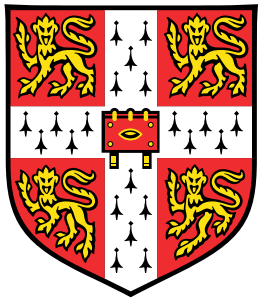
Repurposing Drugs for the Rapid Response to Epidemics and Pandemics

Using Batch Active Learning



Ross Brown

Department of Chemical Engineering and Biotechnology
University of Cambridge

This dissertation is submitted for the degree of
Master of Engineering

Robinson College May 2022

Declaration

The work described in this report is the result of my own research, unaided except as specifically acknowledged in the text, and it does not contain material that has already been used to any substantial extent for a comparable purpose. This report contains 40 pages and 7018 words (excluding this page, the title page, and the safety appendix).

Ross Brown
May 2022

Abstract

The SARS-CoV-2 pandemic saw a high death toll which was reduced as drugs such as Remdesivir were introduced as treatments. This process was slow and few drugs were found capable of reducing the mortality of the virus. A solution to this issue is presented: finding existing, workable treatments using batch active learning. This combines greedy methodologies with clusters and uncertainty analysis, accumulating in a final algorithm *RoDGer*. Algorithms were compared with each other, with the final algorithm learning the datasets at a 10% faster rate than simply randomly sampling.

Table of contents

Nomenclature

Chapter 2

N	Number of features (dimensions) of x
N	Number of data points x
s_g	Sample standard deviation of the predictions
x	Data points where $x = \{x_0, x_1, ..., x_{N-1}\}$
y	Labels for the dataset where $y = \{y_0, y_1, ..., y_{N-1}\}$
AC_{50}	Half maximal effective molar concentration
EC_{50}	Half maximal effective molar concentration
IC_{50}	Half maximal inhibitory molar concentration
Ki	Half maximal molar concentration for half receptor occupancy
LD_{50}	Median lethal dose
XC_{50}	Half maximal effective or inhibitory molar concentration

Chapter 3

X_{test}	Datasets used to provide a score for the algorithms
X_{train}	Datasets used for training the algorithms
x_{known}	Data points where the true label is available to the algorithms used
x_{unknown}	Data points where the true label is not available to the algorithms used
y_{known}	True labels available to the algorithms used
y_{unknown}	True labels unavailable to the algorithms used
n	The number of samples per iteration

Chapter 4

N	The number of datasets

Chapter 1

Introduction

In 2019, human civilisation was on the precipice of a natural disaster: SARS-CoV-2 (COVID-19). First reported to the World Health Organization (WHO) on December 31st, it became officially recognised as a pandemic on March 11th 2020. As of the writing of this passage, 515 million cases and 18 million excess deaths have been recorded [Wan+22; Wor22]. This, however, is not the first time a pandemic has occurred, with the Black Death infamously killing a third of Europe's population and the Spanish Flu causing mass death throughout the world. Likewise, it is unlikely to be the last.

When such a disaster does strike, it is important to react quickly. Vaccinations are developed and manufactured on accelerated timelines, cutting development time from years to months. Trials into potential treatments are encouraged with haste. When speed is not achieved with these measures, misinformation rapidly spreads. Within the first stages of the pandemic, drugs such as hydroxychloroquine and bleach were amongst several that were promoted by the President of the United States of America demonstrating the desperation in finding therapeutic drugs against the virus.

In order to facilitate a more robust approach to finding treatments, the FDA instigated the Coronavirus Treatment Acceleration Program [Cen22]. Here, over 690 drugs are in the development stage with over 450 clinical trials underway to investigate the effectiveness, with 15 drugs currently authorised for emergency use and only one drug, remdesivir, with approval for use against COVID-19 [Cen22]. These results, with the timescale in which they were achieved, are suboptimal. This is due to the slow, laborious, methods used in investigations into pre-existing drugs slow. Flawed selection priorities due to an information overload on scientists. This resulted in delays in treatment. Time many did not have.

A hopeful fulfilment of this problem is the "Robot Scientist" [Spa+10]; a fully automated combination of software and hardware aimed at solving this problem. For the software side, a form of reinforcement machine learning is proposed: batch active learning. Active learning attempts to determine a model on a problem by querying the fewest data points within a set of unlabelled data. This is a methodology suited to fields with large amounts of unlabelled data which is inherently difficult or expensive to label. In this case, the labelling requires chemical and biological experimentation costing both time and money. By using active learning, as few drugs as possible will be labelled within this

stage to accurately predict the best drugs for the given problem. From here, accelerated, targetted clinical trials may begin.

Due to the large importance of time, many drugs may be tested in parallel. This becomes even more practically considering the existence of robotic testing facilities. This presents an additional problem: how does one set up a testing scheme for batches? Can the same techniques used in single sample learning be transitioned across, or are more inventive methodologies required here?

Thus, the purpose of this thesis. To present an algorithm which may be used to discover effective drugs within a short period of time. Additionally, a framework will be developed that allows for different algorithms to be rigorously compared to each other for increased robustness. In doing so, several options are compared to each other, using a variety of techniques in order to produce a methodology more suited to quickly sampling large quantities of drugs against a target, such as the main protease of COVID-19 (3-chymotrypsin-like protease) [Tsu20]. Of the selection compared, RoDGer was found to be best, with the highest rate of initial learning. This method combines a technique called minimisation of the region of disagreement with clusterisation of the available samples and a greedy approach to aggressively find drugs with maximum activity.

To facilitate the combination of these different methods, as well as allowing robust comparison of different algorithms, a testing framework is introduced. This consists of two sections: a training section and a testing section. Using this, parameters may be introduced to allow for the optimal merging of the different methods before final being tested on a sample of datasets which were not used within the training process. This prevents data leakage: a technique commonly used within machine learning.

Using this framework, it was shown that popular algorithms for single sample active learning to perform worse than merely randomly sampling if employed on batches. Instead, a clustering algorithm was shown to be an extremely useful when scaling sample size. This allows commonly used methods to still be useful in batch learning. RoDGer clearly demonstrates this, combining both greedy methodology with regions of disagreement, two methods outperformed by random sampling, and provides a significant improvement on random sampling due to the combination of clustering.

Chapter 2

Previous Work

2.1 Active Learning

In order to assist the understanding of the methodologies used by others within the field of active learning, a toy dataset has been created. It is based upon (2.1) and has been shown in Figure 2.1. The y values used within the algorithms have been combined with errors, $\varepsilon \sim \mathcal{N}(0, 0.01)$.

$$y = \sin(x_0)^{10} + \cos(10 + x_0 x_1)\cos(x_0) \tag{2.1}$$

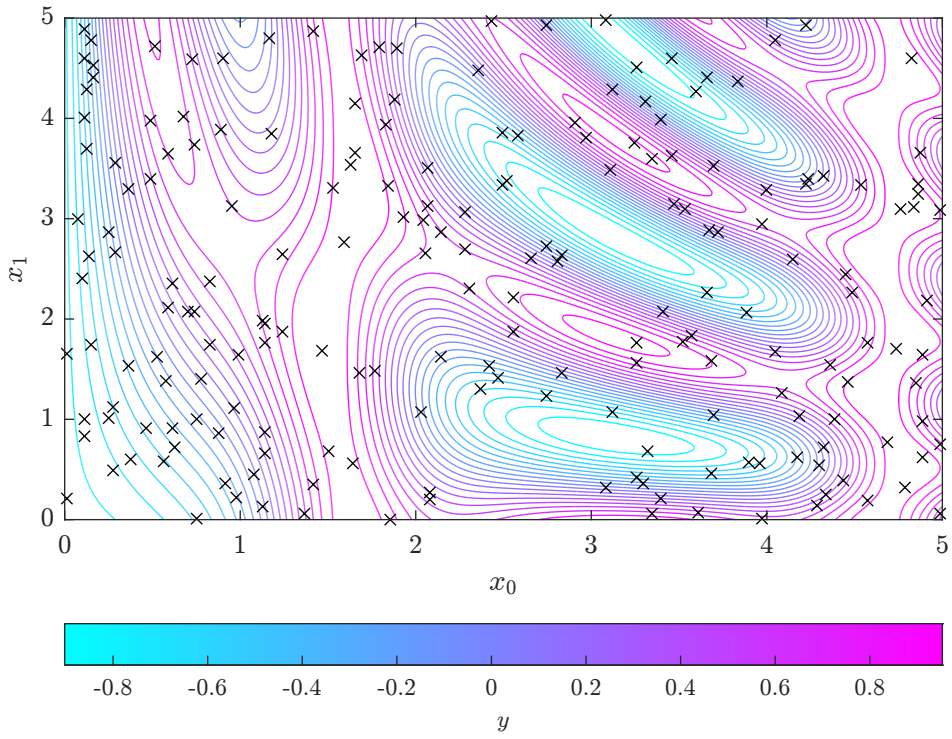


Fig. 2.1 Contour plot of the function used to demonstrate the algorithms presented in previous work. The crosses have been used to show the location of the 200 test data points used within this example.

In order to assess the algorithms, the mean squared error (mse) has been used. Comparisons are made to the naïve approach of Monte Carlo (random sampling). Each algorithm will be given five random starting points from which they will then attenuate to the data set.

There are several schools of thought regarding active learning. These can be separated into two distinct categories: current data and future predictions. The former of these is computationally cheaper, more complex to implement, and less adaptable to model changes.

2.1.1 Current Data

Uncertainty Sampling and Regions of Disagreements

When using probabilistic models, the data points with the highest uncertainty may be sampled first, as described by Settles [Set09]. According to this strategy, (2.2) gives the next point to sample. This, in principle appears to be simple to implement, but upon closer inspection raises some concerns. Most noticeable is the limitation it places upon the model used.

$$x_{\text{next}} = \underset{X}{\operatorname{argmin}} \left[P(y^*|x;\theta) \right] \tag{2.2}$$

Settles [Set09] also notes the use of information theory for probabilistic models (2.2), where $y^* = \operatorname{argmax}_y P(y|x;\theta)$ is the most likely y for x. This derives from the principle that the greatest entropy requires the most information to encode, and thus the least certain. However, Settles [Set09] fails to address non-probabilistic models in this instance, instead converting such models into probabilistic ones.

