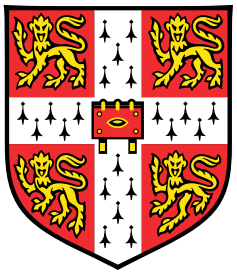
Repurposing Drugs for the Rapid Response to Epidemics and Pandemics

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

I would like to dedicate this thesis to the loss of sleep never to be recovered. Its sacrifice in making this project come to fruition will never be forgotten.

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 39 pages and 9000 words (excluding this page, the title page, and the safety appendix).

Ross Brown
May 2022

Acknowledgements

And I would like to acknowledge ...

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Table of contents

List of figures

List of tables

Nomenclature

Chapter 2

N	Number of features/dimensions of 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 (CTAP) [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, and the timescale in which they were achieved, is suboptimal. This is due to the slow, labourous, 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. This is a methodology suited to fields with large amounts of unlabelled data which is difficult 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 eistence of robotic testing fascilities. This presents an additional problem: how does one set up a testing scheme for batches? Can the same techniques used in single site 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.

Chapter 2

Previous Work

In order to assit 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, $\varepsilon \sim \mathcal{N}(0, 0.01)$.

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

In order to assess the algorithms, the mean squared error (mse) has been used. Comparisons are made to the naive approach of random sampling, i.e. Monte Carlo sampling. Each algorithm will be given five random starting points, and attempted improvement will follow.

2.1 Active Learning

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 t implement, and less adaptable to model changes, as will be apparent on description.

2.1.1 Current Data

Uncertainty Sampling and Regions of Disagreements

The simplest is applicable to cases in which a certainty is provided with each prediction. Settles [Set09] suggests selecting the data point with the largest uncertainty according to the current model. This has been shown with the toy dataset, as demonstrated in Figure 2.2.

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.

As addressed by Settles [Set09], this can be extended to any probabilistic model through 2.2. Settles [Set09] also notes the use of information theory for probabilistic models(**??**), where y_i refers to all possible categorisations 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.

$$x_{\text{next}} = \underset{X}{\text{argmax}} \left[s_{g(X)} \right] \tag{2.2}$$

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 refered to minimising the region of disagreement, refering the spaces of discord within the hypothesis space. By minimising the region of disagreement between various models, a more coherent hypothsis space is sought leading to a more accurate model. Indeed, this was the method used in Figure 2.2. Mathematically, a set of n models $M = \{m_0, \ldots, m_{n-1}\}$, with each model offering a precition of $\hat{m}_i$, $\hat{M} = \frac{1}{n}\sum \hat{m}_i$, and the sample standard deviation of $\hat{m}$ giving the uncertainty.

Settles [Set09] suggests third way of interpreting uncertinty. By taking the approach from information theory, 2.3 is settled upon. This directly states gives the point of highest entropy, suggesting by knowing the point provides the largest information gain. Notably 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.

$$x_{\text{next}} = \underset{x}{\text{argmax}} \left[\phi_A(x) \times \left(\frac{1}{U} \sum \text{sim}\,(x, x_i) \right)^{\beta} \right] \tag{2.3}$$

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). Settles and Craven [SC08] suggest (2.4) which can be broken down into two parts: a function for selection, ϕ_A,

and a function for similarity, sim. The former arises from another method described in this section. The latter requires a function to describe the similarity between data points.

$$x_{\text{next}} = \underset{x}{\text{argmax}} \left[\phi_A(x) \times \left(\frac{1}{U} \sum \text{sim}\,(x, x_i) \right)^{\beta} \right] \tag{2.4}$$

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].

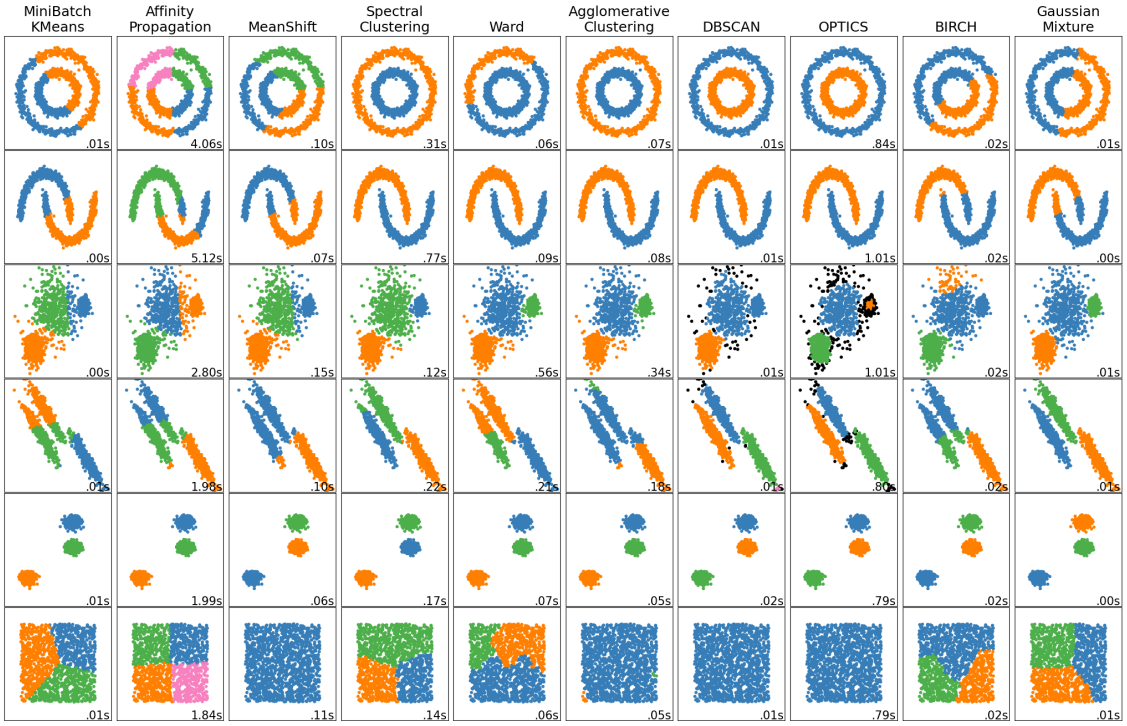


Fig. 2.3 Clusterisation algorithms used on sample two-dimensional data sets to demonstrate resultant clusters.

As Figure 2.3 demonstrates, there are multiple different interpretations of the solution to the problem of clustering. The makers of the Sci-kit 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.

