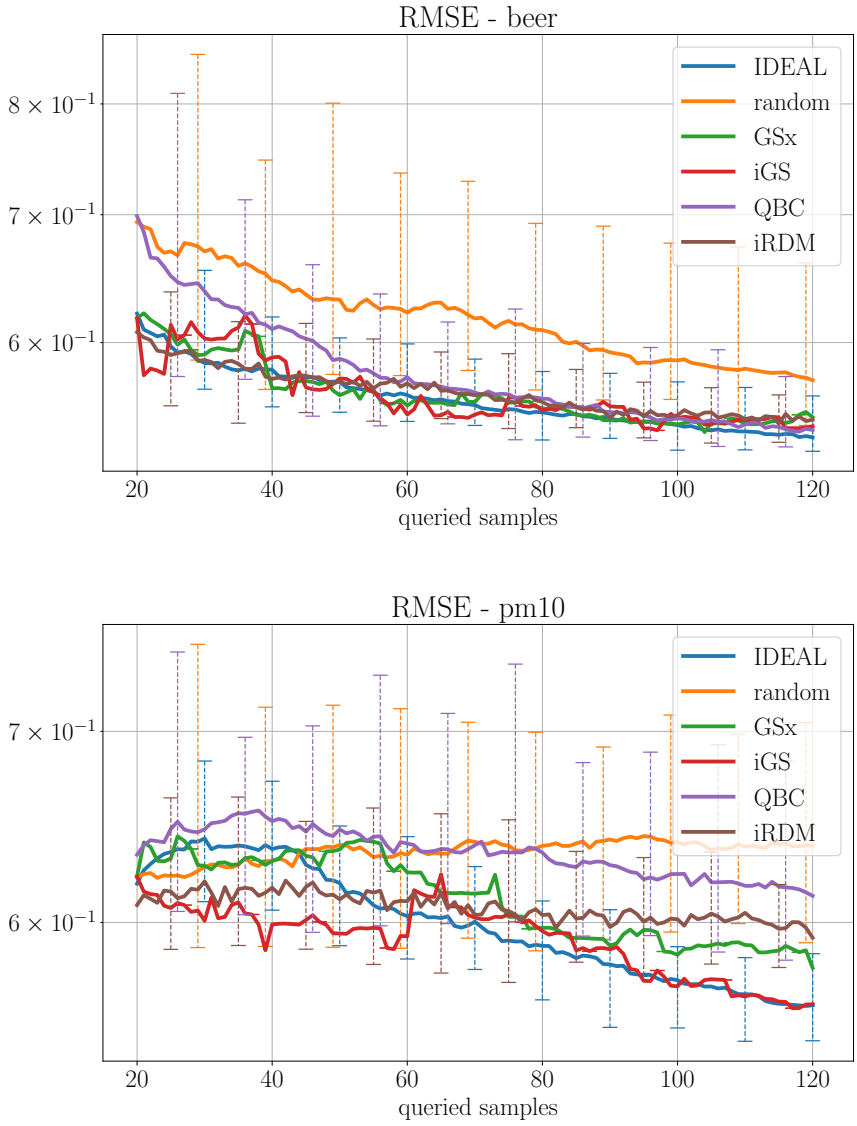

Figure 11: Regression problems, median RMSE results (median and range) on `beer` (left plot), `pm10` (right plot) datasets

We also remark that a rather high variance can be observed when applying all the methods considered in our numerical tests. There are several reasons for this. First, when K-means is applied for initialization, the final cluster centroids found may depend heavily on their initial values, due to the fact that K-means is a coordinate-descent method that is not guaranteed to reach a global minimum. Moreover, in the case of active learning of neural network

models, additional variance is due to the non-convexity of the learning problem, which may lead to largely different prediction models depending on the random initial values of the trained weights/bias terms. Further variance is suffered by QBC due to the random generation of bootstrap samples.

Future research will be devoted to analyze in depth the use of ideal to solve classification problems, to adapt the weight on the exploration term δ automatically while learning, to extend the method to streaming data to support online learning problems, and to active learning for identification of dynamical systems.

References

[1] C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. Yu. Active learning: A survey. In C. Aggarwal, editor, *Data Classification: Algorithms and Applications*, chapter 22, pages 572–605. Chapman and Hall/CRC Press, 2014.

[2] A. Bemporad. A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. *IEEE Trans. on Automatic Control*, 60(11):2892–2903, 2015.