Settles [Set09] suggests another interpretation for uncertainty. By taking another approach from information theory, (2.3) is reached. This directly gives the point of the highest entropy, suggesting by knowing the point provides the largest information gain. Again however, this is difficult to implement with most models, as a probability distribution is required. This could be made simpler by approximating to a normal distribution.

$$x_{\text{next}} = \underset{x}{\operatorname{argmax}} \left[-\sum_i P(y_i|x;\theta) \log P(y_i|x;\theta) \right] \tag{2.3}$$

In order to adapt non-probabilistic models into probabilistic ones, composite models may be used. These are an amalgamation of other models where the standard deviation of the individual models can be taken as the degree of certainty for a given point. This is commonly referred to minimising the region of disagreement, referring to the areas of discord within the hypothesis space. By minimising the region of disagreement between various models, a more coherent hypothesis space is sought leading to a more accurate model. This is the method used in Figure 2.2. This is shown in (2.4).

$$x_{\text{next}} = \underset{X}{\operatorname{argmax}} \left[s_{g(X)} \right] \tag{2.4}$$

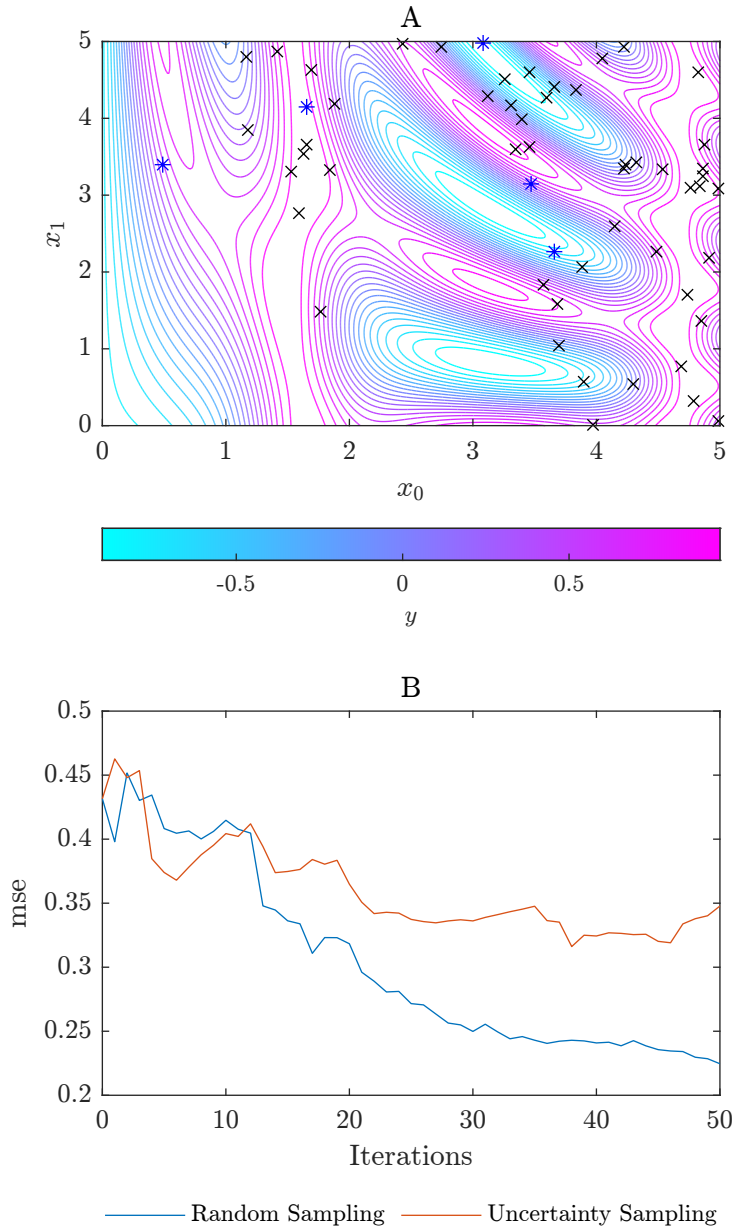


Fig. 2.2 The outcome of the investigating the areas of the highest uncertainty. An initial set of 5 random points (displayed as blue stars in A)) was provided, and 50 further iterations were then carried out of sample size 1. A) demonstrates the final set of points tested by the algorithm and B) shows the mean squared error for the algorithm after each iteration.

Interestingly, Figure 2.2B shows how the mean squared error for the random sampling method performed to worse within the iterations tested. This is likely due to a bias in the use of linear models in fitting leading to large uncertainties surrounding areas with high curvature. Evidence to this is provided in Figure 2.2A with a large proportion of the sampled points at areas of high curvature.

Density Hotspots

Conversely, a density weighted model has been suggested, as it escapes the introduction of error from outliers (i.e. data points far away from alternative data points) and artificially reduces the available number of data points. Settles and Craven [SC08] suggest (2.5) which can be broken down into two

parts: a function for selection, ϕ_A, and a function for similarity, sim. The former represent another function for determining which point to sample next. The latter requires a function to describe the similarity between data points. β is simply a parameter within this algorithm.

$$x_{\text{next}} = \underset{x}{\text{argmax}} \left[\phi_A(x) \times \left(\frac{1}{n} \sum \text{sim}(x, x_i) \right)^{\beta} \right] \tag{2.5}$$

Settles and Craven [SC08] admit that sim is open for interpretation. It must also be recognised that this lays the foundation of a clusterisation algorithm. There exist many forms of these algorithms, with the results of several of these algorithms on toy data sets presented in Figure 2.3 [Sci].

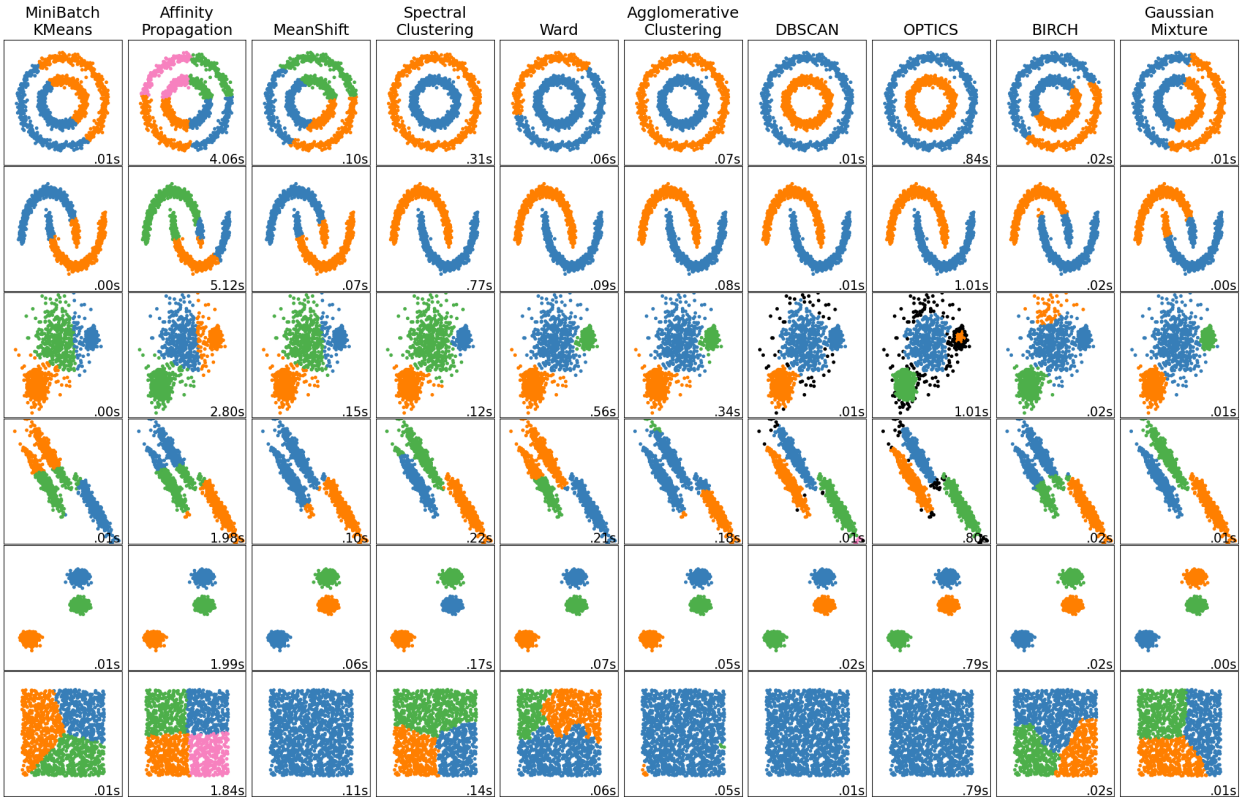


Fig. 2.3 Clusterisation algorithms used on sample two-dimensional data sets to demonstrate resultant clusters.

As Figure 2.3 demonstrates, there are multiple different interpretations of the solution to the problem of clustering. The makers of the Scikit learn package also discuss the scalability of each algorithm [Sci]. In order to prepare a high number of features (beyond the two used within this section for demonstration) and large number of data points, it is required that the algorithm scales accordingly. Further, for an adaptive process, it is more suitable for an algorithm to be adaptive to differing distribution. This limits the suitable algorithms to K-Means, Ward and Birch - columns one, five, and nine of Figure 2.3 respectively. Results for Birch can be seen in Figure 2.4. This appears do well, although it must be noted that this is likely due to the similarity between Monte Carlo (random) sampling, and clusterisation. I.e. areas distant from previously gathered points face a high chance of sampling.

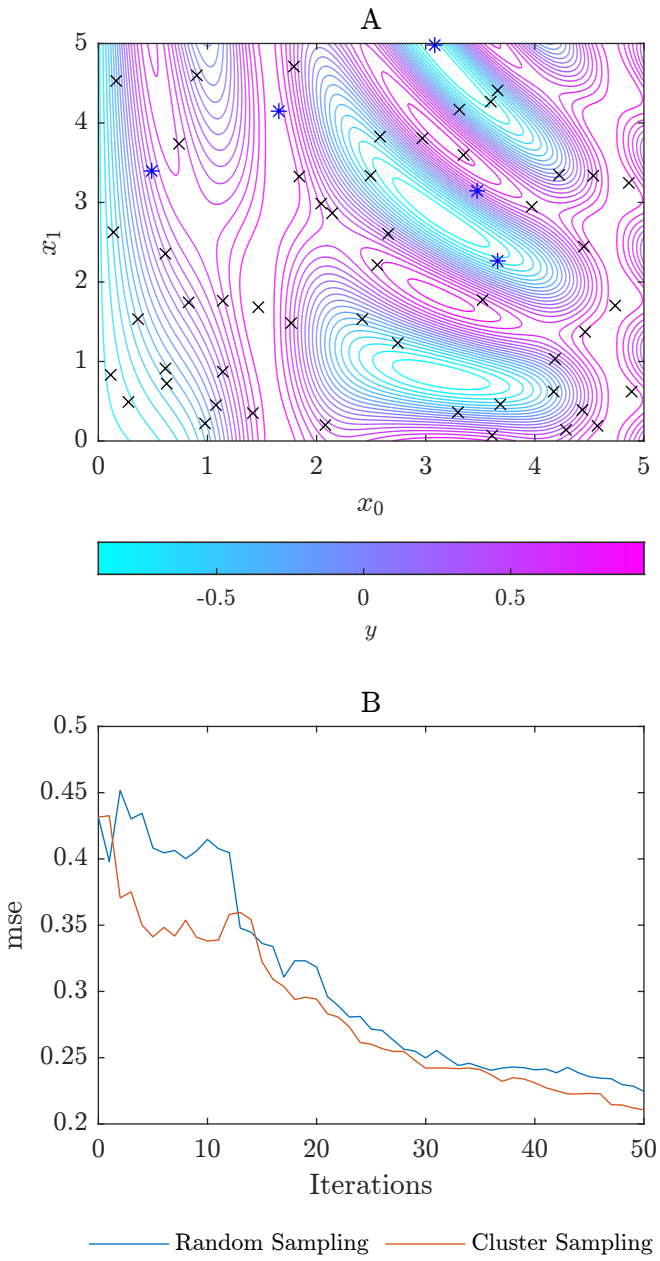


Fig. 2.4 The outcome of the investigating the areas of using a cluster hotspot sampling methodology. An initial set of 5 random points (displayed as blue stars in A)) was provided, and 50 further iterations were then carried out of sample size 1. A) demonstrates the final set of points tested by the algorithm and B) shows the mean squared error for the algorithm after each iteration.