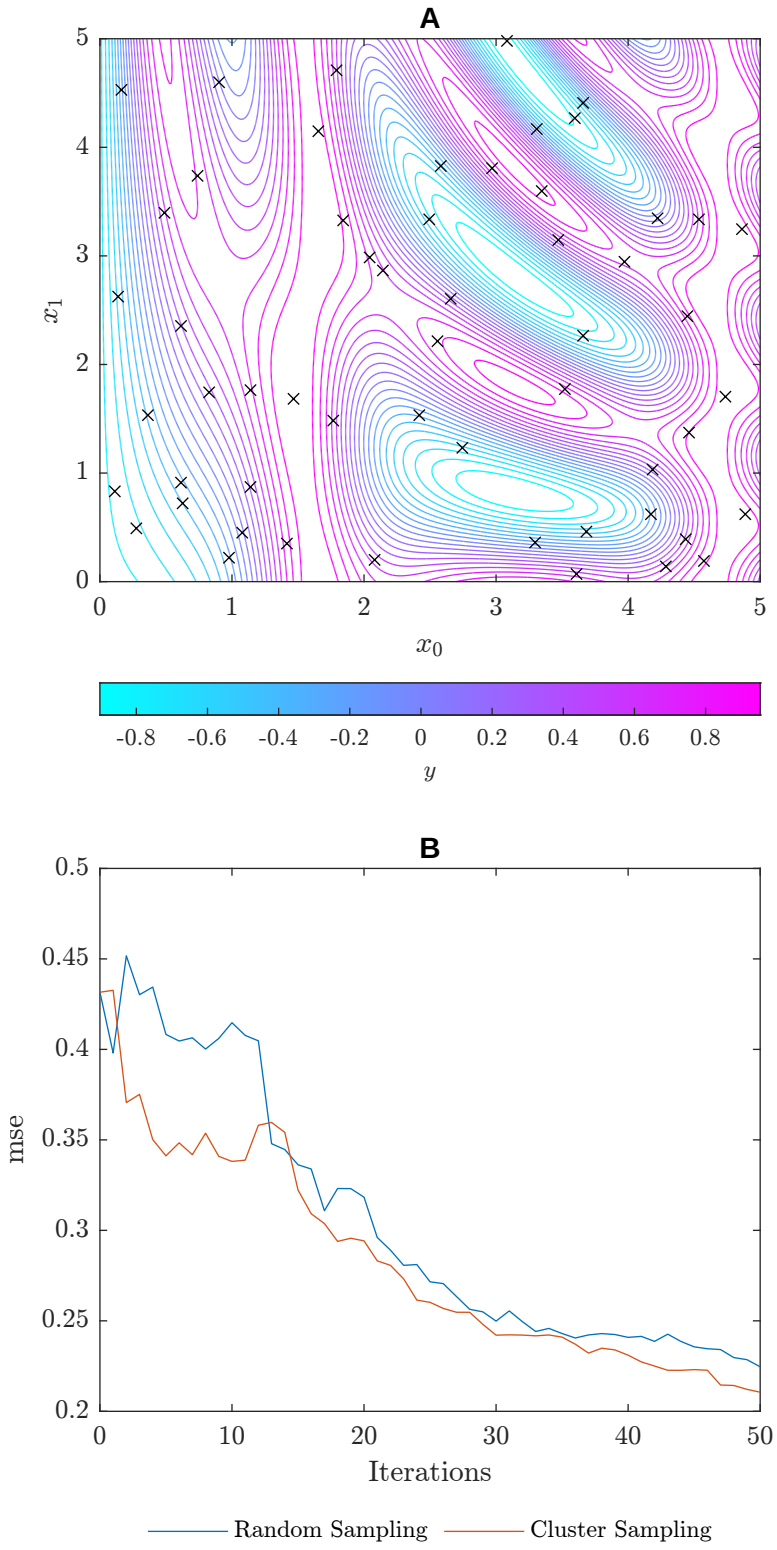


Fig. 2.4 The outcome of the investigating the areas of using a cluster hotspot sampling methodology. An initial set of 5 random points was provided, and 50 further iterations were then carried out of sample size 1. A) Demonstrates the final set of points tested by the algorithm and B) shows the change in 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 lenth 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 2.5.

$$x^*_{EGL} = \underset{x}{\operatorname{argmax}} \sum_i P(y_i|x;\theta) \, \|\nabla l(\langle x, y_i \rangle; \theta)\| \tag{2.5}$$

2.2 Batch Active Learning

Little literature exists with respect to batch active learning. Naive implimentation 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 broad and uncertainty sampling methods, although it is not present within the clusterisation algorithm.

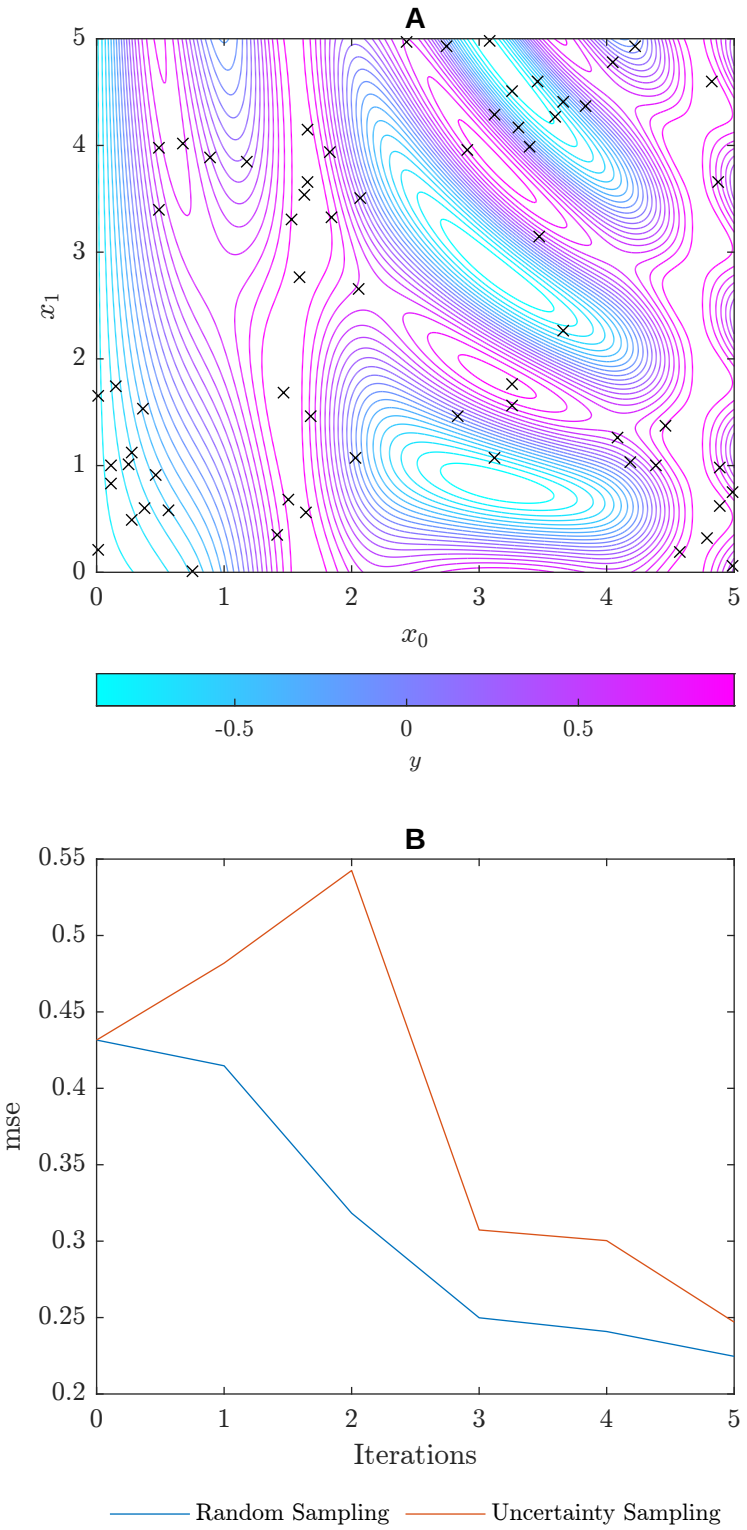


Fig. 2.5 The outcome of the investigating the areas of using uncertainty sampling. An initial set of 5 random points was provided, and 5 further iterations were then carried out of sample size 10. A) Demonstrates the final set of points tested by the algorithm and B) shows the change in the mean squared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

