Population-based Active Learning for Black-Box Regression

M. van Deursen

Technische Universiteit Delft

TU Delft

Population-based Active Learning for Black-Box Regression

by

M. van Deursen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday October 19, 2020 at 10:00 AM.

Student number:	4480848	
Project duration:	November 25, 2019 – October 19, 2020	
Thesis committee:	Dr. M. T. J. Spaan,	TU Delft, supervisor
	Dr. ir. H. Farah,	TU Delft
	Dr. M. Loog,	TU Delft

TU Delft

Preface

This research is the result of my graduation work for the Master Computer Science at the Delft University of Technology.

Unlike any other experiences during my studies, both my thesis and my separate literature survey have shown me what real research entails. Prior experiences focused on learning concepts and then applying them to suitable and solvable problems. Although this definitely allows for knowledge to be obtained from the taught subjects, creating new concepts for problems requires another level of knowledge. During my thesis I have experienced this situation, which seemed like a daunting task at first. I have used literature on related problems together with my own intuition and gathered knowledge to come up with a solution for a problem for which no previous solutions were described. Doing this all without any strict external motivators, was a daunting but in the end rewarding task. As this thesis comes to an end, my appreciation of others that continue in this line of research has increased.

I want to thank all members of my graduation committee for their advice throughout the process of creating this thesis. In particular I would like to thank Matthijs Spaan as my supervisor. Without your insight, critical thinking, and elaborate feedback this thesis would not have achieved the same level it has today.

Lastly, I want to thank my parents for helping me achieve this level of education and all of my friends, family, and Elisa for all their support.

M. van Deursen
Delft, September 2020

Contents

1 Introduction

The construction of large projects, for example a new section of automotive infrastructure, goes paired with a tremendous cost. It is therefore key that the design of these projects is well thought-out and extensively evaluated to make sure that costs are not made in vain. To allow for inexpensive evaluation of these projects, a simplified representation of reality can be used to assess a design's proficiency. This simplified representation can be achieved using simulation software to emulate the realistic response to the design when actually implemented.

The simulation relies on models that approach realistic (human) behaviour whilst requiring a small amount of computation for quick and inexpensive evaluation. In the automotive industry, the field of Traffic Simulation researches these models, each type of model focusing on an aspect of operating a vehicle, such as steering, speed control, or controlling the car's signals [1]. Within this thesis we focus on the car-following aspect of controlling a vehicle [1]. Figure 1.1 depicts the conceptual and minimalistic situation. A single road is considered on which two cars are placed. The car in front is called the *leader* and travels at a constant speed. The following car is in turn called the *follower* and travels at a current speed. A model can then be used to determine the acceleration of the follower in the subsequent timestep, given the vehicles' speeds and the distance between the two vehicles.

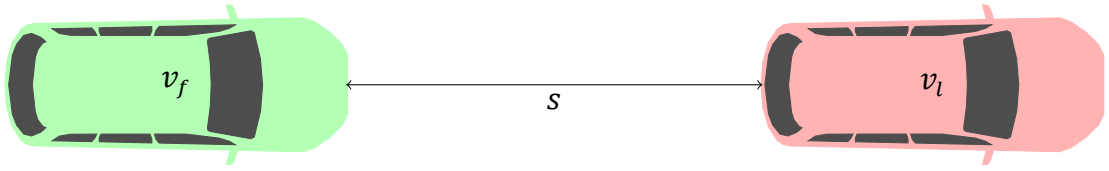


Figure 1.1: Depiction of the car following setup. The red car is the leader travelling at the constant velocity v_l and the green car the follower travelling at the velocity v_f. They are a distance s apart. The task for the model is to predict the acceleration of the follower in the next timestep, using v_l, v_f, and s.

For this model and models alike it is key that realistic behaviour is indeed resembled to allow for meaningful evaluation of designs. Real-life data samples should therefore be used to ensure that the model emulates the reality. In the car following situation, these samples consist of the two speeds and distance, as well as the acceleration of the follower in the subsequent timestep, where the acceleration is deemed the *label* of the sample. These samples can be used to both train the model and evaluate its proficiency in emulating realistic behaviour.

In a general sense the problem at hand is that of *Supervised Machine Learning* (SML). SML deals with extracting information from a large number of *labeled* samples. These labels can be categorical (e.g. whether the sample is an apple or a pear) or numerical, as is the case in the car following situation. A model can then be fitted on this set of labeled samples to mimic this dataset. Moreover, the model should also generalize over this dataset, providing accurate predictions on unseen samples.

The question that inevitably follows is then how to acquire this dataset. It is key that this dataset contains much and useful information to allow for a realistic model. Depending on the origin of the dataset, obtaining samples and especially labels for these samples can be a costly task [2]. When the label for a sample comes at little to no cost, for example a spam label on a forum post, this process'

cost is negligible. In many situations obtaining this label is however a more sophisticated, extensive, and therefore expensive task. To obtain this data within the current setting a vehicle has to be fitted with sensory equipment to gauge its surroundings, and a human driver has to drive this vehicle to allow for data collection. Instead one might also opt for a driving simulator to allow the human driver to drive around in a controlled environment and collect data that way.

In both situations and situations alike, it is clear that collecting a lot of data is a time-consuming task. It is therefore of utmost importance that resources used for obtaining the data are used wisely and efficiently. That is, make sure that the labeled samples are significant for the learning process. This would in turn result in fewer labeled samples required to achieve a certain performance compared to labeling random samples. In our situation this would mean that only those combinations of speeds and distance are considered which provide useful information to the model. For SML, the subfield of *Active Learning* (AL) provides an approach that solves this problem. The aim of AL is to label those samples that allow the model to learn well with as little samples as possible, in turn reducing the labeling effort and providing a more efficient learning process. This is done through an iterative process during which the model is trained. In each iteration, the model is trained on the current training set, a new sample is selected to be labeled, and this labeled sample is added to the training set. The sample that is selected is determined through a *selection strategy*. The label is provided by an *oracle*, an entity which provides the true label to a sample. In most practical applications this oracle is human, such as in the driving simulator. There have however also been cases where instead a robot acts as an oracle, executing a series of actions to obtain the label [3].

One key consideration for AL selection strategies is time efficiency. When compared to sampling randomly and labeling these samples, AL should achieve equal performance in less time to make the whole procedure worthwhile. All AL selection strategies therefore deal with a trade-off between selection quality and execution time; although it is key that the most informative samples are selected, these should be selected in a time efficient way.

Work within the AL research field can be categorized in two groups: *pool-based* and *population-based* AL [4, 5]. In pool-based AL a dataset of unlabeled samples is provided from which samples can be chosen to be labeled. Population-based AL instead provides an infinite sample space, where any sample within this space can be selected and labeled. This thesis considers population-based AL, as the car following situation allows for any speed and distance to be attained within the physical possibility of the considered vehicles. A second categorization can be made based on the possible number of labels that can be used. When a label can take a finite number of (categorical) values, the AL considers a classification task. Regression AL instead deals with a continuous label space. Since our label consists of the acceleration of the follower, which can once again attain any value within physical bounds, regression AL is considered in this thesis.

The research goal of this thesis is to find a sample selection strategy for population-based regression AL. Current work mainly focuses on pool-based solutions [2]. When population-based AL is considered in research, only classification is discussed, hence this thesis pioneers this very category within AL. This thesis should therefore take future extensibility into account for its selection strategy. Moreover, this thesis also pursues a selection strategy that is applicable to arbitrary population-based regression tasks. The research question of this thesis are therefore the following:

Research Question 1: Can a performant, efficient, extensible, and generally applicable selection strategy be created for population-based AL for regression?

- **RQ1.a**: In what fashion can previous research on pool-based regression be leveraged within this strategy?
- **RQ1.b**: How can the selection strategy be structured such that it is extensible by future work?
- **RQ1.c**: How can the selection strategy be formulated such that is generally applicable?

To answer this research question this thesis provides a modular selection strategy for population-based AL for regression. This strategy features an iterative approach to focus on an informative region within the sample space and thereby select interesting samples. It consists of a generator, prioritizer, and restrictor which respectively generates samples, scores them based on their informativeness, and restricts the sample space to an interesting subregion in each iteration. Each of these components is defined as an interface, thereby allowing new variations to be implemented in future work. To allow for this selection strategy to be generally applicable, little information of the model is used, namely only its

parameter format and its output given a sample and parameters. This is known as a *black-box* model and allows for virtually any model to be used with the suggested strategy. Previous work on pool-based regression can be employed as a prioritizer, as long as this work is applied with black-box models in mind. This thesis also provides multiple implementations of the generator, prioritizer, and restrictor interfaces.

To determine the performance of the proposed strategy, first different combinations of implementations are compared against one another to determine the relative performance. The performance of the proposed strategy is then compared against a baseline which randomly selects a sample from the sample space. The selection strategy generally outperforms this baseline, both in performance at a specific number of labeled samples, as well as at a specific execution time. This difference in performance becomes even more apparent when a high labeling effort is considered.

The remainder of this thesis is organized as follows. First, Chapter 2 defines the problem through introducing mathematical notation and relevant terminology which is used in the remainder of this thesis. Chapter 3 then provides an overview of previous work on both population-based AL and pool-based AL for regression. Chapter 4 discusses the proposed modular selection strategy of this thesis in detail. This strategy is then further used in Chapter 5, where the experimental setup is explained and experiments are ran and discussed to create an overview of the strategy's performance. Chapter 6 then features a discussion on this thesis along with possible future extensions. Lastly, Chapter 7 concludes this thesis, summarizing its approach and findings.

2

Problem Definition

In this chapter the framework in which this thesis' selection strategy is placed is defined mathematically. This framework is an extension and adjustment from that provided by Cai et al. [6]. Cai et al. focus their definitions mainly on their strategy, where this chapter provides a more detailed and general definition.

Mathematical notation within this thesis follows a certain format. A scalar value is denoted through an unbold uncapitalized letter, e.g. x. A vector value on the other hand is a bold uncapitalized letter, e.g. $\mathbf{x}$. Matrices are denoted by a capital letter, e.g. X. The scalar i depicts an integer, i.e. $i \in \mathbb{N}$, and is used as the *index* of an element. To ease the notation, this fact is not included in subsequent mentions of the index. Lastly, please note that every time the index i is used, the sets and lists are 1-indexed.

2.1. Supervised Learning

In the field of *Supervised Machine Learning* (SML), the objective is to learn a concept from a set of *samples* and corresponding *labels*. A sample $\mathbf{x}$ is a vector of d values, i.e. $\mathbf{x} = [x_1, x_2, \ldots, x_d]$, where d denotes the dimension of the sample. In essence each individual value x_i can be numerical or categorical. The set $\mathcal{X}$ depicts the sample space and corresponds to the set of all feasible samples $\mathbf{x}$. That is, given a problem, only the samples $\mathbf{x} \in \mathcal{X}$ are relevant to this problem. If $\mathcal{X}$ is finite, the setting is called *pool-based*, as there is only a finite pool of available samples [5]. In the case that $\mathcal{X}$ is instead infinite, the setting is called *population-based*. Similarly the output space is denoted $\mathcal{Y}$, with $y \in \mathcal{Y}$ an element of this space. Note that in the problems considered within this thesis the output space is one-dimensional and in turn the resulting elements are scalar. When an output y belongs to $\mathbf{x}$, y is the *label* of $\mathbf{x}$. Which label belongs to a sample $\mathbf{x}$ is determined through an *oracle* $\mathcal{O}$, an entity that *labels* a sample $\mathbf{x}$ with its corresponding label y. The oracle could therefore be seen as a function mapping from the sample to the output space:

$$\mathcal{O} : \mathcal{X} \to \mathcal{Y}. \tag{2.1}$$

Note that the oracle $\mathcal{O}$ can be noisy, hence the label it provides can differ when asked repetitively. Let $P(y \mid \mathbf{x})$ denote the conditional probability that the oracle $\mathcal{O}$ labels $\mathbf{x}$ with y. The oracle is not noisy iff. there is only one possible label y for each possible $\mathbf{x}$:

$$\forall \mathbf{x} \in \mathcal{X} \exists y \in \mathcal{Y} : P(y \mid \mathbf{x}) = 1 \tag{2.2}$$

A sample $\mathbf{x}$ is *labeled* if the actual label y given by the oracle is known by the algorithm and is *unlabeled* otherwise. When the output space $\mathcal{Y}$ is finite, the concept to learn is a classification concept. In the case where $\mathcal{Y}$ is infinite, instead the problem at hand is instead a regression task. Within this thesis, the problems that are dealt with are population-based regression tasks, i.e. both $\mathcal{X}$ and $\mathcal{Y}$ are infinite.

The SML approach consists of a model M, which is a function that maps from the sample to the output space. To tune the model's behaviour it is parametrized by a set of parameters, which is denoted θ. Note that θ can contain any number of elements, depending on the available parameters of M. Denote all feasible sets of parameters Θ, hence $\theta \in \Theta$. We can therefore denote the model M parametrized by θ as:

$$M^{\theta} : \mathcal{X} \to \mathcal{Y}. \tag{2.3}$$

Within this thesis, the only information that is known about the model is its predicted label y' given a sample $\mathbf{x}$ and a set of parameters θ as well as the feasible parameter set Θ. This is done to make the thesis generally applicable, rather than requiring a specific model to be used. When only this information is known and no further information on how the model's prediction came to be, the model is said to be a black-box model. The relation between $\mathbf{x}$, θ, and y' is denoted:

$$M^{\theta}(\mathbf{x}) = y' \tag{2.4}$$

In this case, we say that the model M^{θ} *predicts* the label for $\mathbf{x}$ to be y' and for y' in turn being the model's *prediction* for $\mathbf{x}$.

The objective within SML is to learn the optimal set of parameters θ^* such that the generalization error over the sample space $\mathcal{X}$ is minimized [6]:

$$\theta^* = \arg\min_{\theta \in \Theta} \int_{(\mathbf{x},y)\in\mathcal{X}\times\mathcal{Y}} \ell(M^{\theta}(\mathbf{x}), y) P(y \mid \mathbf{x}) d(\mathbf{x}, y), \tag{2.5}$$

where $\ell(\cdot,\cdot)$ is a given loss function. In practice the conditional probability of the oracle's labeling is not known. Moreover, it might be that solving the integral is infeasible even when this conditional probability is known. Therefore, θ^* cannot be directly computed and is not known. SML instead attempts to approximate θ^* using a provided training set $\mathcal{D}_t$, which consists of t labeled samples:

$$\mathcal{D}_t = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, 0 < i \leq t\} \tag{2.6}$$

This training set is then used to train the model, i.e. to calculate an optimal approximation of θ^*, which is denoted $\theta'_{\mathcal{D}_t}$:

$$\theta'_{\mathcal{D}_t} = \arg\min_{\theta \in \Theta} \sum_{(\mathbf{x},y)\in\mathcal{D}_t} \ell(M^{\theta}(\mathbf{x}), y) \tag{2.7}$$

The objective of SML is to allow an as good approximation of θ^* as possible:

$$\min |\theta'_{\mathcal{D}_t} - \theta^*| \tag{2.8}$$

To quantify the objective of SML, a fitness of $\theta'_{\mathcal{D}_t}$ can calculated by introducing a test set $\mathcal{T}$ containing m uniformly random samples sampled from the sample space and labeled by the oracle:

$$\mathcal{T} = \{(\mathbf{x_i}, \mathcal{O}(\mathbf{x_i}))\}_{i=0}^{m} \tag{2.9}$$

This test set is not considered during the training of the model and therefore consists of unseen samples. The fitness can then be expressed as the error of the trained model to this test set:

$$f = \sum_{(\mathbf{x},y)\in\mathcal{T}} \ell(M^{\theta'_{\mathcal{D}_t}}(\mathbf{x}), y). \tag{2.10}$$

Hence a lower fitness represents a smaller error, and hence a better approximation of θ^*.

2.2. Active Learning

Note that in Equations 2.8 and 2.10 the quality of the approximation $\theta'_{\mathcal{D}_t}$ is highly dependent on the samples within the training set $\mathcal{D}_t$. It is therefore key that the labeled samples within this set are carefully selected. Moreover it is key that the training set contains as few labeled samples as possible, while still maintaining a good approximation, to minimize labeling effort. This is the case especially if the labeling effort is high, i.e. when $\mathcal{O}(\mathbf{x})$ is expensive to compute. Active Learning aims to select these samples in an iterative manner, such that at each timestep t the SML objective in Equation 2.8 is optimized [4, 7]. At a timestep $t \in \mathbb{N}$ and $t \geq 1$ the previously labeled training set is $\mathcal{D}_{t-1}$, with $\mathcal{D}_0 = \emptyset$. Active Learning selects a new (currently unlabeled) sample $\mathbf{x}_t$. This sample is then labeled as y_t by the oracle ($\mathcal{O}(\mathbf{x_t}) = y_t$), after which it is added to the training set such that:

$$\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\} \tag{2.11}$$

The objective of AL is therefore to sample $\mathbf{x}_t$ such that $\mathcal{D}_t$ provides a good approximation of θ^* as possible:

$$\mathbf{x_t} = \arg\min_{\mathbf{x}\in\mathcal{X}} |\theta'_{\mathcal{D}_{t-1}\cup\{(\mathbf{x},y)\}} - \theta^*|, \tag{2.12}$$

where y is the label of $\mathbf{x}$. The manner in which the AL approach selects this sample is called the *selection strategy*. This selection strategy both has to be performant and time efficient. It has to be time efficient to ensure that the AL procedure is worthwhile. If similar fitness of the model can be achieved with labeling samples in a comparable, or even worse a smaller, amount of time, it is not worth performing the sample selection in the first place. Note that this shifts the objective of AL compared to SML; where SML focuses on providing a good approximation given a training set, this is not necessarily important for AL. AL instead focuses on selecting the best unlabeled sample to be labeled given such approximation approach in a timely fashion.

In Equation 2.12 the label y is not known at the point of selection and therefore it cannot be solved directly. All AL strategies therefore rely on heuristics to select the sample $\mathbf{x}_t$ to be labeled [4]. These heuristics describe criteria to determine which unlabeled sample is the best to label. Wu [4] describes three different categories for pool-based AL strategies into which these criteria fall:

- **Representativeness**: The representativeness category consists of strategies which quantify the sample's representability of the complete sample space $\mathcal{X}$. In the pool-based setting, this sample space is finite, which allows for computations to be made in regards to this criteria, such as clustering of the samples [8]. However, within the population-based learning setting the probabilistic density of the sample space has to be known and non-uniform to take this criteria into account. Since this density is not known within the context of this thesis and uniform density is assumed, this category cannot be applied in this work.

- **Informativeness**: The informativeness category contains strategies which quantify a sample's information gain for the model at that point in time. This strongly relates to the certainty of a model's predictions in a certain region of the sampling space.

  To illustrate this, Figure 2.1 provides an example situation for a one-dimensional sample space. When looking at the available samples A through D to be selected, the informativeness category specifies to choose point D, as the model is uncertain within this region.

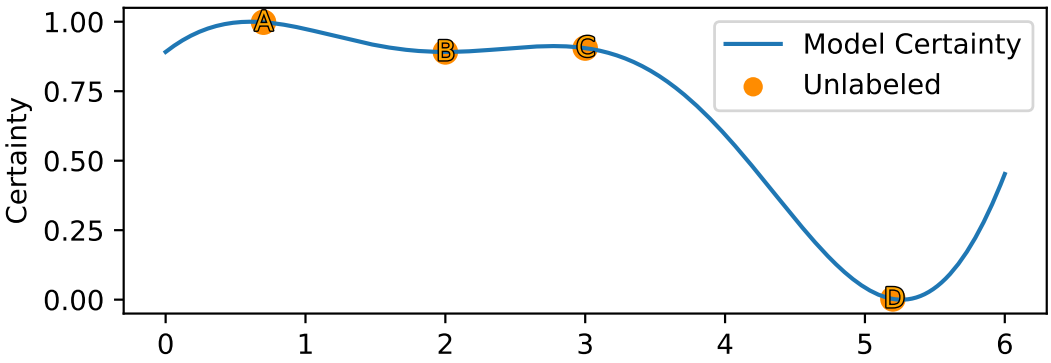


Figure 2.1: Illustration of the notion of informativeness. The horizontal axis represents a one-dimensional sample space and the vertical axis the certainty of the model, where 0 signifies the model being uncertain and 1 the model being absolutely certain. The orange points A through D resemble the choice of unlabeled samples for selection. The blue line resembles the model certainty throughout the sample space. The informativeness strategies will choose D, as the model is most uncertain in this sample.

.

- **Diversity**: The diversity category consists of strategies that quantify the diversity of a sample to the current training set $\mathcal{D}_{t-1}$. The reasoning is to allow for samples to be spread throughout the whole sample space, rather than being concentrated in one subregion of this space. Often a distance metric is used to quantify this diversity, such as the Manhattan or Euclidean distance.

Figure 2.2 illustrates this category of strategies. When dealing with a training set which is clustered, a diversity strategy will choose the sample that is furthest away from this cluster. In Figure 2.2 this furthest away point corresponds to C.

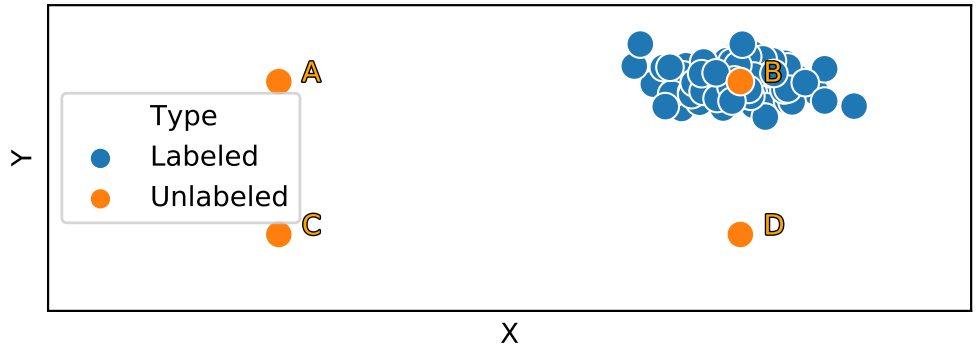


Figure 2.2: Illustration of the notion of diversity. The figure represents a two-dimensional sample space. The blue points are part of the current training set $\mathcal{D}_{t-1}$ and the orange points A through D are the unlabeled samples from which one can be chosen. A diversity strategy will choose C as the distance from the labeled samples is greatest. Note that A and D would also still be possible choices based on the distance metric used. However, since B is within the cluster of labeled samples, it should not be selected in this category of strategies.

Note that an AL strategies can fall into one or more categories. This is often done through a weighting function which factors in criteria from different categories, or a leveled strategy where first one category is taken into account and another one afterwards.

As stated before, opposed to pool-based AL, population-based AL has an infinite sample space. Therefore the calculation of aforementioned heuristics on all samples is infeasible for population-based AL, and instead only a subset of the sample space can be evaluated. Population-based AL strategies therefore face an additional challenge compared to pool-based AL: which samples within the sample space are to be evaluated, whilst still remaining time-efficient. To take care of this additional challenge, this thesis finds these interesting samples in an iterative fashion, where it iteratively narrows down onto an interesting region within the sample space. This is determined through using both diversity and informativeness heuristics, considering a black-box model.