2.1.2 Estimated Future

These methods attempt to minimise a future attribute of the model. This works by predicting changes given with the inclusion of more data with a higher degree f theoretical underpinning that the sampling methods discussed thus far.

Expected Model Change

As the name implies, this method chooses points which are likely to have the largest impact on the final model. By instigating each potential point, the impact on the eventual model can be found. However, this requires a method for quantifying the model change.

Settles and Craven [SC08] and Settles [Set09] investigate models which can be trained "online": i.e. models which can use the previous iteration to reduce the time taken for convergence. They present a method called "Expected Gradient Length" (EGL) which has a couple of prerequisites: **1)** A probabilistic model is used **2)** Linear gradient based optimisation is used **3)** The model can be improved from previous iterations. Given these prerequisites, the problem becomes less computationally inexpensive given a small dataset or extensive parallelisation, and scales as $\mathcal{O}(n)$. However, it does have the distinct drawback of requiring close control of the data models used. Here, the length of the training gradient (the gradient used in re-fitting the parameters with gradient based optimisation) can be used as a measure of model change. In the case of a small model change, as is expected, the length of the training gradient can be written as $\|\nabla l(\langle x, y_i \rangle; \theta)\|$. Combining this with the probability distribution of y, the next sample to undergo labelling is given by (2.6).

$$x^*_{EGL} = \underset{x}{\operatorname{argmax}} \sum_i P(y_i|x;\theta) \, \|\nabla l(\langle x, y_i \rangle; \theta)\| \tag{2.6}$$

2.2 Batch Active Learning

Little literature exists with respect to batch active learning. Naïve implementation exist whereby the methods explored earlier present a stack of data points to be chosen, and the top N are used. However, this method does not take into account the equivalence of the data points. This can be seen by the formation of clusters within the uncertainty sampling method in Figure 2.5, although it is not present within the clusterisation algorithm shown in Figure 2.6.

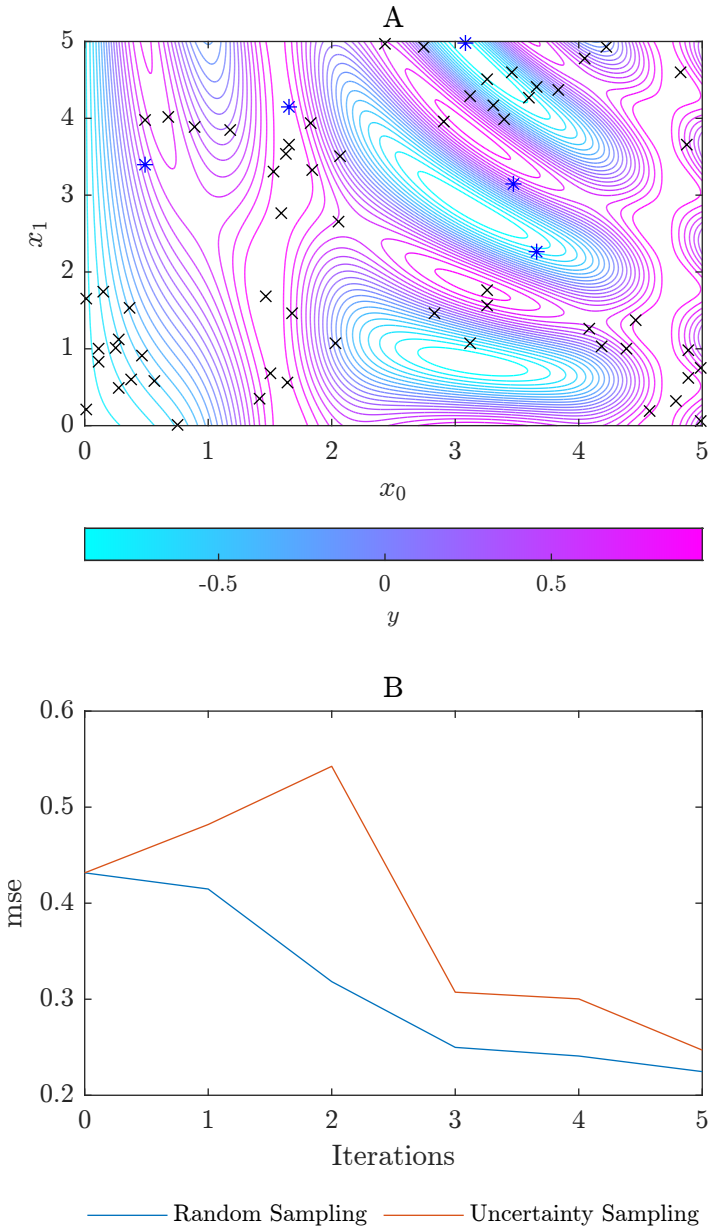


Fig. 2.5 The outcome of the investigating the areas of using uncertainty sampling. An initial set of 5 random points (displayed as blue stars in A)) was provided, and 5 further iterations were then carried out of sample size 10. A) Demonstrates the final set of points tested by the algorithm and B) shows the mean squared error for the algorithm after each iteration.

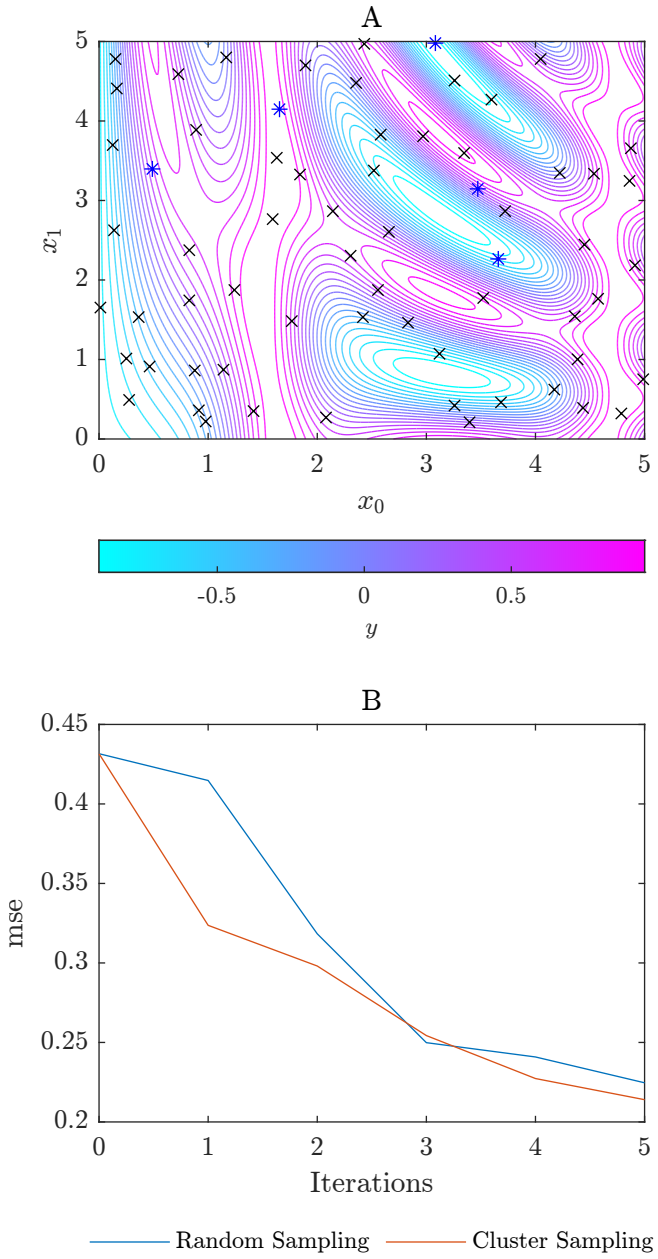


Fig. 2.6 The outcome of the investigating the areas of using cluster sampling. An initial set of 5 random points (displayed as blue stars in A)) was provided, and 5 further iterations were then carried out of sample size 10. A) Demonstrates the final set of points tested by the algorithm and B) shows the mean squared error for the algorithm after each iteration.

It is to be expected that regions of high uncertainty will exist. Within these regions, multiple points will have similar uncertainties. Thus, it is likely in batch querying areas of high uncertainty multiple points from the same area of feature space will be chosen, reducing the information gain by the model. Perhaps the consequences will be even worse due to overfitting. A simple fix would be to simulate the model after 1 iteration, and select the next point from there. By doing this $n-1$ times, a better solution may be found, although this may prove to be computationally expensive.

2.2.1 BatchBALD

Kirsch, Amersfoort, and Gal [KAG19] propose an algorithm named BatchBALD (Batch Bayesian Active Learning by Disagreement). Demonstrating for sample size of 10 an improvement compared to random sampling in optical character recognition. In doing so, they use complex Bayesian Neural Networks and apply the problem to categorical data. They also note it does not work well in the event of noisy data: a common feature of laboratory based labelling.

2.3 Drug Data for Machine Learning

There are numerous data categories that can be used to represent a chemical in a suitable form for machine learning. Indeed, the field of chemoinformatics is dedicated to the pursuit of describing chemicals for computational models. Each of these methods have various strengths and weaknesses. Some are directly based upon the chemical structure whereas others are based upon physical properties. These can be combined to produce models with high predictive capabilities.

2.3.1 Physical Properties

A selection of physical properties from chemicals are known, from melting points to solubility. Many of these provide important aspects for consideration and allow human scientists to predict interactions, especially when determining new drugs. These data are often reported in tables within textbooks such as Perry's Chemical Engineering Handbook or provided through software [EMB09; GS18].

Several of these data can be predicted through theoretical models, although the difficulty increases for larger molecules. For example, models exist for density predictions, but predicting the LD_{50} of a drug is far more challenging task. Indeed, even with animal testing, this property is deemed difficult to truly assess.

Within drug discovery, physical and biological properties are usually the sought after labels. An example of this is supplied by EMBL-EBI [EMB09] with a custom property named pChEMBL, as defined by (2.7) where "|" is synonymous with "or". This provides a measure for a drug against a specific molecular structure, such as proteins. Values are expected to be between 2 and 12, although these are not limited. Indeed, values even be negative, although this is rare and unlikely. The larger the value, the greater the chemical interaction.

$$\text{pChEMBL} = -\log_{10}\left(\text{IC}_{50}|\text{XC}_{50}|\text{EC}_{50}|\text{AC}_{50}|\text{Ki}|\text{LD}_{50}|\text{Potency}\right) \tag{2.7}$$

2.3.2 Fingerprints

Another methodology is to develop a fingerprint: a unique code based on the chemical structure, either of the atomic arrangement, or by the electron cloud distribution. The latter of these is more fundamental to the activity of molecules but far harder to calculate. Indeed, for accurate representation

of the latter, both atomic structure is needed *and* solutions for the Schrödinger equations corresponding to molecule in question.