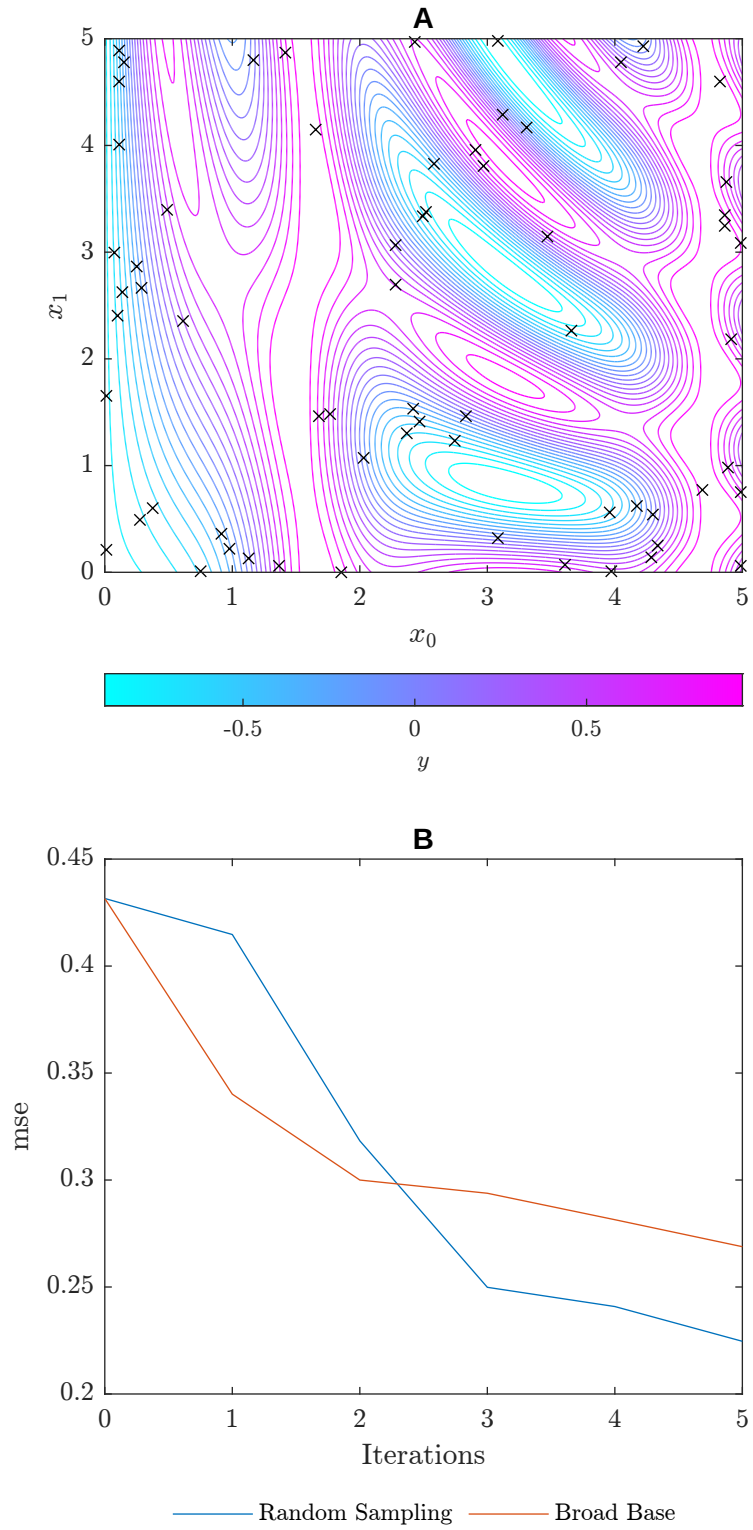


Fig. 2.6 The outcome of the investigating the areas of using broad-base sampling. An initial set of 5 random points was provided, and 5 further iterations were then carried out of sample size 10. A) Demonstrates the final set of points tested by the algorithm and B) shows the change in the mean squared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

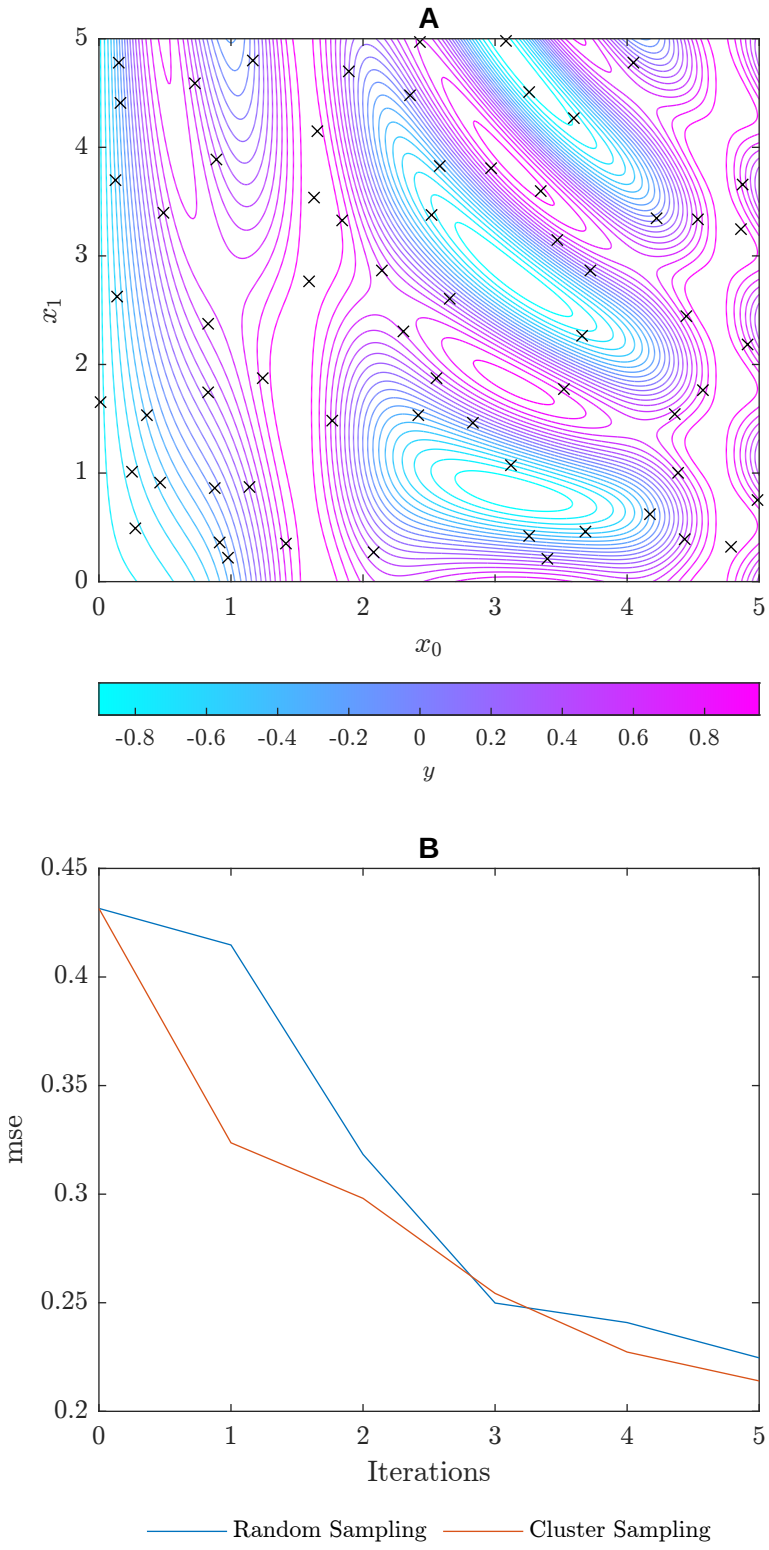


Fig. 2.7 The outcome of the investigating the areas of using cluster sampling. An initial set of 5 random points was provided, and 5 further iterations were then carried out of sample size 10. A) Demonstrates the final set of points tested by the algorithm and B) shows the change in the mean squared error for the algorithm after each iteration.

It stands to reason that the area which has the highest uncertainty will see this for the data points nearest neighbours. Thus, this singular data point suffers the potential of being surrounded by $N-1$ other data points. The benefit this provides in fitting the model is thus extremely limited, and only slightly greater than if one data point had been chosen. A simple fix would be to simulate the model after 1 iteration, and select the next point from here. By doing this $N-1$ times, a better solution may be found, although this may prove to be computationally very expensive.

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 trully 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 2.6 where "|" is synonymous with "or".

$$\text{pChEMBL} = -\log_{10}\left(\text{IC}_{50}|\text{XC}_{50}|\text{EC}_{50}|\text{AC}_{50}|\text{Ki}|\text{LD}_{50}|\text{Potency}\right) \tag{2.6}$$

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].