3

Previous Work

In Chapter 2 the problem of this thesis is defined mathematically. This problem is that of population-based AL for black-box regression. The subject of AL has obviously received attention previously in the scientific community. In this Chapter this interest is related to the problem scenario used in this thesis. Note that this survey uses the extensive survey of Settles [2] as a basis, to which the interested reader is referred.

Although many research focuses on Active Learning, no work has been encountered that covers the complete scenario of both infinite sample and output space, whilst dealing with a black-box model and without knowing any data prior to initialization. Instead this Chapter is therefore separated in two sections, where previous work dealing with either an infinite sample or output space respectively. That is, the first section describes and highlights population-based Active Learning and the second section is instead focused on pool-based Active Learning for regression. Within both sections, both selection strategies for white- and black-box models are discussed, where white-box models are those where all information of a model is known, including the underlying prediction mechanism. Lastly, this previous work is contrasted to this thesis' contribution, namely the modular selection strategy for population-based regression AL.

3.1. Population-Based Active Learning

The work in the population-based AL field focuses on an infinite sample space, from which each iteration a sample is selected [5]. Classically the population-based AL is also named Membership Query Synthesis [9]. Not much work focuses on population-based AL to start with, as it has historically been found to not be applicable to all usecases [10]. That is, when human oracles are used, the selected samples have to be recognizable and distinguishable to the human oracle. Instead it was shown by Lang and Baum [10] that this was not the case for digit classification, where the generated images were often ambiguous to the oracle, as they formed a mix between two digits. This problem has partially been improved on by using Generative Adversarial Networks (GAN) in more recent work [11–13], but even then the problem is not solved for all possible situations [14]. It should therefore be noted that population-based AL can only be applied where all possible labels can be feasibly determined without ambiguity.

As stated previously, there has been no explicit work on regression tasks within the population-based AL setting, instead focusing on classification tasks. In a scientific setting, often a binary classification setting is used to demonstrate the suggested approach, where each sample is either of the positive or negative class [13, 15–18]. Moreover, it is often assumed that the data is separable, i.e. that a boundary can be drawn such that the two classes of samples are separated. From this, a straightforward sample selection strategy is then to select a sample on the boundary [15–18]. Selecting and labeling this sample always finetunes the boundary, since it shows to which side the boundary is to be moved in order to still be consistent with all observations. The main novelty in this type of research is therefore also to be found in applying this strategy to real-life situations [18], by further optimizing the approach by further comparisons of samples on the boundary [15], or by including noise in terms of false positives and negatives [16, 17].

There are some solutions that do allow being applied to regression tasks. However, these papers share the assumption that the sample space density is known or a representative dataset is supplied [19–23], which is a fact this thesis does not assume. The main advantage that this assumption has is that this distribution can then directly be used to generate samples [20, 22], or can be used further in providing a representativeness weight for a sample [23].

3.2. Pool-based Active Learning for Regression

A lot of work in the AL field focuses on the pool-based setting, where the sample space $\mathcal{X}$ is a finite dataset of unlabeled samples, of which each iteration one sample is selected to be labeled. Although not directly applicable to the problem at hand, the methodology and thought process behind the described pool-based regression approaches are still useful. To categorize these approaches, the categories of Wu [4] are used as defined and described in Section 2.2.

Note that some of the discussed approaches are not limited to one of the three categories, instead using a multitude of categories as their selection criteria [4, 8, 24–32]. To combine different categories, either a user weighting is used [24, 25] or are performed sequentially to allow to prioritize one metric over another [4, 27].

3.2.1. Informativeness

Much of the reviewed work focuses on incorporating an informativeness metric within their approach directly. As stated previously in Section 2.2, this metric mostly relates to the certainty of a model's prediction given an unlabeled sample.

Direct Uncertainty Computation

A model on which uncertainty is often computed is a *Gaussian Process* (GP) [28–30, 33–48]. Originally described by Jones et al. [41], the GP uses the training set to arrive at a stochastic representation of the underlying response surface. For a sample $\mathbf{x}$ the GP then provides a prediction consisting of both a mean and variance [49]. Both are computed through correlating the unlabeled sample with the samples in the training set, where the labeled samples in the vicinity are considered to be more important to the prediction, as similar samples are assumed to have similar labels. If there is a labeled sample close to $\mathbf{x}$, the model is already more certain than if this is not the case, as then more distant points have to be taken into account. This is captured within the variance of the GP prediction. This variance therefore forms an easy way to retrieve the uncertainty of the model, hence its popularity in research work. Most commonly this variance is used directly in different usecases [29, 33–35, 37–40, 42–44, 46], such as predicting ground depth from satellite images [40], risk estimation in engineering systems [35], or for estimation of biophysical parameters [29]. Some other work introduces some additional constraints to the problem, such as allowing only safe samples to be sampled in case of radiation [45]. Although the GP's variance is often used without any further measures, it can also be used in combination with other metrics to allow a more extensive procedure. Previous work includes examples of this, such as first applying a clustering algorithm to create a subset of cluster centers which are representative for the dataset before comparing them on their GP variance [28]. Instead these criteria could also be related to the environment in which the approach is used, such as taking the distance to the labeling entity into account to reduce travel time [30]. In this case, the metric is definitely usecase specific and only relates to the samples if these resemble a geographical location. Lastly, when the objective of the task is to in the end find a maximum label value, one can opt to instead use the predicted values to its advantage [36, 47], thus looking at maximum (expected) predicted values.

Other models for which the uncertainty metric has been investigated include Linear Regression models [50–54], binary trees [55], random trees [31], neural networks [56, 57], and support vector machines [8, 32]. Most of these approaches use a measure to estimate the variance of a sample, similar to the GP approaches discussed above. This variance can for example be calculated by random dropout of neural network nodes in neural networks [56] or by including an explicit term for the variance at each point [32, 51, 52, 54, 55].

Expected Model Change

Another take on the informativeness of a sample to the model is to focus on the induced expected change labeling a sample has on the model. This measure is called Expected Model Change Maximization (EMCM) [6, 7, 57–59]. The sample that maximizes this metric is selected since it contributes

most to the Active Learning process of quickly approaching the optimal set of parameters compared to the other unlabeled samples [6]. Although this metric does not require a particular model definition per se, it does require for the model's change in parameters after labeling a sample to be available in closed form for the computation to be feasible [58]. If this is not the case the model's expected change has to be computed directly through fitting the model within each hypothetical value, which is practically impossible given the infinite number of possible label values. Previous work therefore uses models for which this closed-form is available [6, 7]. A similar approach is to choose the point that alters the model's predictions the most for similar reasoning [58].

Black-box Informativeness Metrics

The previous metrics all assume white-box models and are often restricted to specific models or classes of models. There is however one technique that can be used to approximate an uncertainty or expected model change, which is Query By Committee (QBC) [60, 61]. QBC employs a group of models, which are the members of the committee, to estimate informativeness metrics [4]. This is done through subsampling the training set with possible duplicates such that each model has a different training set to fit its parameters on, resulting in different predictions from the models. Note that this has a strong relation with bagging or boosting [62], where the equal subsampling also occurs. However, where bagging or boosting provides a regressor, returning the mean prediction of the committee, QBC instead focuses not on this prediction directly. QBC rather focuses on the informativeness metrics which can be computed from these predictions. Different metrics can then be computed from the predictions of the committee for the unlabeled samples. These metrics include the aforementioned variance [4, 25, 63–65] and EMCM [4], but other work uses additional metrics such as Shannon information criterion [66], Hotellings T^2, or the Q-residual [67]. Apart from a difference in used metric, the subsampling procedure with different subsets of the dimensions of the sample spaces is also investigated [63].

3.2.2. Diversity

The previously discussed informativeness selection criteria is often combined with another selection criteria category [4, 8, 25–27, 29, 32, 64, 65]. Often this combination is done when an approximation for the informativeness criteria is done, for example through QBC. One of these other categories is diversity, as mentioned in Section 2.2. This category includes the metrics that take into account that an unlabeled sample is different from samples in the current training set.

The most intuitive and commonly used diversity metric used in research is that of maximum minimum distance [25, 26, 29, 32, 64, 65, 68, 69]. This means that for an unlabeled sample $\mathbf{x}$ the distances to all samples within the training set are computed [25]. Its score is then the minimum of these distances, i.e. the distance to the labeled samples within the training set. The higher this distance is, the further the sample is from the complete training set and hence the more diverse this point is. Note that in theory any distance metric can be used, although most commonly the Euclidean distance is used [25, 26, 32, 64, 65, 68]. Another distinguishing factor can be the space in which the distance is calculated. Instead of the sample space, one could instead use another space to which the sample space can be mapped and use distances within this space [68, 69]. This can be done by determining this mapping through Principal Component Analysis (PCA) [68] or by using the predicted labels of the model in combination with the sample space [69].

The other encountered method to choose diverse metrics within pool-based methods is to apply clustering on the complete sample space [4, 27]. To accomplish this, take a current labeled training set $\mathcal{D}_t$ of t samples. If one then uses a clustering algorithm to cluster the complete sample space in $t+1$ clusters, there is at least one cluster which does not contain a labeled sample. The clusters which do therefore not contain labeled samples are therefore the diverse samples and based on this samples from this cluster can be selected. To then choose a sample from these clusters additional selection criteria can be used, such as EMCM, QBC [4], or a representativeness criteria [27].

3.2.3. Representativeness

The last discussed category is that of representativeness. As explained in Section 2.2, the representativeness criteria quantify the representability of a sample for the complete sample space. This can be seen as a quantity related to the similarity of samples within the sample space to the sample.

When using a binary tree as a decision tree to split up the sample space, one can use the number of samples in the relevant subspace to define the representability of a sample [31, 55]. Since this is

an approach unique to the sample space representation of a decision tree, in other works a similar approach is not used.

Instead clustering is more often employed to quantify the representativeness of a sample [4, 8, 24, 70]. Contrary to the approach discussed in Section 3.2.2, the clustering used for representativeness does not distinguish between labeled and unlabeled samples. Instead, after the samples are clustered in a defined number of clusters, the center of each cluster is determined. To determine from which cluster should be sampled, one can then look at the number of samples within a cluster or the cluster's density [8]. After it is determined from what cluster should be sampled, the sample closest to the cluster center is sampled [4, 8, 70], or a weighting can instead be created from the distance from the cluster center [24]. Note that this approach only works when either the number of clusters is varied in each subsequent selection [4, 24, 70] or using other selection criteria to further change the outcome of the selection procedure [4, 8, 24].

3.3. Contrast to Contribution

This thesis provides a selection strategy for population-based AL for regression whilst considering a black-box model. As stated previously there is no current work available which focuses on this category specifically, showing a gap in current literature and hence a clear contrast with available solutions. Due to the properties of this problem formulation, the contrast to the available solutions become already apparent on a conceptual level. For population-based classification AL solutions, there is no focus on finding the region which houses the interesting samples, as this region is already known to be around the decision border. This thesis cannot use this notion of a decision border as it deals with a regression problem instead, where interesting regions are not known beforehand.

This allows for a similarity between the thesis' strategy and pool-based regression AL solutions; both use (a combination of) heuristics as described in Section 2.2 to determine interesting regions within the sample space. However, pool-based strategies compute these heuristics over all unlabeled samples under the assumption that this computation is feasible. In that regard this thesis deviates from this assumption since an infinite sample space is assumed and this computation would be infeasible by definition. This is instead solved by performing an iterative search, focusing on an interesting region, allowing for only a relatively small number of samples to be evaluated. Note that with a few adjustments the same notion could be used within the pool-based scenario as well, where there is a large number of samples such that performing exhaustive computation is infeasible.

Lastly, the little reliability of this thesis' strategy on a specific model type is unique, apart from the aforementioned QBC. Using this little information about the model allows this strategy to be used in any application, rather than it being limited to a specific class of models. To conclude, combining all these facts allows the proposed selection strategy to be current, novel, and generally applicable. Since there is no prior research done in population-based regression AL, this thesis moreover fills in a research gap

4

Proposed Selection Strategy

In this chapter, this thesis' selection strategy is completely outlined. This strategy, which is coined a workflow, is executed each time a sample $\mathbf{x_t}$ has to be selected, as described in Chapter 2. Since one of the goals of this strategy is to make it extensible, this workflow is formulated in a modular sense. As stated before, it uses black-box models to allow for general applicability. Firstly the different steps within this modularised workflow are explained conceptually. This is followed by a detailed explanation of the current available variations for the different steps.

4.1. Modular Workflow

As explained in Chapter 2 and more specifically Section 2.2, the goal of AL is to select a sample $\mathbf{x_t}$ which aids in approximating the optimal set of parameters the most within a timely manner. Within population-based AL, however, there is an infinite amount of samples to consider, making direct derivation of this sample infeasible. Instead, the proposed selection strategy searches through the sample space, concentrating on interesting regions within this space. That is, a region which houses samples which score highly on informativeness and/or diversity heuristics discussed in Section 2.2. The motivation behind this is that these regions should contain points that aid in the AL objective, as they are both informative and increase the coverage of the training set over the sample space. This is done in an iterative fashion, where each iteration further concentrates on this interesting region.

In work done on (binary) population-based classification, a similar iterative approach is used to find and select interesting samples [71]. Opposed to regression problems however, the region which houses these samples is known, namely the region close to the decision boundary. Since samples near this boundary are often more ambiguous to the model, selecting and labeling these samples leads to less ambiguity and consequently to a better performing model. Instead the proposed strategy finds these interesting regions in a regression problem through employing an iterative search, narrowing down the sample space to regions with interesting samples.

The proposed strategy, which is coined a *workflow*, is outlined in Figure 4.1. A workflow consists of five components: a generator G, a prioritizer P, a restrictor R, a set of continuation criteria C and a set of stopping criteria S. Each of these components is defined as an interface, allowing for different implementations of each individual component to tailor to different usecases and preferred behaviour. This allows for both pool-based regression AL solutions to be used within the workflow as prioritizers as well, leveraging previous work done in this field. Lastly, using this modularization, future work can easily extend a particular component, without ultimately having to change any of the other components. A specific instance of the workflow is denoted as follows:

$$W(G, P, R, C, S), \tag{4.1}$$

where each symbol denotes the instance of the aforementioned component.

Within an iteration, the generator G generates a set of samples from the current region, possibly taking into account a diversity metric to bias this generation. Note that in the first iteration, this region is the complete sample space. However, in subsequent iterations this region instead is a subset of the sample space, which the previous iterations have deemed to be interesting. The prioritizer P then

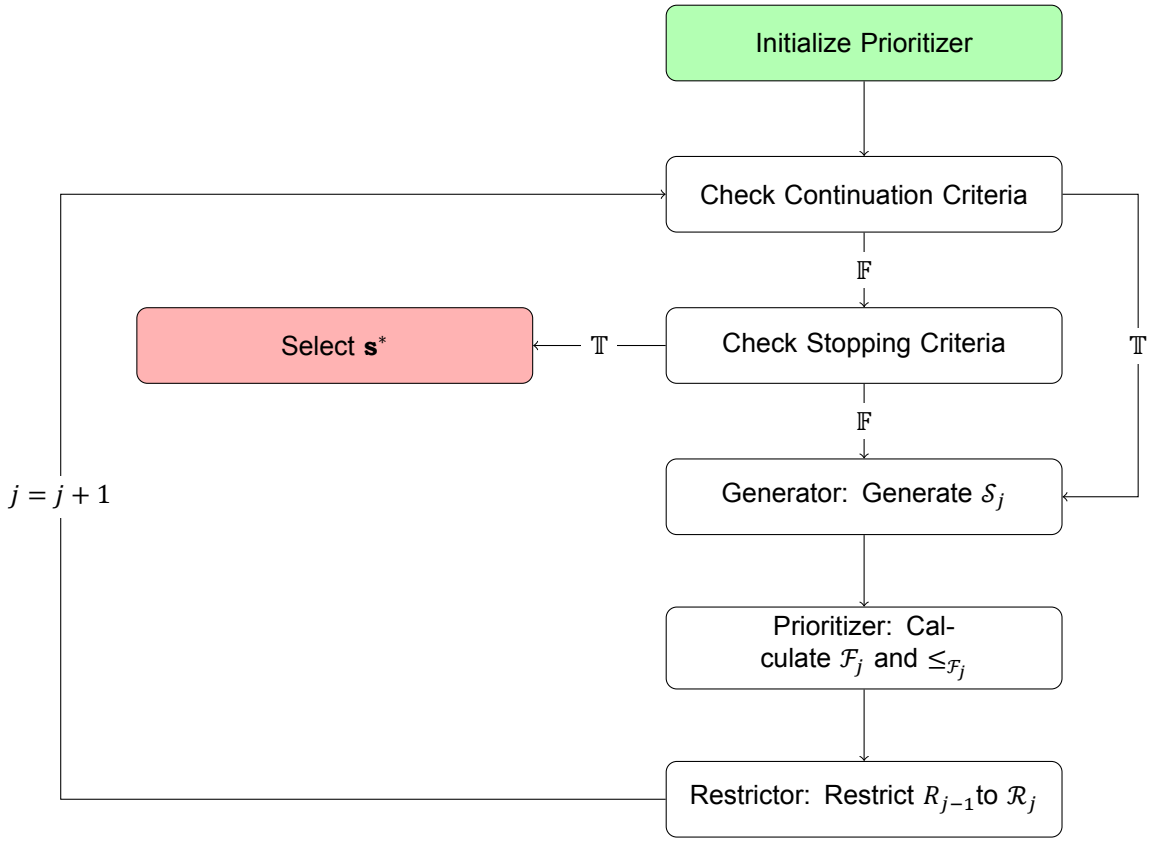


Figure 4.1: Workflow diagram of the proposed workflow. Notice the green state being the starting state where the workflow is invoked. The red state signifies the final state, where the workflow ends and the most prioritized sample is selected. The symbol $\mathbb{F}$ symbol signifies that none of the relevant criteria are true, whereas $\mathbb{T}$ signifies that one or more of these is true.

assigns a fitness score to each of these samples based on an informativeness metric. Based on these fitness scores the restrictor R then restricts the current region to a region in which the (most) fit samples lay. Note that this step also takes diversity into account through the generator's samples. Lastly, the continuation and stopping criteria determine the number of iterations the workflow should go through until the fittest sample is selected. These criteria ultimately control the trade-off between selection quality and runtime. The continuation criteria manage the minimum number of iterations and the stopping criteria conversely manage the maximum number of iterations. Hence, the continuation criteria provide a measure to warrant a minimum selection quality and the stopping criteria warrant a maximum execution time. After it is determined that enough iterations have taken place, i.e. the required amount of concentration to interesting regions is applied, the fittest encountered sample is selected by the strategy, which is then returned.

Describing the strategy more mathematically, let $j > 0$ denote the current iteration. The region that has been concentrated on in the previous iteration is denoted $\mathcal{R}_{j-1}$, with R_0 denoting the complete sample space:

$$\mathcal{R}_0 = \mathcal{X}, \forall j, k : j < k \rightarrow \mathcal{R}_j \subset \mathcal{R}_k \tag{4.2}$$

Note that the region of later iterations is a subset of the current region rather than a single area. A region is therefore not necessarily connected, instead a region of multiple disconnected interesting areas is also valid in this definition.

Within the j-th iteration, the first step is to discretize the region R_{j-1}. As stated before a generator G is used that generates samples within this region. Denote the set of generated samples $\mathcal{S}_j$, containing N samples:

$$\mathcal{S}_j = \{\mathbf{s_i} \in \mathcal{R}_{j-1} \mid 0 < i \leq N\} \tag{4.3}$$

Although discretization is normally employed to resemble a complete space or function into the discrete domain, this is not necessarily the focus of the generator. Namely, the generation step can be used

to take a diversity measure into account to bias the generated samples $\mathcal{S}_j$ to be more distinct. As explained in Section 2.2 this diversity metric is calculated compared to the training set $\mathcal{D}_{t-1}$, hence this set is also available to the generator.

These generated samples $\mathcal{S}_j$ are then provided to the *prioritizer* P. As its name suggests, the prioritizer provides an order to the samples, based on a metric m which computes the fitness f_i of a sample $\mathbf{s_i}$. This metric m is a function that can encompass an informativeness measure to allow prioritization for more interesting samples:

$$m : \mathcal{X} \to \mathbb{R} \tag{4.4}$$

The prioritizer then scores all generated samples in $\mathcal{S}_j$ with their corresponding fitness using the metric m:

$$\mathcal{F}_j = \{(\mathbf{s_i}, f_i = m(\mathbf{s_i})) \mid \mathbf{s_i} \in \mathcal{S}_j\} \tag{4.5}$$

The scored samples $\mathcal{F}_j$ and training data $\mathcal{D}_t$ are then passed onto the final step of the iteration. Within this step the focusing behaviour is done through restricting the current region R_{j-1} to a smaller region R_j. This process is done by the *restrictor* R. This restrictor uses the available information to restrict the current region $\mathcal{R}_{j-1}$ to the subregion R that contains the fittest samples from $\mathcal{F}_j$, compliant with Equation 4.2. This region $\mathcal{R}_j$ is then used in subsequent iterations which further concentrate on an interesting subregion.

As stated before, the continuation and stopping criteria determine the number of iterations which are executed before selecting the most fit sample. The continuation criteria determine the minimum number of iterations, whereas conversely the stopping criteria determine the maximum number of iterations. Each individual criterion can use information of the current process mentioned above: the iteration j, $\mathcal{S}_j$, $\mathcal{F}_j$, and $\mathcal{R}_j$. Based on this information, a criterion either evaluates to `true` or `false`. When a continuation criterion is `true`, it signifies that the workflow must continue the next iteration. As opposed to the continuation criteria, a stopping criterion is `true` when the selection procedure should not continue to the next iteration. As can be seen in Figure 4.1, first the continuation criteria are checked. If any of these criteria are `true`, the workflow continues into its next iteration. If this is not the case, the stopping criteria are subsequently checked. If none of these are `true`, the workflow continues into its next iteration as well. However, when there is a `true` stopping criterion, the workflow is halted and does not continue into another iteration. Note that this means that continuation criteria take priority over stopping criteria. When no more continuation criteria are `true` and there is a stopping criterion which is `true`, the process is stopped. At this point, the fittest sample $\mathbf{s}^*$ from the last iteration is selected and returned by the workflow:

$$\mathbf{s}^*, \_ = \underset{(\mathbf{s},f)\in\mathcal{F}_j}{\arg\min}\, \forall(\mathbf{s}', f') \in \mathcal{F}_j : f > f' \tag{4.6}$$

This sample $\mathbf{s}^*$ is therefore the fittest sample in the concentrated region through the iterative strategy that has taken place before selection and should therefore be a good candidate to aid in the ALs objective.

The following subsections delve deeper into the different variations for each of the described steps. Within this explanation, the value j is used to describe the iteration in which the workflow currently resides.

4.2. Sample Generation

As stated before, the sample generation step is used to discretize the current region $\mathcal{R}_{j-1}$.

As described in Section 4.1, the number of samples N, the current region $\mathcal{R}_{j-1}$, and the training data $\mathcal{D}_{t-1}$ are supplied to the generator. However, a specific variation of a generator can also be initialized with other hyperparameters specific to the variation to further tune its behaviour. As stated above, the generation step allows for a diversity metric to be taken into account.

4.2.1. Random Generator

The simplest and most naive generator is that of the *Random Generator* (RG), whose pseudocode can be found in Algorithm 1. This generator does not take the training data into account and instead samples from the region in an uniformly random manner. The samples returned from this generator are therefore spread across the sample space in an uniform way.