According to Capecchi, Probst, and Reymond [CPR20], the most popular fingerprint in use are Morgan Fingerprints, a form of Extended Chemical Fingerprint (ECFP). ECFPs use a simple algorithm in order to generate a unique identifier, as described by Rogers and Hahn [RH10]:

1. **Initial Assignment**: Each atom has an integer assigned as an identifier.
2. **Iterative Updating**: Updating the identifier assigned to atoms based on adjacent atoms and structural duplications.
3. **Duplicate Removal**: Duplicate features are removed for hashing.

The iteration process involves each atom and adjacent atoms sharing numbers before in an array. A hash function is applied to this array and becomes the atoms new identifier. Fingerprints of this class are labelled according to the number of iterations, n, with the final name given as ECFP_$\langle 2n \rangle$. Morgan fingerprints, the most common form, are thus also called ECFP_4 [CPR20; RH10]. Thus, these come under the remit of fingerprints based upon two-dimensional chemical structure, rather than three-dimensional or even electron distribution. Morgan fingerprints are readily available for millions of compounds from the publicly accessible ChEMBL database [EMB09].

Chapter 3

Methodology

3.1 Data

Each dataset used consists of a 1024-bit Morgan fingerprint for the features and these associated pChEMBL values. The sets used for parameter fitting and score reporting make up a set of 2094 files from EMBL-EBI [EMB09] and compiled by Olier et al. [Oli+18]. These were filtered to prevent datasets with fewer than 1000 entries to be admitted into the main script. Columns were added with the scoring limits, as will be discussed later within the chapter. Consideration was given to reducing larger datasets to 1000 data points, although this notion was disregarded as the data was seen as too valuable to ignore. The data sets used within the scripts is given at https://github.com/rjb255/researchProject/tree/master/data/big/qsar_with_lims.

Morgan fingerprints were chosen due to the ease in which it is to calculate the vectors, the popularity of them within the chemoinformatics sphere, and the success enjoyed by others when using them for predictive purposes. It was decided that physical properties would not be used as this could increase the onus on data sanitation and preparation rather than active learning, although it is unavoidable using physical data for the labels. Here, pChEMBL, as defined in (2.7), is used due to comparability, easy interface with EMBL-EBI [EMB09], and perceived informativeness.

3.2 Computational Methodology

The methodology presents a novel means of assessing different parametrised batch active learning methods on existing data sets, allowing for a robust answer into the use of active learning in drug rediscovery. Results can thus be given with a given belief. This approach has taken principles commonly used in machine learning and applied it to more traditional algorithmic methods. Python was used as the scripting language, with the source code provided at https://github.com/rjb255/researchProject/tree/master/purePython.

Firstly, a collection of pre-existing data sets, X, are used. X is then split into two sub sets: X_{train} and X_{test}. Similarly to classical machine learning methods, the former of these subsets is used in fitting the parameters of the equation, and the latter is used to provide a result without the risk of data leakage

into the training set. Parallelisation is used to efficiently train the algorithms, allowing the time for training to be $\sim \mathcal{O}(c)$ provided an unrestricted number of processors. Datasets used have at least 1000 entries resulting in 164 datasets used for training, and a further 42 used for testing.

Examining the smaller details, each algorithm is provided with the sets x_{known}, y_{known}, and x_{unknown}. Various algorithms are given these sets and allowed to generate a subset of x_{unknown} to be added into x_{known} alongside corresponding y_{known}. This can then repeat until a predefined stopping point is reached. Scores are reported using a weighted mean squared error [] based upon y_{predict} for all x. This is similar to a standard machine learning methodology with a couple of differences. Firstly, no distinction is made between the training and testing set within a dataset contrary to standard practice. This is due to two reasons. Firstly, the datasets are not large enough for an accurate representation of the data within the testing set, and secondly, the scoring to each dataset is not used within the machine learning algorithms to fit parameters as is usually the case. All algorithms used rely upon a simple custom composite model to allow for flexibility and consistency.

3.2.1 Model

The machine learning model is the only custom class used. Here, a similar structure is used when compared with Scikit's machine learning [Ped+], as is demonstrated in Table 3.1. To manage this, it has four methods: __init__ , fit , predict , and predict_error . The last of these is not seen in all Scikit's machine learning models and is usually reserved for those which can report a certainty of prediction. Here, this was achieved by taking a standard deviation of the models.

	Name	Description
Attributes	Models: List	List of models to be used in composite
Methods	fit(X: int[][], Y: double[])	Fits the models in Models
	predict(X: int[][]): double[]	Takes a set of labels and returns mean predicted label from all the models.
	predict_error(X: int[][]): double[][]	Takes a set of labels and returns the mean predicted label from all the models and standard deviations of model predictions.

Table 3.1 Schema for the Model Class.

The models used for the composite model were Bayesian-ridge, k-nearest neighbours, random forest regressor, stochastic gradient descent regressor with Huber-loss, epsilon-support vector regression, and AdaBoost regressor [Ped+]. This was kept consistent during testing, allowing for direct comparison of the algorithms without influence from model selection.

3.2.2 Scoring

This method implements a weighted mean squared error (wmse) given in (3.1) where w is a normalization of the true label such that $\sum w_i = 1$ and $0 \leq w_i \leq 1$. Further modification to this ensures the base case with five data points provides a score $= 1$ and the score if the entire dataset is modelled provides a score $= 0$.

$$\text{wmse} = \frac{1}{n}\sum_{i=0}^{n-1} w_i(y_i - \bar{y})^2 \tag{3.1}$$

This achieves several goals. Firstly, it targets the higher values of pChEMBL, as these are the most beneficial for drug development. Secondly, it reduces the natural spread in results for datasets, preventing those poorly capable of being predicted the model from displacing results from the algorithm. Finally, it allows the results to be given as a fractional improvement instead. It allows a target of "85%" prediction to be given for stopping criteria if desired.

It is also useful having a learning rate metric, as defined in (3.2), where N is the total number of samples sampled. This provides a measure of the rate of learning.

$$\text{score} = -\frac{\Delta\text{score}}{\Delta N} \times 10^4 \tag{3.2}$$

3.2.3 Active Learning Algorithms

The algorithms tested are all provided with x_{known}, x_{unknown}, y_{known}, a model fitted to x_{known} and y_{known}, and a memory object to allow for information to be kept through iterations if required. This is useful for clustering, where online training is possible. It is within the memory object where parameters may also be provided. As a result, it is impossible for the suppressed y_{Yknown} to influence an algorithms scoring process. The algorithms then return a list in the same order as y_{unknown}, with the lowest scores designating higher priority in sampling. This allows uniformity across algorithms and the amalgamation of different algorithms without the duplication of code.

Monte Carlo

The Monte Carlo algorithm employs random sampling. This represents the least computationally expensive approach, and is thus used as a baseline in comparing other algorithms. Since the datasets are shuffled prior to being used, the algorithm is extremely simple, as demonstrated in Algorithm 1.

Algorithm 1: Monte Carlo Sampling

Data: X_{unknown}

Result: An array of priority-scores for sampling

return ones_like(X_{unknown})

Greed

Since the largest activity is sought, a methodology proposed is to simply seek the predicted highest label. Here, the predict() method (see Table 3.1) was used to return a prediction and a standard deviation. The indices of x_{unknown} were then returned, ordered descending with respect to the afore mentioned standard deviations. The algorithm used is given in Algorithm 2.

Algorithm 2: Greed Sampling Selection

Data: X_{known}, Y_{known}, X_{unknown}, Model
Result: An array of priority-scores for sampling
Model.fit(X_{known}, Y_{known});
prediction = Model.predict_error(X_{unknown});
return $-$prediction

Region of Disagreement

Region of **D**isagreement (ROD) uses the predict_error() method (see Table 3.1) to return a prediction and a standard deviation. The prediction is ignored. The negative of the standard deviation is returned to ensure the largest uncertainty has the lowest "score". This is shown with Algorithm 3.

Algorithm 3: RoD Sampling Selection

Data: X_{known}, Y_{known}, X_{unknown}, Model
Result: X ordered according to priority for sampling
Model.fit(X_{known}, Y_{known});
_, error = Model.predict_error(X_{unknown});
return $-$error

Hotspot Clusters

Three clustering algorithms were trialled, all based upon the ideology presented in Section 2.1.1. The function shared by all three algorithms is shown in Algorithm 4. Here, c is the number of clusters sought, and is a parameter that requires fitting. Bounds can be placed upon this. The lower limit can be set as the number of known data points, and the upper as the total number of data points in the data set, although it is hypothesised that beyond the sum of the known points and the samples sought would make little, to no difference. To test this hypothesis, the upper limit will be set at $\text{length}(X_{\text{unknown}}) + 1.5n$. The combined limits have been shown in (3.3). It is important to note the algorithm used for clustering: K-Means with a huber loss function. This follows recommendations from Scikit Learn [Sci] for scalability.

$$\text{length}(X_{\text{known}}) < c < \text{length}(X_{\text{unknown}}) + 1.5n \tag{3.3}$$

Algorithm 4: Uncertainty Sampling Selection

Data: z_{known}, z_{unknown}, c

Result: Score of data points

combined_z = concat(z_{known}, z_{unknown});

clusters = cluster(number_of_clusters=c);

clusters.fit(combined_z);

predicted_custers = clusters.predict(z_{unknown});

distances = clusters.distance_to_nearest_centroid(z_{unknown});

indices = z_{known}.index;

sorted_indices = sort(indices -> By cluster size followed by distance to centroid) ;

high_priority, low_priority = split(sorted_indices, if cluster contains z_{known});

high_priority.riffle();

low_priority.riffle();

order = join(high_priority, low_priority);

return $-$error

Several key steps are involved within the algorithms to fit to the ideology. Firstly, clusters containing samples from x_{known} are given lower priority. These are perceived as known clusters so ideally would not undergo further testing. Secondly, the sorting needs to be addressed. Here, the sample is sorted into the relevant cluster groups. These groups are then ordered by size, with larger cluster favoured. Samples within the cluster are sorted by distance to the equivalent centroid. The clusters are then split into those containing sampled points and those that do not. With each of these groups, a riffling procedure is used. Named after the common card shuffling technique, this ensures the priority is given to different clusters, with the highest priority going to the point from the most populated cluster, and closest to the centroid. The two groups of clusters are then concatenated.

The three versions of clusterisation differ by the z provided. In Cluster I, $z \equiv x$, whereas in Cluster II, y_{known} and y_{unknown} is joined to x_{known} and x_{unknown} respectively. Cluster III takes this a step further by combining $s_{g_{\text{unknown}}}$ into z_{unknown} with 0 being the equivalent value used for z_{known}.

Region of Disagreement with Greed

The first composite algorithm explored is **R**egion **o**f **D**isagreement with **G**reed (RoDG), combining both the greedy sampling, and the uncertainty sampling algorithms. This metric is shown in (3.4).

$$\text{score}_{\text{RoG}} = \text{score}_{\text{Greed}}^{\alpha}\text{score}_{\text{RoD}}^{1-\alpha} \tag{3.4}$$

Here, α is a parameter which needs to be found, bounded as $0 < \alpha < 1$. Note that at the limits, the algorithm reduces to the RoD and Greed algorithms.