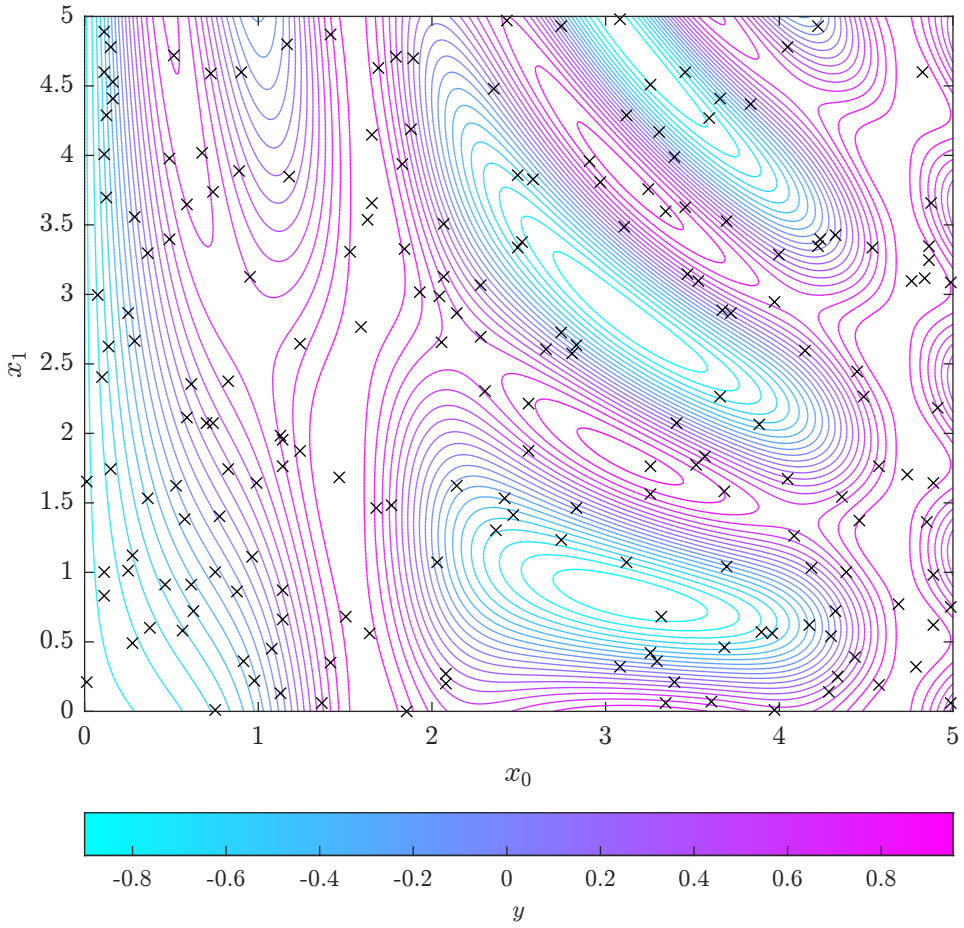


Fig. 2.1 Contour plot of the function used to demonstrate the algorithms presented in previous work. The crosses have been used to show the location of the 200 test data points used within this example.

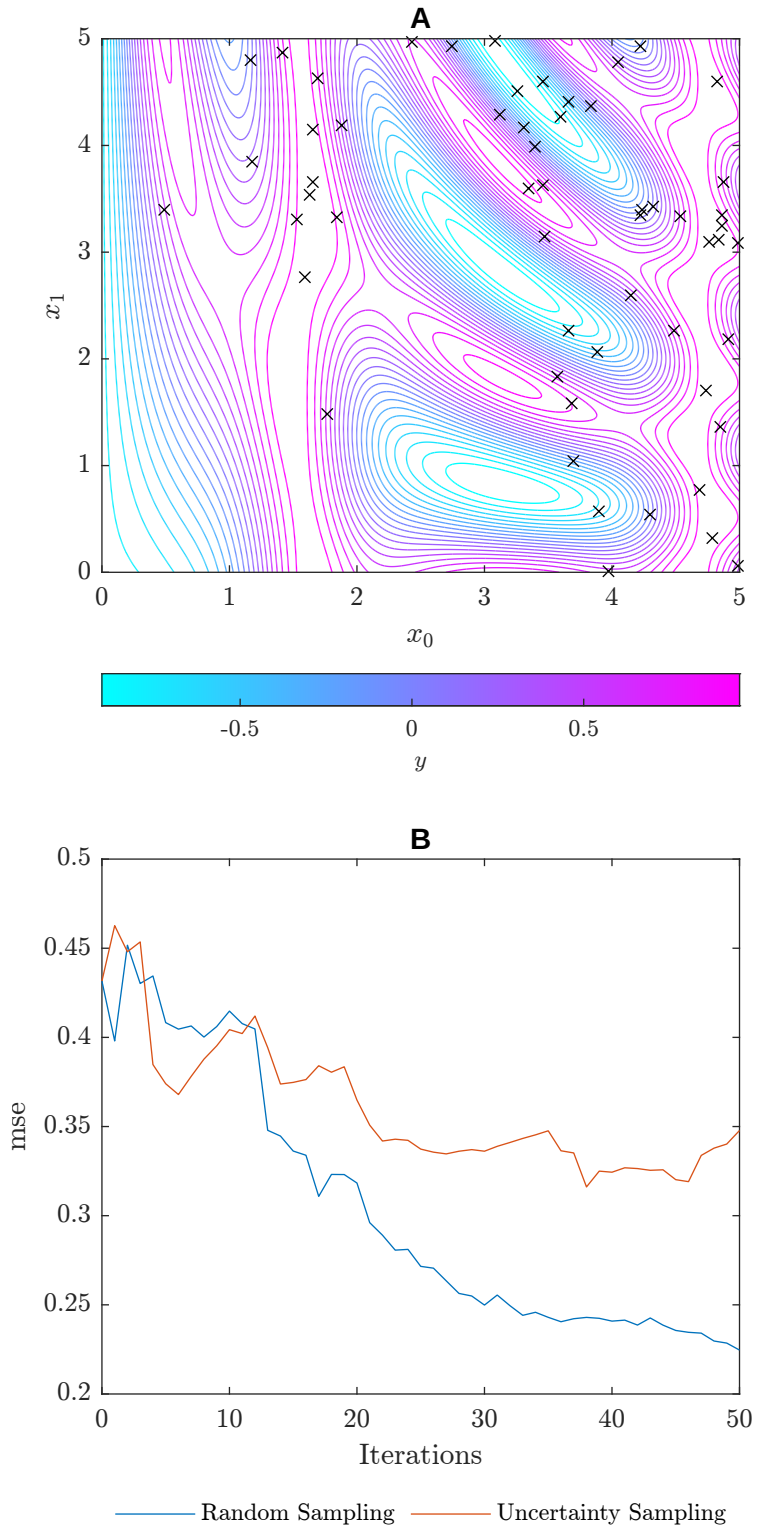


Fig. 2.2 The outcome of the investigating the areas of the highest uncertainty. An initial set of 5 random points was provided, and 50 further iterations were then carried out of sample size 1. A) Demonstrates the final set of points tested by the algorithm and B) shows the change in the mean squared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

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]. 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 added, as will be discussed later within the chapter. 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 2.6, is used due to comparibility and easy interface with EMBL-EBI [EMB09].

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 codebase 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 a large 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 sci-kit'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 sci-kit'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 forrest regressor, stochastic gradient descent regressorwith huber loss, epsilon-support vector regression, and ada-boost regressor [Ped+]. This was kept consistent during testing, allowing for direct comparison of the algorithms.

3.2.2 Scoring

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

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

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.

3.2.3 Active Learning Algorithms

The algorithms tested are all provided with x_{known}, x_{known}, $x_{\text{Y}_\text{u}\text{nknown}}$, a model fitted to x_{known} and y_{known}, and a memory object to allow for information kept over iterations is required.

This is useful for clusterisation, 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 $y_{\mathrm{Y_k nown}}$ to infuence an algorithms scoring process. The algorithms then return a list in the same order as y_{unknown}, with 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 computationally least 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})

Greedy

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 in shown is Algorithm 2.

Algorithm 2: Greedy 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 (RoD)

Similarly to the greedy algorithm, this is a very simple algorithm. Here, the predict_error() method (see Table 3.1) is used to return a prediction and a standard deviation. The prediction

is ignored, and instead the standard deviation is returned, multiplied by -1 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 byall 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 $\text{len}(X_{\text{unknown}}) + 1.5n$. The combined limits have been shown in 3.2.

$$\text{len}(X_{\text{known}}) < c < \text{len}(X_{\text{unknown}}) + 1.5n \tag{3.2}$$

Algorithm 4: Uncertainty Sampling Selection

Data: z_{known}, z_{unknown}, c
Result: Score of datapoints
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});
indicies = z_{known}.index;
sorted_indicies = sort(indicies -> By cluster size followed by distance to centroid) ;
high_priority, low_priority = split(sorted_indicies, 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 percieved as known clusters so ideally would not undergo further testing. Secondly, the sorting needs to be adressed. Here, the sample is sorted into the relavent 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 cluster 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}.

RoD with Greed (RoDG)

This is the first composite function, combining both the greedy sampling, and the uncertainty sampling algorithms. This metric is shown in 3.3.

$$\text{score}_{\text{greedy\&uncertainty}} = \text{score}^{\alpha}_{\text{greedy}} \text{score}^{1-\alpha}_{\text{uncertainty}} \tag{3.3}$$

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

Holy Trinity

This is a second order composite function, involving RoD with greed and Cluster III, as shown in 3.4.

$$\text{score}_{\text{HolyTrinity}} = \text{score}^{\alpha}_{\text{ClusterIII}} \text{score}^{1-\alpha}_{\text{RoDG}} \tag{3.4}$$

Both of the constituant algorithms are paramatised, 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 parallised 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 guid 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-parametised 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 WMSE of 0.184 ± 0.018.