Algorithm 1 Random Sample Generation

1: **procedure** GENERATE SAMPLES(Number of samples N, Region $\mathcal{R}_{j-1}$)
2: **return** $\mathcal{R}_{j-1}.sample\_uniform(num = N)$

4.2.2. Maximum Distance Generator

On the other end of the spectrum lies the *Maximum Distance Generator* (MDG). The MDG is parametrized by $0 \leq \alpha < 100$ set by the user , with which the MDG retrieves N samples above the α-th percentile, based on distance to the training data $\mathcal{D}_{t-1}$. This percentile value is denoted P_α. Note that for a value of α, $100 - \alpha$ percentage of the points have a distance greater than P_α. A distance function δ is supplied such that $\delta(\mathbf{x}, \mathbf{y})$ denotes the distance between the point $\mathbf{x}$ and $\mathbf{y}$. The minimum distance d_i of a sample $\mathbf{x_i}$ is measured as the distance between the sample and the closest training sample to the sample:

$$d_i = \min_{\mathbf{y} \in \mathcal{D}_{t-1}} \delta(\mathbf{x_i}, \mathbf{y}) \tag{4.7}$$

Since dealing with a continuous region, calculating the percentile P_α is an infeasible task since it cannot be derived directly, instead requiring exhaustive computation. Generating samples that have a greater distance than this percentile is an even more troublesome task. The MDG therefore instead approaches this notion of P_α by approximating it, discretizing the region $\mathcal{R}_{j-1}$ with M samples using the RG, which we denote R_M. We denote the set of distances from R_M to be:

$$\Delta = \{d_i \mid \mathbf{x_i} \in R_M\} \tag{4.8}$$

The sample percentile $\widehat{P_\alpha}$ is then the minimum value in Δ such that $\alpha\%$ of the distances in Δ is smaller than $\widehat{P_\alpha}$:

$$\widehat{P_\alpha} = \arg\min_{d \in \Delta} |\{d_k \in \Delta \mid d_k \leq d\}| \geq M \cdot \frac{\alpha}{100} \tag{4.9}$$

We want to return the samples which have a greater distance than $\widehat{P_\alpha}$, of which there are $M \cdot \frac{100-\alpha}{100}$ because of the definition of $\widehat{P_\alpha}$ in Equation 4.9. To allow for N samples to be returned, M is derived to be:

$$\begin{aligned} N &\leq \frac{100 - \alpha}{100} \cdot M \leftrightarrow \\ M &\geq \frac{100}{100 - \alpha} \cdot N \end{aligned} \tag{4.10}$$

Note that the inequalities are used to account for the fact that the fraction does not always result in an integer value, whereas N and M should be integers. Hence in practice M is:

$$M = \lceil \frac{100}{100 - \alpha} \cdot N \rceil \tag{4.11}$$

The set $\mathcal{S}_j$ is returned then returned such that:

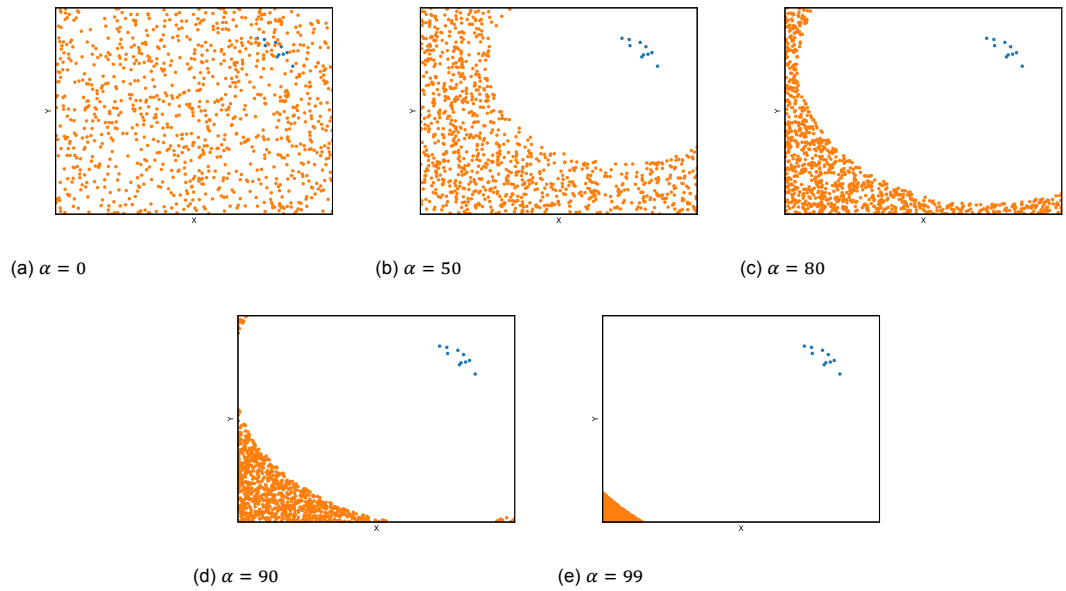
$$\mathcal{S}_j = \{\mathbf{x_i} \mid \mathbf{x_i} \in R_M \wedge d_i \geq \widehat{P_\alpha}\} \tag{4.12}$$

The procedure of the MDG can be found in Algorithm 2.

To illustrate the influence α has on the selection, Figure 4.2 shows the influence of different α's on the returned samples. Note that with $\alpha = 0$, the MDG is reduced to a RG. The higher α is, the more distant the samples in $\mathcal{S}_j$ are from the training data.

4.2.3. Probabilistic Distance Generator

Compared to the RG and MDG, the *Probabilistic Distance Generator* (PDG) provides an approach which could be seen as providing a middle road between the two. The intuition behind PDG is to use the minimum distances computed in Equation 4.7 to derive a probability for each sample x_i to be included in the returned samples $\mathcal{S}_j$, where a higher minimum distance results in a higher probability. Through this measure there is still a bias towards diverse samples, but it does not completely rule out the less diverse samples as the MDG does.



(a) $\alpha = 0$ (b) $\alpha = 50$ (c) $\alpha = 80$

(d) $\alpha = 90$ (e) $\alpha = 99$

Figure 4.2: The influence of different α values on the returned generated set $\mathcal{S}_j$. In each subfigure a two-dimensional sample region is depicted, with the blue points being the training set $\mathcal{D}_t$ and the orange points being $\mathcal{S}_j$. Note that $\alpha = 0$ equals the RG, and the higher α is, the smaller the region in which the points lie is.

Algorithm 2 Maximum Distance Sample Generation

procedure GENERATE SAMPLES(Number of samples N, Region $\mathcal{R}_{j-1}$, Training Data $\mathcal{D}_t$, Percentile α, distance function δ)
  $R_M \leftarrow \mathcal{R}_{j-1}.sample\_uniform(num = \lceil \frac{100}{100-\alpha} \cdot N \rceil))$
  if $D_t = \emptyset$ **then**
    return $R_M.random(num = N)$
  else
    $\Delta \leftarrow \{d_i \mid \mathbf{x_i} \in R_M\}$
    Derive $\widehat{P_\alpha}$ per Equation 4.9
    return $\{\mathbf{x_i} \in R_M \mid d_i \geq \widehat{P_\alpha}\}$

To accomplish this, the PDG first generates $M = \lceil \rho \cdot N \rceil$ uniformly random samples, with $\rho \geq 1$, where ρ is a hyperparameter set by the user. This set of uniform samples is denoted R_M as in the previous section. Δ is defined as in Equation 4.8. Lets denote the minimum and maximum value in Δ as d_{min} and d_{max} respectively. The relative distance compared to Δ can then be defined as:

$$r_i = \frac{d_i - d_{min}}{d_{max} - d_{min}} \tag{4.13}$$

Note that r_i scales the distances from 0 to 1, with the d_{max} being mapped to 1 and d_{min} to 0.

To allow for a sample's draw to be independent of previous draws, its probability should be independent of that of other samples. An initial thought could be to normalize the relative distances and use this as a probability to draw N samples sequentially. This would be accomplished by calculating the sum of relative distances, which is denoted:

$$\mathcal{H} = \sum_{0<i\leq M} r_i, \tag{4.14}$$

and calculating the probability p_i as follows:

$$p_i = \frac{r_i}{\mathcal{H}} \tag{4.15}$$

However, drawing N samples sequentially using the probabilities p_i can only be independent whenever samples can be drawn multiple times and therefore duplicates within $\mathcal{S}_j$ are allowed. Since $\mathcal{S}_j$ is to

consist of N unique samples, this prerequisite is not satisfied and p_i cannot be used in subsequent draws. Instead the sample that has been drawn has to be removed from the set of possible choices and the probabilities of these samples have to be renormalized using Equation 4.14 and 4.15. This operation is costly and has to be done after each draw, further increasing runtime when a large number of samples has to be drawn.

The PDG instead derives a probability q_i from r_i with which $\mathbf{x_i}$ is drawn. If q_i would equal the distance r_i, in expectation the returned set $\mathcal{S}_i$ would contain $\mathcal{H}$ samples. However the generator is to return N samples and hence these probabilities have to be adjusted such that the expected number of drawn samples would equal N, while still factoring in the relative distance of the sample. Define the expected number of returned samples as $\mathbb{E}_{R_M}$:

$$\mathbb{E}_{R_M} = \sum_{0<i\leq M} q_i \tag{4.16}$$

If $\mathcal{H} < N$, this would mean that using the relative distances as probabilities would result in too little samples being drawn in expectation. In this case, assigning a base probability β can be employed to increase the probability of all samples to the point at which N samples are drawn in expectation. Conversely, when $\mathcal{H} > N$, too much samples would be returned in expectation, which could be fixed in similar fashion by lowering the maximum probability γ to the point where one again N samples are drawn in expectation. This results in the probabilities q_i being defined as follows:

$$q_i = \beta + (\gamma - \beta) \cdot r_i \tag{4.17}$$

Where β is the minimum probability and γ is the maximum probability. Note that β and γ denote the range to which the relative distances r_i is mapped. As stated before, when $\beta = 0$ and $\gamma = 1$, we have that $q_i = r_i$ and $\mathcal{H} = \mathbb{E}_{R_M}$.

Using Equation 4.16 and 4.17, we derive that we can write $\mathbb{E}_{R_M}$ as:

$$\mathbb{E}_{R_M} = \sum_{0<i\leq M} q_i = \sum_{0<i\leq M} (\beta+(\gamma-\beta)\cdot r_i) = \sum_{0<i\leq M} \beta+(\gamma-\beta)\cdot \sum_{0<i\leq M} r_i \overset{(4.14)}{=} M\cdot\beta+(\gamma-\beta)\cdot\mathcal{H} = \beta\cdot(M-\mathcal{H})+\gamma\cdot\mathcal{H}. \tag{4.18}$$

Using this derivation, we can now derive the actual values of β and γ in the aforementioned scenarios, where $\mathcal{H} \neq N$:

- $\mathcal{H} < N$: This would mean that in expectation not enough samples from R_M would be selected to equal N. Hence we have to increase the base probability β whilst keeping the upper limit at its maximum value $\gamma = 1$. β is derived to be:

$$\begin{aligned} \mathbb{E}_{R_M} = N \quad &\leftrightarrow \\ \beta \cdot (M - \mathcal{H}) + 1 \cdot \mathcal{H} = N \quad &\leftrightarrow \\ \beta = \frac{N - \mathcal{H}}{M - \mathcal{H}} & \end{aligned} \tag{4.19}$$

- $\mathcal{H} > N$: In this situation the number of samples selected in expectation is greater than N. To account for this γ has to be adjusted, whilst keeping the lower bound at its minimum value $\beta = 0$:

$$\begin{aligned} \mathbb{E}_{R_M} = N \quad &\leftrightarrow \\ 0 \cdot (M - \mathcal{H}) + \gamma \cdot \mathcal{H} = N \quad &\leftrightarrow \\ \gamma = \frac{N}{\mathcal{H}} & \end{aligned} \tag{4.20}$$

Combining the two situations above, using the fact that β derived in Equation 4.19 is less than or equal to 0 when $\mathcal{H} \geq N$ and that γ derived in Equation 4.20 is greater than or equal to 1 when $\mathcal{H} \leq N$, we have that:

$$\begin{aligned} \beta &= \max(0, \frac{N - \mathcal{H}}{M - \mathcal{H}}) \\ \gamma &= \min(1, \frac{N}{\mathcal{H}}) \end{aligned} \tag{4.21}$$

Using the computed probabilities from Equation 4.21 the returned samples $\mathcal{S}_j$ can now be created probabilistically. The sampling is repeated until $|\mathcal{S}_j| \geq N$, choosing N samples from $\mathcal{S}_j$ randomly if $|\mathcal{S}_j| > N$. The complete procedure can be found in Algorithm 3.

Algorithm 3 Probabilistic Maximum Distance Sample Generation

procedure GENERATE SAMPLES(Number of samples N, Region $\mathcal{R}_{j-1}$, Training Data D_t, Ratio ρ, distance function δ)
  $R_M \leftarrow \mathcal{R}_{j-1}.sample\_uniform(num = \lceil N * \rho \rceil)$
  if $|\mathcal{D}_t| = 0$ **then**
    return $R_M.random(num = N)$
  else
    Compute Δ per Equation 4.8
    Compute d_{min} and d_{max}
    Compute r_i for $0 < i \leq M$ per Equation 4.13
    Compute β and γ per Equation 4.21
    Compute q_i for $0 < i \leq M$ per Equation 4.17
    $\mathcal{S}_j \leftarrow \emptyset$
    while $|\mathcal{S}_j| < N$ **do**
      $\mathcal{S}_j \leftarrow \{\mathbf{x_i} \in R_M \mid q_i > u_i \sim \mathcal{U}(0,1)\}$
    return $\mathcal{S}_j.random(num = N)$

4.3. Sample Prioritization

In the prioritization step, the generated samples $\mathcal{S}_j$ are provided, along with some additional variation-specific parameters to further tune the prioritizers behaviour. The prioritizer provides a fitness score for each sample within $\mathcal{S}_j$ enclosed in the set $\mathcal{F}_j$ as described in Equation 4.5. These fitness scores should portray the informativeness of the samples.

4.3.1. Random Prioritizer

To provide a baseline, the *Random Prioritizer* (RP) provides a random ordering through returning an equal fitness score for each sample within $\mathcal{S}_j$. Hence, the fitness scores f_i are set to:

$$f_i = c, \tag{4.22}$$

with c an arbitrary constant, which w.l.o.g. can be set to $c = 0$. The concise piece of pseudocode can be found in Algorithm 4.

Algorithm 4 Random Prioritization

procedure PRIORITIZE SAMPLES(Samples $\mathcal{S}_j$)
  return $\{(\mathbf{s}, 0) \mid \mathbf{s} \in \mathcal{S}_j\}$

4.3.2. Query By Committee

As discussed in Chapter 3, much research has been done in *Query by Committee* (QBC), originally coined by Seung et al. [60]. The metaphor behind the procedure is that of a committee, filled with committee members. These members have different knowledge on the discussed topic at hand, leading to different opinions on these topics. In the end, the discussed topic which triggers the most discussion and hence is the most controversial is selected.

Translating this metaphor into the usecase, the committee consists of C committee members. Each committee member is an instance of the used model M, which we use to denote the set of models $\mathcal{M}$:

$$\mathcal{M} = \{M_k \mid 0 < k \leq C\}, \tag{4.23}$$

where M_k denotes the k-th committee member. Each member M_k is fitted on its own training data $\mathcal{C}^k$, which is a set containing $t-1$ uniformly random samples from $\mathcal{D}_{t-1}$, with duplicates being allowed.

Using $\mathcal{C}^k$ and Equation 2.7, we then define θ_k to be M_k's optimal approximate set of parameters:

$$\theta_k = \theta'_{\mathcal{C}^k} \tag{4.24}$$

These parameters can in turn be used to provide a prediction $p_{k,i}$ on the generated samples $\mathbf{s_i} \in \mathcal{S}_j$:

$$p_{k,i} = M^{\theta_k}(\mathbf{s_i}) \tag{4.25}$$

To measure the controversiality of a sample $\mathbf{s_i}$, the prediction variance σ_i is then computed through the following formula:

$$\sigma_i = \sum_{k=1}^{C} (p_{k,i} - \bar{p_i})^2 \tag{4.26}$$

with $\bar{p_i}$ representing the prediction mean:

$$\bar{p_i} = \frac{1}{C} \cdot \sum_{k=1}^{C} p_{k,i} \tag{4.27}$$

When the prediction variance is high, this means that there is a lot of disagreement between the committee, which stems from a difference in training data. This shows that this point is informative, as small changes in training data result in a big difference in prediction. Hence QBC uses this prediction variance as its returned fitness for a point. That is, the fitness f_i of a sample $\mathbf{s_i}$ is:

$$f_i = \sigma_i \tag{4.28}$$

The process of QBC can be found in Algorithm 5.

Algorithm 5 Query By Committee

1: **procedure** PRIORITIZE SAMPLES(Training Data $\mathcal{D}_t$, Samples $\mathcal{S}_j$, Committee Size C)
2: Sample $\mathcal{C}_t^k$ from $\mathcal{D}_t$ for $0 < k \leq C$
3: Compute θ_k per Equation 4.24.
4: Compute $p_{k,i}$ for $0 < k \leq C, \mathbf{s_i} \in \mathcal{S}_j$ per Equation 4.25.
5: Compute σ_i for $\mathbf{s_i} \in \mathcal{S}_j$ per Equation 4.26.
6: **return** $\{(\mathbf{s_i}, \sigma_i) \mid \mathbf{s_i} \in \mathcal{S}_j\}$

4.4. Region Restriction

The region restriction is the last step in the iteration of the workflow. Using the fitness scores $\mathcal{F}_j$ of the samples $\mathcal{S}_j$ and possible additional parameters specific to the variations, the restrictor restricts the current region $\mathcal{R}_{j-1}$ to a subregion $\mathcal{R}_j$ per Equation 4.2.

4.4.1. No Restrictor

Similar to the RP, the *No Restrictor* (NR) is mainly defined to provide a baseline. As the name suggests, this restrictor does in fact not restrict the region, hence $\mathcal{R}_j = \mathcal{R}_{j-1}$. This restrictor is purely to be used when only one iteration is used to select a sample, since the region is not restricted.

4.4.2. Halfspace Restrictor

The *Halfspace Restrictor* (HR) appropriately halves the sample space through selecting the most fit halfspace. To accomplish this, first a dimension d is selected uniformly random on which the halfspace is split. Within this dimension, the value v on which the space is split is computed as the value between the minimum and maximum value of the samples within that dimension:

$$v = \frac{\min(X_d) + \max(X_d)}{2} \tag{4.29}$$

with X_d being the values of the samples within $\mathcal{S}_j$ in the dimension d:

$$X_d = \{s_d \mid \mathbf{s} \in \mathcal{S}_j\} \tag{4.30}$$

Using this split value the current region is then split into two subregions, A and B, such that:

$$\begin{aligned} A &= \{\mathbf{r} \in \mathcal{R}_{j-1} \mid r_d \leq v\} \\ B &= \mathcal{R}_{j-1} \setminus A \end{aligned} \tag{4.31}$$

Since the objective of this restriction is to zoom in on the most interesting region, we want to select the most fit region out of these two. This fitness is based on the mean fitness score of the scored samples $\mathcal{S}_j$ within the respective halfspace. Although this fitness is therefore only based on the informativeness metric, note that those samples included are also based on a diversity metric through the generator. Define the mean fitness of set C as:

$$f_C = \frac{1}{|\mathcal{S}_j \cap C|} \sum_{(\mathbf{s},f)\in\mathcal{F}_j : \mathbf{s}\in C} f. \tag{4.32}$$

The subregion which is returned is then the one which has the highest mean fitness, i.e. A is returned when $f_A \geq f_B$ and otherwise B is returned. The pseudocode can be found in Algorithm 6.

Algorithm 6 Halfspace Restriction

```
procedure RESTRICT SAMPLE SPACE(Region R_{j-1}, Samples S_j, Fitness F_j)
    Retrieve d randomly from the dimension of R_{j-1}
    Compute v per Equation 4.29
    Compute A and B per Equation 4.31
    Compute f_A and f_B per Equation 4.32
    if f_A ≥ f_B then
        return A
    else
        return B
```

4.4.3. Bounding Box Restrictor

Compared to HR, the *Bounding Box Restrictor* (BBR) instead focuses on the subregion around fit samples which do not contain other samples. This region $\mathcal{R}_j$ is a set of hyperrectangles, i.e. a Cartesian product of closed intervals. A hyperrectangle Q_k follows the following definition:

$$Q_k = \prod_{i=1}^{d} [a_i, b_i] \tag{4.33}$$

Where a_i and b_i are the lower and upper bound for the dimension i respectively. Let n be a user-defined parameter which defines the number of samples to focus a bounding box around. Define $\mathcal{B}_n$ be the n most fit samples from the samples in $\mathcal{S}_j$:

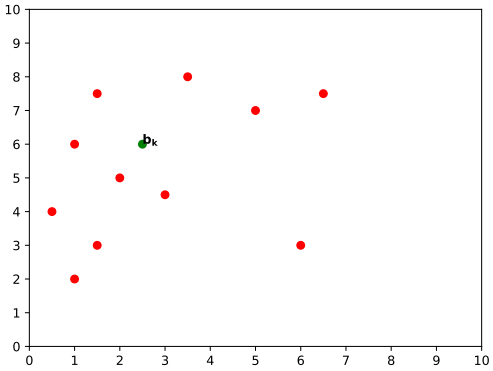
$$\forall \mathbf{b} \in \mathcal{B}_n |\{(\mathbf{s}, f) \in \mathcal{F}_j \mid f_{\mathbf{b}} \leq f\}| \leq n \tag{4.34}$$

For each of the samples $\mathbf{b_k} \in \mathcal{B}_n$, a hyperrectangle Q_k around it is created in which no other samples from $\mathcal{S}_j$ nor $\mathcal{D}_{t-1}$ lie. The returned subregion is defined:

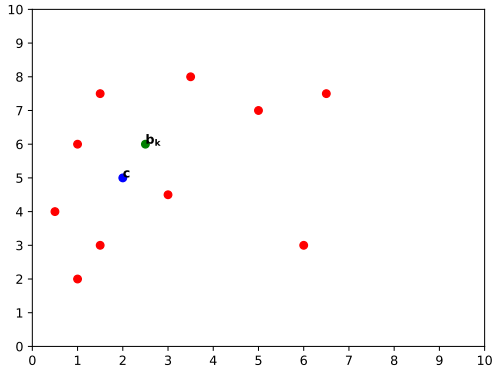
$$\mathcal{R}_j = \bigcup_{\mathbf{b_k}\in\mathcal{B}_n} Q_k \tag{4.35}$$

To describe the procedure to create such bounding box, take a sample $\mathbf{b_k} \in \mathcal{B}_n$. To create Q_k, define $\mathcal{Y}$ to be the set of the samples which should be excluded from the resulting region:

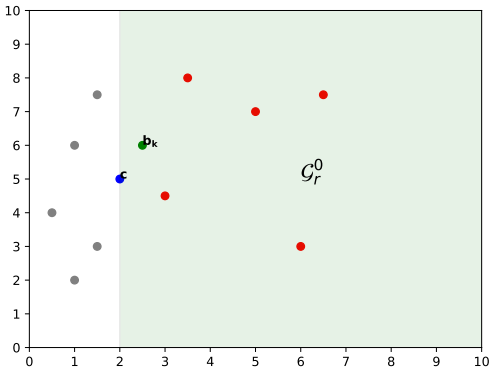
$$\mathcal{Y} = \mathcal{S}_j \cup \mathcal{D}_{t-1} \setminus \{\mathbf{b_k}\} \tag{4.36}$$



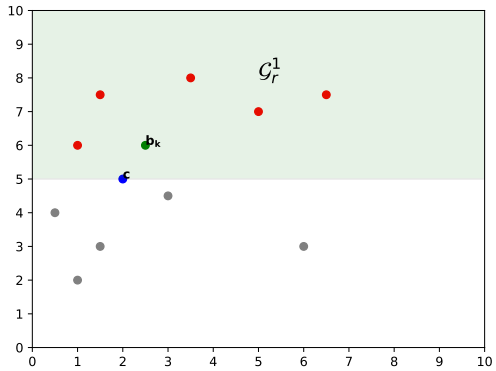
(a) The complete sample space, which shows the green point $\mathbf{b_k}$ around which a bounding box is to be placed, and $\mathcal{Y}$ as the red points.