Region of Disagreement with Greed and Clustering

Region **o**f **D**isagreement with **G**reed and Clust**er**ing (RoDGer) is a second order composite function, involving RoD with Greed and Cluster III, as shown in (3.5).

$$\text{score}_{\text{RoDGer}} = \text{score}^{\alpha}_{\text{ClusterIII}} \text{score}^{1-\alpha}_{\text{RoDG}} \tag{3.5}$$

Both of the constituent algorithms are parameterised, implying a total of three parameters. Bounds on initial estimates will be provided by the results of these algorithms taken individually.

3.2.4 Parallelisation

The large number of datasets used presents a problem: time. Indeed, each iteration sees a new fitting of a machine learning model. Within the training stage, this would correspond to a minimum of 1000 models trained: a considerable number. Thus, by exploiting parallelisation, the time can be reduced in execution to the case, where given an infinite number of processes, the training and testing framework would scale as $\mathcal{O}(c)$. This requires circumventing pythons global interpreter lock, accomplished using Pathos due to several shortcomings found with the default multiprocessing package [McK+12; MA10].

3.2.5 Minimisation

Due to the available parallelisation, only one iteration was performed in minimisation. This approach consisted of generating a uniform distribution of test parameters, testing upon the datasets in one parallelised step, and selecting the best performing parameter combination.

Chapter 4

Results

Results are presented for the algorithms discussed in Chapter 3. Where possible, errors have been provided by taking the sample standard deviation of the results provided and dividing by $\sqrt{N-1}$. This allows for robust discussion and comparison of each method used. Within figures, lines are added to guide the eye to changes.

4.1 Non-Parametric

Non-parametric equations have the benefit of not requiring the minimisation function. Due to this, all testing of these algorithms were undertaken on a standard laptop. These also tend to be the easiest to implement, as uncovered in Chapter 3. Particularly important is the Monte Carlo method as this allows shows what should be a minimum baseline to achieve.

4.1.1 Monte Carlo

The first non-parametrised algorithm discussed in Chapter 3 was the Monte Carlo method. Due to the non-parametric nature of this algorithm, execution was simply carried out on the test data set. Results are presented in Figure 4.1, demonstrating a final score of 0.184 ± 0.018 and an overall score of 16.32 ± 0.36.

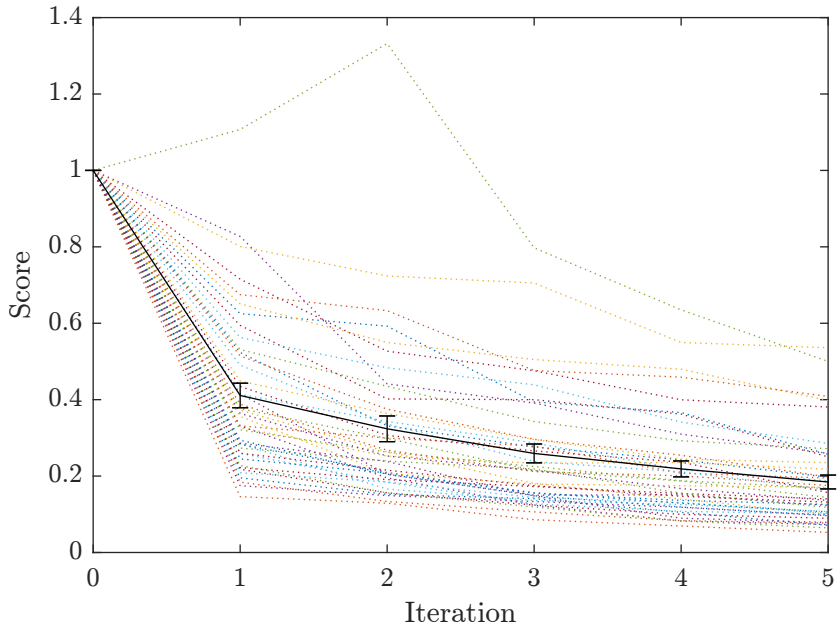


Fig. 4.1 Results of Monte Carlo sampling on the test datasets. Dotted lines represent the individual scoring for the datasets and the solid line shows the mean results at each iteration in order to guide the eye.

4.1.2 Greed

Likewise, the Greedy algorithm was tested, with results presented in Figure 4.2. Here, a final score of 0.323 ± 0.039 and an overall scòre of 13.5 ± 0.8, indicating a worse performance than the base case.

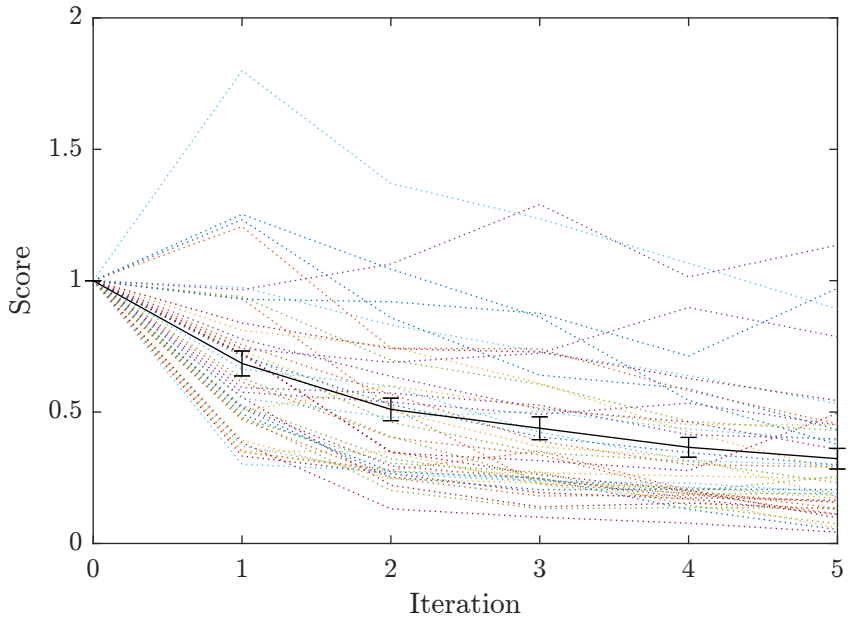


Fig. 4.2 Results of greedy sampling on the test datasets. Dotted lines represent the individual scoring for the data sets and the solid line shows the mean results at each iteration in order to guide the eye.

4.1.3 RoD

The final non-parametric algorithm to be tested was RoD. A final score of 0.211 ± 0.022 and an overall score of 15.78 ± 0.44, leading to a middling position between the other two parametric algorithms. The improvement in each iteration is shown in Figure 4.3.

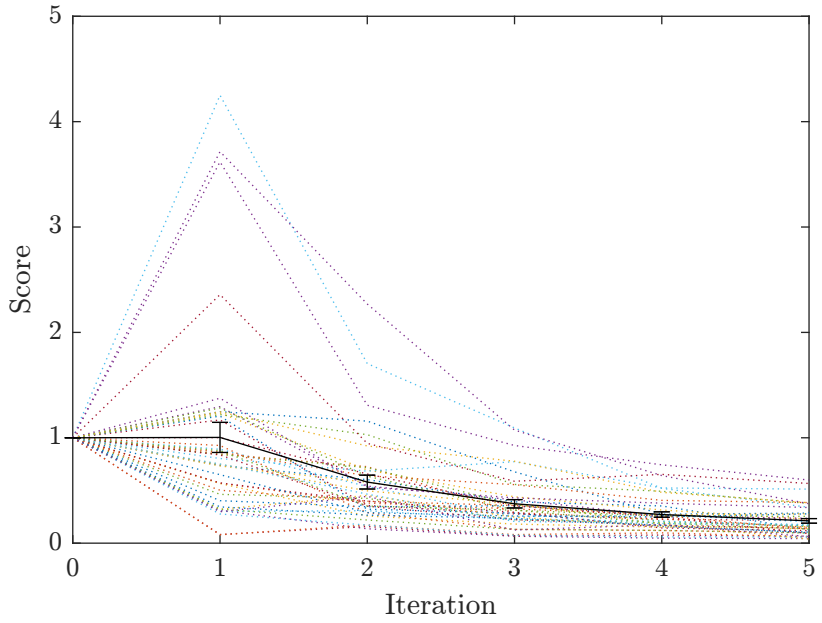


Fig. 4.3 Results of RoD sampling on the test datasets. Dotted lines represent the individual scoring for the datasets and the solid line shows the mean results at each iteration in order to guide the eye.

4.2 Parametric

Parametric algorithms require a minimisation procedure on the training set. This is computationally challenging, and for this the author is grateful for the services provided by the HPC [Uni22].

4.2.1 Clusters

The first set of algorithms tested were the clusters. Each of these outperformed all three of the other algorithms, with Cluster I, Cluster II, and Cluster III giving scores of 0.155 ± 0.020, 0.145 ± 0.009, and 0.143 ± 0.016 respectively. This corresponds to score of 16.90 ± 0.40, 17.10 ± 0.18, and 17.14 ± 0.32. Due to the results from Cluster III, this is the one that will be used within the Holy Trinity. A variety of optimal c were found when comparing to Algorithm 4. An additional cluster size of 45, 40, and 60 were found to be optimal for Cluster I, II, and III respectively.

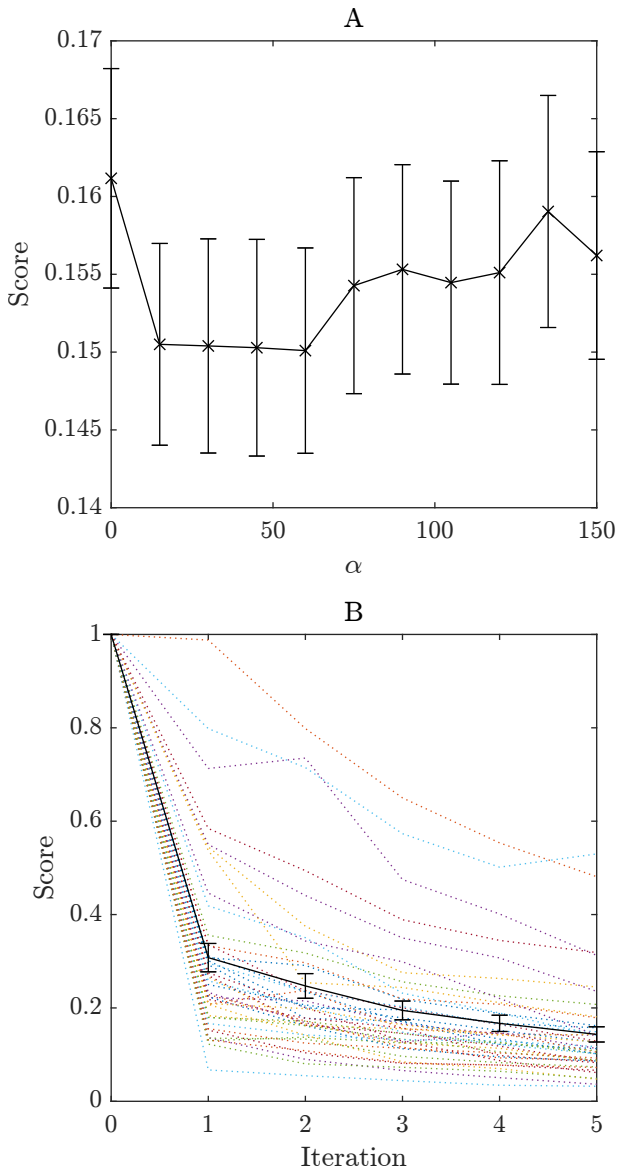


Fig. 4.4 The results of fitting c to the Cluster III algorithm. A) shows the results from parameter fitting and B) shows the learning process on the test set. Dotted lines show performance of individual datasets and solid lines guide the eye to the trend.