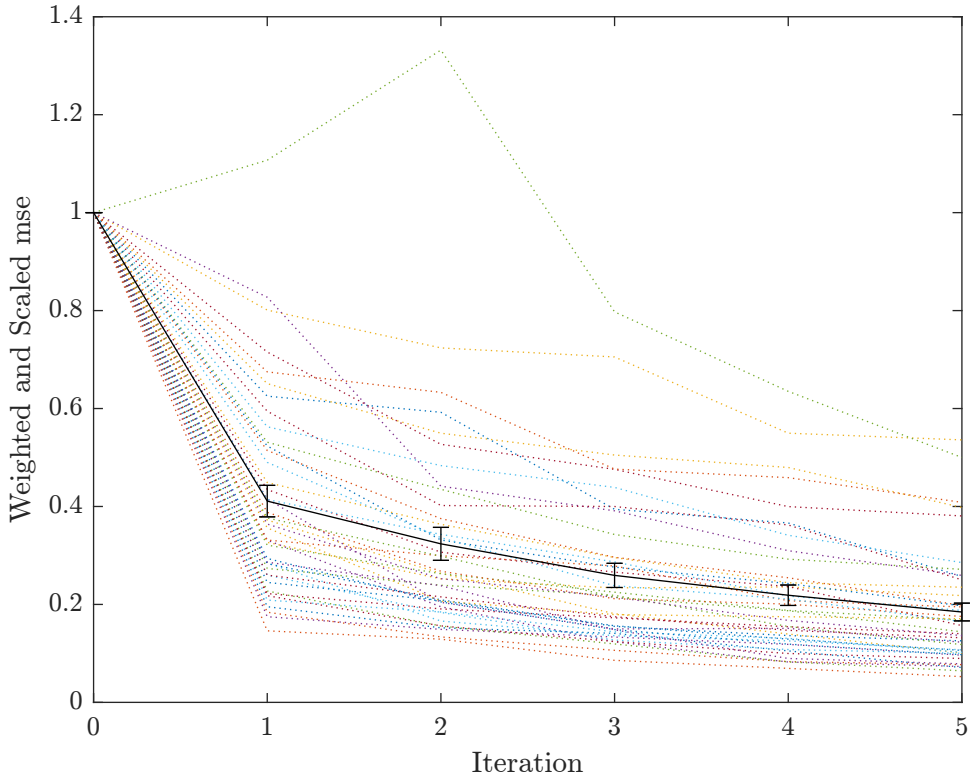


Fig. 4.1 Results of Monte Carlo sampling on the test datasets. Dotted lines represent the individual scoring for the data sets and the solid line shows the mean results at each iteration with error bars of $\frac{\sigma}{\sqrt{n-1}}$.

4.1.2 Greedy

Likewise, the Greedy algorithm was tested, with results presented in Figure 4.2. Here, a final WMSE of 0.323 ± 0.039 was found indicating a worse scoring than the base case.

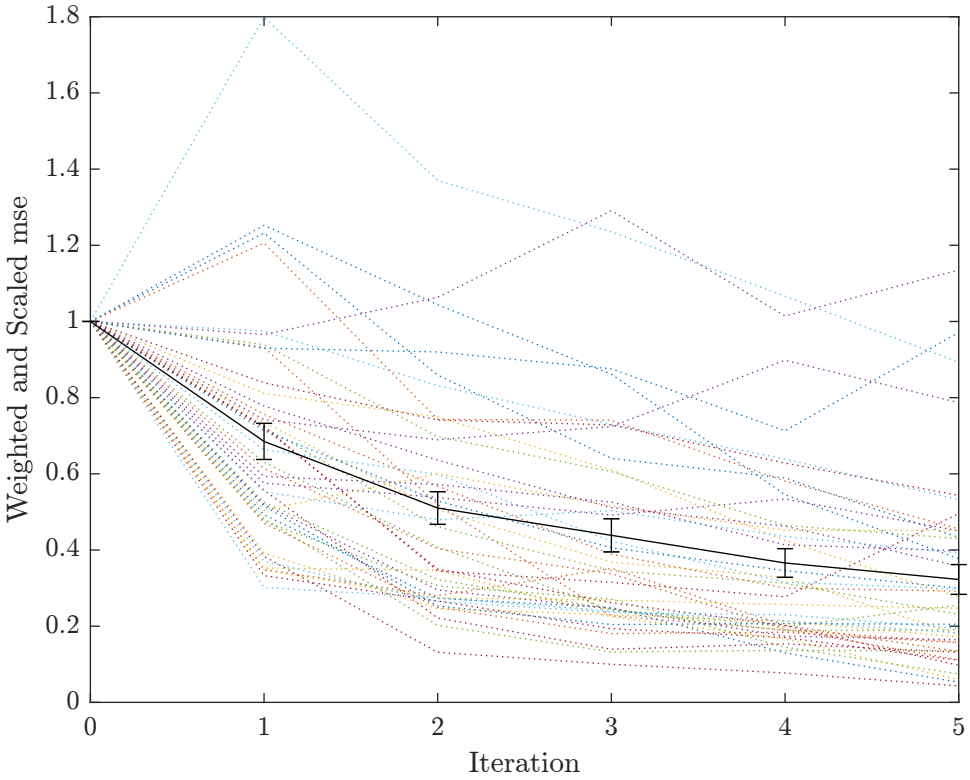


Fig. 4.2 Results of greedy sampling on the test datasets. Dotted lines represent the individual scoring for the data sets and the solid line shows the mean results at each iteration.

4.1.3 RoD Sampling

The final non-parametic algorithm to be tested was RoD. A final WMSE of 0.211 ± 0.022, leading to a middling position between the other two parametric algorithms. The imporvement in each iteration is shown in Figure 4.3.