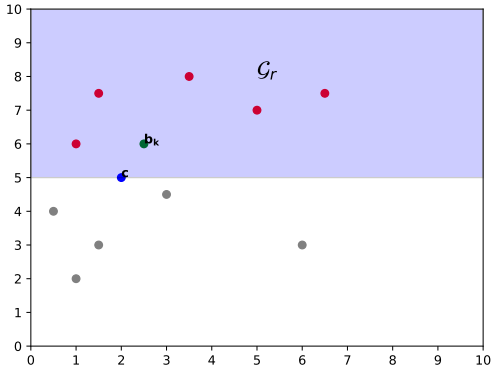
(b) First, the closest point $\mathbf{c}$ is determined, using the infinite norm as a distance metric. In this case, the point left below $\mathbf{b_k}$ is the closest with its distance to $\mathbf{b_k}$ being 1. Note that $\mathbf{c}$ is colored blue.



(c) To determine in which dimension a side supported by $\mathbf{c}$ is placed for both dimensions the score of the created subspace is computed (see Equation 4.42. $\mathcal{G}_r^0$ is created by splitting the space through a vertical line through $\mathbf{c}$, i.e. a vertical line at $x = 2$. The resulting volume of $\mathcal{G}_r^0$ is $(10-2)\cdot(10-0) = 80$. There are 5 points in $\mathcal{G}_r^0 \cap \mathcal{Y}$, hence $E_{\mathcal{G}_r^0} = \frac{80}{5} = 16$. The distance of $\mathbf{b_k}$ to the line is 0.5, hence the score of $\mathcal{G}_r^0$ is $s_{\mathcal{G}_r^0} = 16 \cdot 0.5 = 8$.



(d) Similar to Subfigure 4.3c, $\mathcal{G}_r^1$ is created by splitting the space with a horizontal line through $\mathbf{c}$, i.e. a horizontal line through $y = 5$. The resulting volume of $\mathcal{G}_r^1$ is $(10-0)\cdot(10-5) = 50$. There are also 5 points in $\mathcal{G}_r^1 \cap \mathcal{Y}$, hence $E_{\mathcal{G}_r^1} = \frac{50}{5} = 10$. The distance of $\mathbf{b_k}$ to the line is 1, hence the score of $\mathcal{G}_r^1$ is $s_{\mathcal{G}_r^1} = 10 \cdot 1 = 10$.



(e) The subspace with the highest score is selected and used in subsequent recursive calls to determine further sides. Since $s_{\mathcal{G}_r^1} > s_{\mathcal{G}_r^0}$, the resulting subspace $\mathcal{G}_r = \mathcal{G}_r^1$, as can be seen now highlighted in blue.

Figure 4.3: An example of one recursive call of the Bounding Box Restrictor. The green point is $\mathbf{b_k}$, and all other points are in $\mathcal{Y}$. The sample space currently is the rectangle with lower and upper bounds of 0 and 10 in both dimensions respectively.

This problem is an instance of the bichromatic hyperrectangle problem [72]. In this problem there are two colored sets B and C, whose elements fall inside a hyperrectangle E. The objective is to find a hyperrectangle such that no points from B and as much points from C are included. In the current situation, $B = \mathcal{Y}$ and $C = \{\mathbf{b}\}$. However, Backer and Mark Keil [72] found that computing an exact solution for this problem is NP-hard and can therefore not be found in polynomial time (as long as $P \neq NP$). Instead this thesis relies on heuristics to create a sizeable region around $\mathbf{b}_k$, using a recursive procedure. To do this, we employ the notion of sides of the hyperrectangle $\mathcal{Q}_k$, of which there are $2d$ when d dimensions are considered. Backer and Mark Keil [72] state that a side is *supported* whenever its interior points touches a point in B or E itself. The BBR uses a recursive procedure to add a supported side each recursive call, resulting in a maximum of $2d$ recursive calls. Note that adding a side can be seen as restricting the region $\mathcal{R}_{j-1}$ in a dimension d with either a lower or an upper bound.

Within a certain recursive call r, we have that $r-1$ sides have already been created that restrict the region $\mathcal{R}_{j-1}$, resulting in the intermediary subregion $\mathcal{G}_{r-1}$ s.t.:

$$\mathcal{G}_r \subset \mathcal{G}_{r-1} \subset ... \subset \mathcal{G}_0 = \mathcal{R}_{j-1} \tag{4.37}$$

When $\mathcal{G}_{r-1}$ does not contain any samples that have to be excluded, i.e. $\mathcal{G}_{r-1} \cap \mathcal{Y} = \emptyset$, we have found a valid hyperrectangle and $\mathcal{Q}_k = \mathcal{G}_{r-1}$.

When this is not the case, there are one or more samples that have to be excluded in $\mathcal{G}_{r-1}$, which can be done by adding another supported side to the hyperrectangle. An example illustration can be found in Figure 4.3. In this Figure, the initial configuration can be seen in Subfigure 4.3a, where $\mathbf{b_k}$ is colored green and the sample in Y are colored red. Since $\mathcal{Q}_k$ should be based around $\mathbf{b}_k$, the side that is created should go through the closest point c. Since we are dealing with a hyperrectangle, the infinity norm is used as a distance function between two points:

$$\|\mathbf{x} - \mathbf{y}\|_\infty = \max_d(|x_d - y_d|) \tag{4.38}$$

With the distance function, $\mathbf{c}$ can now be defined as:

$$\mathbf{c} = \arg\min_{\mathbf{y} \in \mathcal{G}_{r-1} \cap \mathcal{Y}} \|\mathbf{b_k} - \mathbf{y}\|_\infty \tag{4.39}$$

This step of determining $\mathbf{c}$ is visualized in Subfigure 4.3b. Since $\mathbf{c}$ is derived in this manner, there is no point $\mathbf{y} \in \mathcal{G}_{r-1}$ within the space between $\mathbf{c}$ and $\mathbf{b_k}$. We can therefore create a supported side by placing a boundary in any dimension d through c_d safely, reducing the problem. Note that the bound c_d is an upper bound when $b_{k,d} < c_d$ and a lower bound otherwise. Define the subregion of $\mathcal{G}_{r-1}$ bound by c_d in dimension d to be:

$$\mathcal{G}_r^d = \{\mathbf{g} \in \mathcal{G}_{r-1} \mid sign(g_d - c_d) = sign(b_{k,d} - c_d)\} \text{ with } sign(x,y) = \begin{cases} 1 & \text{if } x \geq y \\ -1 & \text{otherwise} \end{cases} \tag{4.40}$$

In Subfigure 4.3c and 4.3d, the subregions for each dimension are drawn.

To determine in which dimension d the side will be added, the notion of weighted expected volume of the resulting subregion $\mathcal{G}_r^d$ is used. Since the heuristic should create the subregion with a large volume, the expected volume $\mathbb{E}_{\mathcal{G}_r^d}$ approximates the further volume reduction in subsequent recursive calls:

$$\mathbb{E}_{\mathcal{G}_r^d} = \frac{Vol(\mathcal{G}_r^d)}{|\mathcal{G}_r^d \cap \mathcal{Y}|} \tag{4.41}$$

where $Vol(\cdot)$ denotes the volume of the provided space. This expected volume is weighted with the distance of $\mathbf{b_k}$ to the boundary c_d to prioritize hyperrectangles which center around $\mathbf{b_k}$. Hence, the scoring of the subregions is:

$$s_{\mathcal{G}_r^d} = \mathbb{E}_{\mathcal{G}_r^d} \cdot |b_{k,d} - c_d| \tag{4.42}$$

These scores are also calculated in Subfigures 4.3c and 4.3d, which are 8 and 10 for $\mathcal{G}_r^0$ and $\mathcal{G}_r^1$ respectively.

Using this score, we then define the resulting subregion in the iteration r as:

$$\mathcal{G}_r = \arg\max_{\mathcal{G}_d^r} s_{\mathcal{G}_r^d} \tag{4.43}$$

Then a further recursive call is made to determine the other sides of the hyperrectangle. As can be noted in Subfigure 4.3e, in the example $G_r = G_r^1$, as it has the highest score. This procedure is depicted in Algorithm 7.

This procedure is then repeated until no more samples of $\mathcal{Y}$ are found in $\mathcal{G}_r$. Another example showing the different recursive calls is shown in Figure 4.4, where a complete bounding box is constructed.

Algorithm 7 Bounding Box Restriction

procedure RESTRICT SAMPLE SPACE(Region $\mathcal{R}_{j-1}$, Samples $\mathcal{S}_j$, Fitness $\mathcal{F}_j$, Training Data $\mathcal{D}_t$, Number of Boxes n)
    Calculate $\mathcal{B}_n$ per Equation 4.34.
    return $\cup_{\mathbf{b_k} \in \mathcal{B}_n}$ BOUNDING BOX($\mathcal{R}_{j-1}, \mathbf{b_k}, \mathcal{S}_j \cup \mathcal{D}_t \setminus \{\mathbf{b_k}\}$)
procedure BOUNDING BOX(Restricted Region $\mathcal{G}_{r-1}$, Sample $\mathbf{b_k}$, Excluded Samples $\mathcal{Y}$)
    if $\mathcal{Y} = \emptyset$ **then**
        return $\mathcal{G}_{r-1}$
    Calculate **c** per Equation 4.39.
    Define $\mathcal{G}_r^d$ per Equation 4.40 for each dimension d.
    Calculate $s_{\mathcal{G}_r^d}$ per Equation 4.42 for each dimension d.
    Calculate $\mathcal{G}_r$ per Equation 4.43
    return BOUNDING BOX($\mathcal{G}_r, \mathbf{b_k}, \mathcal{Y} \cap \mathcal{G}_r$)

4.5. Continuation and Stopping Criteria

Apart from the steps discussed previously which are used within iterations, the Continuation and Stopping Criteria determine the number of iterations which the workflow goes through. As explained in Section 4.1, the continuation criteria specify the minimum and the stopping criteria specify the maximum amount of effort taken in each procedure. These criteria can be categorized in another way, namely the criterion being static or dynamic. A criterion is static when its judgment or state is independent of the results of the iteration and is dynamic if this is the case.

4.5.1. Static Criteria

The included static criteria in this thesis are those that count the number of iterations and base their judgment on this number. Using this number of iterations one can specify a continuation criterion with a minimum number of iterations and a stopping criterion through a maximum number of allowed iterations. That is, given a number of iterations j, the *Minimum Iteration Continuation Criterion* (MFCC) allowing j_{min} iterations is:

$$j < j_{min} \tag{4.44}$$

. Note that this criterion remains true until j_{min} iterations have been completed, as defined in Section 4.1.

Similarly, the *Maximum Iteration Stopping Criterion* (MISC) allowing j_{max} iterations can be defined:

$$j_{max} > j, \tag{4.45}$$

Where the criterion remains falls until j_{max} iterations have been completed.

4.5.2. Dynamic Criteria

Opposed to the static criteria, the dynamic criteria's judgment is dependent on the results obtained within the iteration. That is, when iteration j has been completed, the dynamic criterion bases its judgment based on $\mathcal{S}_j$, $\mathcal{F}_j$, and/or $\mathcal{R}_j$.

The *Minimum Fitness Criterion* (MFC) is a stopping criterion that determines its outcome through the fitness scores found in $\mathcal{F}_j$. When the values within $\mathcal{F}_j$ are similar, this means that the workflow has focused sufficiently on a small region where similar interesting samples are present. At this point, the workflow can be stopped and the most informative sample can be selected. Hence, the MFC bases its judgment on the difference $\Delta_{\mathcal{F}_j}$ between the minimum and maximum fitness score in $\mathcal{F}_j$:

$$\Delta_{\mathcal{F}_j} = \max(\mathcal{F}_j) - \min(\mathcal{F}_j) \tag{4.46}$$

The MFC can then be formulated as follows:

$$\Delta_{\mathcal{F}_j} < c \tag{4.47}$$

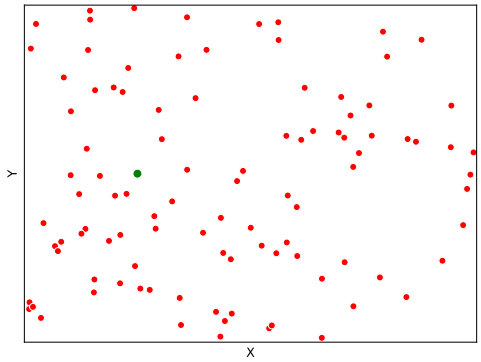
where the workflow is stopped when $\Delta_{\mathcal{F}_j}$ is below a user-defined boundary c.

This boundary can be defined either as an absolute value or as a percentage of the highest fitness value $f_{max} = \max(\mathcal{F}_j)$:

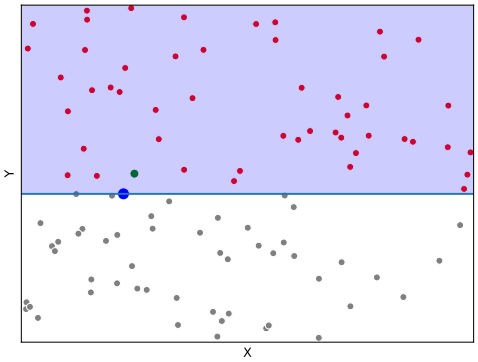
$$c = p \cdot q \text{ with } q = \begin{cases} 1 & \text{if absolute value} \\ f_{max} & \text{if percentual value} \end{cases} \tag{4.48}$$

Note that p is user defined and either describes the absolute value or the percentage of the highest fitness score.

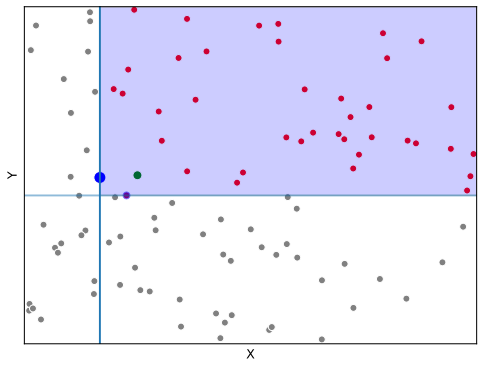
Using (a combination of) these criteria, together with the aforementioned components, a complete workflow is instantiated. When different implementations are selected and parametrized, the workflow can be applied to a general problem instance and select interesting samples from the specified sample space in a timely fashion.



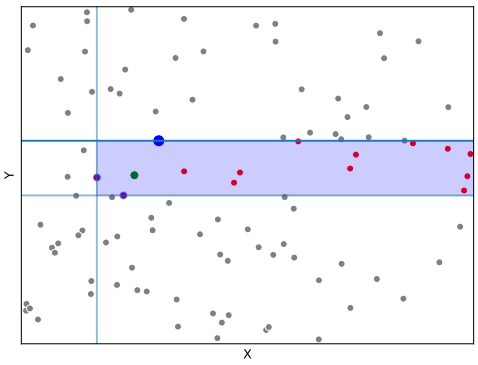
(a) $r = 0$



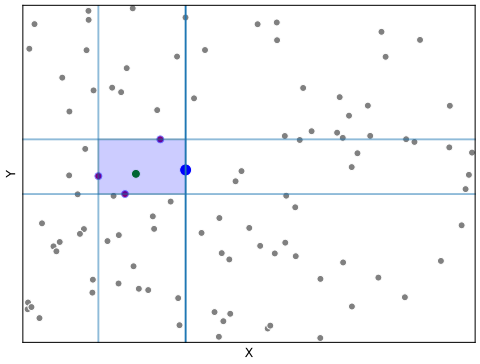
(b) $r = 1$



(c) $r = 2$



(d) $r = 3$



(e) $r = 4$

Figure 4.4: An example creation of a bounding box by the BBR. Each subfigure depicts the same two-dimensional region, with each point within the graph depicting a sample. $\mathbf{b_k}$ is colored green. Both grey and red samples are within $\mathcal{Y}$, with red samples being those that still have to be excluded from $\mathcal{G}_r$, whereas grey samples are already excluded. $\mathcal{G}_r$ is depicted in light blue, where the opaque blue lines are the bounds. The highlighted blue point is $\mathbf{c}$ in that specific iteration and remains slightly highlighted in subsequent iterations.

5

Experimental Results

To accomplish the goal for this thesis a strategy which should be both performant and time efficient is proposed in Chapter 4. To evaluate these two claims, this chapter provides an analysis of the performance of this strategy. This performance analysis is two-phased: determine the change in performance due to different components and their hyperparameters and compare the strategy's performance to a baseline. The hyperparameter and component performance is done within a scientific setting using a low-dimensional polynomial function as the oracle. The same function is also used to provide a comparison to the baseline. Lastly we employ a simulation of the car-following situation depicted in Figure 1.1 to evaluate the strategy in a more realistic scenario.

Each experiment makes use of the same model which is trained throughout the iterations. The model used is the Gaussian Process Regressor (GPR) of `Scikit Learn`, a Python module that provides machine learning tools within Python[1]. The implementation of this GPR is based on work of Carl Edward Rasmussen [49]. The GPR requires a kernel which is used to determine the covariance between two samples. Within this thesis the Radial Basis Function kernel is used to determine this covariance:

$$k(\mathbf{x}, \mathbf{x}') = \exp(\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}) \tag{5.1}$$

Where $\|\mathbf{x}\|$ resembles the Euclidean distance. Notice that the GPR has one parameter within its kernel, namely the length scale ℓ. Within experiments the set of parameters θ which is to be fitted using Equation 2.7 therefore only consists of this length scale. For finding this approximate optimal θ', the internal optimization process provided by `Scikit Learn` is used, namely the L_BFGS algorithm provided by `Scipy`, another scientific Python module. This algorithm is based on the theory described by Byrd et al. [73].

As stated in Equation 4.1, a specific workflow is denoted $W(G, P, R, C, S)$. Each of the components within the workflow are denoted in the following way:

$$A(\cdot), \tag{5.2}$$

where A is the abbreviation of the variation of the component mentioned in Chapter 4 and additional component-specific arguments are placed within the parentheses. As an example, $QBC(C = 10)$ denotes a Query By Committee Prioritizer with a committee size of 10.

The baseline which is used to compare the performance of the suggested workflow is not one from previous work, as there is no such selection strategy that solves the regression problem in a population-based setting with a black box model. To allow for comparison, instead a naive baseline W_{base} is adopted, which selects a random sample from the sample space, without considering any other information nor any iterative behaviour:

$$W_{base} = W(RG(N = 1), RP, NR, [], [MISC(j_{max} = 1)]) \tag{5.3}$$

[1] `https://scikit-learn.org/stable/`

To evaluate the fitness of a workflow instance, a test set $\mathcal{T}$ of 10000 uniformly random samples are sampled from the sample space and labeled by used oracle $\mathcal{O}$:

$$\mathcal{T} = \{(\mathbf{x_i}, \mathcal{O}(\mathbf{x_i})\}_{i=0}^{10000}. \tag{5.4}$$

When an identical oracle configuration is used, the test set is also kept equal to allow for meaningful comparison. The fitness of a workflow instance W in iteration t is measured as the Root Mean Squared Error, which is often used in regression tasks [4, 6, 7, 26, 33, 54, 55, 58, 65, 68, 69]:

$$f_{W,t} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x},y)\in\mathcal{T}} (y - M_W^t(\mathbf{x}))^2}, \tag{5.5}$$

where M_W^t is shorthand notation for the model M used by W with the fitted parameters $\theta'_{\mathcal{D}_t}$ on the training set $\mathcal{D}_t$ in iteration t, i.e. $M_W^t = M_W^{\theta'_{\mathcal{D}_t}}$. Note that a lower fitness is better, as this means that the prediction error to the test set is small. To compare the runtime of the different workflow, the selection time for each iteration is used.

Wherever significance is considered Welch's t-test [74] is employed to determine whether the difference in a metric is significant. This thesis defines two workflows A and B to be significantly different in an aggregate measure when the calculated p-value is smaller than 0.05. That is, the chance of A and B being equal in performance is smaller than five percent. This probability p is symmetrical, a fact which is used implicitly when reporting the relevant $p-$values. To employ the significance test for both the fitness and runtime metric we consider the aggregated fitness through the area under curve (AUC) [4, 56, 66, 69]:

$$AUC_W = \sum_{t=1}^{T} f_{W,t} \tag{5.6}$$

Note that for runtime this measures the total runtime, and for fitness the AUC measures the speed at which $\theta'_{\mathcal{D}_t}$ approaches the optimal parameters, since this coincides with a low fitness. Therefore a relatively low AUC indicates that the workflow allows for quick approachment of the optimal parameters and the higher the AUC metric is, the slower this approachment is.

5.1. Hyperparameter and Component Analysis

In this set of experiment a simple polynomial oracle with three dimensions is used to label the selected samples. The oracle $\mathcal{O}$ is defined as follows:

$$\mathcal{O}(\mathbf{x}) = \sum_{i=1}^{dim(\mathbf{x})} i \cdot x_i^{\lceil\sqrt{i}\rceil} + \mathcal{N}(dim(\mathbf{x}) + 1, 100), \tag{5.7}$$

where $dim(\cdot)$ denotes the dimension of a sample and $\mathcal{N}(\mu, \sigma^2)$ resembles a normal distribution with the mean μ and variance σ^2. This random factor is introduced to offset the equation with μ and create label variability as would be the case in real-life labels using σ^2.

Within these experiments a three dimensional sample space is used and as such $dim(\mathbf{x}) = 3$. This sample space is bounded by -1000 and 1000 in each dimension. To quickly examine the initial performance difference the number of samples T which are selected in total is 250. Initial experiments have shown that this number of iterations is enough to exhibit performance differences between different workflows. The goal of this set of experiments is to determine the performance of different combinations of components and their hyperparameters. This can in turn be used to analyse what combination of components performs well, which can be used to compare the suggested workflow to the aforementioned baseline.