4.2.2 RoDG

When testing the RoDG sampling algorithm, it was found that despite the weighting towards higher value targets in scoring, no significant improvement was seen over RoD. The minimisation procedure returned $\alpha = 0.06$, with α defined in (3.4). However, the tolerance at small α, as shown in Figure 4.5A. This method gave a final a score of 0.206 ± 0.011 and overall score $= 15.88 \pm 0.22$, a slight improvement over simply using RoD.

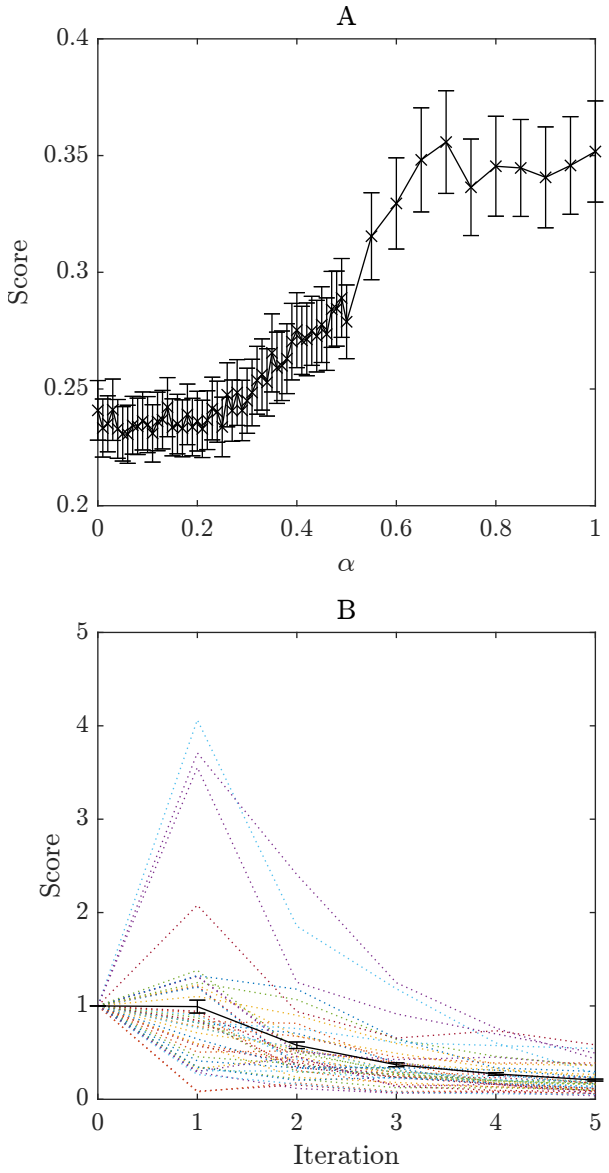


Fig. 4.5 The results of fitting α to the RoDG algorithm. A) shows the results from parameter fitting and B) shows the learning process on the test set. Dotted lines show performance of individual datasets and solid lines guide the eye to the trend.

4.2.3 RoDGer

By sampling multiple values for α, a final set of $\alpha = [60, 0.47, 0.22]$ was reached where $[\alpha_0, \alpha_1, \alpha_2]$ correspond to the constants used in Algorithm 4, (3.4), and (3.5) respectively. When validated against the testing datasets, a final score of 0.115 ± 0.013 was found with an overall score $= 17.70 \pm 0.26$; the best result.

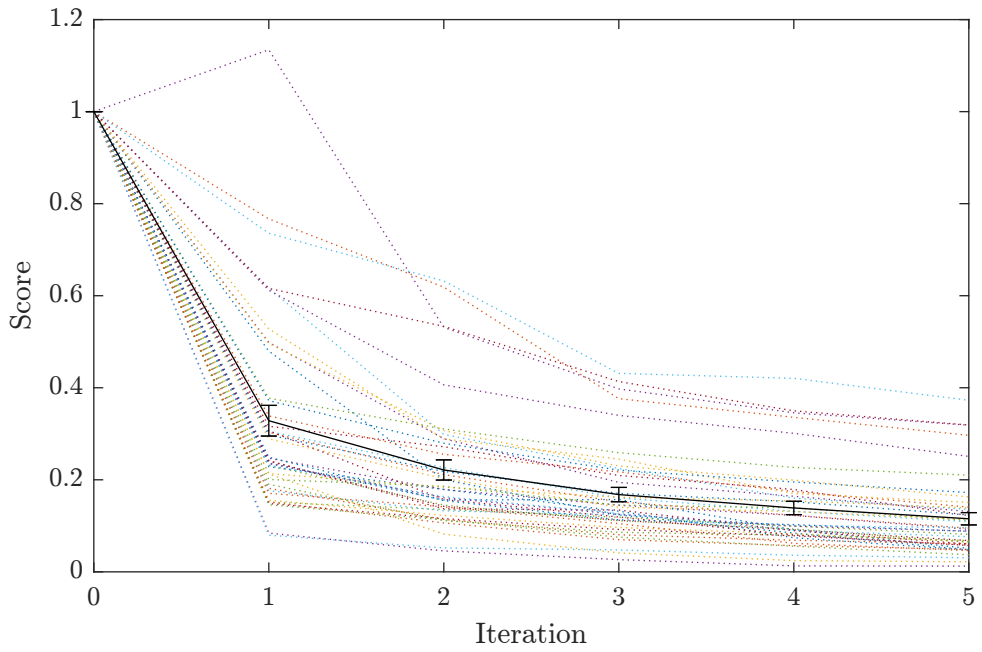


Fig. 4.6 The results after fitting α to RoDGer, showing the learning process on the test set. The learning of each dataset has been added with dotted lines for illustrative purposes, and solid lines guide the eye to overall trends.

Chapter 5

Discussion

5.1 Non-Parametric

The three non-parametric algorithms produced several noticeable results. Firstly, Monte Carlo sampling outperformed both Greedy and RoD sampling. This is demonstrated convincingly through Figure 5.1, where results from the greedy results suggest the worst accuracy.

Despite the greedy algorithm demonstrating the worst accuracy, interesting results were shown with RoD sampling. Poor selection is evidently present with the first sample set, although rapid improvement quickly follows. Indeed, after the first iteration, the learning rate is superior to the other two algorithms, boasting a score $= 41$. An extra iteration may indeed have seen RoD surpassing Monte Carlo. This behaviour is expected as the RoD algorithm specifically attempts to target regions of the model which cause the largest changes towards optimal fitting.

Both RoD and greedy sampling are suspected to suffer from clusterisation whereby data points similar to each other in the feature space are sampled within the same batch, thus reducing the total information conveyed per batch operation. This is believed to be particularly costly with the first iteration as the model will heavily overfit to the new cluster it has sampled. The random nature of Monte Carlo reduces this prospect, hence the apparent promising performance of a random sampling methodology. Evidence to this is shown in Figure 4.1, Figure 4.2, and Figure 4.3.

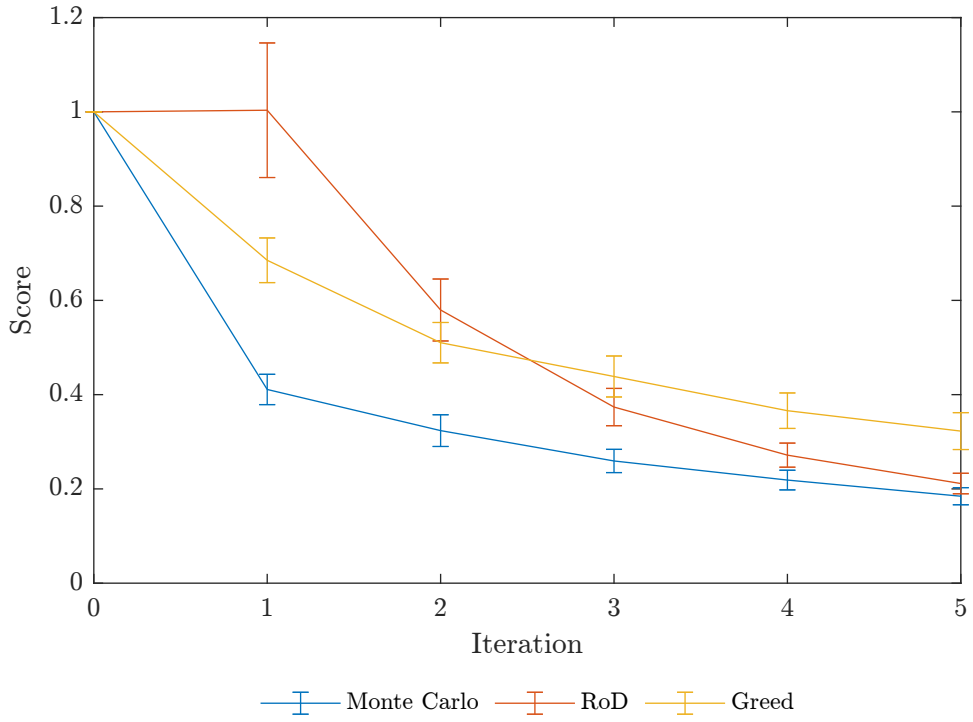


Fig. 5.1 Comparison of different non-parametric algorithms with interpolation lines to guide the eye.

This demonstrates a danger with Batch Active Learning. It is very easy to produce a learning algorithm which actually performs worse than random screening.

5.2 Parametric

The first of these is a first order composite algorithm, RoDG, which uses different active learning algorithms as a base. The second is a clustering algorithm with the number of clusters left as a parameter. The third is a second order composite active learning algorithm which combines the other two parametric functions, named RoDGer. An improvement is seen throughout these algorithms, with RoDGer performing the best.