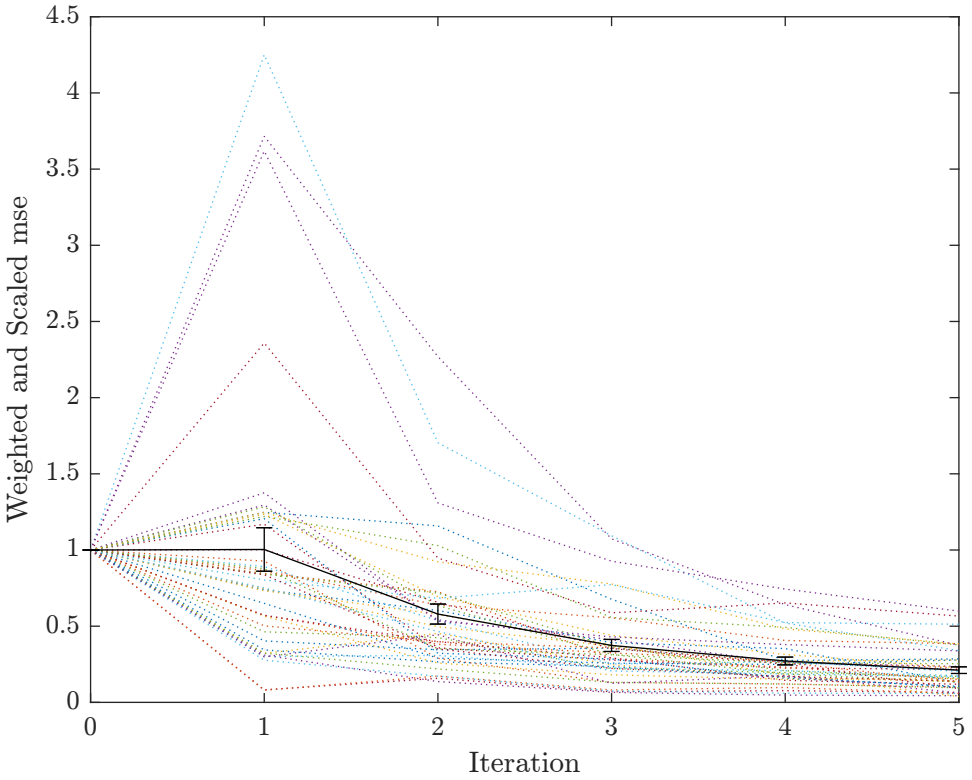


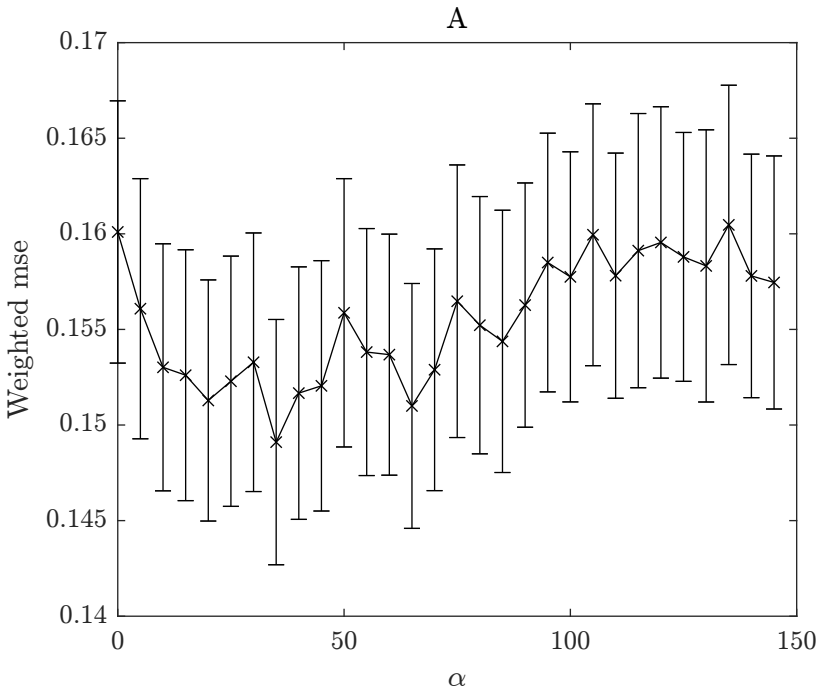
Fig. 4.3 Results of RoD sampling on the test datasets. Dotted lines represent the individual scoring for the data sets and the solid line shows the mean results at each iteration.

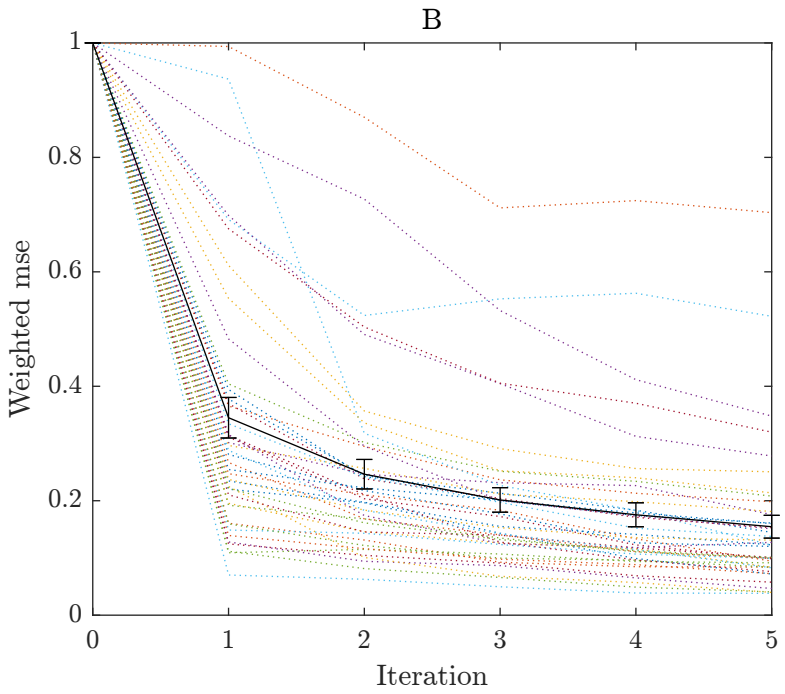
4.2 Parametric

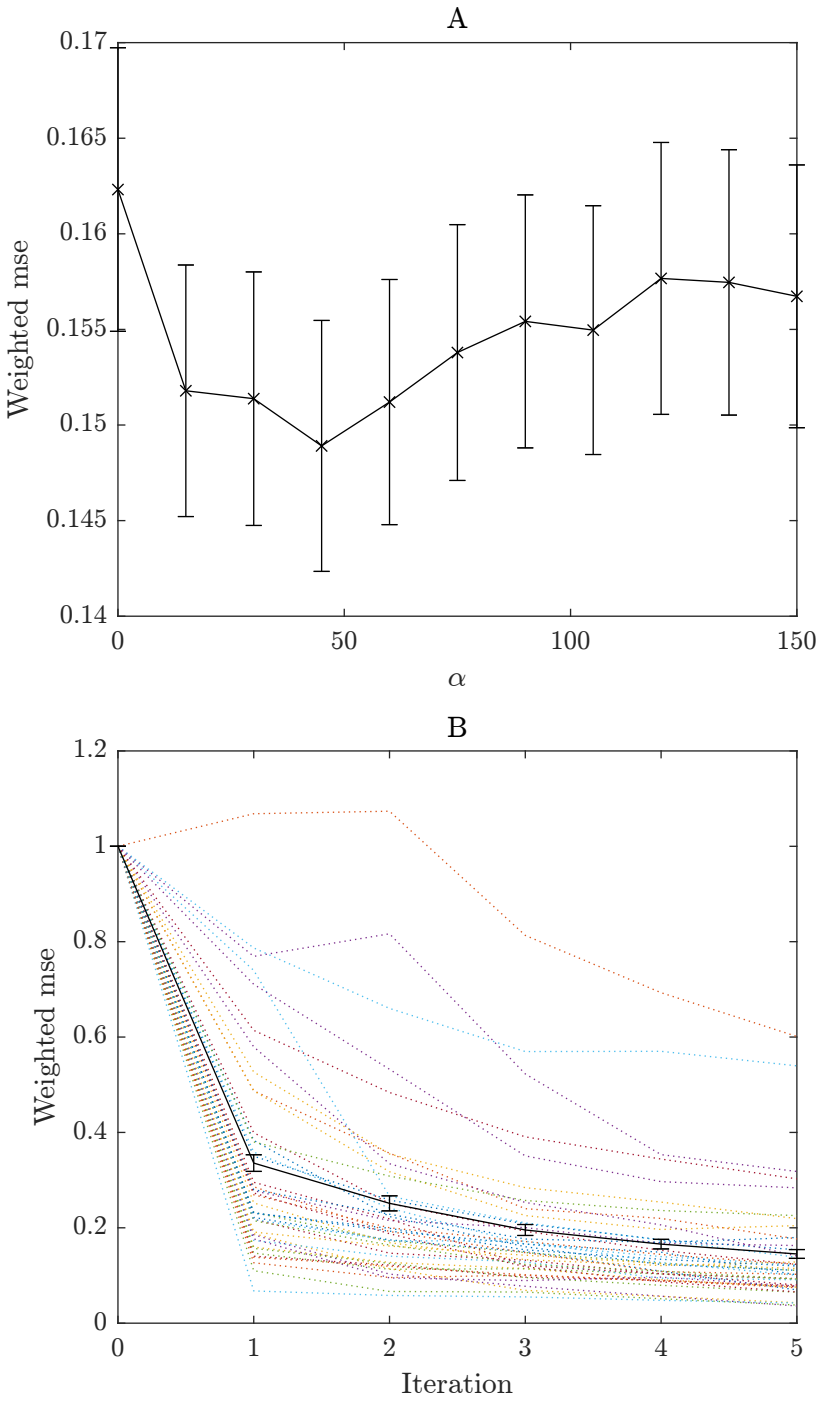
Parametric algorithms require a minimisation procedure on the training set. This leads to a computationally challenging script, and for this the author is grateful for the services provided by the HPC [Uni22].

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 WMSEs of 0.155 ± 0.020, 0.145 ± 0.009, and respectively 0.143 ± 0.016. 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 aditional cluster size of 45, 40, and 60 were found to be optimal for Cluster I, II, and III respectively.