5.1.1. Number of Samples and Committee Size

The first experiment reviews the influence of the number of generated samples N within $\mathcal{S}_j$ and the number of committee members C within the QBC Prioritizer. The following workflow is used within the experiments:

$$W_{n,c} = W(RG(N = n), QBC(C = c), NR, [], [MISC(j_{max} = 1]), \tag{5.8}$$

that is, we generate n samples uniformly random and prioritize using QBC with c committee members, but allow no restriction to take place. The hypotheses for both these variables is that a higher number of respectively samples or committee members positively influences the workflow's fitness. Of course, because a higher number also requires more computation it is also expected that the runtime increases with bigger numbers. To test this situations with little generated samples, a lot of samples, and a middle road are considered, through varying n with an order of magnitude:

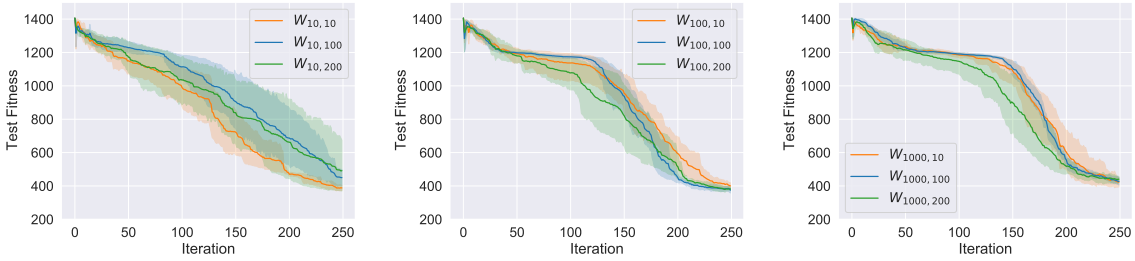
$$n \in \{10, 100, 1000\} \tag{5.9}$$

The number of committee members c is varied in similar fashion:

$$c \in \{10, 100, 200\} \tag{5.10}$$

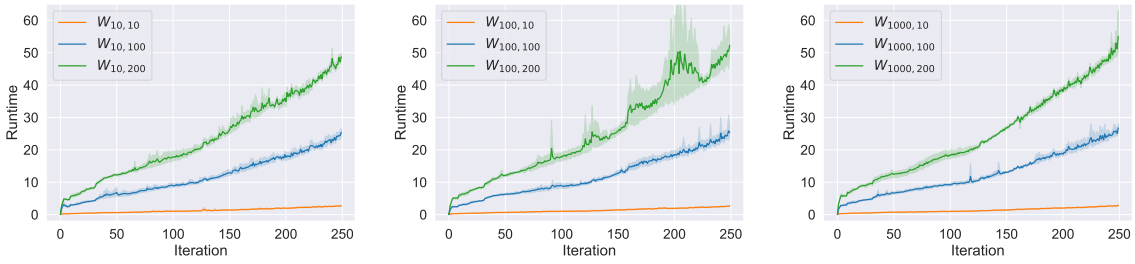
The greatest value 200 is mainly there to observe whether the increase in fitness is worth the additional required runtime. All combinations of these two values are ran and hence a total of 9 different workflow instances are present.

Figures 5.1 and 5.2 show the fitness and runtime of all individual solutions respectively. It can be noted that no big differences in fitness can be observed, though $W_{10,10}$ achieves the lowest mean fitness throughout the iterations. The significance results can be found in Table 5.1a, which shows there is nearly no performance difference between the individual workflows, except for $W_{10,10}$ outperforming both $W_{1000,10}$ and $W_{1000,100}$, the latter also being outperformed by two other workflows. Looking at the runtimes of the individual workflows in Figure 5.2, we notice that three clear groups of variations can be distinguished: one that has nearly 0 runtime throughout, one group that runs up until 20 seconds in later iterations and the last group that nearly takes a minute in later iterations to select a sample. This difference is also significant as can be noticed from Table 5.1b, where the committee size seems to be the responsible factor for the difference in runtime, resulting in $p-$values smaller than $1 \cdot 10^{-4}$ between individuals with different committee sizes.



(a) N=10 (b) N=100 (c) N=1000

Figure 5.1: Test Fitness for the individual workflows defined in Equation 5.8. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal committee size.



(a) N=10 (b) N=100 (c) N=1000

Figure 5.2: Runtime for the individual workflows defined in Equation 5.8. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal committee size.

To further examine the two parameters' influence on the runtime and fitness we aggregate the results so that the differences in parameters' performance is shown. Let $W_{\_,c}$ denote all variations that

Table 5.1: Pair-wise $p-$values for the workflows with parameterized number of generated samples and committee size as defined in Equation 5.8. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{10,10}$	$W_{10,100}$	$W_{10,200}$	$W_{100,10}$	$W_{100,100}$	$W_{100,200}$	$W_{1000,10}$	$W_{1000,100}$	$W_{1000,200}$
$W_{10,10}$	1.0000e+00	8.9811e-02	3.8567e-01	5.9850e-02	1.0950e-01	5.0998e-01	**1.5260e-02**	**1.4440e-02**	1.3939e-01
$W_{10,100}$	-	1.0000e+00	6.6349e-01	7.4344e-01	4.5557e-01	2.2533e-01	6.6002e-01	5.6216e-01	5.8424e-01
$W_{10,200}$	-	-	1.0000e+00	7.9380e-01	9.8961e-01	6.4105e-01	4.4331e-01	3.9524e-01	9.4421e-01
$W_{100,10}$	-	-	-	1.0000e+00	4.7701e-01	2.1462e-01	2.6782e-01	1.2131e-01	7.2923e-01
$W_{100,100}$	-	-	-	-	1.0000e+00	3.8078e-01	7.8084e-02	**1.0142e-02**	8.6343e-01
$W_{100,200}$	-	-	-	-	-	1.0000e+00	5.6692e-02	**4.2587e-02**	3.9450e-01
$W_{1000,10}$	-	-	-	-	-	-	1.0000e+00	8.6552e-01	2.2084e-01
$W_{1000,100}$	-	-	-	-	-	-	-	1.0000e+00	1.4206e-01
$W_{1000,200}$	-	-	-	-	-	-	-	-	1.0000e+00

(b) $p-$values comparing the aggregated total runtime.

	$W_{10,10}$	$W_{10,100}$	$W_{10,200}$	$W_{100,10}$	$W_{100,100}$	$W_{100,200}$	$W_{1000,10}$	$W_{1000,100}$	$W_{1000,200}$
$W_{10,10}$	1.0000e+00	**8.0116e-06**	**7.8982e-07**	7.4121e-01	**3.8322e-06**	**7.1617e-05**	4.4110e-01	**9.9173e-07**	**3.6462e-07**
$W_{10,100}$	-	1.0000e+00	**4.2907e-08**	**8.2653e-06**	8.5743e-01	**4.3245e-04**	**8.5772e-06**	1.5285e-01	**8.4136e-09**
$W_{10,200}$	-	-	1.0000e+00	**8.1437e-07**	**6.8518e-08**	3.5241e-01	**8.3705e-07**	**3.3303e-07**	9.3503e-02
$W_{100,10}$	-	-	-	1.0000e+00	**4.0259e-06**	**7.1604e-05**	1.9721e-01	**1.0927e-06**	**3.8104e-07**
$W_{100,100}$	-	-	-	-	1.0000e+00	**5.0614e-04**	**4.2255e-06**	1.5874e-01	**8.3553e-09**
$W_{100,200}$	-	-	-	-	-	1.0000e+00	**7.2140e-05**	**7.0312e-04**	7.8552e-01
$W_{1000,10}$	-	-	-	-	-	-	1.0000e+00	**1.1808e-06**	**3.9483e-07**
$W_{1000,100}$	-	-	-	-	-	-	-	1.0000e+00	**3.5158e-08**
$W_{1000,200}$	-	-	-	-	-	-	-	-	1.0000e+00

have a committee size of c. Within these experiments therefore $W_{-,10}$ are the aggregated results of the variations $W_{10,10}$, $W_{100,10}$, and $W_{1000,10}$. Using this aggregation we can further focus on the influence of each of the two parameters individually rather than the combination of the two. Figure 5.3 shows the results aggregated based on the committee size. The main observation is that the committee size c is leading in the observed difference in runtime, as can be seen in Figure 5.3b. From Table 5.2b it can also be seen that this difference in runtime is indeed very much significant, with all probabilities $p < 10^{-14}$. On the other hand, there is no apparent difference in fitness as seen in Figure 5.3a, where all different committee sizes seem to perform similarly, being confirmed further by the $p-$values in Table 5.2a. The hypothesis of improved performance when using more committee members is therefore rejected. These observations do show that there is an increase in runtime from an increase in the number of members in the committee. From these results it can further be concluded that a committee size of $c = 10$ should be used in future exploration, as the runtime for is relatively small, whilst there is no sacrifice in fitness compared to larger committee sizes.

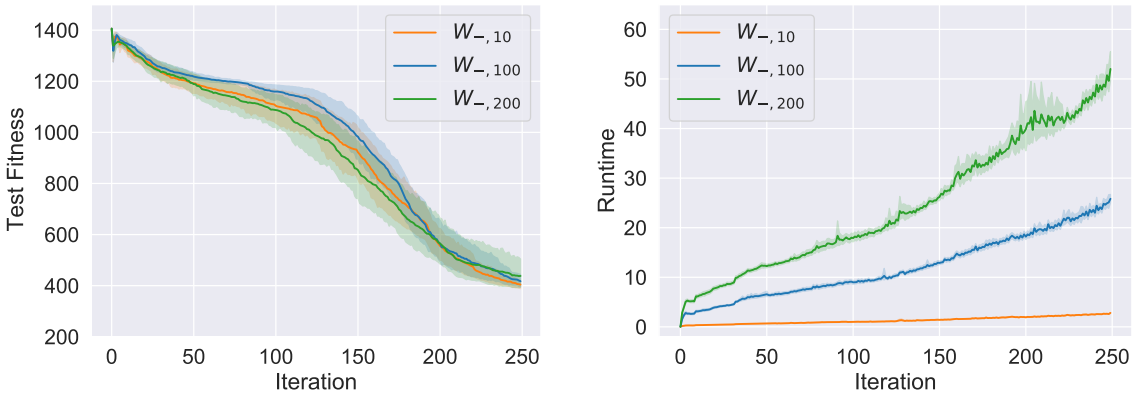
Table 5.2: Pair-wise $p-$values for the aggregated workflows based on the committee size. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{-,10}$	$W_{-,100}$	$W_{-,200}$
$W_{-,10}$	1.0000e+00	2.5637e-01	6.8571e-01
$W_{-,100}$	-	1.0000e+00	1.6383e-01
$W_{-,200}$	-	-	1.0000e+00

(b) $p-$values comparing the AUC of the runtime.

	$W_{-,10}$	$W_{-,100}$	$W_{-,200}$
$W_{-,10}$	1.0000e+00	**3.4126e-18**	**9.7964e-17**
$W_{-,100}$	-	1.0000e+00	**9.2878e-15**
$W_{-,200}$	-	-	1.0000e+00



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.8.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.3: Results of the aggregation based on committee size.

To view the influence of the number of generated samples a similar notation $W_{n,-}$ is employed to signify all results for variations with n generated samples. Hence $W_{10,-}$ denotes the aggregated results from $W_{10,10}$, $W_{10,100}$, and $W_{10,200}$. Figure 5.4 shows the results for these aggregated workflows' observations. Contrary to the committee size, the number of generated samples does not significantly increase the runtime of the variations as can be seen in Figure 5.4b and Table 5.4b, where all variations display a similar trend which does not differ significantly. For the fitness in Figure 5.4a we notice that $W_{100,-}$ follows the same trend as $W_{1000,-}$, although $W_{100,-}$ consistently outperforms $W_{1000,-}$ throughout

the iterations. This is also seen in Table 5.3a, where it is shown that the $p-$value is significant ($p \approx 0.025$. Opposed to this $W_{10,-}$ does not display a significant improvement to either of the other variations in terms of fitness, performing similarly to $W_{100,-}$ in terms of AUC. The choice between 10 or 100 as the best number of generated samples is therefore fairly arbitrary, as both display similar fitness and runtime characteristics. Since the committee size $c = 10$ is considered the best choice for our usecase, we employ the difference in performance in Figure 5.1 to determine the chosen number of samples. Looking at the mean performance of $W_{10,10}$ compared to $W_{100,10}$ it can be noted that $W_{10,10}$ performs better on average than $W_{100,10}$. Moreover, $W_{10,10}$ outperforms most other workflows as can be observed in Table 5.1a. The decision is therefore made to use $n = 10$ in subsequent experiments, as this provides a slight edge over $n = 100$ in our observations.

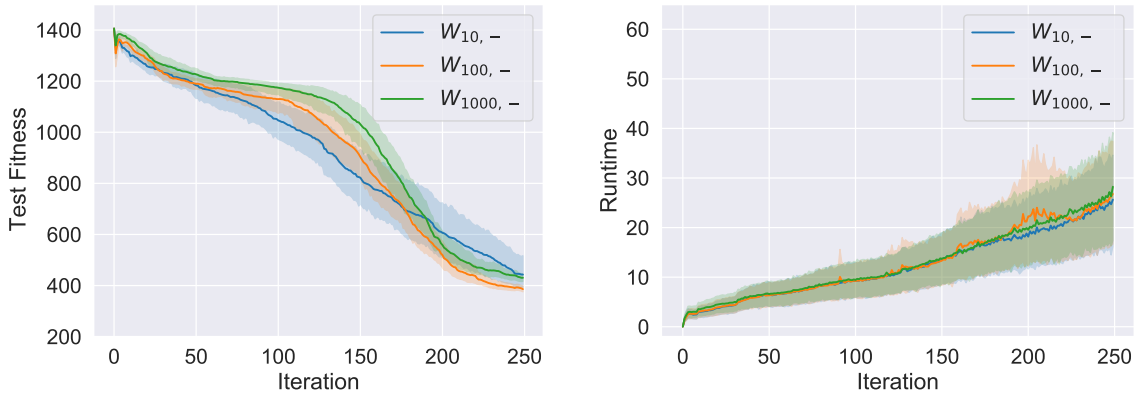
Table 5.3: Pair-wise $p-$values for the aggregated workflows based on the number of generated samples. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{10,-}$	$W_{100,-}$	$W_{1000,-}$
$W_{10,-}$	1.0000e+00	8.7796e-01	1.1150e-01
$W_{100,-}$	-	1.0000e+00	**2.5668e-02**
$W_{1000,-}$	-	-	1.0000e+00

(b) $p-$values comparing the AUC of the runtime.

	$W_{10,-}$	$W_{100,-}$	$W_{1000,-}$
$W_{10,-}$	1.0000e+00	8.8343e-01	8.6424e-01
$W_{100,-}$	-	1.0000e+00	9.8337e-01
$W_{1000,-}$	-	-	1.0000e+00



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.8.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.4: Results of the aggregation based on number of generated samples.

5.1.2. Generators and Restrictors

To determine the relative performance of different combinations of components of the workflow, the best-performing number of samples N and committee size C observed in Subsection 5.1.1 are used, i.e. $N = 10, C = 10$. Using this configuration, we define the workflow with generator G and restrictor R:

$$W_{G,R} = W(G(N = 10), QBC(C = 10), R, [MICC(j_{min} = 3)], [MFC(percent, p = 0.001)]) \quad (5.11)$$

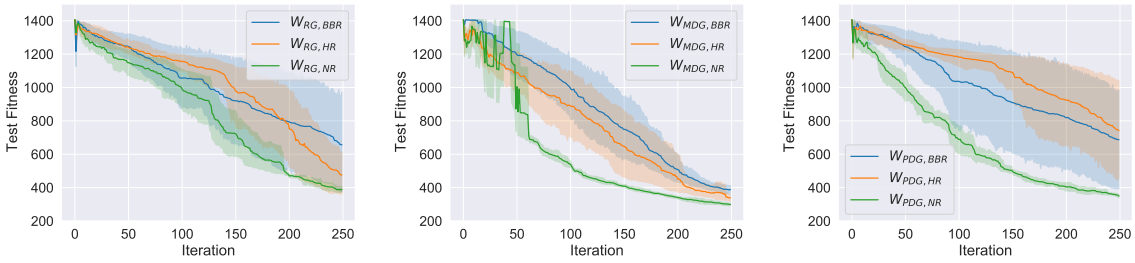
The combination of the continuation and stopping criteria allows for the focusing behaviour of the workflow to become apparent. To observe the influence of different combinations of components, G and R

encompass all varieties described in Chapter 4:

$$\begin{aligned} G &\in \{RG, MDG(\alpha = 20), PDG(\rho = 5)\} \\ R &\in \{NR, HR, BBR(n = 1)\}, \end{aligned} \tag{5.12}$$

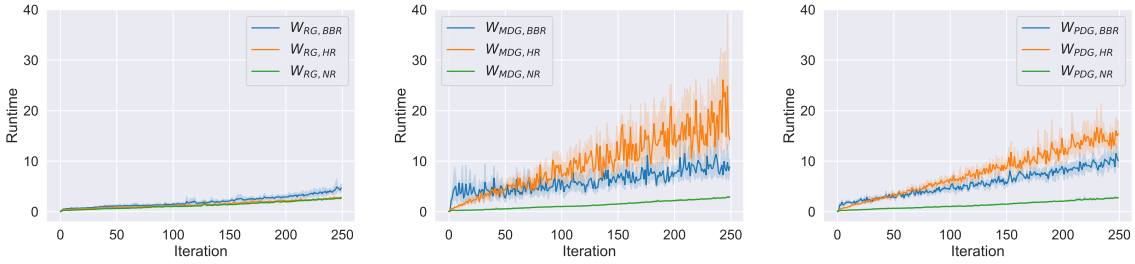
where the values of α, ρ, and n are determined by initial experiments which are not reported. Moreover, note that for all combinations that use the No Restrictor, the combination of continuation and stopping criteria is equal to that in Equation 5.3, since forcing multiple iterations without restricting the region would not be efficient and not change anything in performance. This variation is included to observe the behaviour of the workflow without its concentrating behaviour. Note that this further means that $W_{RG,NR}$ is equal to $W_{10,10}$ from the previous section. In total, this results in nine different combinations of components.

The hypotheses for these experiments is that the diversity metric through the generator improves performance, but takes longer in runtime compared to random generation. This also means that it is expected that the MDG outperforms the PDG, as it provides a more extreme diversity measure. In terms of restrictors the expectation is that both restrictive varieties outperform the non-restricting workflows, which obviously will take a longer time.



(a) Random Generator (b) Max Distance Generator (c) Probabilistic Generator

Figure 5.5: Test Fitness for the individual workflows defined in Equation 5.11. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal restrictor.



(a) Random Generator (b) Maximum Distance Generator (c) Probabilistic Generator

Figure 5.6: Runtime for the individual workflows defined in Equation 5.11. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal restrictor.

In Figures 5.5 and 5.6 the results in terms of test fitness and runtime can be viewed respectively. First thing to note is that both $W_{PDG,NR}$ and $W_{MDG,NR}$ outperform the results from the previous experiments ($W_{RG,NR}$) significantly, as can be observed from Table 5.4a. This could portray a positive influence of the diversity metric to the fitness of the resulting model as was hypothesized. In fact, $W_{MDG,NR}$ significantly outperforms all other workflows apart from $W_{MDG,HR}$, which shows similar performance in some of the runs.

The hypothesis of the restrictors however seems to be invalidated, as non-restricting workflows result in fitter models than when restricting behaviour is applied. Although this difference in performance is not significant in all cases for workflows using the halfspace restrictor, this can be explained by the observed variability in the fitness for these workflows. However, in each of the subfigures in Figure 5.5 it can be seen that the workflows using NR outperform the other two.

Table 5.4: Pair-wise p–values for the workflows with parameterized number of generated samples and committee size as defined in Equation 5.11. Note that significant values are highlighted with bold text.

(a) p–values comparing the AUC of the fitness.

	$W_{MDG,BBR}$	$W_{MDG,HR}$	$W_{MDG,NR}$	$W_{PDG,BBR}$	$W_{PDG,HR}$	$W_{PDG,NR}$	$W_{RG,BBR}$	$W_{RG,HR}$	$W_{RG,NR}$
$W_{MDG,BBR}$	1.0000e+00	3.4723e-01	**1.6548e-02**	3.6356e-01	**3.9679e-02**	**4.5236e-02**	3.1255e-01	1.3601e-01	7.2225e-01
$W_{MDG,HR}$	-	1.0000e+00	8.4097e-02	1.5040e-01	**9.5383e-03**	2.6500e-01	1.1115e-01	**2.9001e-02**	4.3138e-01
$W_{MDG,NR}$	-	-	1.0000e+00	**3.4266e-02**	**1.3307e-03**	**1.7111e-02**	**2.0726e-02**	**1.4634e-03**	**2.8447e-03**
$W_{PDG,BBR}$	-	-	-	1.0000e+00	5.5335e-01	5.9336e-02	9.9077e-01	9.7141e-01	2.6038e-01
$W_{PDG,HR}$	-	-	-	-	1.0000e+00	**2.3476e-03**	4.9378e-01	3.6190e-01	**1.3251e-02**
$W_{PDG,NR}$	-	-	-	-	-	1.0000e+00	**3.7673e-02**	**2.9986e-03**	**1.2381e-02**
$W_{RG,BBR}$	-	-	-	-	-	-	1.0000e+00	9.5490e-01	2.0460e-01
$W_{RG,HR}$	-	-	-	-	-	-	-	1.0000e+00	**3.9691e-02**
$W_{RG,NR}$	-	-	-	-	-	-	-	-	1.0000e+00

(b) p–values comparing the aggregated total runtime.

	$W_{MDG,BBR}$	$W_{MDG,HR}$	$W_{MDG,NR}$	$W_{PDG,BBR}$	$W_{PDG,HR}$	$W_{PDG,NR}$	$W_{RG,BBR}$	$W_{RG,HR}$	$W_{RG,NR}$
$W_{MDG,BBR}$	1.0000e+00	**3.3864e-05**	**9.9636e-05**	2.5696e-01	**3.2501e-03**	**1.0350e-04**	**2.5503e-05**	**9.0736e-05**	**9.4930e-05**
$W_{MDG,HR}$	-	1.0000e+00	**5.0152e-06**	**3.6737e-06**	**1.4145e-03**	**5.4699e-06**	**9.0425e-08**	**3.5758e-06**	**4.8494e-06**
$W_{MDG,NR}$	-	-	1.0000e+00	**2.6027e-05**	**1.3286e-05**	9.2421e-01	**1.2925e-02**	1.2278e-01	2.0023e-01
$W_{PDG,BBR}$	-	-	-	1.0000e+00	**2.2985e-04**	**2.8969e-05**	**1.3366e-06**	**1.7853e-05**	**2.4513e-05**
$W_{PDG,HR}$	-	-	-	-	1.0000e+00	**1.4343e-05**	**5.0505e-07**	**1.0036e-05**	**1.2765e-05**
$W_{PDG,NR}$	-	-	-	-	-	1.0000e+00	**1.3121e-02**	1.0461e-01	1.7569e-01
$W_{RG,BBR}$	-	-	-	-	-	-	1.0000e+00	**2.0698e-02**	**9.7627e-03**
$W_{RG,HR}$	-	-	-	-	-	-	-	1.0000e+00	**3.1553e-02**
$W_{RG,NR}$	-	-	-	-	-	-	-	-	1.0000e+00

In terms of runtime the hypotheses partially hold. Although from Figure 5.6 it can be concluded that using additional diversity measures result in longer runtimes, this difference is not obvious when no restriction is used. This can also be observed from Table 5.4b, where the runtime differences between non-restricting workflows are insignificant ($W_{RG,NR}$, $W_{MDG,NR}$, and $W_{PDG,NR}$). For the restrictors there is an obvious difference between not applying any restriction or either using a BBR or HR.