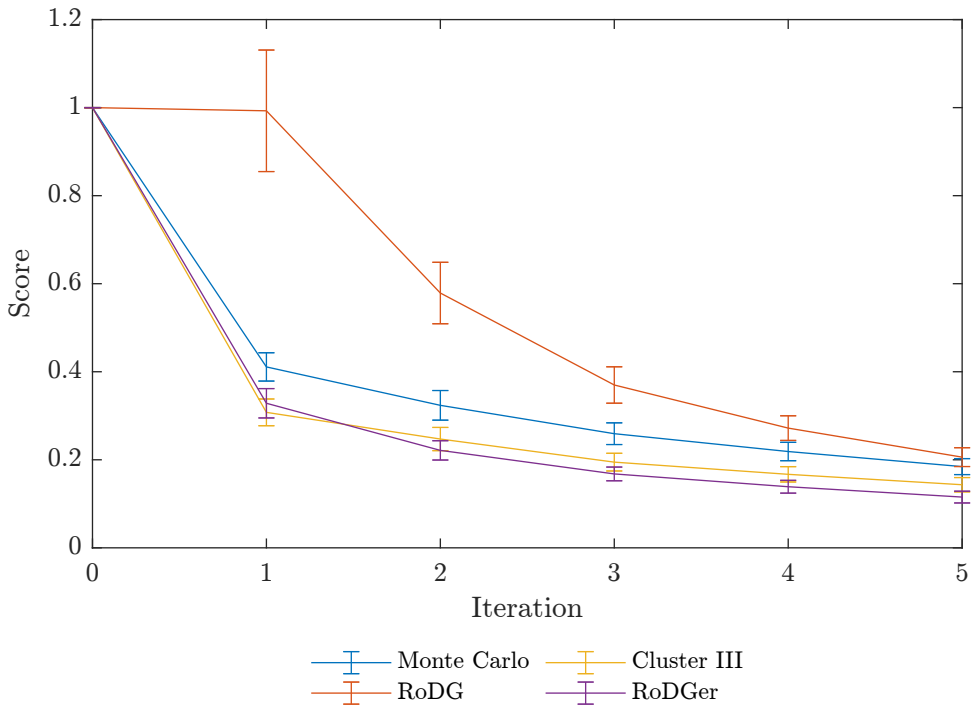


Fig. 5.2 Comparison of different parametric algorithms with interpolation lines to guide the eye.

Several points of interest are highlighted with these results. Firstly, despite improving upon RoD, RoDG is still beaten by Monte Carlo sampling. Again, the methodology suffers from clustering of points.

The sampling process of the Cluster III algorithm demonstrates its ability at outperforming the random sampling methodology of Monte Carlo even within the first iteration. The RoDGer algorithm demonstrates a long term strategy, falling within error of Cluster III in the first iteration before steadily improving beyond the capabilities of Cluster III.

Upon investigation of the parameters settled upon within these several points arise. Firstly with RoD with Greed, at low α, the score appears uncorrelated with α, only experiencing a significant rise with $\alpha > 0.3$. Thus, it can be surmised that RoD is the driving force, with evidence given by the final scores for RoD and RoD with Greed algorithms arising within error of each other.

On the other hand, the sensitivity of Cluster III is low to cluster size, demonstrated by Figure 4.4. Here, no significant change is observed within the significant parameter range. It is believed this is due to the highest ranking points remaining within the top clusters as large clusters are likely to remain the largest, even with an increased number of clusters. Thus, the top candidates are likely to remain in the same region of the feature space.

Most surprising are the parameters found for the RoDGer, shown in Figure 5.3. Here, it appears the highest sensitivity is based on α_2, the exponent to the RoD with Greed score. This indeed shows a difference from Rod with Greed, where the Greed aspect appeared to be borderline irrelevant, with final results within error of RoD. Comparatively, α_0 and α_1 appear to have a linear fit.

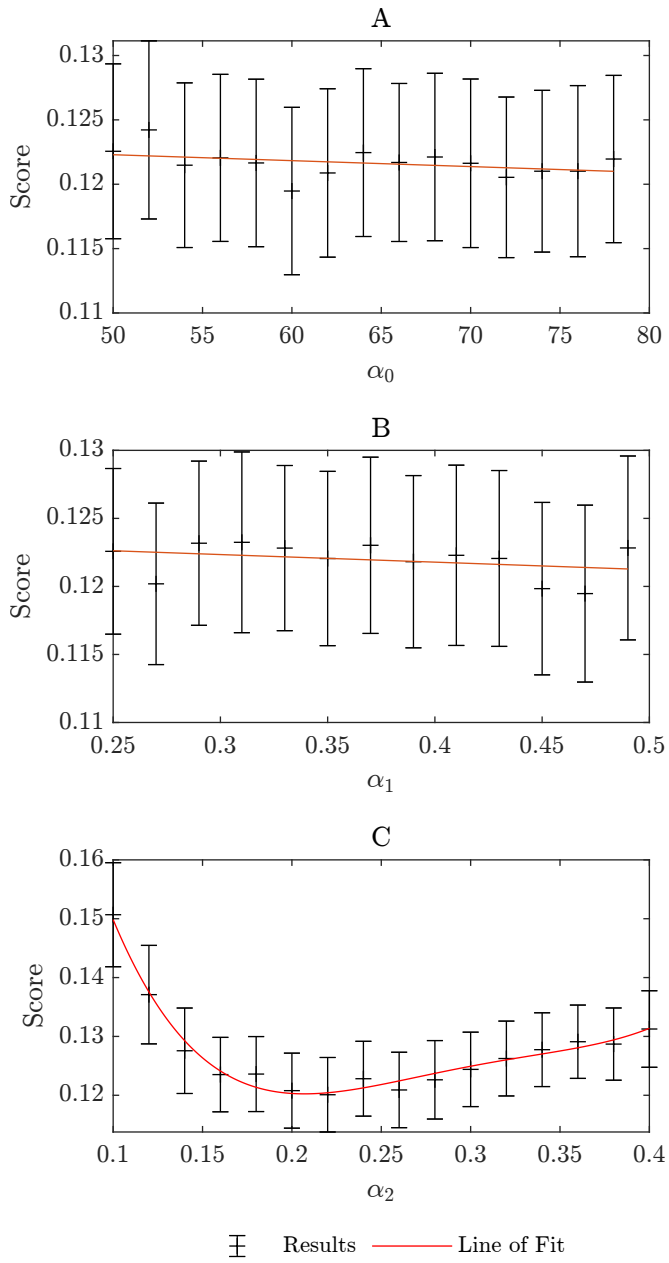


Fig. 5.3 Training results for determining parameters for the RoDGer algorithm. A) shows the deviation of α_0, B) shows the deviation of α_1, and C) shows the deviation of α_2. Curves have been added to demonstrate the trends observed.

Due to the empirical nature of the curves, linear interpolation has been used to guide the eye of the reader to the trend. Take Figure 4.3 where the first iteration produces no significant change. By fitting a curve, the results here could be misrepresented, so interpolation has been used in most cases to guide the eye. In Figure 5.3, it was considered essential to present the different in form between α_0, α_1, and α_2 so a curve has been added to guide the reader to the differences in form. Polynomial fits were used for each subfigure, with A and B using 1st order and C using 4th order.

5.3 Problem Sensitivity

The problem set up provides an estimation of a fixed number of iteration and sample size. What if these change? Are the parameters overfitted to this specific problem? If deviations from this are taken, are the intelligent methods worse than Monte Carlo? This is important to determine, as there is a significant potential cost to such an inefficiency.

Monte Carlo is tested compared to RoDGer and RoD, using the already fitted parameters where required. Monte Carlo is included to demonstrate the baseline case, while RoDGer is included due to producing the best results in previous testing. RoD has been included to assess if the learning rate maintains significantly higher than Monte Carlo in an extended testing range. Results have been presented in Figure 5.4. Convergence of RoD to Monte Carlo is observed, with RoDGer consistently outperforming the other two algorithms.

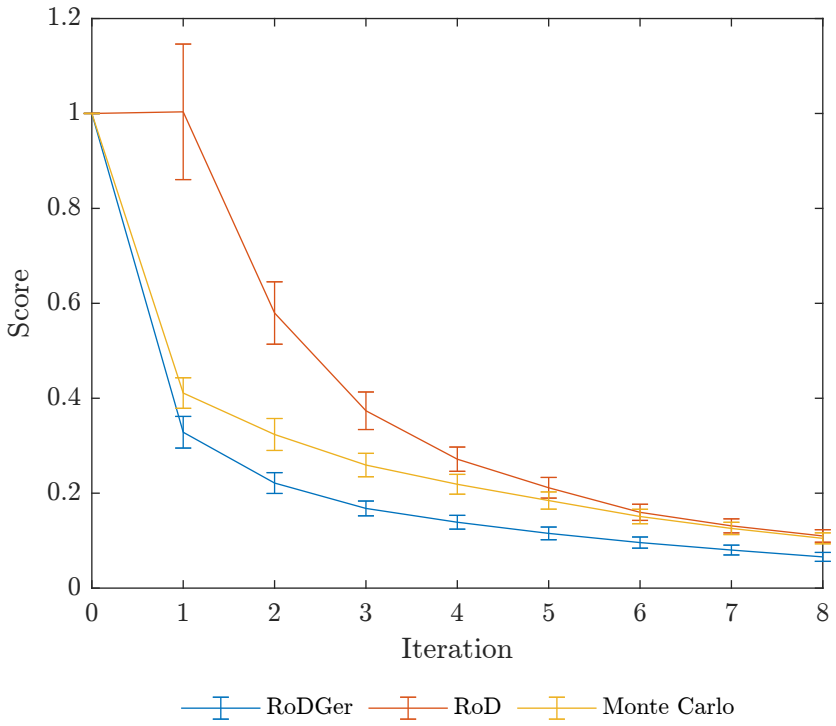


Fig. 5.4 Comparison of Monte Carlo, RoD, and RoDGer over 8 iterations.

Using sample sizes of 50 and 150 on Monte Carlo and RoDGer supports the power of RoDGer over simply using Monte Carlo.

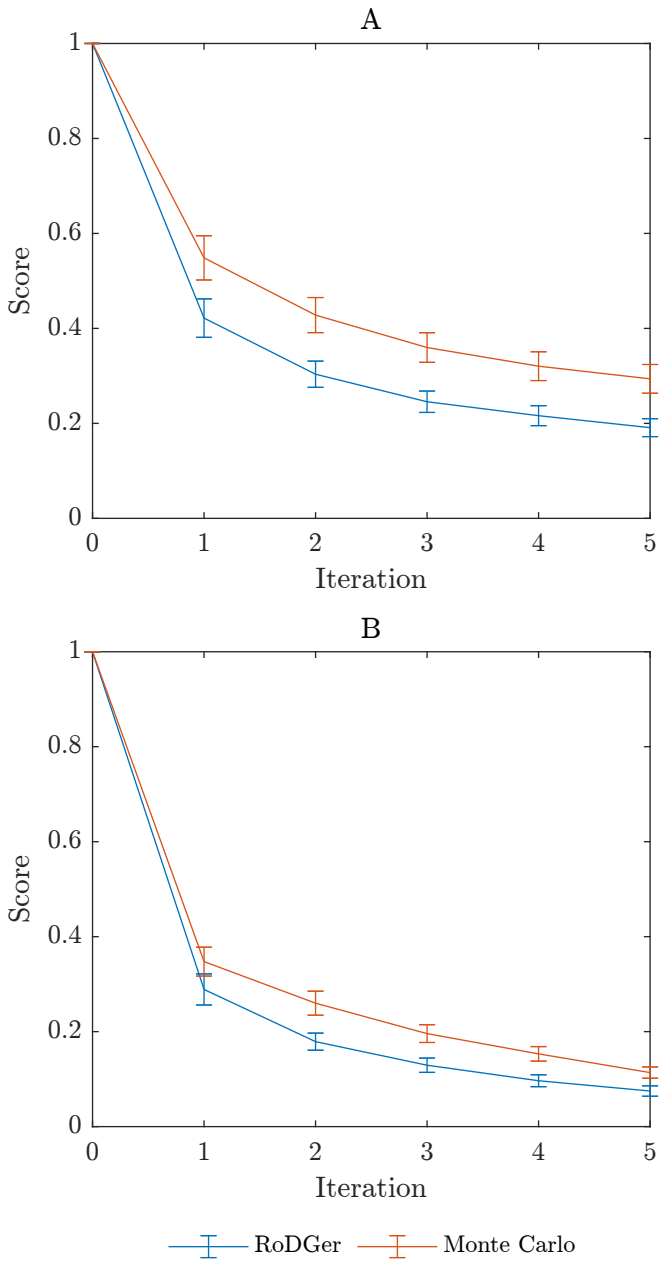


Fig. 5.5 Comparison of Monte Carlo and RoDGer with different sample size. A) shows batches of 50 and B) shows results for batches of 150.