A
Weighted mse
0.17
0.165
0.16
0.155
0.15
0.145
0.14
0
50
100
150
α


B
Weighted mse
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
Iteration


A
Weighted mse
0.17
0.165
0.16
0.155
0.15
0.145
0.14
0
50
100
150
α
B
Weighted mse
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
Iteration

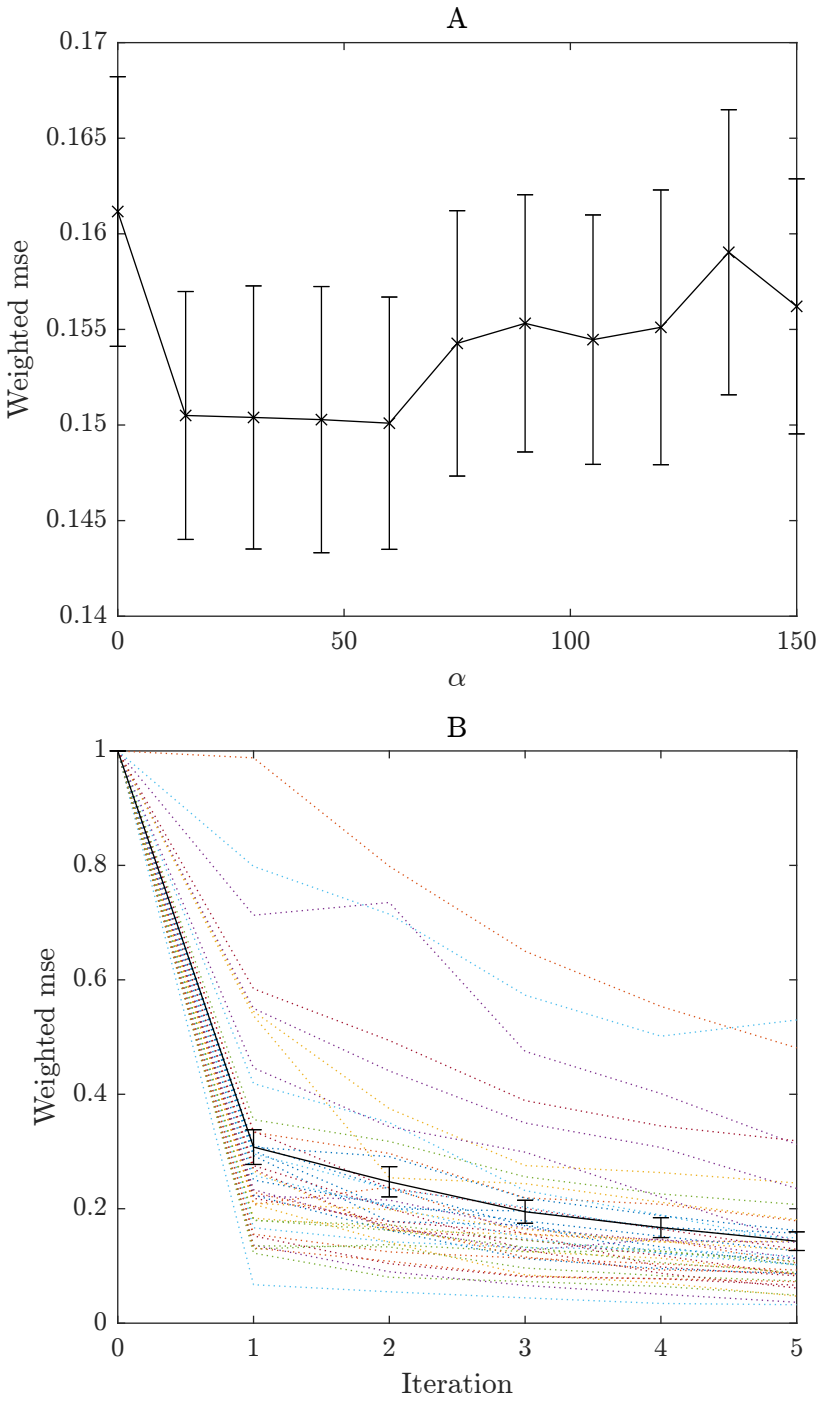


Fig. 4.4 The results of fitting c to the Cluster III algorithm. A) shows the results from parameter fitting and B) shows the learning process on the test set.

4.2.2 RoD with Greed

When testing the RoD with greed sampling method, it was found that despite the weighting towards higher value targets, no improvement was seen over ROD with $\alpha = 0.06$, with α defined in 3.3. However, the tolerance at small α, as shown in Figure 4.5A. This mehtod gave a final a score of 0.206 ± 0.011, a slight improvement over simply using RoD.

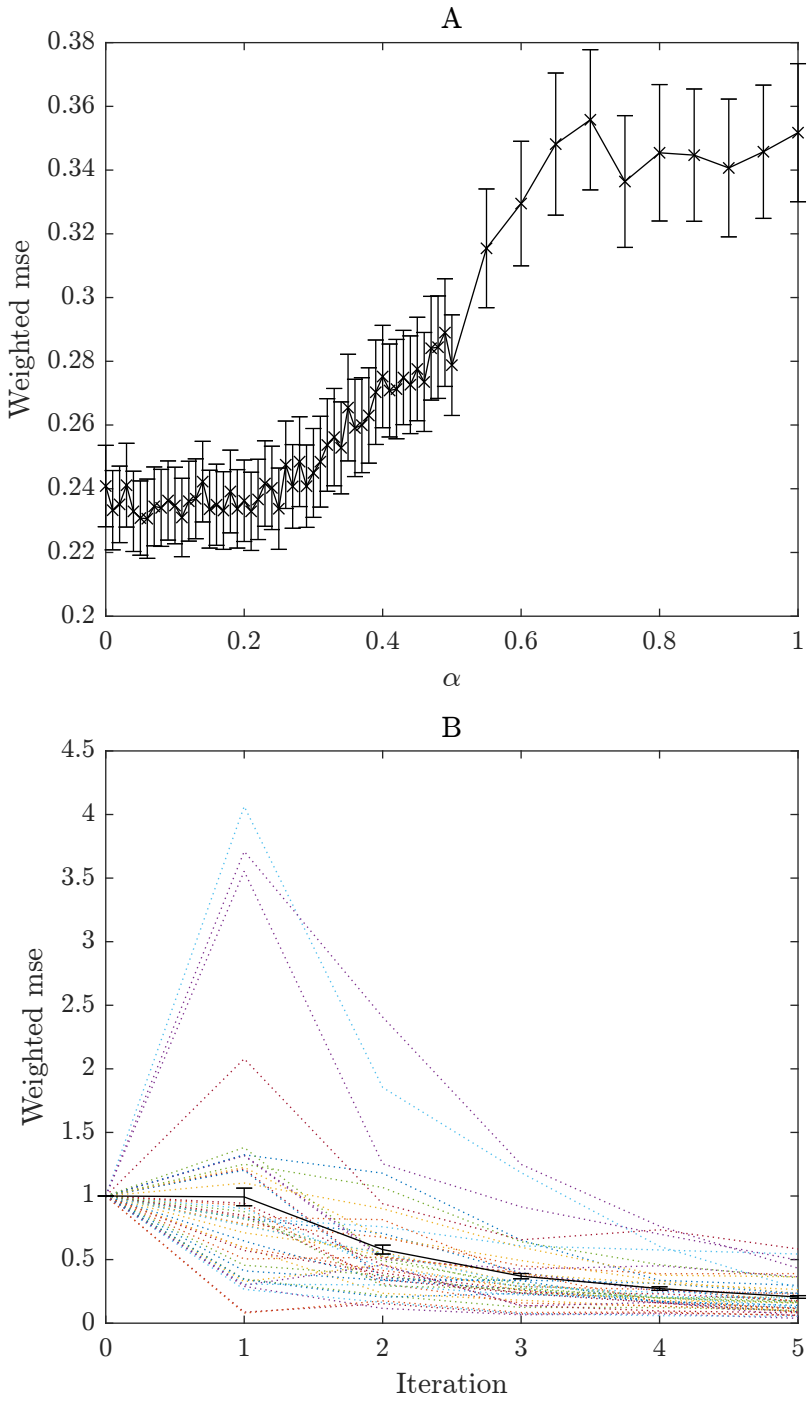


Fig. 4.5 The results of fitting α to the RoD with Greed algorithm. A) shows the results from parameter fitting and B) shows the learning process on the test set.

4.2.3 Holy Trinity

By sampling multiple values for α, a final set of $\alpha = [60, 0.4, 0.2]$ was reached where $[\alpha_1, \alpha_2, \alpha_3]$ correspond to the constants used in Algorithm 4, 3.3, and 3.4 respectively. When validated against the testing datasets, a final result of wmse $= 0.116 \pm 0.014$ was found; the best result.