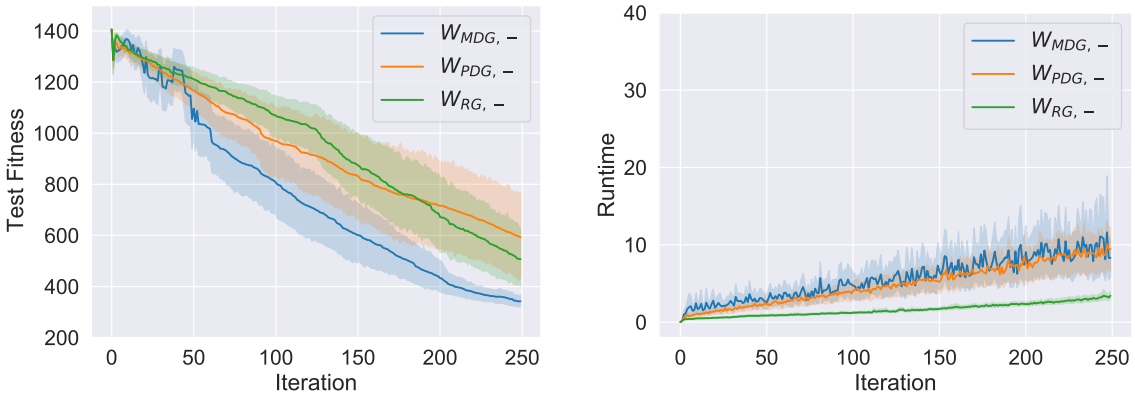
Table 5.5: Pair-wise $p-$values for the aggregated workflows based on the generator variation. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{MDG,-}$	$W_{PDG,-}$	$W_{RG,-}$
$W_{MDG,-}$	1.0000e+00	**3.0324e-02**	**2.5567e-03**
$W_{PDG,-}$	-	1.0000e+00	7.6005e-01
$W_{RG,-}$	-	-	1.0000e+00

(b) $p-$values comparing the AUC of the runtime.

	$W_{MDG,-}$	$W_{PDG,-}$	$W_{RG,-}$
$W_{MDG,-}$	1.0000e+00	5.2960e-01	**4.9960e-04**
$W_{PDG,-}$	-	1.0000e+00	**3.9735e-04**
$W_{RG,-}$	-	-	1.0000e+00



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.11.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.7: Aggregated results of the different varieties of generators.

To further conclude the influence of the separate components, the individual results are once again aggregated in similar fashion to Subsection 5.1.1. Hence $W_{G,-}$ denotes all combinations with G as its generator and similarly $W_{-,R}$ denotes all combinations with R as its restrictor.

In Figure 5.7 the aggregated results related to the generator varieties is displayed. As noted before, the MDG performs best in the experiments, outperforming the other two generators significantly, as displayed in Table 5.5a ($p < 0.031$). This is in line with the hypothesis. On the other hand, the PDG's fitness is not in line with this hypothesis, as it performs similar to the RG, as can be noted from both Figure 5.7a and Table 5.5a.

In terms of runtime in Figure 5.7b, there is a clear significant difference between the RG and the other two generators, which was expected, as also noted in Table 5.5b. The MDG and PDG however cannot be distinguished in terms of runtime. To allow for well performing sample selection it can be concluded that MDG is the generator to select, as it implements the diversity metric most extreme, resulting in better results.

Equal to prior reporting of results, Figure 5.8 reports on the aggregated results based on the restrictor used in the workflow. As Subfigure 5.8a shows, not applying any restriction clearly outperforms

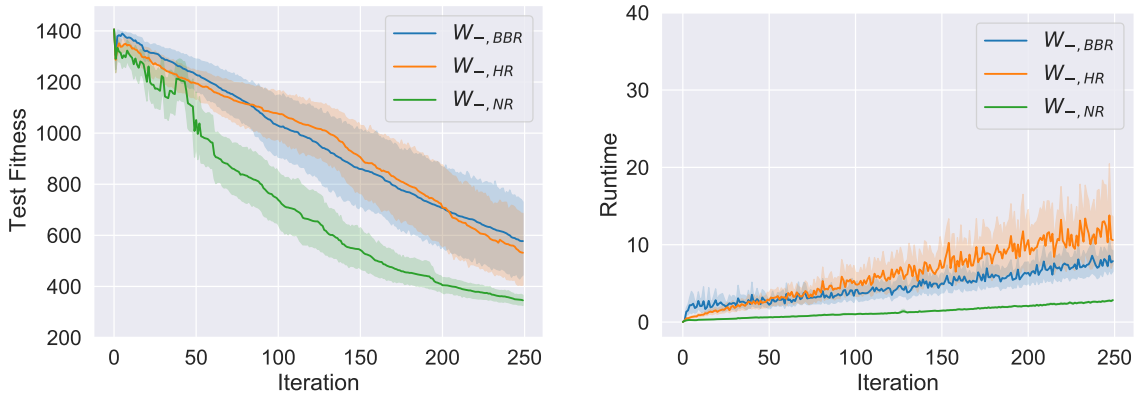
Table 5.6: Pair-wise $p-$values for the aggregated workflows based on the restrictor variation. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{-,BBR}$	$W_{-,HR}$	$W_{-,NR}$
$W_{-,BBR}$	1.0000e+00	9.9995e-01	**9.1369e-04**
$W_{-,HR}$	-	1.0000e+00	**2.8251e-04**
$W_{-,NR}$	-	-	1.0000e+00

(b) $p-$values comparing the AUC of the runtime.

	$W_{-,BBR}$	$W_{-,HR}$	$W_{-,NR}$
$W_{-,BBR}$	1.0000e+00	1.2822e-01	**1.5081e-05**
$W_{-,HR}$	-	1.0000e+00	**1.1945e-04**
$W_{-,NR}$	-	-	1.0000e+00



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.11.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.8: Aggregated results of the different varieties of restrictors

concentrating on interesting regions with the used settings. Table 5.6a further confirms that this difference indeed is significant for both restrictors, both having a $p-$value smaller than 0.001. What is interesting to see is that the performance of the two restrictor varieties is approximately equal, with a 99.99 percent chance that the observations had come from an equal workflow according to the $t-$test.

Similar results are observed in terms of runtime. There is a similarly significant difference in runtime when the NR or the other two restrictors are considered, as can be noticed in Figure 5.8b and Table 5.6b. Although not as extreme as the fitness, the two restrictors can not be separated significantly by runtime either.

From these results, one could conclude that no restriction should be used, instead only relying on a single iteration to sample and consequently selecting the most informative sample.

5.1.3. Stopping and Continuation Criteria

To further investigate the fact the results in Subsection 5.1.2 and particularly the restriction performing worse than those workflows without restriction, the influence of the stopping and continuation criteria is examined. What is expected is that the results in Subsection 5.1.2 are mainly due to over-restricting the sample space, therefore resulting in worse workflows. To evaluate this hypothesis these experiments vary both the minimum number of iterations through the MICC, as well as the maximum fitness difference through the MFC. Using this setup, it is moreover expected that the runtime increases as the minimum number of iterations is increased and the maximum fitness difference is decreased. In terms of fitness the hypothesis is that a workflow with a lower number of iterations and a high maximum

fitness difference outperforms other workflows, as is apparent in previous results. To further observe the number of iterations j to restrict the sample space whilst selecting a sample, this statistic is also reported. The hypothesis is that this number of iterations aligns with the runtime, such that the variables influence the number of iterations in similar fashion.

Since Subsection 5.1.2 did not show any conclusive observations whether the HR or BBR performs significantly better there is no clear choice for a restrictor to use within these experiments. Since it has shown that the MDG outperforms the other two generators, instead the restrictor that performs best with this generator is chosen. As can be seen in Figure 5.5, the HR in this case slightly outperforms the BBR as far as mean test fitness over multiple runs is concerned. Although the t-test in Table 5.4a does not show that this is significant ($p \approx 0.347$), the HR is considered in this experiment.

To test the influence of the present criteria, therefore the following workflow is used:

$$W_{i,v} = W(MDG(N = 10), QBC(C = 10), HR, [MICC(j_{min} = i)], [MFC(percent, p = v)]), \tag{5.13}$$

where i configures the minimum number of iterations and v the maximum percentual difference between the fitness of the best and worst sample to stop further iterations.

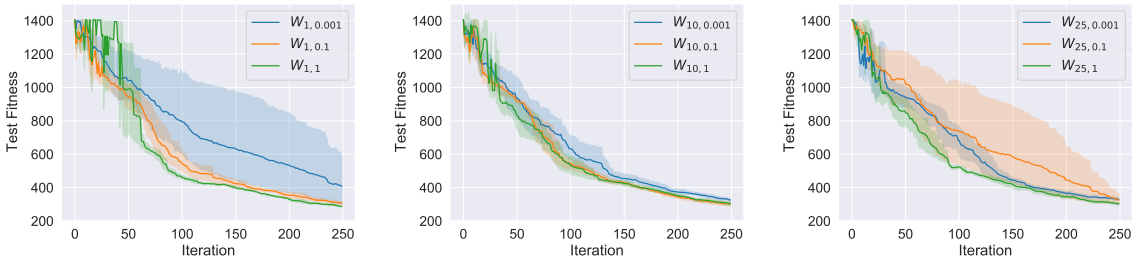
Since Section 5.1.2 shows a worse performance when using restriction and it is suspected that this is because of over-restriction, the value v is varied to higher percentual values, hence allowing for a less overfocused workflow:

$$v \in \{1, 0.1, 0.001\} \tag{5.14}$$

Note that $v = 1$ results in the workflow stopping as soon as no continuation criteria are true and that $v = 0.001$ is the configuration used in Subsection 5.1.2. To test whether a minimum iterative effort aids or restrains the workflow's effectiveness, the values i are varied as follows:

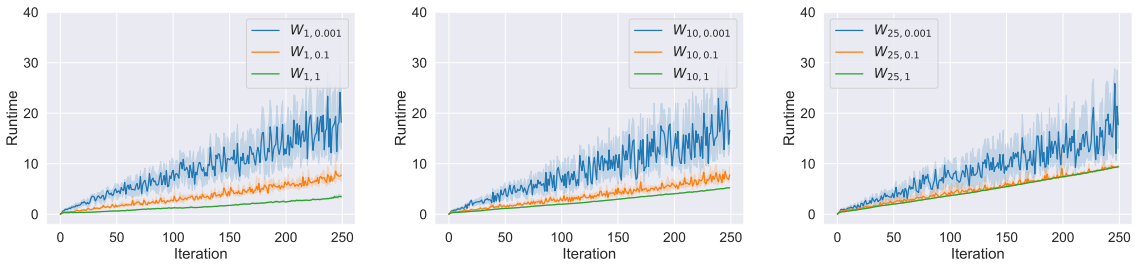
$$i \in \{1, 10, 25\}, \tag{5.15}$$

which resembles minimum effort, medium effort, and high effort respectively. Note that the workflow $W_{1,1}$ corresponds to the $W_{MDG,NR}$ observation in Subsection 5.1.2, and equally $W_{1,0.001} = W_{MDG,HR}$.



(a) $i = 1$ (b) $i = 10$ (c) $i = 25$

Figure 5.9: Test Fitness for the individual workflows defined in Equation 5.13. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal maximum fitness difference.



(a) $i = 1$ (b) $i = 10$ (c) $i = 25$

Figure 5.10: Runtime for the individual workflows defined in Equation 5.13. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal maximum fitness difference.

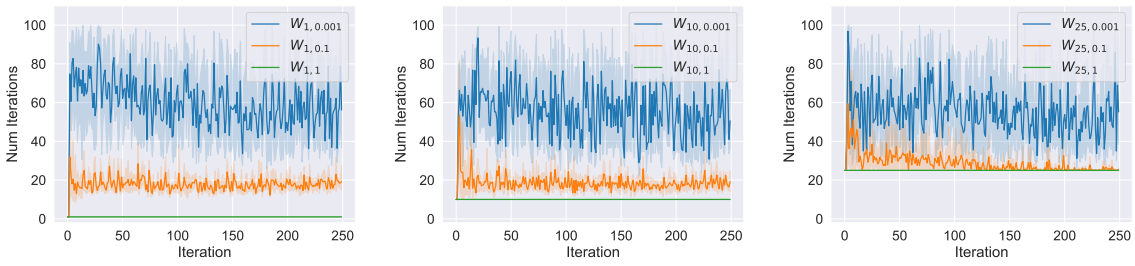
Table 5.7: Pair-wise p–values for the workflows with parameterized number of generated samples and committee size as defined in Equation 5.13. Note that significant values are highlighted with bold text.

(a) p–values comparing the AUC of the fitness.

	$W_{1,0.001}$	$W_{1,0.1}$	$W_{1,1}$	$W_{10,0.001}$	$W_{10,0.1}$	$W_{10,1}$	$W_{25,0.001}$	$W_{25,0.1}$	$W_{25,1}$
$W_{1,0.001}$	1.0000e+00	2.7268e-01	2.4636e-01	3.6567e-01	2.4533e-01	2.3658e-01	3.6804e-01	7.4660e-01	2.0719e-01
$W_{1,0.1}$	-	1.0000e+00	5.3866e-01	2.6269e-01	5.3604e-01	4.5759e-01	1.7754e-01	3.7689e-01	1.3118e-01
$W_{1,1}$	-	-	1.0000e+00	1.2004e-01	9.6409e-01	7.5866e-01	5.5095e-02	3.3447e-01	1.1077e-01
$W_{10,0.001}$	-	-	-	1.0000e+00	1.1886e-01	1.0406e-01	9.6821e-01	5.2578e-01	**4.3126e-02**
$W_{10,0.1}$	-	-	-	-	1.0000e+00	7.9497e-01	5.4727e-02	3.3290e-01	1.5382e-01
$W_{10,1}$	-	-	-	-	-	1.0000e+00	5.1045e-02	3.1920e-01	3.7620e-01
$W_{25,0.001}$	-	-	-	-	-	-	1.0000e+00	5.2952e-01	**1.5143e-02**
$W_{25,0.1}$	-	-	-	-	-	-	-	1.0000e+00	2.7266e-01
$W_{25,1}$	-	-	-	-	-	-	-	-	1.0000e+00

(b) p–values comparing the aggregated total runtime.

	$W_{1,0.001}$	$W_{1,0.1}$	$W_{1,1}$	$W_{10,0.001}$	$W_{10,0.1}$	$W_{10,1}$	$W_{25,0.001}$	$W_{25,0.1}$	$W_{25,1}$
$W_{1,0.001}$	1.0000e+00	**5.9768e-05**	**1.8824e-05**	2.8901e-01	**6.6738e-05**	**3.2798e-05**	1.8777e-01	**1.6876e-05**	**1.4409e-04**
$W_{1,0.1}$	-	1.0000e+00	**1.8244e-08**	**1.1296e-07**	2.7080e-01	**1.5950e-06**	**3.4645e-06**	**1.9690e-03**	**5.4245e-07**
$W_{1,1}$	-	-	1.0000e+00	**1.5185e-07**	**2.1878e-13**	**4.0757e-11**	**1.4524e-06**	**5.3329e-05**	**4.3008e-16**
$W_{10,0.001}$	-	-	-	1.0000e+00	**5.0640e-07**	**3.2427e-07**	4.9760e-01	**1.4096e-06**	**1.4230e-06**
$W_{10,0.1}$	-	-	-	-	1.0000e+00	**1.6672e-09**	**5.6072e-06**	**2.4762e-03**	**5.2076e-11**
$W_{10,1}$	-	-	-	-	-	1.0000e+00	**2.7740e-06**	**2.1520e-04**	**5.4972e-13**
$W_{25,0.001}$	-	-	-	-	-	-	1.0000e+00	**4.0189e-07**	**1.4049e-05**
$W_{25,0.1}$	-	-	-	-	-	-	-	1.0000e+00	2.1047e-01
$W_{25,1}$	-	-	-	-	-	-	-	-	1.0000e+00



(a) $i = 1$ (b) $i = 10$ (c) $i = 25$

Figure 5.11: Number of iterations of the individual workflows defined in Equation 5.13. Note that these are split in three figures to allow for an easier interpretation of each individual workflow, where equal colors indicate equal maximum fitness difference.

Figures 5.9 through 5.11 shows the individual results of the different combinations of i and v in terms of fitness, runtime, and number of iterations respectively. As can be seen in Figure 5.9, the workflow $W_{1,1}$ is approached in performance by a multitude of other configurations, some of which show less unstable behaviour in the earlier iterations than $W_{1,1}$. It is specifically noted that $W_{10,1}$ and $W_{25,1}$ perform in similar, but more stable fashion, also showing similar $p-$values in Table 5.7a. That being said, no workflow significantly outperforms $W_{1,1}$ (notably $p \approx 0.76$ against $W_{10,1}$ and $p \approx 0.11$ against $W_{25,1}$), whilst $W_{1,1}$ selects the samples significantly faster than all other variations ($p < 6 \cdot 10^{-5}$). This is in line with the hypothesis and previous experiments, although the hypothesis that $W_{1,1}$ significantly outperforms the other solutions is not significant judging from Table 5.7a.

In terms of runtime, it is definitely clear that the allowed difference v is the main influence for the runtime, where a smaller v seems to result in a more unstable runtime and fitness. Nearly all workflow exhibit significantly different runtimes, as can be noted from Table 5.7b with all highlighted cells. However, those pairs that are not significantly different do not differ in v, but only in number of minimum iterations, suggesting even more that v is the influencing parameter.

As hypothesized, the runtime results obtained coincide with the number of iterations, as can be seen in Figure 5.11. In these graphs the instability of the workflows with a low v becomes even more apparent, where this clearly results in not only more iterations, but also a more varying amount of iterations. What also is interesting is that for $W_{1,0.1}$ and $W_{10,0.1}$, the number of iterations seems to be around 20 throughout the experiment. However, when the minimum iterations is set to 25, this number is further increased to around 35, as becomes apparent in $W_{25,0.1}$'s results.

Table 5.8: Pair-wise $p-$values for the aggregated workflows based on the minimum number of iterations. Note that significant values are highlighted with bold text.

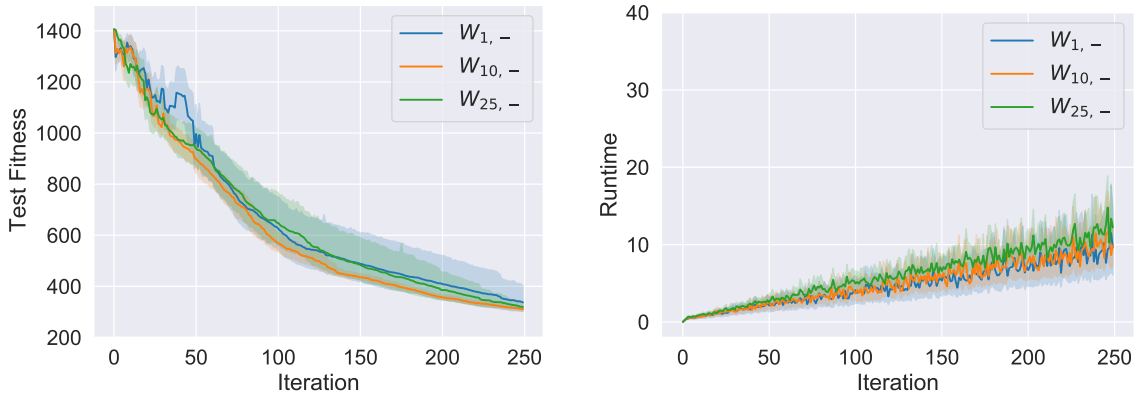
(a) $p-$values comparing the AUC of the fitness.

	$W_{1,-}$	$W_{10,-}$	$W_{25,-}$
$W_{1,-}$	1.0000e+00	2.6234e-01	7.5292e-01
$W_{10,-}$	-	1.0000e+00	3.7056e-01
$W_{25,-}$	-	-	1.0000e+00

(b) $p-$values comparing the AUC of the runtime.

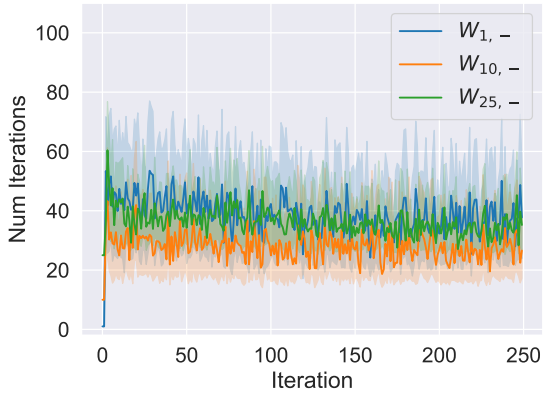
	$W_{1,-}$	$W_{10,-}$	$W_{25,-}$
$W_{1,-}$	1.0000e+00	8.0961e-01	1.9992e-01
$W_{10,-}$	-	1.0000e+00	2.3843e-01
$W_{25,-}$	-	-	1.0000e+00

To further examine the influence of the individual parameters, similarly to previous sections, we employ the $W_{i,-}$ to depict all results of varieties with that particular value for i, and $W_{-,v}$ for all results of combinations with the specified value v. From Figure 5.12 the aggregated data in terms of minimum number of iterations can be observed. As stated before, barely any difference in performance



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.11.

(b) Runtime in seconds of the selection process for each variation.



(c) The number of iterations j it took to select a sample.

Figure 5.12: Aggregated results of the different number of minimum iterations.

can be observed in Figure 5.12a when varying i. The same can be observed from the $p-$values in Table 5.8a, where no significant differences can be observed, instead showing that all observations are quite similar. It does appear that a minimum of 10 iterations does however improves the fitness of the model, especially in the earlier iterations of the experiments. That is, with this number of iterations the instability observed in $W_{1,-}$ and $W_{1,1}$ is stabilized.

The runtime in Figure 5.12b and number of iterations in Figure 5.12c show near similar levels of similarity, barely allowing any difference to be observed. This set of observations definitely invalidates the set hypotheses, as there is no significant difference to be noted by varying i.

This opposed to the results of the aggregated results of the maximum fitness difference depicted in Figure 5.13. Figure 5.13b especially shows the earlier noticed difference in the time itself as well as the stability of the curve. When Figure 5.13c is considered, it can moreover be observed that the instability indeed is caused by a lower value of v. That being said, in Figure 5.13a can be seen that although $W_{-,1}$ seems to be performing best as hypothesized, it is not significantly better than $W_{-,0.1}$, but significantly better than $W_{-,0.001}$. This supports the hypothesis that a higher v indeed increases performance, but also shows that the previous results in Section 5.1 might indeed be due to over-restriction.