Finally, an investigation into using a reduced set of models is undertaken, using Bayesian ridge, k-nearest neighbours, and a multi-layer perceptron regressor (MLPR). This last model is a neural network, chosen since it represents a different class of machine learning model. The other two were used within other trial in order to allow a sample standard deviation to be returned, as required by algorithms relying on the predict_error method of the Model class. Results are shown in Figure 5.6, demonstrating the robustness of RoDGer against model changes.

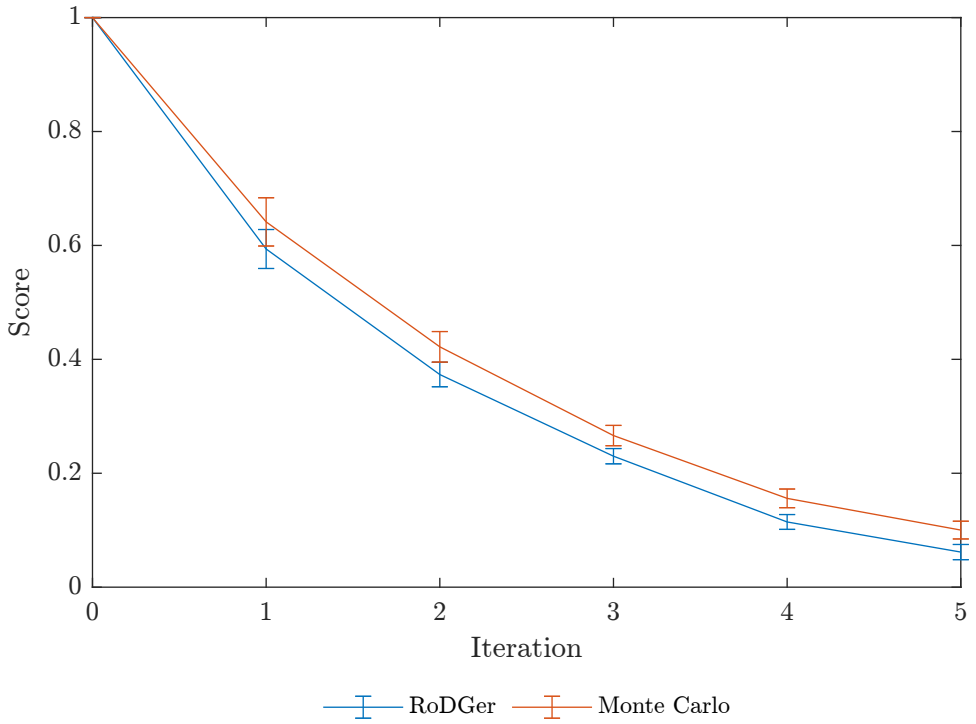


Fig. 5.6 Comparison of Monte Carlo and RoDGer using different machine learning models.

5.4 Refinement

There are several points for progression that are worth discussing. Firstly, the training system requires improvement. Despite the observations of parallelisation leading to a situation of $\mathcal{O}(c)$ given an infinite number of processing units, an infinite number of processing units unfortunately do not exist. In the event of three parameters, the current methodology requires $N_{\text{train}} \prod(n_{\alpha_i})$ executions of the learning algorithm, where n_{α_i} is the number of different values tested for α_i. Even with simply 3 different values tested for each parameter in the RoDGer algorithm, this equates to ~ 4500 executions of the algorithm. Thus, as the maximum number of CPUs available was 380, the training process became $\mathcal{O}\left(N_{\text{train}} \prod n_{\alpha_i}\right)$. A potential solution would be segmentation, where the datasets are split into smaller groups with which are tested on a sample of the parameter space, with results from these smaller groups amalgamated at the end. Additionally, active learning could be used in the determination of the minima. Here, the goal would be to find the minima using as few parameter values as possible.

The current implementation of the scoring system is simple. Although this would please Occam's barber, it is believed that this is not satisfactory for the aggressive pursuit of effective drugs. One possibility is only scoring samples which show a pChEMBL greater than seven. This puts a greater weight on these samples, although a multidisciplinary discussion is required here. This could potentially be aided by categorising the data set instead, instead separating results into pChEMBL > 7 and pChEMBL ≤ 7.

5.5 SARS-CoV-2

Given the need for current need for treatments against COVID-19, there is not a better and more suitable time to trial the algorithm within this context. Tsuji [Tsu20] suggest 3-chymotrypsin-like protease as a target for antiviral treatments. Several amendments to the current code base will be required to ensure successful operation within a robot scientist. Fortunately, the code base was written with this in mind, featuring several custom functions which may be rewritten to create an application interface (API).

Chapter 6

Conclusion

There are about 20000 drugs approved for pharmaceutical use, with more being trialled every year. However, there is hesitancy in using these drugs as treatments for emerging diseases. Instead, focus is given to designer drugs which may take years to develop. It is estimated 80% of the cost in drug design and manufacture arises simply from the developing process. This was shown to be ineffective at the time of the SARS-CoV-2 outbreak whereby the use of existing drugs was slow, and often polluted with misinformation.

In order to combat these issues, the role of active learning cannot be understated. Reviewing drugs already in commercial production allows a fast role out upon approval. Secondly, research and development costs will reduce significantly due to low wastage. Thirdly, clinical trials could safely be accelerated by providing phase one trial data from the first trials the drug will have had to pass.

Thus, this thesis targetted the issue of an algorithm capable of executing batch active learning with batch sizes of 100. In doing so, several algorithms were tested and compared consisting of both parametric and non-parametric algorithms.

References

Capecchi, Alice, Daniel Probst, and Jean-Louis Reymond (June 12, 2020). "One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome". In: *Journal of Cheminformatics* 12.1, p. 43. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00445-4. URL: https://doi.org/10.1186/s13321-020-00445-4 (visited on 05/06/2022).

Center for Drug Evaluation and Research (Apr. 25, 2022). "Coronavirus Treatment Acceleration Program (CTAP)". In: *FDA*. Publisher: FDA. URL: https://www.fda.gov/drugs/coronavirus-covid-19-drugs/coronavirus-treatment-acceleration-program-ctap (visited on 05/05/2022).

EMBL-EBI (2009). *ChEMBL Database*. URL: https://www.ebi.ac.uk/chembl/ (visited on 05/06/2022).

Green, Don W. and Marylee Z. Southard (2018). *Perry's Chemical Engineering Handbook*. 9th. McGraw-Hill.

Kirsch, Andreas, Joost van Amersfoort, and Yarin Gal (Oct. 28, 2019). "BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning". In: *arXiv:1906.08158 [cs, stat]*. arXiv: 1906.08158. URL: http://arxiv.org/abs/1906.08158 (visited on 05/12/2022).

McKerns, Michael and Michael Aivazis (2010). *pathos: a framework for heterogeneous computing*. URL: http://uqfoundation.github.io/project/pathos.

McKerns, Michael M. et al. (Feb. 6, 2012). "Building a Framework for Predictive Science". In: *arXiv:1202.1056 [cs]*. arXiv: 1202.1056. URL: http://arxiv.org/abs/1202.1056 (visited on 05/07/2022).

Olier, Ivan et al. (Jan. 1, 2018). "Meta-QSAR: a large-scale application of meta-learning to drug design and discovery". In: *Machine Learning* 107.1, pp. 285–311. ISSN: 1573-0565. DOI: 10.1007/s10994-017-5685-x. URL: https://doi.org/10.1007/s10994-017-5685-x (visited on 05/13/2022).

Pedregosa, Fabian et al. (n.d.). "Scikit-learn: Machine Learning in Python". In: *MACHINE LEARNING IN PYTHON* (), p. 6.

Rogers, David and Mathew Hahn (Feb. 4, 2010). "Extended-Connectivity Fingerprints | Journal of Chemical Information and Modeling". In: *Journal of Chemical Information and Modeling* 50.5. DOI: 10.1021/ci100050t. URL: https://pubs.acs.org/doi/10.1021/ci100050t (visited on 11/01/2021).

Scikit Learn (2022). *2.3. Clustering*. scikit-learn. URL: https://scikit-learn.org/stable/modules/clustering.html (visited on 05/05/2022).

Settles, Burr (2009). *Active Learning Literature Survey*. Technical Report. Accepted: 2012-03-15T17:23:56Z. University of Wisconsin-Madison Department of Computer Sciences. URL: https://minds.wisconsin.edu/handle/1793/60660 (visited on 11/01/2021).

Settles, Burr and Mark Craven (Oct. 25, 2008). "An analysis of active learning strategies for sequence labeling tasks". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. USA: Association for Computational Linguistics, pp. 1070–1079. (Visited on 05/01/2022).

Sparkes, Andrew et al. (Jan. 4, 2010). "Towards Robot Scientists for autonomous scientific discovery". In: *Automated Experimentation* 2.1, p. 1. ISSN: 1759-4499. DOI: 10.1186/1759-4499-2-1. URL: https://doi.org/10.1186/1759-4499-2-1 (visited on 05/09/2022).

Tsuji, Motonori (June 2020). "Potential anti-SARS-CoV-2 drug candidates identified through virtual screening of the ChEMBL database for compounds that target the main coronavirus protease". In: *FEBS open bio* 10.6, pp. 995–1004. ISSN: 2211-5463. DOI: 10.1002/2211-5463.12875.

University of Cambridge (2022). *Research Computing Services*. URL: https://www.hpc.cam.ac.uk/ (visited on 05/07/2022).

Wang, Haidong et al. (Apr. 16, 2022). "Estimating excess mortality due to the COVID-19 pandemic: a systematic analysis of COVID-19-related mortality, 2020–21". In: *The Lancet* 399.10334. Publisher:

Elsevier, pp. 1513–1536. ISSN: 0140-6736, 1474-547X. DOI: 10.1016/S0140-6736(21)02796-3. URL: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(21)02796-3/fulltext (visited on 05/06/2022).

World Health Organization (May 6, 2022). *WHO Coronavirus (COVID-19) Dashboard*. URL: https://covid19.who.int (visited on 05/06/2022).