[3] A. Bemporad. Global optimization via inverse distance weighting and radial basis functions. *Computational Optimization and Applications*, 77:571–595, 2020. Code available at `http://cse.lab.imtlucca.it/~bemporad/glis`.

[4] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1): 3–20, 2002.

[5] A. Bemporad and D. Piga. Active preference learning based on radial basis functions. *Machine Learning*, 110(2):417–448, 2021. Code available at `http://cse.lab.imtlucca.it/~bemporad/glis`.

[6] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[7] G. Box, W. Hunter, and J. Hunter. An introduction to design, data analysis, and model building. *Statistics for experimenters*, pages 374–434, 1978.

[8] R. Burbidge, J. Rowland, and R. King. Active learning for regression based on query by committee. In *Int. Conf. on Intelligent Data Engineering and Automated Learning*, pages 209–218, 2007.

[9] W. Cai, M. Zhang, and Y. Zhang. Batch mode active learning for regression with expected model change. *IEEE Transactions on Neural Networks and Learning Systems*, 28(7):1668–1681, 2017.

[10] W. Cai, Y. Zhang, and J. Zhou. Maximizing expected model change for active learning in regression. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 51–60, 2013.

[11] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[12] B. Demir and L. Bruzzone. A multiple criteria active learning method for support vector regression. *Pattern recognition*, 47(7):2558–2567, 2014.

[13] F. Douak, F. Melgani, and N. Benoudjit. Kernel ridge regression with active learning for wind speed prediction. *Applied energy*, 103:328–340, 2013.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD-96*, pages 226–231, 1996.

[15] R. Fisher. *The Design of Experiments*. Oliver & Boyd, Edinburgh, 1935.

[16] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowledge and information systems*, 35(2):249–283, 2013.

[17] Q. Jin, M. Yuan, S. Li, H. Wang, M. Wang, and Z. Song. Cold-start active learning for image classification. *Information Sciences*, 616:16–36, 2022.

[18] V. Joseph and L. Kang. Regression-based inverse distance weighting with applications to computer experiments. *Technometrics*, 53(3):255–265, 2011.

[19] B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

[20] S. Kee, E. Del Castillo, and G. Runger. Query-by-committee improvement with diversity and density in batch active learning. *Information Sciences*, 454:401–418, 2018.

[21] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[22] P. Kumar and A. Gupta. Active learning query strategies for classification, regression, and clustering: a survey. *Journal of Computer Science and Technology*, 35(4):913–945, 2020.

[23] H. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[24] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[25] Z. Liu, X. Jiang, H. Luo, W. Fang, J. Liu, and D. Wu. Pool-based unsupervised active learning for regression using iterative representativeness-diversity maximization (iRDM). *Pattern Recognition Letters*, 142:11–19, 2021.

[26] S. Lloyd. Least square quantization in PCM. *Bell Telephone Laboratories Paper. Also published in IEEE Trans. Inform. Theor., vol. 18, n. 2, pp. 129–137, 1982*, 1957.

[27] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[28] M. McKay, R. Beckman, and W. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[29] T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] T. RayChaudhuri and L. Hamey. Minimisation of data collection by active learning. In *Proc. Int. Conf. on Neural Networks*, volume 3, pages 1338–1341, 1995.

[32] L. Rios and N. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[33] N. Roy and A. McCallum. Toward optimal active learning through Monte Carlo estimation of error reduction. In *Proc. 18th Int. Conf. Machine Learning (ICML)*, volume 2, pages 441–448. Williamstown, MA, USA, 2001.

[34] B. Settles. Active learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, number 18. Morgan & Claypool Publishers, 2012.

[35] H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. 5th Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

[36] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[37] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proc. ACM National Conference*, pages 517–524. New York, 1968.

[38] M. Sugiyama and S. Nakajima. Pool-based active learning in approximate linear regression. *Machine Learning*, 75(3):249–274, 2009.

[39] L.-L. Sun and X.-Z. Wang. A survey on active learning strategy. In *Int. Conf. on Machine Learning and Cybernetics*, volume 1, pages 161–166, 2010.

[40] L. Wang, X. Hu, B. Yuan, and J. Lu. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 147:426–434, 2015.

[41] D. Wu. Pool-based sequential active learning for regression. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1348–1359, 2019.

[42] D. Wu, C.-T. Lin, and J. Huang. Active learning for regression using greedy sampling. *Information Sciences*, 474:90–105, 2019.

[43] H. Yu and S. Kim. Passive sampling for regression. In *IEEE Int. Conf. on Data Mining*, pages 1151–1156, 2010.