To both conclude and summarise the findings in this section, the best-performing workflow is discussed. In Section 5.1.1 it is concluded that a bigger number of generated samples and committee members does not increase performance, but only the runtime. Therefore it was found that the workflow should generate 10 samples and that a committee size of 10 is to be used. The combination of components that then allows for the quickest conversion of the fitness is found in Section 5.1.2. It consists of the MDG, which provides the most extreme diversity measure, together with QBC, but without any restrictive behavior. When restriction is considered one should opt for the HR, for which the criteria

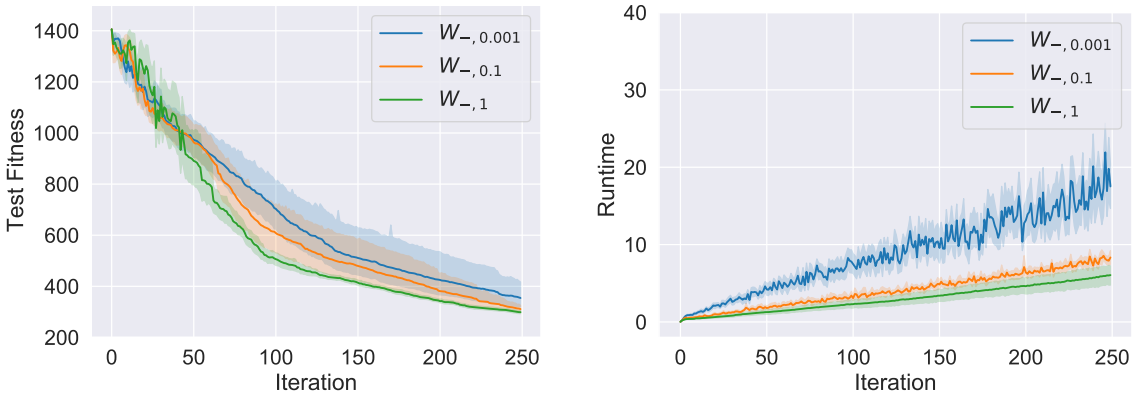
Table 5.9: Pair-wise $p-$values for the aggregated workflows based on the maximum fitness difference v. Note that significant values are highlighted with bold text.

(a) $p-$values comparing the AUC of the fitness.

	$W_{-,0.001}$	$W_{-,0.1}$	$W_{-,1}$
$W_{-,0.001}$	1.0000e+00	4.4704e-01	**4.9357e-02**
$W_{-,0.1}$	-	1.0000e+00	1.9178e-01
$W_{-,1}$	-	-	1.0000e+00

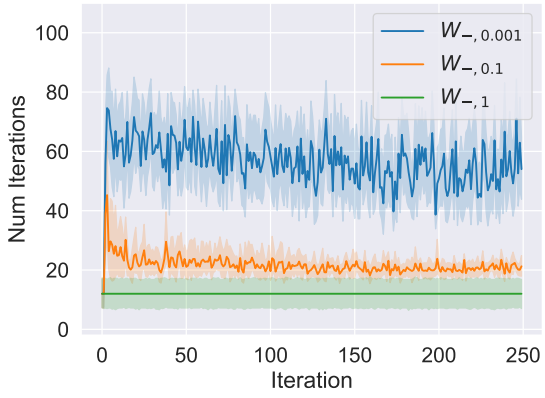
(b) $p-$values comparing the AUC of the runtime.

	$W_{-,0.001}$	$W_{-,0.1}$	$W_{-,1}$
$W_{-,0.001}$	1.0000e+00	**1.0395e-18**	**3.0601e-12**
$W_{-,0.1}$	-	1.0000e+00	**6.7207e-03**
$W_{-,1}$	-	-	1.0000e+00



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.11.

(b) Runtime in seconds of the selection process for each variation.



(c) The number of iterations j it took to select a sample.

Figure 5.13: Aggregated results of the different maximum fitness difference.

parameters are investigated in Section 5.1.3. From this, it shows that a set static amount of minimum iterations as well as a mild (high) maximum fitness criterion to approach the non-restricted behaviour is best.

5.2. Performance Comparison

In this set of experiments the performance of best-performing workflow from Section 5.1 is tested. The workflow using this combination is compared against W_{base} as defined in Equation 5.3. From Section 5.1 it can be concluded that the MDG with 10 generated samples achieves the best performance and is therefore used in this experiment as well. The committee used by QBC in these experiments is 10 as well as no considerable fitness difference can be observed in Section 5.1.1. Section 5.1.3 shows that no significant can be observed by adding restriction behaviour. However, the results show that initial stability is better achieved when some concentration on a subregion is added through specifying a standard number of iterations. To allow for this stability in these experiments, we use a middle road and use 5 minimum iterations. To make sure that there are similar samples in the generated set in terms of informativeness, a mild maximum fitness criterion is added as a stopping criterion of $p = 0.5$. Using these settings, the following workflow is derived:

$$W_{new} = W(MDG(N = 10), QBC(C = 10), HR, [MICC(j_{min} = 5)], [MFC(percent, p = 0.5)]) \quad (5.16)$$

In the following two subsections W_{new} and W_{base} are compared using two different oracles, namely the oracle used previously, as well as an oracle corresponding to a car-following situation.

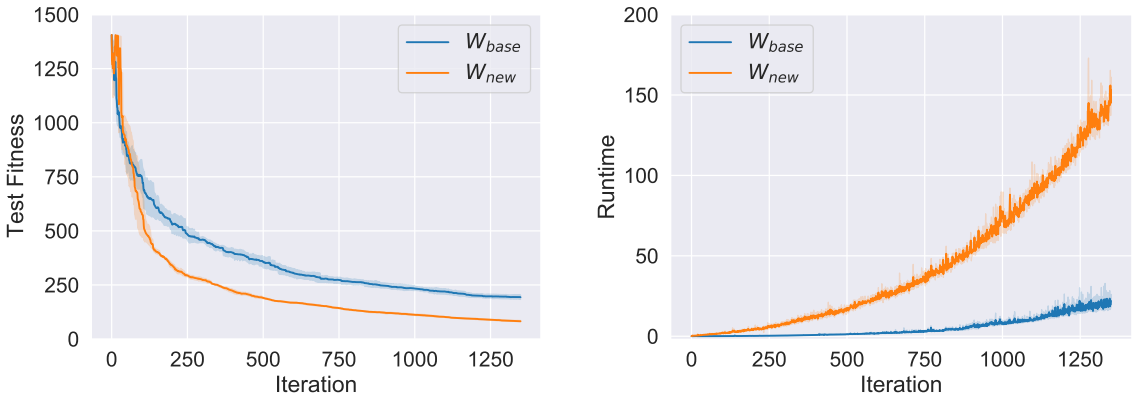
5.2.1. Polynomial Oracle

In this subsection the oracle mentioned in Equation 5.7 is used to determine the performance differences between the two workflows. Equal to Section 5.1 the three-dimensional variant is adopted, using the same lower and upper bounds for each dimension of -1000 and 1000 respectively. Each run selects 1350 samples, as initial experiments have shown that after this point there is no change in a model's fitness.

Since W_{new} is designed to outperform the baseline and also optimized to do so, the main hypothesis to consider is that in terms of iterations it will outperform W_{base}. The base workflow should retrieve samples randomly and therefore the distribution of these samples should be close to uniform in each dimension. W_{new} should however consider regions that account for high error rates, which are the regions close to the upper and lower bounds in each dimension. That being said, W_{new} should not only choose samples within this region as diversity is also to be considered, but definitely be biased to these areas. The baseline will however only incur runtime from training the model, where the actual selection procedure's runtime can be neglected, as this only consists of generating three random numbers. Opposed to that, W_{new} does require at least 10 times more runtime, as it instead has to train 10 committee members each, as well as the model used for evaluation. This is without even introducing the generator or restrictive behaviour to the equation. Therefore, the proposed workflow definitely has a higher runtime than that of the baseline at a certain iteration. In terms of runtime to achieve a certain fitness is concerned it is expected that the runtime of W_{new} still exceeds that of W_{base} when no labeling time is taken into account. When a labeling time per iteration is however introduced, it may be the case that W_{new} is then the best choice.

In Figure 5.14 the fitness and runtime results are displayed for both W_{base} and W_{new}. What is immediately clear is that the proposed workflow outperforms the baseline throughout the runs in terms of fitness, as is visualized in Figure 5.14a. This difference in performance is very much significant, with a $p-$value of roughly $8.4 \cdot 10^{-8}$. As hypothesized, the runtime of the two workflows is also significantly different ($p \approx 3.55 \cdot 10^{-15}$), obviously showing that the proposed workflow requires a longer time to run than the baseline

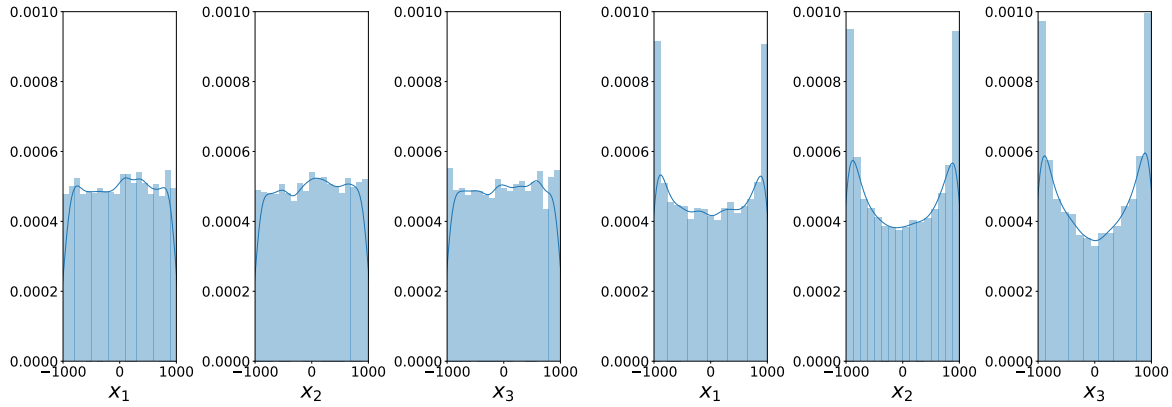
The difference in performance can be explained by the distribution of the samples, which can be found in Figure 5.15. From the distributions of W_{base} in Figure 5.15a it an be seen that it is uniformly distributed in all dimensions, as is expected. However, as explained before, the extreme regions, i.e. those near the lower and upper bounds, contribute more to the fitness. The predictions in these areas should therefore be accurate to achieve a well performing model. Figure 5.15b shows that W_{new} does just this; the samples are definitely more biased to these extreme values. It should be noted that although this bias is visible, there is still a coverage of the complete range through the implemented



(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.3 and Equation 5.16.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.14: Results of the baseline and proposed workflow using the three-dimensional polynomial oracle defined in Equation 5.7



(a) W_{base}

(b) W_{new}

Figure 5.15: Distribution of selected samples by both workflows per dimension.

diversity metric. Lastly, we note that the higher dimensions contribute even more than the lower dimensions to the error, which can also be seen in the proposed strategy's distribution, where the higher numbered dimensions receive more bias than the lower numbered dimensions.

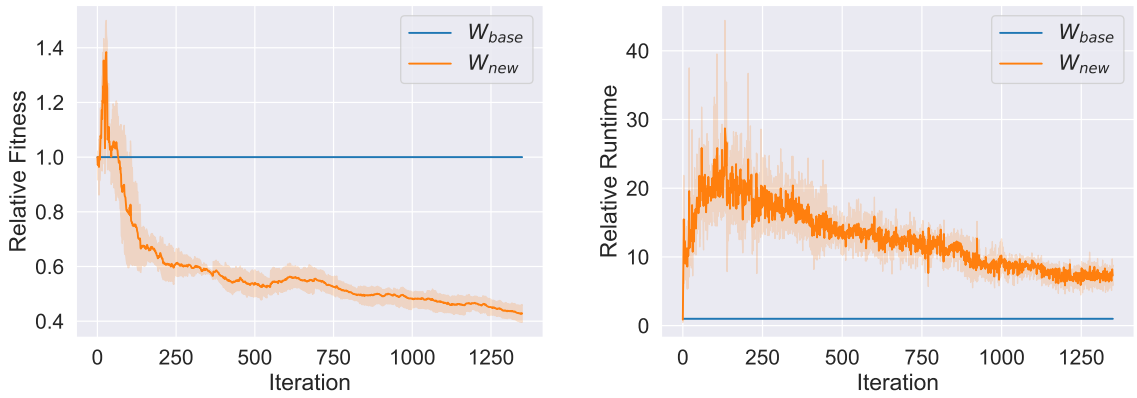
To compare the relative scores, Figure 5.16 reports the relative measures for both the fitness and runtime. In Figure 5.16a it becomes apparent that for the first 120 iterations the proposed workflow is outperformed by the baseline. However, after this number of iterations the workflow has settled and consistently outperforms the baseline with as much as 50% in later iterations.

The relative difference in runtime depicted in Figure 5.16b shows that this is indeed roughly in line with the set hypothesis. Although earlier iterations show that the relative difference is a lot more than 10 times, this is mainly because of the sample generation and restrictive behaviour still having impact on the runtime. In later iterations it shows that this relation indeed converges to around 11 times as hypothesized.

To determine whether using the proposed workflow is worth it when runtime and fitness are considered together, we visualize the time to reach a certain fitnesss value. We consider three different situations in terms of labeling time t_l: one where this does not take any time, one where it takes some seconds to label a sample, and one where it takes a minute. At a certain iteration i, consider the aggregate runtime t_i^a of a workflow:

$$t_i^a = t_{i-1}^a + t_i + t_l, \tag{5.17}$$

where t_i is the runtime reported previously, consisting of the selection process and the fitting of the model, and $t_0^a = 0$. Figure 5.17 visualizes the time to achieve a certain test fitness for the t_l values

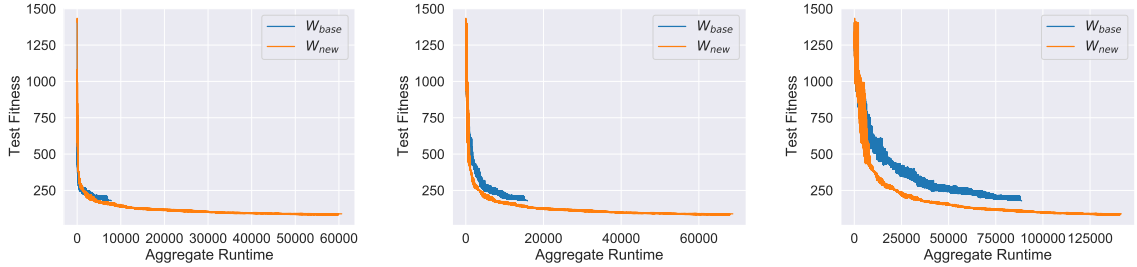


(a) Relative Test Fitness. The workflows correspond to the definition in Equation 5.3 and Equation 5.16.

(b) Relative Runtime.

Figure 5.16: Relative results compared to W_{base} per iteration for the polynomial oracle. Values higher than 1 mean higher metric values at that point, and therefore lower values mean lower values at that point than W_{base}'s values.

$0, 6$, and 60 respectively. What can be noted from this figure is that it is almost always worth using the proposed workflow to achieve a certain fitness, even when no labeling time is considered. The lower this certain fitness is, the better it is to use the proposed workflow in this situation. Looking at Figure 5.17a, when a fitness of 250 is to be achieved, both solutions take equal time. However, when a minute of labeling time is considered, as in Figure 5.17c, using W_{base} results in roughly 3 times as much time being consumed by W_{base} opposed to W_{new}. This therefore shows that the proposed workflow performs well, but is even more worthwhile when labeling takes some time.



(a) $t_l = 0$ (b) $t_l = 6$ (c) $t_l = 60$

Figure 5.17: Aggregated runtime as defined in Equation 5.17 required to achieve a certain fitness, using a different labeling time t_l in each subfigure for the polynomial oracle.

5.2.2. Car-Following Oracle

To allow for a situation which depicts a real-life use-case of the model, we use an oracle which represents a car-following environment explained in Figure 1.1. To simulate this situation, the simulation software SUMO is used[2]. SUMO allows for very extensive, complex, and large networks to be simulated with different types of entities including bicycles, pedestrians, and trains. There are many car following models included in SUMO, where in this comparison the Intelligent Driver Model (IDM) is used [1]. The acceleration a within the IDM is formulated as follows:

$$a = a_{max} \cdot (1 - (\frac{v_f}{v_0})^{\delta} - (\frac{s^*(v_f, v_f - v_l)}{s})^2), \tag{5.18}$$

[2] https://sumo.dlr.de/

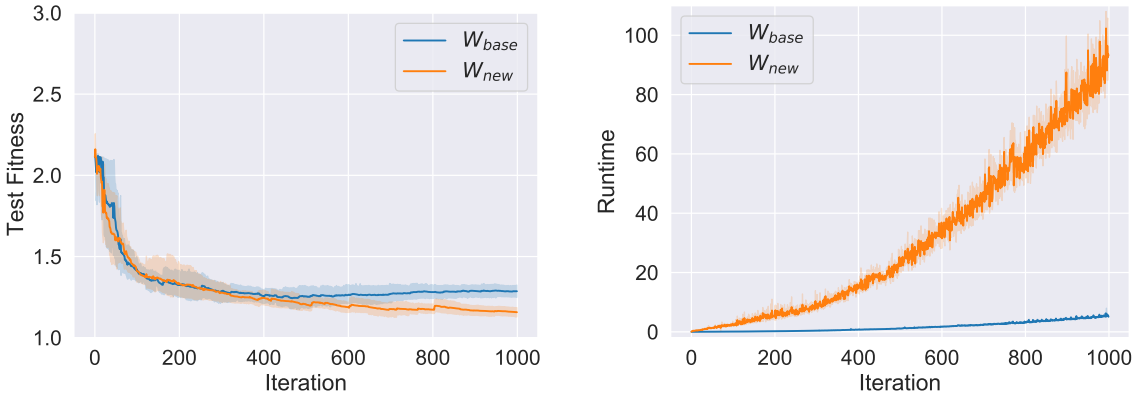
where a_{max} is the maximum acceleration of the vehicle, v_0 is the desired speed, δ is a control parameter, and $s^*(v, \Delta v)$ denotes the desired distance:

$$s^*(v, \Delta v) = s_0 + \max(0, v \cdot T + \frac{v \cdot \Delta v}{2\sqrt{a_{max} \cdot b}}), \tag{5.19}$$

where s_0 denotes the minimum distance gap, T the time gap, and b the comfortable deceleration of the vehicle. For a_{max}, b, and s_0 the passenger vehicle supplied by SUMO is used, where $a_{max} = 2.6ms^{-2}$, $b = 7.5ms^{-2}$, and $s_0 = 2.5m$ respectively. For T and δ, the SUMO defaults of 1 and 4 are used respectively. For v_0 the maximum speed of the vehicle is used, which is $50ms^{-1}$. To make sure the simulation is more like the real world, SUMO also provides the driver's imperfection parameter $0 \leq \sigma \leq 1$, where 0 is driving exactly according to the model. For this parameter, the default of SUMO is used, i.e. $\sigma = 0.5$

The sample space consists of a three-dimensional space, where a sample $\mathbf{x} = [v_l, v_f, s]$. Since both v_l and v_f are constrained by the maximum speed of the passenger vehicle and are not going backwards, the first two dimensions are the ranges $[0, 50]$. For the distance, any distance between 0 and 200 can be selected. The sample space $\mathcal{X}$ can therefore be defined as $S = [0, 50] \times [0, 50] \times [0, 200]$. The output space $\mathcal{Y}$ consists of all possible accelerations, that is $[b, a_{max}]$. Note that b is a negative number which represents a deceleration.

Since the results of Subsection 5.2.1 are from a scientific setting due to the oracle used and since the current experiment approaches real-life more, the performance difference is assumed to be smaller than that observed in Subsection 5.2.1. We do hypothesize that the proposed strategy does again exhibit a bias to the interesting values. In the car-following setting, these are those values with a high following speed v_f, a slow leader speed v_l, and a short distance s. In all these situations the reaction of the follow vehicle should be most extreme and these situations should therefore be prioritized on. It is however still expected that the runtime behaves in a similar fashion to what is observed previously and hence a 10 time difference between the two workflows is expected. Due to the difference in fitness mentioned above, the hypothesis further is that a longer labeling time t_l is required to make the proposed workflow worthwhile compared to the time observed in Subsection 5.2.1.

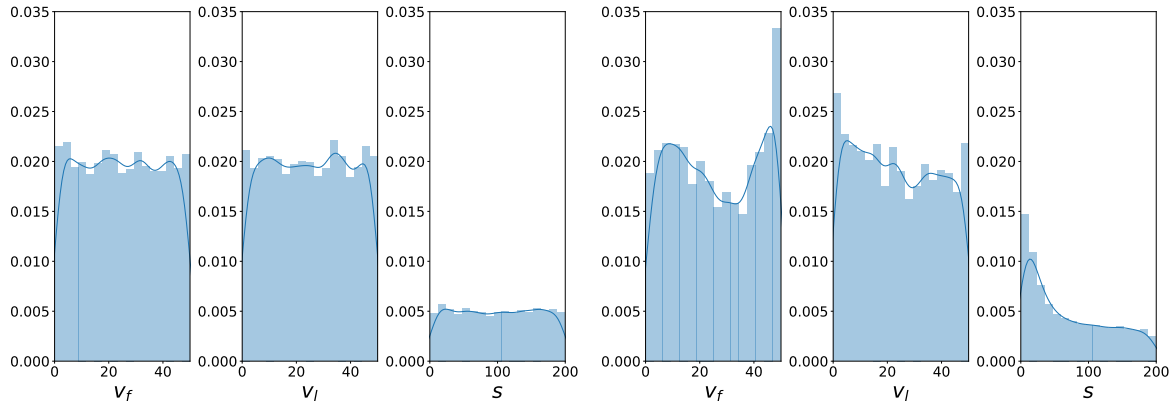


(a) Test Fitness as defined in Equation 5.5. The workflows correspond to the definition in Equation 5.3 and Equation 5.16.

(b) Runtime in seconds of the selection process for each variation.

Figure 5.18: Results of the baseline and proposed workflow using the car-following oracle defined in Equation 5.18

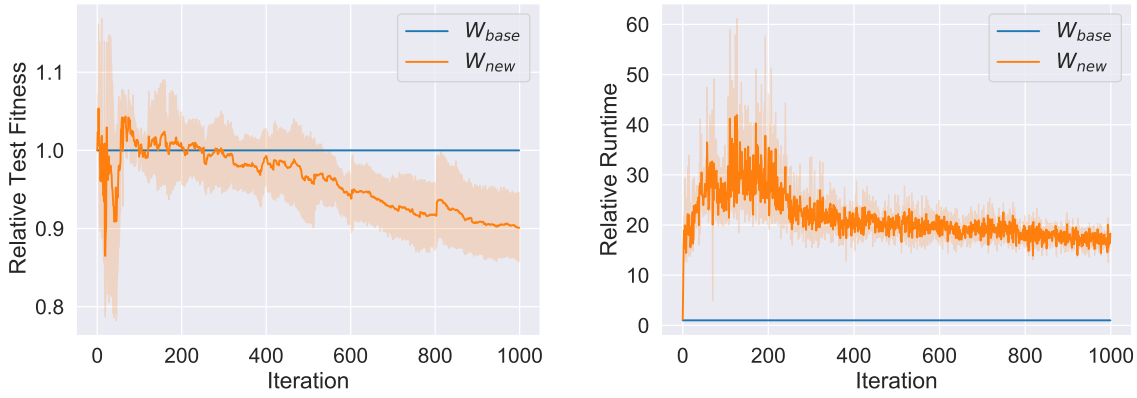
Figure 5.18 shows the fitness and runtime results obtained by this experiment. Note that only 1000 iterations are used in this experiment over 1350 in the previous experiment since initial experiments have shown no further improvements are obtained after running the workflows for longer. What becomes immediately apparent is that the fitness featured in Figure 5.18a shows less difference between the two workflows than previously observed, especially noting that in this case this difference is deemed insignificant ($p \approx 0.179$). This is in line with the hypothesis in a sense that the difference is less extreme, but it is still invalidated as this difference is not significant. The runtime still aligns with previous observations as expected, where the proposed workflow definitely requires more time to select samples than the baseline, as is to be expected.