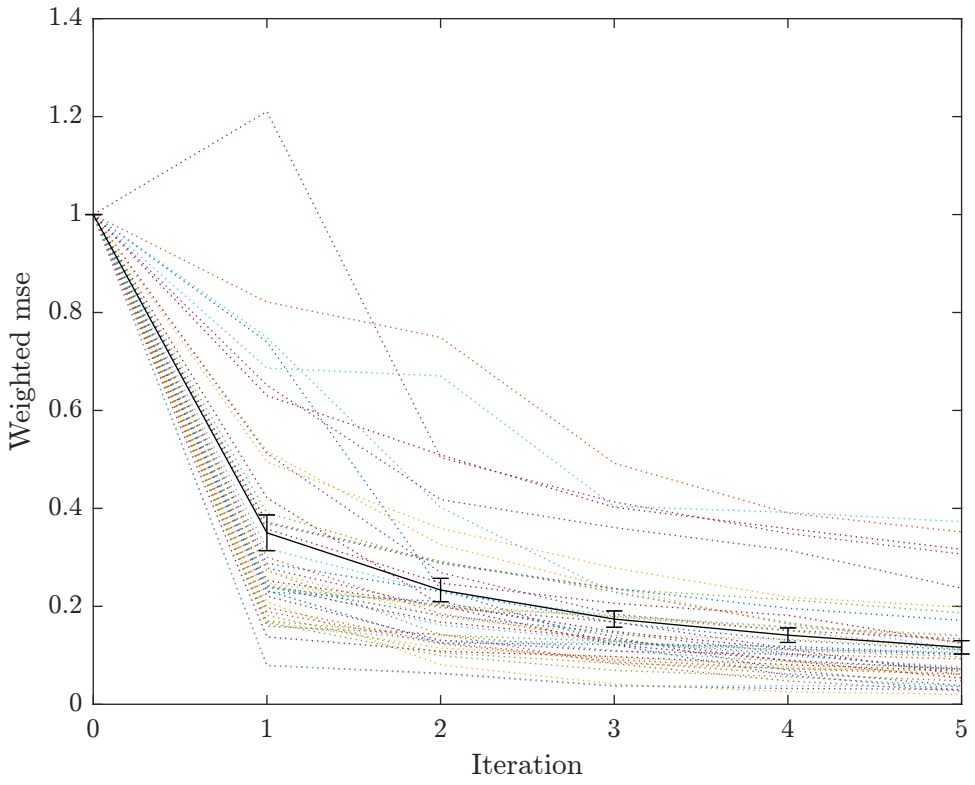


Fig. 4.6 The results of fitting α to the Holy Trinity algorithm, showing the learning process on the test set. The learning of each data set has been added with dotted lines for illustative purposes.

Chapter 5

Discussion

5.1 Non-Parametric

Three algorithms tested of non-parametric variety producing 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. An extra iteration may indeed have seen RoD surpassing Monte Carlo. This is expected as the ROD algorithm specifically targets regions of the model which are challenging causing the largest changes towards proper 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.

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

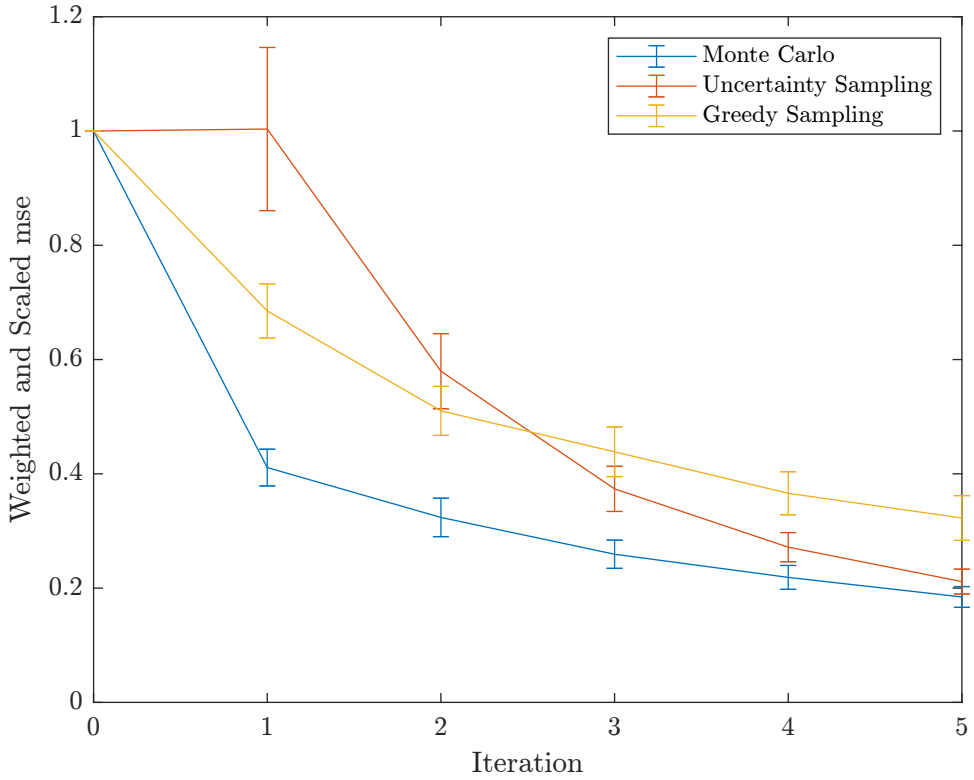


Fig. 5.1 Comparison of different non-parametric algorithms with standard deviations represented as error bars.

5.2 Parametric

Different classes of parametric algorithms were tested. The first of these is a first order composite algorithm, RoD with Greed: i.e. 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, affectionately named the Holy Trinity. A constant improvement is seen throughout these algorithms, with the Holy Trinity performing the best.

Several points of interest are highlighted with these results. Firstly, despite improving upon RoD, RoD with Greed is still beaten by Monte Carlo sampling. Again, the methodology suffers from clustering of points. This is shown with the ability of Cluster III outperforming Monte Carlo. The progression through the different cluster algorithms also sees improvement with progression, as anticipated.

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 Holy

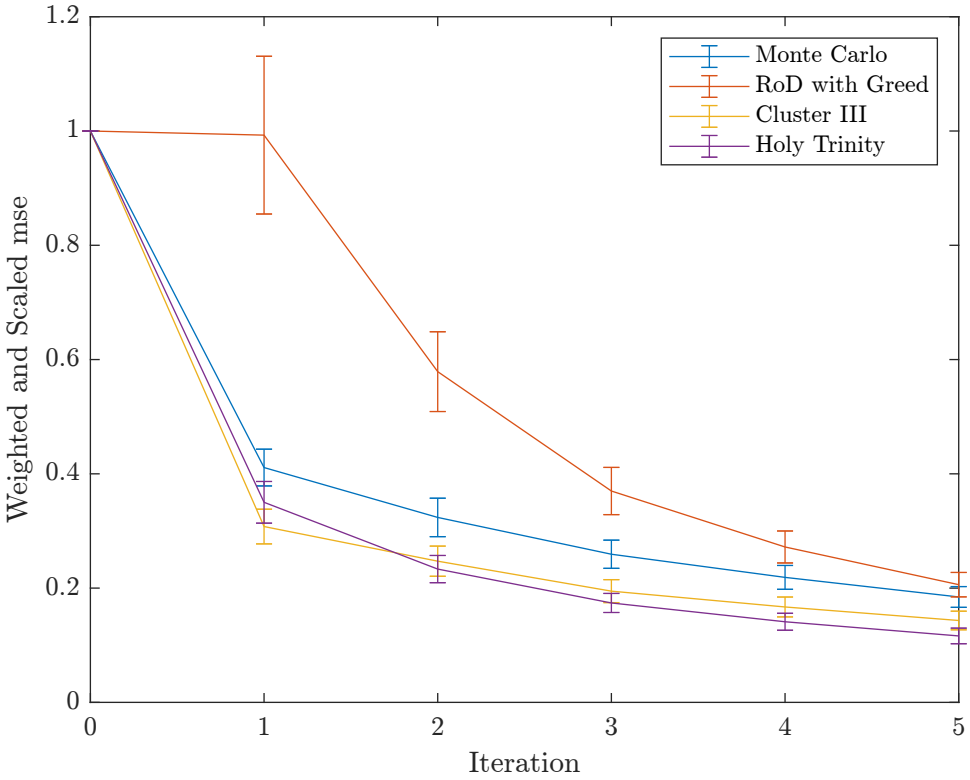


Fig. 5.2 Comparison of different parametric algorithms with standard deviations represented as error bars.

Trinity algorithm sacrifices some of this initial performance for longer term gain, as shown by the worse performance after the first iteration. By the second iteration, the difference becomes insignificant, with the error for each algorithm. By the end of the fifth iteration, the Holy Trinity algorithm convincingly outperforms the other algorithms.

Upon investigation of the parameters settled upon within these several points arise. Firstly with RoD with Greed, at low α, WMSE appears uncorrelated with α, only experiencing a significant rise with $\alpha > 0.4$. 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 extremely 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 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.

Chapter 6

Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam,

luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

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.

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).

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).

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).