(a) W_{base} (b) W_{new}

Figure 5.19: Distribution of selected samples by both workflows per dimension.

Once again, the distribution of samples is in line with the hypothesis as depicted in Figure 5.19. Similar to the previous section, Figure 5.19a shows that W_{base} indeed draws samples uniformly random. Moreover, we do see the difference with W_{new}'s distribution in Figure 5.19b, which is indeed biased towards the aforementioned situations. However, the distribution of W_{new}'s v_f dimension also shows that focus is put on a low following speed, which is interesting.

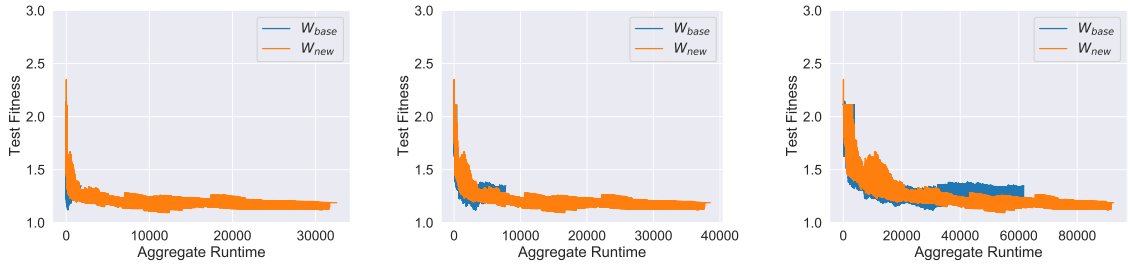


(a) Relative Test Fitness. The workflows correspond to the definition in Equation 5.3 and Equation 5.16.

(b) Relative Runtime.

Figure 5.20: Relative results compared to W_{base} per iteration for the Car-Following oracle. Values higher than 1 mean higher metric values at that point, and therefore lower values mean lower values at that point than W_{base}'s values.

When looking at the relative metrics displayed in Figure 5.20, it becomes more apparent that there is a difference between the current observations and those in Subsection 5.2.1. The relative fitness displayed in Figure 5.20a shows that although the proposed workflow does provide a small performance increase smaller than 10 percent, it is nowhere near the 50 percent performance increase observed with the polynomial oracle. It also appears that the relative runtime of the workflow is different, converging to a 20 times increase over the 10 times previously observed. This can be due to the different training times for the models using this oracle, as can be seen comparing the runtimes of W_{base} in Figure 5.14b and Figure 5.18b. In the first figure, in final iterations W_{base} takes roughly 20 seconds, whereas in the latter it only takes 10. This would mean that the sample generation and further selection overhead has a bigger factor in the runtime, hence the relative runtime being higher than in Figure 5.16b. This might also explain the smaller difference in performance, as it might be the case that the car-following concept is easier to learn than the polynomial concept, hence the lower runtimes. Similar to Subsection 5.2.1, we once again employ the concept of the aggregate runtime defined in Equation 5.17 with different labeling times t_l to determine whether it's worthwhile using the proposed workflow over random sam-



(a) $t_l = 0$ (b) $t_l = 6$ (c) $t_l = 60$

Figure 5.21: Aggregated runtime as defined in Equation 5.17 required to achieve a certain fitness, using a different labeling time t_l in each subfigure for the Car-Following oracle.

pling. The results are displayed in Figure 5.21. Opposed to previous observations it seems that there is no distinction to be made between the two workflows when the labeling time is either non-existent or small (0 and 6 respectively), as can be seen in Subfigures 5.21a and 5.21b. However, even with the minor increase in performance it is still worth to use the proposed workflow when a minute of labeling time is considered, as can be seen in Figure 5.21c. This is in line with our expectations, as longer labeling times still make the proposed workflow perform better, although this being less extreme than the results observed previously.

6

Discussion

The overall goal of this thesis was to create a selection strategy for population-based regression AL. This strategy had to be performant, efficient, extensible, and generally applicable. The latter two properties are satisfied by making the strategy modular and using black-box model. Work in pool-based approaches is leveraged in both generators and prioritizers. Similar to previous work this approach takes advantage of the maximum minimum distance metric [4, 8, 25–27, 29, 32, 64, 65] and QBC [4, 25, 63–67], providing both a diversity and informativeness measure to the workflow. For restriction the BBR specifically uses the conceptual notion of supported sides defined by Backer and Mark Keil [72]. To determine whether the strategy is performant and efficient, Chapter 5 describes a number of experiments to determine the performance of different component combinations and its hyperparameters, as well as compare this performance to a baseline. The results indicate that there is no performance increase when a larger number of samples is generated or a larger number of committee members is considered, although the runtime is influenced by the latter hyperparameter. In terms of generators it the results suggest that using a diversity metric improves the performance significantly. For restrictors it showed that not using any restrictive behaviour is best for both runtime and fitness and that no clear difference between the HR and BBR can be found in either of the two categories. When further investigating the influence of the stopping criteria it showed that a small number of minimum iterations increases stability in the starting sample selection iterations compared to not having this continuation criterion. This can best be combined with a lenient maximum fitness difference, as using a more strict difference decreases performance and increases both runtime and number of iterations. Comparing the proposed workflow to a random baseline further indicates that the strategy generally outperforms the baseline. Depending on the problem this difference in performance differs, showing increases as big as fifty percent in one situation, whereas another situation only shows a five percent increase in later iterations. This difference is hypothesized to be due to the simplicity of the concept to learn, where the workflows perform more similar the easier this concept is. Considering that the proposed strategy takes considerably longer to select a sample than the baseline, the results suggest that it is best applied the longer the expected labeling time is. In this situation the quicker convergence in terms of number of labeled samples of the proposed strategy can be fully leveraged.

As the population-based regression problem used in this thesis is not discussed prior in previous work, this thesis provides initial results of an selection strategy in this problem setting. The oracles used for the performance evaluation are therefore also novel, as no prior ways of evaluation are described in literature. The generalizability of the results therefore is partially influenced and limited by choice of these oracles. Most notably, the current used oracles are relatively low-dimensional, both using three dimensions. The results cannot be generalized to situations where many more dimensions are considered, as this might influence the performance of both baseline and the suggested strategy. Moreover, in the results the underlying model used is the Gaussian Process Regressor. Similar to the oracles used, the observed results are therefore not generalizable to the use of other underlying models. That being said, the strategy does allow for other models to be used in identical fashion without further complications.

A self-evident direction of further research would therefore be to further research the generalizability of the observed results. This can be done by employing higher dimensional oracles, differentiating

between different types of polynomial functions, or testing the performance using other real-life situations. The latter can also be done by transforming a real-life dataset to an infinite sample space, through performing interpolation between the existing labeled samples. This way one does not have to rely on expressing the situation through a mathematical function and instead the workflow can be applied in a broader fashion. Similarly further research can also employ different types of models to also specifically find out whether switching out models preserves the same obtained trends in fitness in runtime.

The current thesis describes a modular workflow consisting of a generator, prioritizer, restrictor, and criteria to guide the restriction effort. For each of these components a multitude of different implementations are discussed. The set of varieties can also be further extended in future work. For generators, one could look in the direction of space discretization, optionally including a diversity metric. All pool-based regression solutions which do not have assumptions on the model can be used as a prioritizer, or the QBC can be implemented using a different informativeness metric than the prediction variance.

When using this thesis' strategy in practice, one should note that results on hyperparameters provide limited generalizability. It is therefore key that the user first confirms that similar performance is achieved on their situation, or a simplified representation thereof. Moreover, the situation should be suitable for applying a population-based strategy in the first place. As explained previously, this should include a situation where every point in the sample space can be labeled without ambiguity.

7 Conclusion

In this thesis, a modular approach to population-based AL for regression is presented, which is applied to a car following situation. Firstly, the problem is defined mathematically, namely to select those samples such that at each point in time, the fitted set of parameters is as close to optimal as possible. This is done through the iterative AL procedure, where each iteration a sample is selected from the sample space, labeled, and then added to the training set. After these groundworks are laid out this thesis continues with examining the current state-of-the-art research and approaches to population-based regression. Since no prior research is found that covers both categories without further assumptions, instead the work is split in works on population-based AL and pool-based AL for regression.

With this basis, this thesis' modular selection strategy is explained thoroughly. Through this modularization, future extensions can easily be implemented. This strategy assumes a black-box model allowing for general applicability. For each sample selection, the strategy gradually concentrates on an interesting region within the sample space using an iterative process. Within each iteration, three steps take place. Firstly, the generator generates a set amount of samples from the current region, whilst possibly including a diversity measure. This thesis implements three variants: one which generates these samples randomly and two which generate more than the required samples and select using the minimum distance to samples in the training set. From the latter two, the Maximum Distance Generator uses this distance in selecting samples above a set approximate distance percentile, while the Probabilistic Distance Generator uses these distances to derive a probability. The second step is the prioritization, in which the generated samples are scored and ordered based on an informativeness criterion. In this step, previous work on pool-based regression AL is leveraged. For the prioritizer two varieties are described: random prioritization or Query by Committee. The latter prioritizes the samples using a group of models, each trained on a subsample of the training set. The prediction variance is then used to order the samples. Lastly, the current region is restricted, focusing on the most interesting region based on the prioritizer's order and scores. The Halfspace restrictor restricts the current region through splitting it in two and focusing on the split with the highest mean prioritizer score. Instead the Bounding Box Restrictor bases a big hyperrectangle around different fittest points and restricts the current region this way. To control the number of focusing iterations, both continuation and stopping criteria are defined which determine the minimum and maximum effort respectively. This can be done either through specifying the minimum and maximum iterations explicitly, or specifying a maximum allowed difference in the prioritizer's scores required in order to stop further iterations. After this procedure is stopped, the fittest sample is selected and in turn labeled.

Using a three-dimensional polynomial function as oracle, the combinations of different components and their hyperparameters is evaluated using a series of experiments. From these results, it followed that the combination of the Maximum Distance Generator and Query By Committee without any restriction effort performs best and provides the best fitness-runtime tradeoff when a small number of samples are generated and the committee consists of a small number of models. When considering restriction, similar and more stable results are obtained with a small number of minimum iterations and a lenient maximum fitness difference.

To test the performance and efficiency of the strategy, it is compared to a baseline that randomly selects a sample. Using two different oracles, the results show that the proposed strategy is more

performant in terms of fitness per number of labeled samples, showing increases in performance up to fifty percent. Moreover, when instead accumulated runtime is considered, it is noted that the proposed strategy is especially worthwhile when the labeling effort is high and therefore takes a large amount of time.

Overall it should be noted that this thesis' selection strategy therefore forms the answer to **RQ1**, as it has shown all required properties. Since this thesis lays the groundworks for population-based AL for regression, the generalizability of its results is restricted to both low-dimensional sample spaces and the use of Gaussian Process Regressors as a model. Future work can be focused on extending this generalizability, both through applying the same methodology to other (higher-dimensional) problems and using different models in the process. This thesis approach can also be further extended by providing additional varieties to the steps within its workflow, allowing for more custom and well-performing workflows.

Bibliography

[1] M. Treiber and A. Kesting, *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 2013.

[2] B. Settles, “Active learning literature survey,” University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[3] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, A. Sparkes, K. E. Whelan, and A. Clare, “The automation of science,” *Science*, vol. 324, no. 5923, pp. 85–89, 2009.

[4] D. Wu, “Pool-Based Sequential Active Learning for Regression,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1348–1359, 2019.

[5] M. Sugiyama and S. Nakajima, “Pool-based active learning in approximate linear regression,” *Machine Learning*, vol. 75, no. 3, pp. 249–274, 2009.

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

[7] 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*, vol. 28, no. 7, pp. 1668–1681, 2017.

[8] B. Demir and L. Bruzzone, “A multiple criteria active learning method for support vector regression,” *Pattern Recognition*, vol. 47, no. 7, pp. 2558–2567, 2014.

[9] D. Angluin, “Queries and concept learning,” *Machine Learning*, vol. 2, no. 4, pp. 319–342, Apr 1988.

[10] K. Lang and E. Baum, “Query learning can work poorly when a human oracle is used,” *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 335–340, 1992.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, vol. 3, no. January, pp. 2672–2680, 2014.

[12] J.-J. Zhu and J. Bento, “Generative adversarial active learning,” 2017.

[13] M. Huijser and J. C. Gemert, “Active Decision Boundary Annotation with Deep Generative Models,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 5296–5305, 2017.

[14] R. Schumann and I. Rehbein, “Active learning via membership query synthesis for semi-supervised sentence classification,” in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, Nov. 2019, pp. 472–481.

[15] I. Alabdulmohsin, X. Gao, and X. Zhang, “Efficient active learning of halfspaces via query synthesis,” in *Proceedings of the National Conference on Artificial Intelligence*, vol. 4, 2015, pp. 2483–2489.

[16] Y. Wang and A. Singh, “Noise-Adaptive margin-based active learning and lower bounds under tsybakov noise condition,” *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2180–2186, 2016.

[17] L. Chen, H. Hassani, and A. Karbasi, “Near-Optimal Active Learning of Halfspaces via Query Synthesis in the Noisy Setting,” *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 1798–1804, 2017.

[18] A. Farooqui and M. Fabian, “Synthesis of supervisors for unknown plant models using active learning,” in *IEEE International Conference on Automation Science and Engineering*, vol. 2019-Augus, 2019, pp. 502–508.

[19] D. P. Wiens, “Robust weights and designs for biased regression models: Least squares and generalized M-estimation,” *Journal of Statistical Planning and Inference*, vol. 83, no. 2, pp. 395–412, 2000.

[20] A. D. Bull, “Spatially-adaptive sensing in nonparametric regression,” *Annals of Statistics*, vol. 41, no. 1, pp. 41–62, 2013.

[21] T. Akiyama, H. Hachiya, and M. Sugiyama, “Efficient exploration through active learning for value function approximation in reinforcement learning,” *Neural Networks*, vol. 23, no. 5, pp. 639–648, 2010.

[22] K. Chaudhuri, S. M. Kakade, P. Netrapalli, and S. Sanghavi, “Convergence rates of active learning for maximum likelihood estimation,” in *Advances in Neural Information Processing Systems*, vol. 2015-Janua, 2015, pp. 1090–1098.

[23] M. Sugiyama, “Active learning in approximately linear regression based on conditional expectation of generalization error,” *Journal of Machine Learning Research*, vol. 7, pp. 141–166, 2006.

[24] S. H. Park and S. B. Kim, “Active semi-supervised learning with multiple complementary information,” *Expert Systems with Applications*, vol. 126, pp. 30–40, 2019.

[25] L. Fusani and A. C. Cabrera, “Active learning strategies with COMBINE analysis: new tricks for an old dog,” *Journal of Computer-Aided Molecular Design*, vol. 33, no. 2, pp. 287–294, 2019.

[26] J. Yang, X. Zhao, H. Wei, and K. Zhang, “Sample selection based on active learning for short-term wind speed prediction,” *Energies*, vol. 12, no. 3, 2019.

[27] A. Appice, C. Loglisci, and D. Malerba, “Active learning via collective inference in network regression problems,” *Information Sciences*, vol. 460-461, pp. 293–317, 2018.

[28] S. Mohamad, A. Bouchachia, and M. Sayed-Mouchaweh, “A Bi-Criteria Active Learning Algorithm for Dynamic Data Streams,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 74–86, 2018.

[29] E. Pasolli and F. Melgani, “Gaussian process regression within an active learning scheme,” in *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2011, pp. 3574–3577.

[30] P. R. Silveira, D. D. F. Naiff, C. Pereira, and R. Schirru, “Reconstruction of radiation dose rate profiles by autonomous robot with active learning and Gaussian process regression,” *Annals of Nuclear Energy*, vol. 112, pp. 876–886, 2018.

[31] J. Goetz, A. Tewari, and P. Zimmerman, “Active learning for non-parametric regression using purely random trees,” in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, 2018, pp. 2537–2546.

[32] Y. Son and J. Lee, “Active learning using transductive sparse Bayesian regression,” *Information Sciences*, vol. 374, pp. 240–254, 2016.

[33] J. Proppe, S. Gugler, and M. Reiher, “Gaussian Process-Based Refinement of Dispersion Corrections,” *Journal of Chemical Theory and Computation*, vol. 15, no. 11, pp. 6046–6060, 2019.

[34] S. Zhang, G. Deng, and F. Wang, “Active learning strategy for online prediction of particle size distribution in cobalt oxalate synthesis process,” *IEEE Access*, vol. 7, pp. 40 810–40 821, 2019.

[35] H. Azizsoltani and E. Sadeghi, “Adaptive sequential strategy for risk estimation of engineering systems using Gaussian process regression active learning,” *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 146–165, 2018.

[36] L. Bassman, P. Rajak, R. K. Kalia, A. Nakano, F. Sha, J. Sun, D. J. Singh, M. Aykol, P. Huck, K. Persson, and P. Vashishta, “Active learning for accelerated design of layered materials,” *npj Computational Materials*, vol. 4, no. 1, 2018.

[37] K. Kandasamy, J. Schneider, and B. Póczos, “Query efficient posterior estimation in scientific experiments via Bayesian active learning,” *Artificial Intelligence*, vol. 243, pp. 45–56, 2017.

[38] ——, “Bayesian active learning for posterior estimation,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2015, pp. 3605–3611.

[39] Z. Ge, “Active probabilistic sample selection for intelligent soft sensing of industrial processes,” *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 181–189, 2016.

[40] G. Jun, J. Ghosh, V. Radosavljevic, and Z. Obradovic, “Predicting ground-based aerosol optical depth with satellite images via Gaussian processes,” in *KDIR 2010 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, 2010, pp. 370–375.

[41] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient Global Optimization of Expensive Black-Box Functions,” *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[42] E. Pasolli, F. Melgani, N. Alajlan, and Y. Bazi, “Active learning methods for biophysical parameter estimation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 10 PART2, pp. 4071–4084, 2012.

[43] D. Gu and H. Hu, “Active learning of Gaussian processes for spatial functions in mobile sensor networks,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 44, no. 1 PART 1, 2011, pp. 13 564–13 569.

[44] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, “Safe exploration for active learning with gaussian processes,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9286, pp. 133–149, 2015.

[45] R. Garnett, M. A. Osborne, and P. Hennig, “Active learning of linear embeddings for gaussian processes,” in *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*, 2014, pp. 230–239.

[46] E. Burnaev and M. Panov, “Adaptive design of experiments based on gaussian processes,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9047, pp. 116–125, 2015.

[47] C. Kim, A. Chandrasekaran, A. Jha, and R. Ramprasad, “Active-learning and materials design: The example of high glass transition temperature polymers,” *MRS Communications*, vol. 9, no. 3, pp. 860–866, 2019.

[48] H. Deng, Y. Liu, P. Li, and S. Zhang, “Active learning for modeling and prediction of dynamical fluid processes,” *Chemometrics and Intelligent Laboratory Systems*, vol. 183, pp. 11–22, 2018.

[49] C. K. I. W. Carl Edward Rasmussen, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, nov 2005.

[50] D. Elreedy, A. F. Atiya, and S. I. Shaheen, “A novel active learning regression framework for balancing the exploration-exploitation trade-off,” *Entropy*, vol. 21, no. 7, 2019.

[51] J. Zheng and X. Li, “Active regression with compressive-sensing based outlier mitigation for both small and large outliers,” in *2016 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2016 - Proceedings*, 2017, pp. 172–176.

[52] D. Padmanabhan, S. Bhat, D. Garg, S. Shevade, and Y. Narahari, “A Robust UCB scheme for active learning in regression from strategic crowds,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016, 2016, pp. 2212–2219.

[53] C. Riquelme, R. Johari, and B. Zhang, “Online active linear regression via thresholding,” *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 2506–2512, 2017.

[54] K. Chaudhuri, P. Jain, and N. Natarajan, “Active heteroscedastic regression,” in *34th International Conference on Machine Learning, ICML 2017*, vol. 2, 2017, pp. 1156–1172.

[55] M. Golfarelli, S. Graziani, and S. Rizzi, “An active learning approach to build adaptive cost models for web services,” *Data and Knowledge Engineering*, vol. 119, pp. 89–104, 2019.

[56] E. Tsymbalov, M. Panov, and A. Shapeev, “Dropout-based active learning for regression,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11179 LNCS, pp. 247–258, 2018.

[57] G. Cuesta, C. Rio, and S. Jankowski, “Active learning of neural networks based on influential statistics,” in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 8903, 2013.

[58] C. Kading, E. Rodner, A. Freytag, O. Mothes, B. Barz, and J. Denzler, “Active learning for regression tasks with expected model output changes,” in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[59] X. Li, Y. Chen, and K. Zeng, “Integration of machine learning and human learning for training optimization in robust linear regression,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2016-May, 2016, pp. 2613–2617.

[60] H. S. Seung, M. Opper, and H. Sompolinsky, “Query by committee,” *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, no. September 2014, pp. 287–294, 1992.

[61] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, “Selective Sampling Using the Query by Committee Algorithm,” *Machine Learning*, vol. 28, no. 2-3, pp. 133–168, 1997.

[62] H. Drucker, “Improving regressors using boosting techniques,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 107–115.

[63] J. Chen, S. Li, B. Dai, and G. Zhou, “Active learning for age regression in social media,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10035 LNAI, pp. 351–362, 2016.

[64] F. Douak, F. Melgani, N. Alajlan, E. Pasolli, Y. Bazi, and N. Benoudjit, “Active learning for spectroscopic data regression,” *Journal of Chemometrics*, vol. 26, no. 7, pp. 374–383, 2012.

[65] F. Douak, F. Melgani, and N. Benoudjit, “Kernel ridge regression with active learning for wind speed prediction,” *Applied Energy*, vol. 103, pp. 328–340, 2013.

[66] D. L. Ly and H. Lipson, “Optimal experiment design for coevolutionary active learning,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 394–404, 2014.

[67] R. Nikzad-Langerodi, E. Lughofer, C. Cernuda, T. Reischer, W. Kantner, M. Pawliczek, and M. Brandstetter, “Calibration model maintenance in melamine resin production: Integrating drift detection, smart sample selection and model adaptation,” *Analytica Chimica Acta*, vol. 1013, pp. 1–12, 2018.

[68] Z. Ge, “Active learning strategy for smart soft sensor development under a small number of labeled data samples,” *Journal of Process Control*, vol. 24, no. 9, pp. 1454–1461, 2014.

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

[70] K. Ueki, M. Sugiyama, and Y. Ihara, “A semi-supervised approach to perceived age prediction from face images,” *IEICE Transactions on Information and Systems*, vol. E93-D, no. 10, pp. 2875–2878, 2010.

[71] L. Wang, X. Hu, B. Yuan, and J. Lu, “Active learning via query synthesis and nearest neighbour search,” *Neurocomputing*, vol. 147, no. 1, pp. 426–434, 2015.

[72] J. Backer and J. Mark Keil, “The bichromatic rectangle problem in high dimensions,” *Proceedings of the 21st Annual Canadian Conference on Computational Geometry, CCCG 2009*, pp. 157–160, 2009.

[73] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995.

[74] B. L. Welch, “The generalization of student’s’ problem when several different population variances are involved,” *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.